Robust balancing of production lines : MILP models and pre-processing rules

Aleksandr Pirogov

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'École Nationale Superieure Mines-Télécom Atlantique Bretagne Pays de la Loire - IMT Atlantique
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Aleksandr PIROGOV

Équilibrage robuste de lignes de production

Modèles de programmation linéaire en variables mixtes et règles de pré-traitement

Thèse présentée et soutenue à IMT Atlantique, le 20 Novembre 2019
Unité de recherche : Le Laboratoire des Sciences du Numérique de Nantes
Thèse N° : 2019IMTA0158

Composition du Jury :

Présidente :	Marie-Laure ESPINOUSE	Professeur, G-SCOP, Université Grenoble Alpes
Rapporteurs :	Alexis AUBRY	Maître de conférences HDR, CRAN, Université de Lorraine
	Olga BATTAÏA	Professeur, KEDGE Business School, Talence
Examinateurs :	Öncü HAZIR	Maître de conférences, Rennes School of Business
	Mikhail Y. KOVALYOV	Professeur, UIIP, NASB, Minsk, Belarus
Dir. de thèse :	Alexandre DOLGUI	Professeur, LS2N, IMT Atlantique, Nantes
Co-dir. de thèse :	André ROSSI	Professeur, LAMSADE, Université Paris-Dauphine
Co-encadrant :	Evgeny GUREVSKY	Maître de conférences, LS2N, Université de Nantes

Remerciements

I would like to thank Professors Alexis Aubry and Olga Battaïa, reviewers of my thesis, for their time and their feedback on my work. I thank Professor Marie-Laure Espinouse, for agreeing to be an examiner and president of my thesis defence. I am grateful to Professors Öncü Hazır and Mikhail Y. Kovalyov for their interest in my work and participation as examiners. I have a special thank to Professors Jean-Charles Billaut and Marc Sevaux or having followed my work regularly within the framework of the thesis individual monitoring committee (comité de suivi individuel).

My thesis would not have been possible without my supervisors. Evgeny Gurevsky, who helped me a lot with general and administrative questions during my first year in France and provided daily guidance in research for all three years. André Rossi, which despite the fact that we are not in the same city (Angers at the beginning and Paris in the end), has always been available to discuss our work, with the optimism that characterises it. Alexandre Dolgui for every piece of advice and inspiration after every work meeting.

Here, I also want to mention my maths teacher at high school – Tatyana Alexandrovna Dyachenko; and my supervisor during the master thesis – Anna Anatolyevna Romanova. They helped me to make my first steps into science and research.

I want to extend my thanks to the team SLP (Systèmes Logistiques et de Production) at LS2N (Laboratoire des Sciences du Numérique de Nantes) for hospitality and the atmosphere. I don't forget Axel, Guillaume and Oussama without which I would have lost myself in some professional and personal situations. I also thank them for their daily welcome and good humour.

I appreciate partial financial support by the council of the french region "Pays de la Loire".

I am pleased to everyone I met throughout my thesis: Ekaterina, Ka Yu, Jiuchun, Xiao, Igor, Konstantin, Maxim, Ilhem, Ziwei, Houda, Karim, Ehsan, Hamidreza, Pooya, Mahsa, Samim, Loubna, Hiba, Paula and many others. Thank you, friends, for all the time and moments we spent together.

Finally, but not least, I thank my family for their support. My parents for their wise advice and their help in difficult times, since always. My Russian friends for their support, over a distance of 6000 kilometres.

À mes parents
et à mes amis.

Résumé de la thèse

Contexte

Dans l'industrie contemporaine, les systèmes de fabrication jouent un rôle crucial. La population mondiale augmente, créant une demande croissante de produits et de services. Pour répondre à cette demande et relever le défi de la concurrence, chaque entreprise doit concevoir avec soin son système de fabrication qui soit à la fois efficace du point de vue de la qualité des produits fabriqués et des capitaux investis. L'un des systèmes de fabrication les plus courants dans l'industrie est la chaîne de production, elle est constituée de stations alignées de manière séquentielle. Les décisions relatives à la conception des lignes de production revêtent une grande importance car le coût de la mise en place d'une ligne de production se chiffre en millions d'euros.

La conception des systèmes de production comprend plusieurs étapes. Nous nous sommes focalisés sur l'étape consistant à équilibrer la chaîne de production. Pour un ensemble donné de postes de travail, cette étape suit l'analyse des produits à fabriquer et la planification du processus de fabrication, qui déterminent les informations sur le traitement du travail dans le système conçu, c'est-à-dire un ensemble de tâches indivisibles liées par certaines contraintes. Ces contraintes proviennent de considérations technologiques, économiques et environnementales ou de facteurs ergonomiques pour les travailleurs. L'équilibrage de ligne consiste à choisir les ressources et à les affecter aux postes de travail, de manière à ce que toutes les contraintes soient satisfaites et que les objectifs prédéfinis soient atteints. Traditionnellement, ces objectifs peuvent être, par exemple, de minimiser le coût total ou de maximiser la productivité de la ligne. Les problèmes rencontrés au stade de la configuration des chaînes de fabrication sont de nature combinatoire. Le concepteur de la ligne recherche la ou les solutions optimales parmi d'autres solutions possibles. La tâche est ardue car le nombre de solutions envisageables est énorme et le choix d'une méthode appropriée pour résoudre ce problème est décisif. Il est attendu de la méthode retenue qu'elle permettre d'obtenir des solutions de haute qualité à un coût de calcul relativement faible. De plus, certaines incertitudes, suscepti-

bles de compromettre la faisabilité d'une solution, doivent être prises en compte afin de protéger la solution contre les conséquences de ces incertitudes.

Motivation

Dans cette thèse, nous considérons le problème d'équilibrage de la ligne de production sous incertitude. Elle consiste à affecter des tâches de production à un nombre fixe de stations, de sorte que la ligne fonctionne avec une productivité donnée déterminée par le temps de cycle. Le temps de cycle est la durée qui sépare la production de deux produits consécutifs, en régime permanent. Toutes les tâches sont spécifiées par un temps de traitement estimé et un ensemble de contraintes déterminant leur ordre sur la ligne. Nous supposons que les temps de traitement des tâches peuvent différer dans le futur en fonction des changements de caractéristiques du produit, de la fatigue des employés, de leur expérience, ou pour d'autres raisons. Ces événements peuvent compromettre la faisabilité de la solution ou détériorer les performances. Le problème consiste donc à maximiser la stabilité de la ligne de production. Nos objectifs sont de développer une méthode efficace pour maximiser la capacité de la ligne à maintenir la faisabilité malgré l'incertitude affectant la durée des tâches, et de montrer son application pour des lignes de montage et de transfert simples. La méthode proposée consiste à étudier une nouvelle approche, introduite dans des travaux récemment publiés sur l'optimisation robuste, qui est plus appropriée pour traiter les incertitudes dans le domaine des systèmes de production que celles utilisées jusqu'à présent. Cela nous conduit à étudier de nouveaux problèmes d'optimisation industrielle représentant un objet d'étude essentiel, à la fois d'un point de vue pratique et algorithmique.

Contributions

Les contributions de cette thèse s'appuient sur les observations présentées dans le paragraphe précédent. Le problème d'équilibrage des lignes de production est le problème support de ces travaux. Le manuscrit s'articule de la manière suivante:

- Le Chapitre 1 présente l'état de l'art sur les problèmes d'équilibrage sous incertitudes. Les applications et les méthodes de résolution dédiées sont décrites.
- Le Chapitre 2 présente le problème d'équilibrage robuste de lignes d'assemblage simples. Plusieurs versions des modèles MILP sont distinguées, en fonction de la

représentation du rayon de stabilité. Les améliorations pour les modèles et la mise en évidence de cas résolvables en temps polynomial sont également proposées.

- Le Chapitre 3 présente le problème d'équilibrage robuste de lignes de transfert. Les formules du calcul du rayon de stabilité pour deux métriques classiques sont introduites. Les modèles de PLNE sont construits et des règles de pré-traitement pour les améliorer sont proposées.

- Le Chapitre 4 présente une approche hybride pour le problème d'équilibrage de ligne de transfert. Cette approche utilise des techniques avancées de solveurs commerciaux pour guider le processus de résolution. Deux types de techniques dans deux solveurs populaires sont considérés et comparés.

- Enfin le dernier chapitre présente un résumé général des contributions et dresse des perspectives ouvertes par ces travaux.

Les Chapitres 2, 3 et 4, qui présentent les contributions de cette thèse, sont détaillés dans les sections suivantes.

Équilibrage robuste de lignes d'assemblage simples

Dans ce type de lignes de production, les stations sont synchronisées par un système unique de convoyage des produits. Les pièces de production qui y sont placées se déplacent du début à la fin de la ligne en passant par toutes les stations dans l'ordre de leur installation. Le problème consiste à affecter les tâches d'assemblage aux stations satisfaisant toutes les contraintes de fabrication. Le nombre de stations est limité et le temps de cycle est fixé. L'objectif de production est de maximiser la mesure de la robustesse, c'est-à-dire le rayon de stabilité. Nous analysons deux métriques classiques pour le calcul du rayon de stabilité et en introduisons une nouvelle. Des théorèmes sont établis et confèrent un cadre formel aux résultats. Nous proposons des modèles de programmation linéaire en variables mixtes, destinés à aider les décideurs à basculer facilement d'un modèle à l'autre en fonction de situations particulières et des connaissances disponibles.

Les modèles de PLNE proposés sont programmés avec C++ à l'aide du solveur commercial Gurobi. Des tests sont effectués sur des instances de la bibliothèque de tests en ligne avec des données supplémentaires générées aléatoirement. Deux paramètres sont utilisés pour établir le niveau d'incertitude entre les tâches et les stations. Les résultats obtenus montrent que les modèles présentés peuvent être utilisés pour des instances industrielles sous certaines conditions de densité du graphe de précédence.

Équilibrage robuste de lignes de transfert

La constitution de lignes de transfert permet de réaliser plusieurs tâches en même temps par la même tête d'usinage. Chaque poste (machine) contient une ou plusieurs têtes d'usinage (blocs), les têtes de chaque poste sont activées de manière séquentielle, les stations comme précédemment sont synchronisées par un convoyeur unique commun. Ainsi, le problème de l'équilibrage de la ligne de transfert consiste en l'attribution de tâches de production à des têtes d'usinage et à la distribution de têtes aux machines répondant à toutes les contraintes. Le critère d'optimisation est la robustesse mesurée par le rayon de stabilité. Seules les métriques ℓ_1 et ℓ_∞ sont prises en compte pour son calcul. Nous formalisons le calcul de ces métriques par l'intermédiaire d'une série de lemmes et de théorèmes. Ensuite, deux nouveaux modèles mathématiques sont présentés. Leur complexité nécessite quelques améliorations pour une exécution efficace. Sur la base de pratiques bien connues concernant les problèmes d'équilibrage de ligne, nous introduisons des intervalles d'affectation des tâches sur des lignes de transfert. Cet intervalle affiche les index des blocs pouvant exécuter la tâche considérée. Cinq règles sont élaborées pour calculer ces intervalles pour deux métriques différentes et sont appliquées dans une approche appelée Reduction.

Une méthode heuristique pour les lignes de transfert est élaborée. Outre l'idée, qui s'apparente aux méthodes classiques de lignes d'assemblage, nous devons prendre en compte la construction de blocs. Pour éviter l'attribution répétitive de tâches, quatre procédures avec différentes restrictions sont construites. Ensemble, elles composent un algorithme heuristique à démarrages multiples qui, à chaque itération, recherche un équilibrage linéaire avec un rayon de stabilité supérieur à la plus grande valeur connue. Tous les développements présentés constituent l'approche globale de pré-traitement, qui génère non seulement une solution initiale et une borne inférieure, mais crée également deux familles d'inégalités pour enrichir les formulations PLNE.

Tous les modèles et algorithmes de pré-traitement sont programmés avec C++ et résolus par le solveur commercial Gurobi. Les données d'instance ont été importées de la bibliothèque d'exemples générés aléatoirement pour les lignes de transfert. Les résultats prouvent l'hypothèse centrale selon laquelle la densité du graphe de précédence a un impact significatif sur la résolution du problème. En général, la méthode présentée montre une capacité à résoudre des instances dont la taille est équivalente aux résultats précédemment rapportés dans la littérature relative aux lignes de transfert.

Approches de solutions hybrides pour le problème d'équilibrage de ligne de transfert

Cette approche consiste en une application combinée d'un solveur PLNE commercial et d'algorithmes de génération de coupes basés sur l'optimalité. Nous utilisons des techniques de callback qui permettent de gérer et de guider le processus d'optimisation. Deux solveurs commerciaux sont considérés afin de trouver celui qui conduit à la résolution la plus efficace. Plusieurs types de callbacks sont étudiés. Nous avons choisi deux approches possibles pour la génération de coupes avec les callbacks. Ils reposent sur la technologie dite des lazy constraints et le MIP restart, respectivement. L'approche par les lazy constraints fonctionne avec un ensemble de coupes qui est introduit progressivement dans le modèle à résoudre. Dans ce cas, les callbacks n'interrompent pas la résolution, et le solveur reprend son travail après l'insertion des coupes basées sur l'optimalité. Les métriques ℓ_1 et ℓ_∞ ont nécessité des implémentations distinctes en raison d'une interprétation différente du rayon de stabilité. Comme pour la méthode de prétraitement globale, nous utilisons la méthode REDUCTION pour déterminer les intervalles d'affectation et les blocs inutilisés sur les machines.

La solution hybride est programmée en C++ et résolue par le solveur commercial GUurobi pour les tests finaux. Des expériences ont été menées sur un nombre plus réduit d'instances qu'au chapitre 3. L'approche MIP restart a montré son avantage par rapport aux lazy constraints et aux opportunités de recherche et développement futurs.

Perspectives

Tous les résultats obtenus révèlent de nombreuses perspectives pour les recherches futures sur l'optimisation robuste des chaînes de production.

- Tout d'abord, nous avons plusieurs options pour améliorer les modèles PLNE en utilisant des techniques avancées de solveurs commerciaux. Mais même dans ce cas, il existe un risque de se heurter à des limites strictes qui ne permettront pas de trouver une solution optimale pour des instances de grande taille. L'une des clés de la résolution de ce problème réside dans le développement de méta-heuristiques telles que : les calculs évolutifs, les algorithmes génétiques et l'optimisation d'essaims de particules.
- D'un autre côté, nous pouvons développer des méthodes exactes efficaces pour trouver une solution optimale au problème. La littérature comprend de nombreux exem-

ples d'algorithmes de branch-and-price et de branch-and-bound pour les problèmes d'équilibrage de ligne.

- Une estimation de bornes superieures pour le problème d'équilibrage de ligne de transfert pourrait être divisée en deux étapes : la détermination des blocs et leur affectation aux machines. Cela rend impossible l'application de certaines formules classiques au calcul puisque la première étape consiste en général en un problème $\mathcal{NP}$-difficile.

- Notre technique de pré-traitement du problème d'équilibrage de la ligne de transfert fait particulièrement référence à la méthode bien connue appelée kernelisation. Il s'agit de pré-traitements au cours desquels les entrées de l'algorithme sont remplacées par des entrées plus petite (noyau). La simplification de la saisie aide les décideurs, car ceux-ci ne disposent pas toujours d'informations exhaustives au stade de la conception. Ainsi, les développements futurs dans ce domaine ont une signification pratique indéniable.

- Un certain nombre de méthodes de recherche opérationnelle ont fait leur preuves dans la programmation par contraintes. Cette évolution est tout à fait naturelle, car leurs objetcifs ont souvent les mêmes. D'autres schémas d'hybridation décomposent le problème afin que CP et OR puissent attaquer les parties du problème auxquelles ils conviennent le mieux. Cette combinaison peut apporter des avantages informatiques substantiels.

- Nous avons pris en compte deux mesures (ℓ_1 et ℓ_∞) pour calculer le rayon de stabilité dans le problème d'équilibrage de la ligne de transfert et une autre (résilience relative) dans le problème d'équilibrage de la ligne d'assemblage simple. Cependant, sa définition classique est basée sur le concept de norme, qui a diverses expressions en science. Par exemple, il existe de nombreuses manières de mesurer la différence entre deux séquences (voir les distances de Levenshtein, Hamming, Lee et Jaro–Winkler). Elles ouvrent des possibilités d'estimation de la stabilité des systèmes de fabrication reconfigurables (y compris les chaînes de production), dont la popularité ne cesse de croître ces dernières années.

Dans cette thèse, nous traitons le problème de la conception de lignes de production en présence de temps de traitement sujets à des incertitudes. Cependant, le problème de la prise en compte de l'incertitude des données au stade de la conception ne se limite pas à la prise en compte des variations de ces données numériques. Ainsi, des modifications possibles peuvent également se produire au niveau de diverses contraintes, telles que les contraintes de précédence. Jusqu'à présent, cet aspect n'est pas encore bien été traité dans la littérature. Ceci est une piste de nos recherches futures dans ce domaine.

Contents

List of Figures

List of Tables

Introduction

In the contemporary industry, manufacturing systems play a crucial role. The world's population is increasing, creating a growing demand for products and services. To meet this demand and take the challenge of competition, each company must carefully design its manufacturing system that will be both effective for the quality of manufactured products and the capital expended. One of the most common manufacturing systems in the industry is the production line, it consists of stations aligned sequentially. Decisions related to the design of the production lines are of great importance because the cost of setting up a production line is in the millions of euros.

The design of production systems includes several steps. We are particularly interested in the stage of balancing the production line. For a given set of workstations, this step follows the analysis of the products to be manufactured and the planning of the manufacturing process which determine the information about the work processing in the designed system, *i.e.*, a set of indivisible tasks related by some constraints. These constraints come from technological, economical and environmental considerations or ergonomic factors for the workforce. Line balancing consists of choosing the resources of and assign them to the workstations, so that all constraints are satisfied and predefined objectives are met. Traditionally, these objectives can be, for example, to minimise the total cost or to maximise the productivity of the line. Problems occurring at the configuration stage manufacturing lines are combinatorial in nature. The designer of the line is looking for the optimal solution(s) among other feasible solutions. It is a difficult task as the number of feasible solutions is huge, so the choice of an appropriate method to tackle this problem is decisive. The selected method is expected to return high-quality solutions at a reasonably low computational cost. Moreover, some uncertainties, which may compromise the feasibility of a solution, must be taken into account in order to avoid this pitfall.

In this thesis, we consider the production line balancing problem under uncertainty. It consists of assigning of production tasks to the fixed number of stations, such that the line is going to work with a given production rate determined by cycle time. The cycle time shows how fast the line produces a new piece of product. All tasks are supported by an estimated processing time and a set of constraints determining their order on the line. We assume that task processing times can deviate in the future according to product characteristic changes, employee fatigue and experience or something else. These events

may compromise solution feasibility, or deteriorate performance, so the stability of the line balance is the problem objective. Our goals are to develop an effective method for maximising the line ability to maintain feasibility despite task time deviation, and to show its application for simple assembly and transfer lines. The proposed method consists in studying a new alternative approach, introduced in recently appeared works on robust optimisation, that is more appropriate for handling uncertainty in the field of production systems than those used up to now. This leads us to investigate new industrial optimisation problems representing an essential object of study from both practical and algorithmic points of view. In what follows, we present the structure of the thesis, which is made of four chapters.

Chapter 1 presents the study of existing production lines, their parameters and attributes. We show which types of production lines exist and why it is important to know how to balance them. Three common objective functions minimising the number of stations, cycle time and line efficiency are highlighted. Events modifying processing times of tasks are considered. Their impact on a line balance and its productivity is pictured in series of examples. Possible task time deviations have to be taken into account at the design stage in order to keep the line working after installation. We consider uncertainties that can be faced during the line balancing process. In the second part of the chapter, we analyse existing approaches to model data uncertainty in combinatorial optimisation problems. We identify the robust one that best fit the context of the design of production lines. It is based on the stability radius, which shows the maximal value of processing time deviations that do not violate solution feasibility or optimality. In the end, we introduce a new robust formulation for the production line balancing problem.

Chapter 2 is related to Simple Assembly Lines, which consist of stations synchronised by a common conveyor. Robust versions of the simple assembly line balancing problem consider that the number of stations and the cycle time are fixed parameters. The problem is to allocate assembly tasks to stations satisfying all manufacturing constraints and maximising the robustness measured by the stability radius. We analyse two classic metrics for its calculation and introduce a new one representing the case of the proportional robustness. Respective theorems and formulas are established. Based on them, we propose combined mixed-integer linear programming models, which are aimed to help decision-makers easily switch from one model to another according to the particular situation and available knowledge. Several enhancements are following the studied models. Among them, a greedy heuristic method, some combinatorial upper bounds, and a cut generation method. Computational results and conclusion are finishing the chapter.

In Chapter 3 we adapt our method for the generalised case of assembly lines, referred to in the literature as Transfer Lines where several tasks can be performed simultaneously by a single machining head. Each station (machine) contains one or more machining heads (blocks), the heads of each station are activated sequentially, the stations as before are synchronised by a common conveyor. This major difference from simple assembly lines determines new properties that makes mathematical models more complex. The transfer line balancing problem consists in assignment of production tasks to machining heads and

distribution of heads to machines satisfying all given constraints and maximising the value of the stability radius. We discover new formulas for measurement of line robustness in Manhattan and Chebyshev metrics and prove them within a series of lemmas and theorems. Two Mixed Integer Linear Programming (MILP) models are developed according to all problem properties. To handle their complexity, we proposed some enhancements based on well-known practices for line balancing problems. First of all, we introduce assignment intervals of tasks and introduce five rules to calculate them under both the considered metrics. Together with a reworked heuristic algorithm they compose a global pre-processing approach which not only generates an initial solution and a lower bound, but also creates two groups of constraints for MILP formulations. The chapter is ending by computational experiments.

Chapter 4 presents a new hybrid approach for transfer line balancing problem. It is aiming to reduce the search space during the optimisation process by generation of optimality based cuts. We study advanced technologies of commercial solvers called *callbacks*. Solvers IBM CPLEX and Gurobi are considered for a better understanding of possible options and approaches. Two implementations (MIP restart and lazy constraints) are analysed and explained in details. Within both of them, we explain applications for developed models. Computational results compare implementations with each other and with enhanced MILP models.

We finish this work with a general conclusion and a bibliography.

Chapter 1

Line Balancing under Uncertainty

Contents

1.1 Production lines

Production lines are present in different industrial environments and are widely used to manufacture a large variety of products. Primarily, they are used to produce consumer goods such as cars, engines, domestic appliances, television sets, computers, smart-phones, and other electrical devices. Since the production constraints can vary a lot according to the products, the corresponding production systems can present major differences. Consequently, the literature includes a wide range of balancing problems. In this section, we examine characteristics that determine various types of production lines.

1.1.1 Basic terms of production lines

A **task** is a simple portion of the total work content in a production line. The time required to perform a task is called *processing time (of a task)*. We assume that tasks are indivisible, *i.e.*, they cannot be split into smaller work elements without creating unnecessary additional work.

A **station** (or workstation) is a segment of the production line where a certain amount of tasks is executed. It is characterised by its dimension, the machinery, and equipment as well as the kind of assigned work. There are two types of stations: manual and automated. The manual station or workplace has to be occupied by a human operator, who performs work using either a simple tool or semi-automated machines which have to be controlled by an operator. Automated stations or machines can work by themselves. We do not consider independently a group of collaborative stations, which are hybrids with human and machine (robot) working together. Their applications in manufacturing are usually presented by co-existence or sequential collaboration. In both, human and robot do not work on a production part at the same time. Thus, mathematical representation may interpret a collaborative station as two independent: one manual and one automated. The set of tasks assigned to a station is referred to as *station load*, the time necessary to perform the work as *load time (of a station)*.

The **material handling system** is the mechanical equipment used for the movement, storage, control and protection of workpieces or production parts throughout the process of manufacturing. A conveyor belt is a common choice for production lines.

The **cycle time** represents the duration that separates the release of two products in a permanent regime. On a paced production line this duration is equivalent to (or at least not smaller than) the load time of the most charged station. On an unpaced one, this value has no direct references to stations and their load times. See the differences between paced and unpaced lines below in this chapter. The *production rate* is the division of a one-time unit by the cycle time. The distance between the cycle time and the load time is called *idle time (of a station)*.

Precedence Constraints show some technological restrictions between tasks. They are usually represented by an acyclic oriented graph, where each node models a task. A given precedence graph may have several source and final nodes if there is no special type asked by an industrial environment. The interpretation of arcs depends on particular cases, but, in general, an arc (i, j) indicates that the task j must not be completed before the task i can start, and cannot be assigned to an earlier station than i. An example of a precedence graph appears in Figure 1.1. Here the task 4 cannot be performed earlier than any of its predecessors 1, 2 or 3; as well as later than any of its successors 6, 7 or 8. Similarly, there is no connection with task 5; thus, the order of their appearance on the line is free.

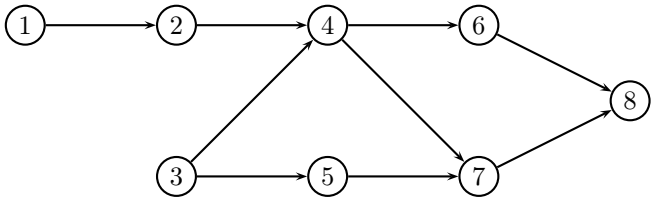

Figure 1.1: Precedence graph

1.1.2 Types of production lines

Number of products

The first characteristic is the number of different products released by the same production line. The literature distinguishes the following line types.

Single-model lines: A single product type is manufactured at the line in large quantities. A set of production tasks is known. All stations repeatedly have to perform the same subset of tasks on identical workpieces.

Mixed-model lines: several models of a basic product are manufactured on the same line (but not at the same time). The models differ from the primary one only concerning some attributes and optional features. The production processes are quite similar in this case and, consequently, the same set of basic operations is necessary to produce any of the models. The presence or absence as well as processing time of a task depends on an optional part being installed or not and on its parameters. Minor setup activities are applying to switch from one model to another.

Multi-model lines: several products are manufactured on the same line in separate batches. Significant differences in production processes require major setup activities (or rearrangements) of the line equipment when products are changing. Besides processing time variations, setup times are taking into account.

Product Flow Manner

A production line consists of stations and a material handling system, which connects stations and moves production parts from one station to the next one. It also determines how elements flow through the line.

Paced lines: material handling systems as conveyor belts flexibly connect the stations. Production parts are steadily moved between stations by the conveyor belt at constant speed immediately after being processed. Each station has the same amount of available working time (the cycle time) to perform the assigned tasks on all incoming

workpieces. Figure 1.2 illustrates a paced production line with four stations.

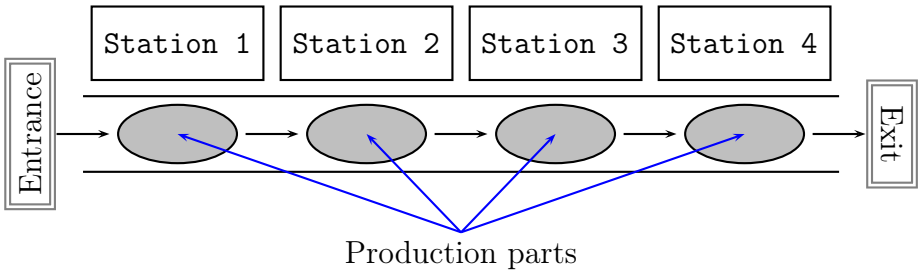

Figure 1.2: Scheme of a paced production line

Unpaced lines: the stations are separated by buffer stocks which hold production parts when previous items still occupy the following stations. Usually, buffers have limited capacities. Because of this, a station may be *blocked* when the following buffer is full. Conversely, a station is *starving* when the preceding buffer is empty after terminating its assigned tasks. Thus, buffer sizes directly impact the production rate of the line. An illustration of an unpaced line is presented in Figure 1.3. There are three stations with two buffers in between. Station 1 could be blocked since the following buffer is full, while station 3 is going to be idle if station 2 did not release its production part in time.

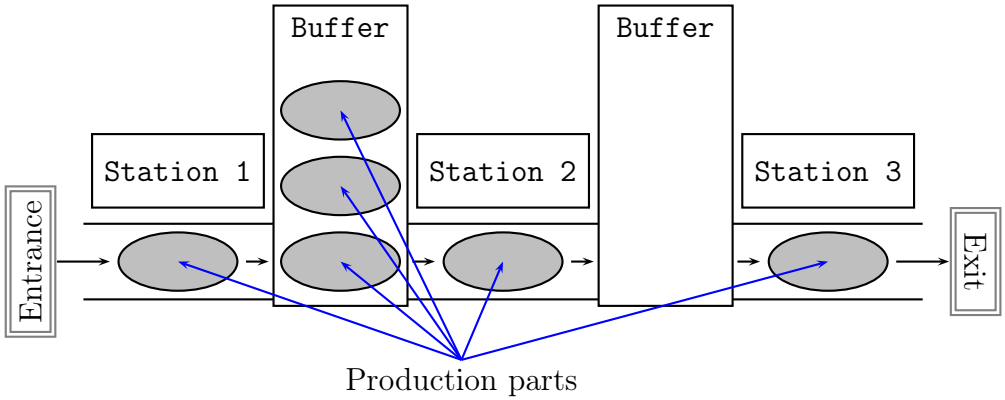

Figure 1.3: Scheme of an unpaced production line

Line layout

Basic Straight: this is a traditional layout of production lines. Each workpiece visits a series of stations in order of their installation. The tasks on stations are supposed to be executed sequentially one by one.

Straight with multiple stations: the advantages of parallelisation can be used to increase productivity in a single line by installing multiple stations, *i.e.*, the production parts are distributed among several operators or automated tools which perform the same

tasks. It helps to reduce the (actual) cycle time of the entire system if some tasks have processing times longer than the desired cycle time. Such tasks should be assigned to parallel stations.

U-shaped: these lines have both the entrance and exit in the same place. They are commonly manual. Workers placed between two legs of the line are allowed to walk from one leg to another. Therefore, they can work on two (or more) workpieces during the same cycle. Stations are closer together such that visibility of the production process, and communications between operators are improved.

Lines with circular transfer: stations are dispatched around a rotating table which is used for loading-unloading and moving the part from a station to another. Concerning the number of turns during which a part stays on the table before being completed, the lines with a single circular transfer can be distinguished. If only one part side is handling at each station and a single turn is sufficient for completing a product, then this configuration is equivalent to a basic straight line. If several sides of the part could be treated simultaneously, then this configuration is equivalent to a line with multiple stations. For the case of multi-turn transfer, the set of tasks assigned to a station must be partitioned into the different cycles corresponding to the number of turns of the table.

Asymmetric: they can be used to postpone the differentiation of products to maintain a typical line configuration for all manufactured products as long as possible. This strategy reduces the risks associated with increasing product variety, but the corresponding line balancing problem must be solved conjointly with a layout optimisation problem in order to determine the final line configuration.

Multiple lines using multiple lines can offer several advantages. Investments can be deferred because additional lines can be installed one by one as they are needed. Production can be adapted more quickly to meet changes in demand. Failure of one line does not necessarily adversely affect the rhythm of other lines, and production costs may be reduced, among other advantages. One drawback is the increased investment cost compared to a single line. The consequences for worker productivity can be diverse: a longer line cycle time in the case of multiple lines enriches the work and improves motivation. However, this may limit the learning effect, since the variation of operations performed by a worker increases. Therefore, the choice between a single line with short cycle time and many lines with longer cycle times is not straightforward and should be considered as a part of an integrated decision-making process.

Industrial environment

Assembly: a final product is assembled from different components. These lines have many different applications from manual with worker assignment to fully automated.

Disassembly: the objective is to separate end-of-life (EOL) products subassemblies and components for recycling, remanufacturing and reuse. Such lines are used to carry out disassembly operations with higher productivity rate. The structure and quality of EOL products are strongly uncertain and even the number of components in such products can not be predicted. Moreover, the precedence graph cannot be obtained by reversing a relevant one for the assembly line. That makes disassembly process more complex than assembly. Due to technical or economic restrictions such as irreversible connections of components of a product and low revenue obtained from retrieved parts, disassembly lines are mostly manual today.

Machining: a product is completed by a series of machining operations like drilling, milling, reaming, and other. In general, there may be much fewer precedence relations between such operations than in the case of an assembly or disassembly process. However, there may exist many tolerance constraints that impose assigning operations to the same workstation and incompatibility constraints that forbid the assignment of specific tasks to the same station because of technological incompatibilities. Such lines are usually highly automated.

1.1.3 Line balancing

Line balancing is a stage of the design that consists in finding an assignment of production tasks among stations that satisfies all the constraints of the problem in order to optimise one or several goals.

A **Line Balance** is a feasible solution to a balancing problem. For the feasibility, a solution has to satisfy several constraints: each task is assigned to exactly one station; precedence constraints are fulfilled; the load time of any station does not exceed the cycle time. A line balance is optimal if, besides the restrictions, it also maximises (minimises) given production goals (objectives).

Example 1. A line balance is shown in Figure 1.4 as a Gantt chart, where four stations process nine tasks. One layer includes all tasks assigned to the same station. Processing times are equivalent to lengths of rectangles. The most loaded station determines the cycle time (9, by station S2). The idle times are $1, 0, 3$ and 2 for stations S1, S2, S3 and S4 respectively. The balance also presents the order of tasks execution on the stations. Thus, station S1 (or a worker assigned to it) performs the task 1, then task 2, and task 4 in the end. At time 9, a conveyor belt moves a production part to station S2. Station S2 starts the task 3, and then task 5, and so on until the end of the production line.

The installation of a production line is a long-term decision that requires significant

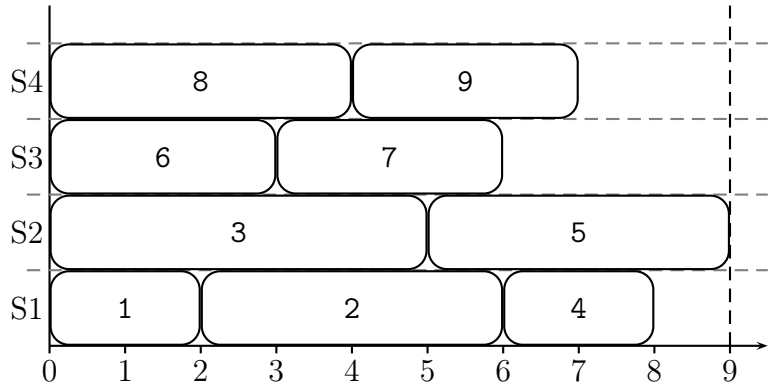

Figure 1.4: Line balance

capital investments. Therefore, it is vital for a designed system to be as efficient as possible. A new line balanced before launching has to be re-designed later according to a periodically (planned) or single changes in the production process. The objectives have to be carefully chosen, according to the goals of the company to minimise the long-term effect of balancing (re-balancing) decisions. Traditionally, there are three types of line balancing problems:

- **type-1:** minimise the number of stations for a given cycle time;
- **type-2:** minimise the cycle time for a given number of stations;
- **type-E:** minimise the line efficiency represented by the non-linear function – the product of the number of stations and the cycle time, while they are given by lower and upper bounds.

Now, when we have an understanding about line balance and production goals, let introduce some helpful notations:

- $V = \{1, 2, \ldots, n\}$ is a set of production tasks to be assigned;
- t_j is a nominal processing time of the task j, $j \in V$;
- m is a number (maximal) of available stations;
- T is a cycle time;
- $G = (V, A)$ is a precedence graph, where A is a set of arcs (i, j), such that $i, j \in V$.

Then, deterministic models can be introduced with following constraints:

$$\sum_{k=1}^{m} x_{j,k} = 1, \ \forall j \in V, \tag{1.1}$$

$$\sum_{k=1}^{m} kx_{i,k} \leq \sum_{k=1}^{m} kx_{j,k},\ \forall (i,j) \in A, \tag{1.2}$$

$$\sum_{j \in V} t_j x_{j,k} \leq T y_k,\ \forall k = 1, 2, \ldots, m, \tag{1.3}$$

$$y_k \in \{0, 1\},\ \forall k = 1, 2, \ldots, m,$$

$$x_{j,k} \in \{0, 1\},\ \forall j \in V, \forall k = 1, 2, \ldots, m. \tag{1.4}$$

Restrictions 1.1 assure that any production task assigned exactly to one station. The precedence order is modelled in 1.2. The load time of any station does not exceed the cycle time, as it is shown in 1.3. Two last groups represent constraints on variables:

- $x_{j,k}$ is a decision variable, that equals to 1 if and only if the task j is assigned to the station k;
- y_k states if the station k is presented on the line.

Objective goals mentioned above can be written in mathematical form as:

- **type-1:** minimise $\sum_{k=1}^{m} y_k$;
- **type-2:** minimise T;
- **type-E:** minimise $T \cdot \sum_{k=1}^{m} y_k$.

Regardless of the type of production line and objectives, new challenging conditions or changes may arise during the manufacturing process. Their prediction or, at least, anticipation is a substantial part of the line balancing stage. The changes that affect stations availability and task processing time posses the major challenges. Indeed, a human operator cannot work with constant speed due to several reasons, such that: personal experience, proficiency, education, fatigue, health conditions, partial dissatisfaction with something, thrill before the holidays, or something else. Even tasks executed by a machine may not have a stable processing time because of overheating, material or product specification changes, malfunctioning, the input lag of software. All of these events may inflict costly damage and should be prevented within the line balancing. For this purpose, we assume that all tasks are belonging to one of two subsets: *certain* and *uncertain*. Certain ones have a fixed processing time that is not affected by any event. On the contrary, uncertain tasks are vulnerable, and their processing time is changing with an occasion on the line.

Example 2. Consider the line balance from Figure 1.4. Assuming that tasks 1, 3, 8 and 9 are uncertain, they are coloured in grey, as shown in Figure 1.5. In paced mode the line produces a new piece of product every 9 units of time. If a unit of time equals to one second, then the production rate is 400 pieces per hour. Now, imagine that because of a worker replacement or mistaken estimation the processing time of the task 3 is increasing from 5 to 6 units of time. Figure 1.6 shows the deviated line balance. The new cycle time is 10, which is equivalent to the production rate of only 360 pieces per hour, which amounts to a significant throughput loss. Moreover, if the line is automatic and the cycle time is strongly fixed, then such deviation will cause the line interruption. Thus, anticipation of task time uncertainties should be taken into account at the design stage.

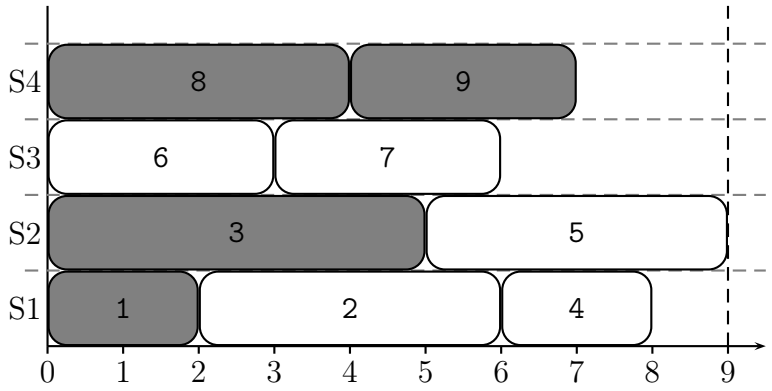

Figure 1.5: Line balance with certain and uncertain tasks

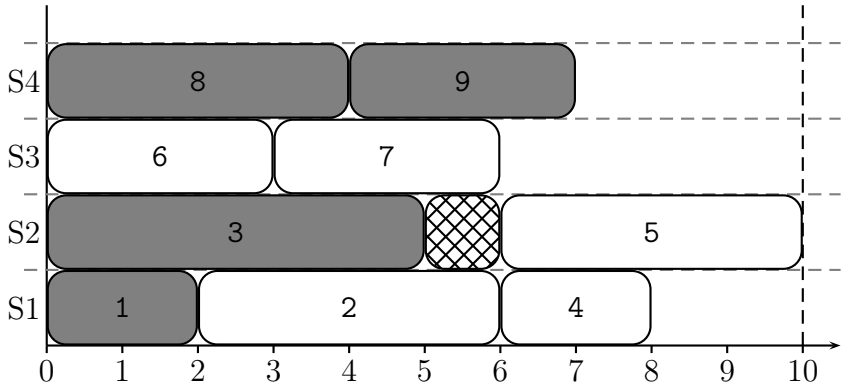

Figure 1.6: Deviated line balance

1.2 Modelling of uncertainty

In this section, we discuss some approaches which are used to handle task time uncertainties in line balancing problems. Detailed general observations on a line balancing are given by Scholl and Becker [63], Boysen et al. [10], Battaïa et al. [4]. Earlier comprehensive surveys about line balancing under task time uncertainty was published by Bentaha et al. in [9] and [8]. The literature usually considers three major approaches: stochastic, fuzzy and robust methods.

1.2.1 Stochastic approaches

Stochastic approaches are the most popular ones and consist of a representation of task times as independent random variables with known characteristics, usually symmetric distribution.

Chance-constrained programming

This approach presents the processing time of tasks as normally distributed independent random variables $t_j \sim \mathcal{N}(\mu_j, \sigma_j^2), j \in V$. The mean μ_j and variance σ_j^2 are known for all tasks. Let us introduce $Y = \sum\limits_{j \in V} t_j$, *i.e.*, a normally distributed random variable $Y \sim \mathcal{N}(\sum\limits_{j \in V} \mu_j, \sum\limits_{j \in V} \sigma_j^2)$. This variable can be easily converted into a standard distribution using the transformation $Z = \frac{Y - \sum\limits_{j \in V} \mu_j}{\sqrt{\sum\limits_{j \in V} \sigma_j^2}}$. A designed line balance should respect the cycle time with a given probability α. In practice and literature, this value usually is greater than 0.9. Thus, $\mathbb{P}(Y \leq T) \geq \alpha$, and using mentioned transformation we get the next formula:

$$\mathbb{P}\left(Z \leq \frac{T - \sum\limits_{j \in V} \mu_j}{\sqrt{\sum\limits_{j \in V} \sigma_j^2}} \right) \geq \alpha \tag{1.5}$$

$\mathbb{P}(Z \leq z_\alpha) = \alpha$, z_α is (100α)-percentile of the standard normal distribution. Expression (1.5) holds if and only if

$$z_\alpha \leq \frac{T - \sum\limits_{j \in V} \mu_j}{\sqrt{\sum\limits_{j \in V} \sigma_j^2}} \tag{1.6}$$

or

$$\sum_{j \in V} \mu_j + z_\alpha \sqrt{\sum_{j \in V} \sigma_j^2} \leq T. \tag{1.7}$$

Let $x_{j,k}$ be a binary variable that is set to one if and only if the task j is allocated to the station k. Then inequality (1.7) can be written as follows:

$$\sum_{j \in V} \mu_j x_{j,k} + z_\alpha \sqrt{\sum_{j \in V} \sigma_j^2 x_{j,k}} \leq T. \tag{1.8}$$

This expression represents the cycle time constraint that should be satisfied during the design in any type of line balancing problem.

It is evident that a mathematical model with the constraint (1.8) is non-linear which may prevent to successfully solve large problem instances. Thus chance-constrained programming requires an additional linear transformation for real-life problem instances. Non-linear part can be eliminated by a squaring procedure by introducing new binary variables and constraints.

The reason why the developed chance-constraint model is not linear is the term $\sqrt{\sum_{j \in V} \sigma_j^2 x_{j,k}}$ with task variable. For that reason by eliminating the square-root, linearising will be possible. So, by squaring both side of the inequality (1.6), including variables from (1.8), the next inequality has been obtained.

$$(z_\alpha)^2 \leq \frac{(T - \sum_{j \in V} \mu_j x_{j,k})^2}{\left(\sqrt{\sum_{j \in V} \sigma_j^2 x_{j,k}}\right)^2} \tag{1.9}$$

While the non-linear situation caused by the term $\sqrt{\sum_{j \in V} \sigma_j^2 x_{j,k}}$ has been eliminated by inequality (1.9), a new non-linear term has appeared with term $(T - \sum_{j \in V} \mu_j x_{j,k})^2$. The open form of this term is given by

$$(T - \sum_{j \in V} \mu_j x_{j,k})^2 = T^2 - 2T \sum_{j \in V} \mu_j x_{j,k} + \begin{pmatrix} \mu_1^2 x_{1,k}^2 + \mu_1 x_{1,k} \mu_2 x_{2,k} + \cdots + \mu_1 x_{1,k} \mu_n x_{n,k} \\ \mu_1 x_{1,k} \mu_2 x_{2,k} + \mu_2^2 x_{2,k}^2 + \cdots + \mu_2 x_{2,k} \mu_n x_{n,k} \\ \cdots \\ \mu_1 x_{1,k} \mu_n x_{n,k} + \mu_2 x_{2,k} \mu_n x_{n,k} + \cdots + \mu_n^2 x_{n,k}^2 \end{pmatrix} \tag{1.10}$$

Since the variable $x_{j,k}$ is $0-1$ integer, $x_{j,k}^2$ is equal to $x_{j,k}$. The term $(\mu_i x_{i,k} \mu_j x_{j,k})$ is the non-linear part of the equation (1.10). Thus, a current transformation technique in the literature has been used to make this part linear

$$u_{i,j,k} = x_{i,k} x_{j,k}. \tag{1.11}$$

After the variable transformation given in (1.11), the variable u_{ijk} is linked to x_{ik} and x_{jk} by the following inequalities:

$$x_{i,k} + x_{j,k} - u_{i,j,k} \leq 1, \tag{1.12}$$

$$x_{i,k} + x_{j,k} - 2u_{i,j,k} \geq 0. \tag{1.13}$$

Another situation to be considered on inequality (1.9) is that the inequality is valid in two different cases. These can be expressed by the inequality below.

$$|z_\alpha| \leq \left| \frac{T - \sum\limits_{j \in V} \mu_j}{\sqrt{\sum\limits_{j \in V} \sigma_j^2}} \right| \tag{1.14}$$

Assuming that z_α is larger than 0.9 implies that z_α is positive, and so the right-hand side of the inequality (1.14) must be positive, too. For that reason, we have to introduce an additional constraints which is the same as the cycle time constraint at the basic deterministic model (compare to 1.3):

$$T - \sum_{j \in V} \mu_j x_{j,k} \geq 0. \tag{1.15}$$

Then, the inequality (1.8) is replaced by the constraint (1.15), the equality (1.11) together with (1.12–1.13) and:

$$T^2 - 2T \sum_{j \in V} \mu_j x_{j,k} + \sum_{j=1}^{n} \mu_j^2 x_{j,k} + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n-1} \mu_i \mu_j u_{i,j,k} - z_\alpha^2 \sum_{j=1}^{n} \sigma_j^2 x_{j,k} \geq 0. \tag{1.16}$$

This stochastic approach is well-developed and applied almost for all existing types of production lines: straight ([13], [22], [62], [64]), U-shaped ([1], [2], [6], [29], [55]), two-sided ([14], [53], [54]).

Simulation of problem characteristics and re-balancing

The second category involves the work performed to examine the problem characteristics via simulation and to compare the deterministic and stochastic versions of the problem. In these procedures, an initial balance is given, and the objective is to rearrange the tasks to fulfil a substitution with a new item due to customers' unique requirements. It can be observed that the definition of the category is deeply related to a particular case well known in the literature as a re-balancing problem.

Rearrangements may require the modifications of the whole initial parameters, as well as the modification of a subset only. Because of this, the main difficulty in practical cases is

the precise estimation of costs related to all possible tasks movements, a weighted multiple objective function involving both completion and tasks movement associated costs is not introduced. Preferably, two separate objective functions, concerning expected completion costs (EC) and the degree of similarity between initial and new tasks assignments (shortly, mean similarity factor or MSF), are separately introduced. EC is calculated in a similar way as the objective function for type-E balancing problem:

$$\mathrm{EC} = mT + \sum_{j \in V} p_j I_j, \tag{1.17}$$

where as before m is the number of machines, and T is the cycle time of a new balance, p_j is the probability of not completing task j in the assigned station, I_j is the total incompletion cost of task j. It includes the value of the task itself and those of its followers in the precedence graph, *i.e.*, the task staying earlier in the order is more worthy. In the same manner as in chance-constrained programming, the probability p_j is calculated using the transformation into a standard form:

$$\begin{aligned} p_j &= \mathbb{P}(Z \leq z_j) \qquad && \forall j \in V, \\ z_j &\leq \frac{T - \sum\limits_{i \in V_{jk}} \mu_j}{\sqrt{\sum\limits_{i \in V_{jk}} \sigma_j^2}} \qquad && \forall j \in V, k \mid j \in V_k \end{aligned} \tag{1.18}$$

where z_j represents the station idle time, with respect to task j. Consequently, V_k is a subset of tasks assigned to station k; V_{jk} is a subset of V_k (such that $j \in V_k$) including both the task j and tasks executed before it in the station k. Here, T is exactly the task time which is not always equivalent to the cycle time. As above, the non-linear part can be eliminated by the squaring procedure which consists in introducing new binary variables and constraints.

The second objective function showing the mean similarity factor (MSF), as it follows from the name, is aiming to compare a given initial line balance with a re-designed one. Two special subsets are introduced for this purpose:

$$\begin{aligned} \mathrm{TIB}_j &= \{i \in V_k^0 \mid j \in V_k^0 \text{ and } i \neq j\} \qquad && \forall j \in V^0 \\ \mathrm{TNB}_j &= \{i \in V_k \mid j \in V_k \text{ and } i \neq j\} \qquad && \forall j \in V \end{aligned}$$

TIB_j is the subset of initial tasks, other than j, belonging to the station k in an initial balance. In a similar manner, TNB_j is also the subset of tasks, but for a new balance. As it was mentioned above, V may differ from V^0 because of product changes and another requirements. The similarity factor (SF_j) of the task j is obtained in

$$\mathrm{SF}_j = \frac{|\mathrm{TIB}_j \cap \mathrm{TNB}_j|}{|\mathrm{TIB}_j|}. \tag{1.19}$$

It is the ratio of number of tasks assigned to the same station as task j in the initial and in the new configuration to the number of tasks assigned to the same station in the initial balance.

When the task j is alone in a station in the given configuration, SF_j assumes an indefinite form $0/0$. In this case, the SF_j value is set to 0 in assignment procedures so as to make it less likely that this task will be assigned to a station where tasks belonging to other sets are present. Whilst, in the final MSF evaluation step, where the similarity of the line as a whole is computed, SF_j is set to 1 in case the task j is alone both in the initial and in the new balance, and to 0 otherwise.

Notice that SF_j takes a value within the range $0 \leq \mathrm{SF}_j \leq 1$. SF_j takes the value 0 if none of the tasks assigned to the same station as task j in the initial line balance are assigned to the current station. SF_j takes the value 1 if all the tasks assigned initially to the same station as task j are assigned to the current station. SF_j takes a value within the range $(0, 1)$ for any other subset of the tasks. The more numerous is this subset, the higher is SF_j.

The mean similarity factor (MSF) between the new re-balanced line and the initial one is finally evaluated as in

$$\mathrm{MSF} = \frac{\sum_{j \in V} \mathrm{SF}_j}{|V|}. \tag{1.20}$$

In the presence of several new feasible balances, the highest MSF values address the choice of balances with less tasks movements.

Since one of the first appearance in [57], this approach is generally used for assembly line balancing and can be found in [7], [26], [45], [49], [50], [69].

Alternative procedures

In the third category, there are procedures specifically developed for the stochastic problem. They may partially include ideas from either chance-constrained or simulation approach. For example, Saif et al [61] consider a bi-objective model, where the main goal is to minimise the cycle time (as in type-2 problems), and the supporting goal is to maximise the probability of stations to withstand the limit both individually and together. Zhang et al. [74] also present a bi-objective optimisation method aiming to minimise the cycle time as well as the total processing cost. Kao in his works [42] and [43] optimises a type-1 assembly line constructing the preference order for task assignment under stochastic processing times. Carter and Silverman [12] develop a model with off-line repairs for uncompleted tasks partially based on chance-constrained programming. Another approach with off-line completion is proposed in [22] by Erel et al. Minimisation of the smoothness index together with the design cost associated with labor and equipment requirements is given by Cakir et al. in [11] for a line with parallel stations. Chiang et al. [14] model two-sided assembly lines taking into account mated-stations and optimising a weighted sum of the line length and the number of station. Zheng et al. [76] propose a distribution-free models for disassembly line balancing problem, but still dealing with stochastic tasks.

Bowl-phenomenon

Besides these three categories, we can find an approach based on the *bowl-phenomenon* that arises in unpaced systems. This phenomenon consists in the fact that workload should be allocated according to a strictly concave function (stations in the middle are less utilised than those at the start or the end of the line). It helps to find a size of buffers and their locations in an unpaced system that improves a production rate especially when processing times of tasks are not normally distributed or when they are significantly different from each other. Among the latest works, we refer to Das et al. [15], Tiacci [67], Urban et al. [69].

1.2.2 Fuzzy approaches

Another type of task times uncertainty is called fuzzy. The term *fuzzy* is meant to represent variables, expressions and judgements that have no clear (crisp) values or boundaries. In this case, the given information about the processing times is better introduced as a fuzzy set or a fuzzy number. They can appropriately represent imprecise parameters and can be manipulated through different operations defined on them.

A **fuzzy set** is defined by a membership function (all the information about a fuzzy set is described by its membership function). The membership function maps elements (crisp inputs) in the universe of discourse (interval that contains all the possible input values) to elements (degrees of membership) within a certain interval, which is usually $[0, 1]$. Then, the degree of membership specifies the extent to which a given element belongs to a set or is related to a concept. Figure 1.7 shows the membership function of a fuzzy set that represents the concept of "a number close to m". The crisp input m has a degree of membership in the fuzzy set of one, which means that m is totally related to the concept (it is close to m). Any other number from $[a, b] \setminus \{m\}$ has a degree of membership between 0 and 1, which means that a number is partially related to the concept.

The most commonly used range for expressing degree of membership is the unit interval $[0, 1]$. Thus, for a fuzzy set A, this can be expressed by

$$f_A(x) : X \to [0, 1], \tag{1.21}$$

which means that the membership function f_A assigns to each element x of the universal set X, which is a crisp set, a value within the range $[0, 1]$. If the value assigned is 0, the element does not belong to the set (it has no membership). If the value assigned is 1, the element belongs completely to the set (it has total membership). Finally, if the value lies within the interval $(0, 1)$, the element has a certain degree of membership (it belongs partially to the fuzzy set). A fuzzy set, then, contains elements that have different degrees

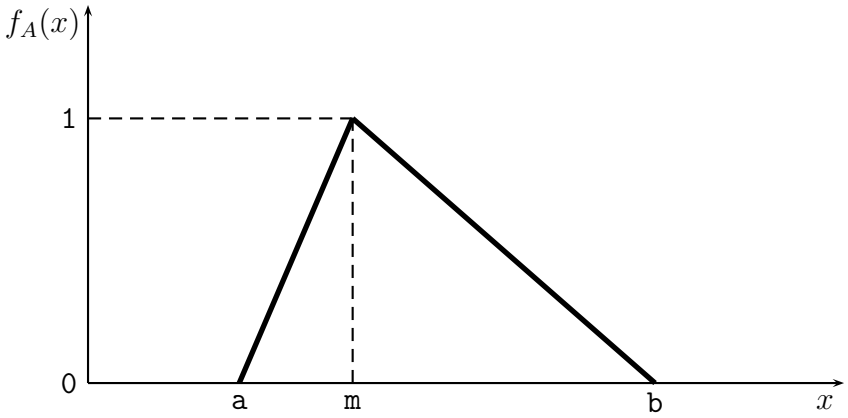

Figure 1.7: A membership function for a fuzzy set

of membership in it. More concrete example, related to Figure 1.7 is written as follows:

$$f_A(x) = \begin{cases} 0, & \text{if } x \leq a, \text{or } x \geq b, \\ \frac{\lambda(x-a)}{(m-a)}, & \text{if } a \leq x \leq m, \\ \frac{\lambda(b-x)}{(b-m)}, & \text{if } m \leq x \leq b. \end{cases} \tag{1.22}$$

Here, the fuzzy set A is presented by a triplet (a, m, b) of real numbers, such that $0 \leq a \leq m \leq b$. The parameter λ shows the maximal possible degree of membership. Sometimes, the inverse of half the length of the segment is used, *i.e.*, $\lambda = \frac{2}{(b-a)}$. In this case, the membership function represents the triangular density of probability.

Initially, necessity of this approach for real production lines was explained by the fact that it is often impractical to use a well-known probability distribution (like Gaussian) to characterise task times. Indeed, for manual production systems where a lot of employees are involved, it is extremely difficult to estimate these densities. Nevertheless, in practice, as a rough estimate, managers are able to provide a usual task time m and 2 values a and b that are, respectively, the lowest and greatest task times. Usually, evaluating a task time using the above three parameters can be enough for most management purposes.

A **fuzzy number** is a special case of a fuzzy set. Different definitions and properties of fuzzy numbers are encountered in the literature, but they all agree on that a fuzzy number represents the conception of a set of "real numbers close to r", where r is the number being fuzzified. To classify as a fuzzy number, a fuzzy set A on $\mathbb{R}$ must satisfy the following:

- at least one element of the fuzzy set A has full membership;
- A_α must be a closed interval for every $\alpha \in (0, 1]$, where

$$A_\alpha = \{x \in X \mid f_A(x) \geq \alpha\}; \tag{1.23}$$

- The support or scope of A must be bounded.

The concept of α-cut of a fuzzy set is useful for defining the arithmetic operations on fuzzy numbers. As it follows from (1.23), the α-cut is the (crisp) set A_α of elements x, such that their degree of membership in the set A is at least equal to α ($0 \leq \alpha \leq 1$).

To apply this approach, suitable fuzzy arithmetic and an appropriate method dedicated to comparing these fuzzy sets (numbers) have to be introduced. Because of this, not only the processing times but also another problem data has to be presented by fuzzy sets (the cycle time or number of machines).

In the literature, several fuzzy comparison methods have been proposed such as pseudo-order fuzzy preference model (see Roy and Vincke [59]), fuzzy-weighted average (Vanegas and Labib [70]), signed distance method (Yao and Wu [72]). The last one is the most popular among them, but generally, the fuzzy approach becomes less popular. The following work can be noted: a heuristic method in Hop [40]; a meta-heuristic algorithm for line efficiency maximisation in Zacharia and Nearchou [73]; genetic methods in Tsujimura et al. [68]; binary fuzzy goal programming approach in Kara et al. [44].

During the last four years, the only work that was related to the fuzzy approach is Babazadeh et al. [3]. Authors provide mixed integer linear problem formulations for straight and U-shaped assembly lines and develop a non-dominated sorting genetic algorithm for resolution.

1.2.3 Robust approaches

Evoked in the late 1960s [30], the idea of robustness is attracting increasing interest from both practitioners and theorists in operational research. Originally reflecting a concern for flexibility in a context of uncertainty, this concept now seems to fit a much broader spectrum of situations. In the robustness literature, uncertain data is mainly associated with a set, continuous or discrete, of possible values, with no associated probability [46]. In the continuous case, the sets are often intervals, hence the notion of interval approach. In the discrete case, meanwhile, we are talking about the scenario model. Whatever the case, the Cartesian product of these sets defines the possible instances of a problem. After modelling the uncertain data in the form of possible instances, the problem is to find a solution that is "good" for all these instances or robust. We say that a solution is robust if its performance is somehow insensitive to data uncertainty and hazards.

All above-mentioned stochastic techniques require some a priori information to construct an appropriate probability distribution and membership function. However, such information is not always available at the design stage due to, for example, originality of the considered line. Since stochastic and fuzzy approaches are hardly applicable, some robust ones have to be used in this case. These approaches assume that only a discrete

set of scenarios or closed intervals of tasks time realisations are available without any distribution on it.

Set of scenarios

Following this definition, a robust solution is then designed to limit or absorb the effects of uncertain data. In practice, it is often considered that the robust solution is the best solution in the worst case. Most of the works on robustness uses the concept of distance between the robust solution calculated a priori and the optimal solution of the instance actually achieved according to a certain criterion (satisfaction of constraints or a particular requirement). In the search for robust solutions, most authors use the criteria of absolute robustness (min-max), maximum regret (min-max regret) or relative regret [60]. For a minimisation problem, if $\mathcal{S}$ is the set of possible solutions, $\mathcal{I}$ is the set of instances, $f(s, I)$ gives the value of the solution $s \in \mathcal{S}$ for the instance $I \in \mathcal{I}$ and f_I^* the value optimal for instance $I \in \mathcal{I}$, then the three criteria are defined as follows:

- absolute robustness: $\min\limits_{s\in\mathcal{S}} \max\limits_{I\in\mathcal{I}} f(s, I)$;
- maximum regret: $\min\limits_{s\in\mathcal{S}} \max\limits_{I\in\mathcal{I}} (f(s, I) - f_I^*)$;
- relative regret: $\min\limits_{s\in\mathcal{S}} \max\limits_{I\in\mathcal{I}} \frac{f(s,I)-f_I^*}{f_I^*}$;

Consider a more precise example for production lines. Let $t_{j,k}$ be the processing time of the task j on the station k, $k \in W$ (W is a set of available stations), and let t be the respective matrix of processing times. It is assumed that t belongs to a set S of scenarios. The set of scenarios S is a finite collection of subsets of interval scenarios: $S = S^{(1)} \bigcup S^{(2)} \bigcup \ldots \bigcup S^{(l)}$. Since the set of task processing times is not unique, a question then appears as to which solution is appropriate. The best solution for one specific set can be the worst for another set. One of the classic approaches to hedge against uncertainty is to construct the best solution for the worst scenario.

The cycle time for this formulation is modelled using the following expression:

$$\max_{t\in S} \max_{k\in W} \left\{ \sum_{j\in V_k} t_{j,k} x_{j,k} \right\} \leq T, \tag{1.24}$$

where V_k is a subset of tasks assigned to the station k. This constraint chooses the most loaded station among all scenarios and checks that its load time does not exceed the cycle time. In case of the type-2 line balancing problem, this approach is exactly representing absolute robustness (or min–max) approach. Its detailed explanation can be found in [21]. Another comprehensive work with new exact and heuristic methods based on the min-max regret approach was given in [56].

In the work [71], a mixed-model assembly line balancing problem is considered. Authors Yang and Gao propose combinatorial lower and upper bounds for studied problem formulation and develop a multi-scenario greedy procedure as well as a branch, bound and remember algorithm. The processing time of tasks in this work varies according to a scenario of an assignment. A similar situation happens in robotic assembly lines, where a set of chosen robot determines an assignment scenario and a respective processing time for tasks. Multi-objective optimisation for U-shaped robotic assembly lines is presented in [75]. Zhang et al. discover an artificial bee colony algorithm for this purpose. Another group of metaheuristic algorithms for robotic assembly lines are compared in [41] by Janardhanan et al.

Moreira et al. propose a comparison of three heuristic procedures for multi-objective assembly line worker integration and balancing problem in [52].

Several heuristic procedures are compared in He et al. [38] for a multi-objective automated transfer line balancing problem. Authors introduce a new robust dominance criterion which they use for the construction of a feasible solution.

Interval uncertainty

Similar to the set of scenarios, tasks processing times can be represented as closed intervals $t_j = [a_j, b_j]$ dealing with nominal and worst case values which are used for borders of ranges. Here, these intervals are fixed and do not depend on the scenario or a station. This approach uses the special parameter Γ that shows the number of tasks involving a processing times variability. When $\Gamma = 0$, the consequences of the variability are ignored and the deterministic problem with nominal time values is obtained. By contrast, high values of this parameter indicate risk-averse behaviour. Variability of processing times is presented in the model as the following function:

$$g_k(x) = \max \left\{ \sum_{j \in V} d_j x_{j,k} u_{j,k} : \sum_{j \in V} u_{j,k} \leq \Gamma \right\}. \tag{1.25}$$

More precisely, the function (1.25) defines the maximal time deviation for the station k. The value d_j equals to the interval's range of the task j, *i.e.*, $d_j = b_j - a_j$. The set of tasks that are subject to uncertainty are determined by the binary vector $u = \{j : u_{j,k} = 1, \forall k\}$. As aforementioned, these tasks will have processing times equal to the upper bounds. However, total possible deviation is bounded using the parameter Γ, which also reflects the pessimism level of the decision maker. A larger value of Γ corresponds to a larger amount of deviation that is taken into account. The time deviation (1.25) transforms the cycle time constraints into:

$$\sum_{j \in V} a_j x_{j,k} + g_k(x) \leq T. \tag{1.26}$$

Again, in type-2 line balancing problems, this approach uses techniques of the min-max optimisation. Also, it should be noted that the function (1.25) may be introduced in a

different way which depends on another representation of time intervals and a definition of the parameter Γ, see [36]. In the article of Hazir and Dolgui [37], mixed integer programming formulation for a U-type assembly lines is presented, and a decomposition algorithm is proposed for its resolution. Moreira et al. [51] present two mixed integer formulation as well as a heuristic method for assembly line worker assignment and balancing problem. Other formulations and exact solution methods have been presented in [33] by Gurevsky et al.

The evolutionary simulated annealing algorithm for mixed-model multi-robotic disassembly line balancing problem can be found in the work of Fang et al. [23].

Stability radius

Among the robust approaches, there is one related to **stability analysis**. This theory is concerned with the stability of solutions of a problem (system of constraints) under small perturbations of initial conditions. Thus, it is not a question of solving an optimisation problem, but of evaluating the behaviour of the solutions already found in the face of disturbances of the initial data. In this way, uncertainty is introduced into the phase after solving the problem.

Sensitivity analysis was initially used to evaluate the influence of changes in one of the parameters of the linear programming problems solved using the simplex algorithm. The goal was to analyse the behaviour of the objective function without having to solve the problem again. Sensitivity analysis for combinatorial optimisation problems appeared a little later. Attempts to directly employ the methods developed for continuous linear problems on these problems have not resulted in a satisfactory result. Since the 1970s, many studies have been devoted to the study of sensitivity for combinatorial optimisation problems. The first state of the art was published by Geoffrion and Nauss in 1977 [27].

In order to determine the stability (or robustness), it is necessary to use a measure that would confront a value to a given solution. Sotskov, Dolgui and Portmann [66] introduce a specific indicator named *stability radius* for application in balancing problems. According to the definition, it is a maximal deviation of uncertain tasks from their nominal values that do not violate solution feasibility. The stability radius is a suited measure of robustness for known solutions in the presence of task processing time deviations. Gurevsky, Battaïa and Dolgui [31] show results of its application for simple assembly line balancing problem. Stability analysis of feasible and optimal solutions of generalised line balancing problem for the case of processing time variations for some tasks have been developed in [32]. Sotskov et al. [65] also investigate the stability of simple assembly lines with respect to variations of processing times and propose algorithms for constructing feasible and stable optimal line balances. Their work is continued in [47]. Authors are discovering of stability radius' properties and solution approaches.

The majority of papers presented above are considering a line balancing problem

supported by one of the following goals: minimisation of the line cost (through the minimal number of stations); minimisation of the cycle time (*i.e.*, maximisation of the production rate). It is clear that from a practical point of view, these goals are most important. However, an obtained solution may not stand all incoming events as well as it assumes to be. Availability for the production systems to be reconfigured may solve this issue but as the last review in this field shows (see Galizia et al. [25]), companies still prefer to use traditional systems because important questions are still open in flexible systems:

1. In the current production environment, do industrial companies have a clear view of the emerging market trends and of the need of adopting these emerging production systems?
2. Is it possible to make reconfigurable a production system not designed to be? Is it necessary to include reconfigurability principles just in the design phase of such systems?
3. From an economic point of view, is it possible to introduce reconfigurability principles in an existing production system without making any new substantial investments?

The ability to keep a line working in changing circumstances without reconfigurations or, at least, to find the moment when the line has to be reconfigured could be important for modern production systems. Because of this, we propose a new combinatorial optimisation problem, whose main goal is to maximise the robustness of the designed production line.

1.3 Stability radius maximisation

Classic formulations assume that either the cycle time or the number of stations are fixed. From one point of view, we can find a solution (optimal) and use the stability radius for post-optimal analysis. It will help us to understand its level of robustness and, in case of the presence of several optimal balances, to choose the one which is more stable. But from a different perspective, the real challenge lies in the fact that usually an optimal solution may be unstable (see Lai et al. [48]), *i.e.*, the optimal solution may not withstand even slight processing time deviations, which would make it unpractical in a real-life context.

Example 3. Let see again the example of the line balance of Figure 1.5. The stability radius equals to zero since any nonzero deviation of the processing time of the task 3 violates the cycle time. Thus, this balance is unstable. The solution pictured in Figure 1.6 also cannot be considered as stable, because nothing can prevent future

time deviations. However, we can imagine alternative line constitutions, for example the one shown in Figure 1.8. This configuration may provide the production rate in 360 pieces per hour and protect from some time variations. For any uncertain task the threshold is determined within the idle time of the station. The task 1 may increase its processing time from 2 up to 4; task 3 – from 5 up to 10; tasks 8 and 9 share the total possible deviation in 3 time units.

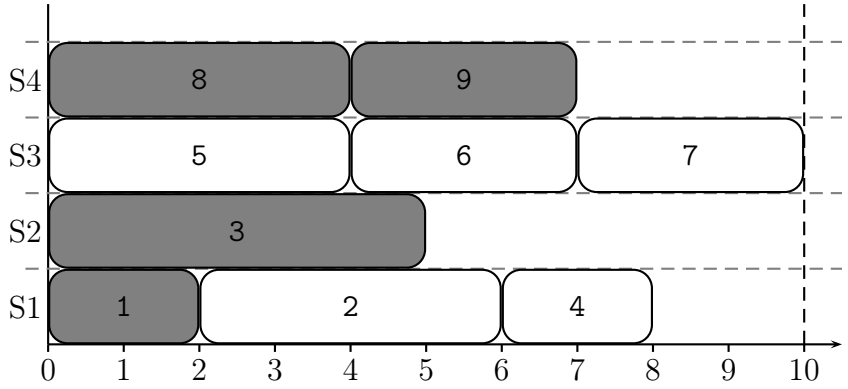

Figure 1.8: Robust line balance

Example 3 shows that the capability of the line balance to stand tasks processing times deviation should be taken into account in the design stage. A configuration with the greatest value of the stability radius has to be chosen as an optimal one. Eventually, both the cycle time and number of stations have to be fixed or, at least, are given as known input parameters. In practice it is possible according to the budget limitations, which generally are not so variant to provide both bounds [$m_{\min}$ and $m_{\max}$], but precise enough for a concrete number. Another limitation comes from an available working surface. We can divide it by measurements of stations to see a total capacity. Alternatively, the cycle time represents the reversed value of the production rate. Indeed, the release of a modern product often consists of two parts: pre-order and foremost one. Using pre-order, manufacturers can gauge how much demand there will be and thus the size of initial production runs, and sellers can be assured of minimum sales. Additionally, high or low pre-order rates can be used to form sales for the foremost part. Since customer needs are known (exactly or approximately), calculation of the respective boundaries does not seem to be a difficult challenge.

The paper of Rossi et al. [58] shows the first attempt to address the aforementioned problem. The authors introduce the idea of stability radius maximisation for simple assembly line balancing under tasks time uncertainties with the fixed cycle time and number of workstations. Mixed integer linear programming (MILP) models are presented for two classic metric: Manhattan (l_1) and Chebyshev (l_∞).

1.4 Conclusion

We consider line balancing problems under uncertainty where production tasks have to be assigned to a linearly ordered set of stations without buffers. The number of machines, as well as the cycle time, are available as parts of input data for the problem and do not have to be minimised. The production tasks are supporting by nominal values of processing time. Nevertheless, there is a subset of uncertain tasks which are sensitive to different events during manufacturing. To model these variations, we use a robust approach based on stability radius. The optimisation goal is aimed at maximising the robustness of the designed line.

Chapter 2

Robust Balancing of Simple Assembly Lines

Contents

In this Chapter we study the simple assembly line balancing problem under task processing time deviations. Mathematical models are presented for three metrics according to introduced formulas for the stability radius calculation.

Assembly lines considered here are basic straight paced production systems, aimed to manufacture a large quantity of product and consist of a set of linearly ordered workstation linked by a conveyor belt. During manufacturing, the units are sequentially injected at the beginning of the line, then they are transferred from one workstation to another. At each workstation a set of tasks is performed on the units by operators or automated equipment. Furthermore, the workstations operate simultaneously and only one unit can be processed at the same time at a workstation. As for tasks, they are indivisible and subject to strict precedence constraints, usually represented by a directed acyclic graph that contains a node for each task, an arc (i, j) means that the task i must be done before the task j.

In the following, we consider simple assembly line, where there is no buffer stock nor parallel workstation. Therefore the units are transferred in a synchronised manner based on the *cycle time*, defined as the time period between the transfer of a unit to another workstation. Since a simple assembly line is conceived to run in a long-term and may require a large capital investment, its balancing is an important issue that arises in its design so that the line works as efficiently as possible. The balancing of a simple assembly line consists in partitioning a set of tasks among workstations in an optimal way with respect to a given production goal. Due to space restriction, the number of workstations is limited. On manual workstations, operators' work rate depends on their skills, fatigue and motivation, thereby generating variation on tasks processing time. Otherwise tasks processing time variation can be caused by delays and micro-stoppages during the manufacturing cycle or by changes of product specifications and workstation characteristics.

The uncertainty on task processing time was tackled in [58] for simple assembly line with a limited number of workstations and fixed cycle time. A robust approach was used and mixed-integer linear programs were formulated to seek a simple assembly line configuration, which is able to support the maximum amplitude of task time variability know as *stability radius*. Two well-known norms l_1 and l_i were used to evaluate the stability radius. Following on this previous work, in this thesis we also measure the robustness of a simple assembly line as the maximum proportion increases of all uncertain tasks processing time.

2.1 Constitution of an assembly line in introduced notations

Let us introduce the following notations for the robust formulation of simple assembly line balancing problem (SALBP-R): the set $V = \{1, \ldots, n\}$ of production tasks; the set $W = \{1, \ldots, m\}$ of workstations; the vector $t = (t_1, \ldots, t_n)$ of nominal processing times; and the cycle time T. An acyclic oriented graph $G = (V, A)$ pictures manufacturing and processing order of tasks. If arc (i, j) belongs to A, then the processing of task j cannot start before the end of task i.

We are calling *solution* a combination of subsets V_p of tasks assigned to the workstation p such that the expressions hold:

$$V = \bigcup_{p \in W} V_p, \tag{2.1}$$

$$V_p \cap V_q = \emptyset, \quad \forall p, q \in W, \text{ where } p \neq q. \tag{2.2}$$

If a solution satisfies the precedence (2.3) and cycle time (2.4) constraints, then it is called feasible. If two tasks i and j are assigned to the same workstation, then the task j has to

be performed later than task i.

$$(i,j) \in A \Rightarrow p \leq q, \text{ where } i \in V_p, j \in V_q, \tag{2.3}$$

$$\sum_{j \in V_p} t_j \leq T, \quad \forall p \in W. \tag{2.4}$$

It is supposed here that there exist two sets of *uncertain tasks*: a non-empty set $\widetilde{V}^1$ ($\widetilde{V}^1 \subseteq V$) of *a priori uncertain tasks* whose processing time may deviate from its nominal value with regard to time without any additional information and a set $\widetilde{V}^2$ ($\widetilde{V}^2 \subseteq V$) of *a posteriori uncertain tasks* whose uncertainty is caused by a set $\widehat{W}$ ($\widehat{W} \subseteq W$) of *uncertain workstations*. These workstations are such that any task allocated to them becomes uncertain (even if it belongs to $V \setminus \widetilde{V}^1$). Hereinafter, the set of all uncertain tasks is denoted as $\widetilde{V} = \widetilde{V}^1 \cup \widetilde{V}^2$ and any workstation from $W \setminus \widehat{W}$ is called *certain*. The presence of certain and/or uncertain workstations can be explained by the existence of assembly lines having simultaneously two types of workstations: workstations with automatic tasks executed by robots or machines and workplaces where tasks are operated in a manual manner, respectively.

To evaluate the robustness of a feasible solution, we use the concept of *stability radius* whose formal definition requires some supplementary notations:

- $F(t)$ is the set of feasible solutions ((2.1)–(2.4) hold) with respect to a given vector $t \in \mathbb{R}^n_+$;
- Ξ is the set of vectors, where each of which presents possible non-negative processing time deviations for the uncertain tasks, *i.e.*, $\{\xi \in \mathbb{R}^n_+ \mid \xi_j = 0,\ j \in V \setminus \widetilde{V}\}$[1];

Thus, the stability radius of a feasible solution $s \in F(t)$ can be defined as follows (see [66]):

$$\rho(s,t) = \max\{\varepsilon \geq 0 \mid \forall \xi \in B(\varepsilon)\ s \in F(t+\xi)\}, \tag{2.5}$$

where $B(\varepsilon) = \{\xi \in \Xi \mid \|\xi\| \leq \varepsilon\}$.

In other words, $\rho(s,t)$ is determined as the value of the radius of the greatest closed ball $B(\cdot)$, called *stability ball*, representing the deviations of the uncertain task nominal processing times, for which s remains feasible. Any element ξ of $B(\cdot)$ is evaluated based on a given norm $\|\cdot\|$ defining the distance between vectors t and $t+\xi$ (or the amplitude of deviations from t).

We may interpret the stability radius in l_1-metric as the maximal total deviation of nominal processing times of all uncertain tasks which does not violate a solution feasibility

[1]Since any decrease of task processing time cannot compromise the solution feasibility, it is sufficient to consider only non-negative task time deviations, *i.e.*, $\xi_j \in \mathbb{R}_+$ for any $j \in \widetilde{V}$.

or optimality. On another side, the stability radius in l_∞-metric shows the maximal deviation of nominal processing time of any uncertain task which does not violate a solution feasibility or optimality.

The work [58] proposes two classic norm for calculation of the stability radius: $\|\cdot\|_1$ and $\|\cdot\|_\infty$, where by the definition $\|\xi\|_1 = \sum_{j\in\widetilde{V}} \xi_j$ $\|\xi\|_\infty = \max_{j\in\widetilde{V}} \xi_j$. Notations $\rho_1(\cdot,\cdot)$, $B_1(\cdot)$ and $\rho_\infty(\cdot,\cdot)$, $B_\infty(\cdot)$ are used to distinguish respective values and variables.

The next two theorems show how to calculate the exact value of the stability radius for a given feasible solution for two introduced norms. They prove that the stability radius in the ℓ_1-norm is equal to the minimum idle time among the workstations containing uncertain tasks. While for the ℓ_∞-norm, it needs to seek the workstation that provides the minimum value of the idle time divided by the number of its uncertain tasks.

Please note that notations $\widetilde{W}$ and $\widehat{W}$ have a different meaning. While $\widehat{W}$ includes initially uncertain workstations, $\widetilde{W}$ collects both initially uncertain and those from $W\setminus\widehat{W}$ which have at least one uncertain task. In general, we have the following relation on workstations: $\widehat{W}\subseteq\widetilde{W}\subseteq W$.

Theorem 1 (**Rossi et al. [58]**). *The stability radius $\rho_1(s,t)$ of $s\in F(t)$ is calculated as follows:*

$$\rho_1(s,t) = \min_{p\in\widetilde{W}} \left(T - \sum_{j\in V_p} t_j \right) \tag{2.6}$$

Theorem 2 (**Rossi et al. [58]**). *The stability radius $\rho_\infty(s,t)$ of $s\in F(t)$ is calculated as follows:*

$$\rho_\infty(s,t) = \min_{p\in\widetilde{W}} \frac{T - \sum\limits_{j\in V_p} t_j}{|\widetilde{V}_p|} \tag{2.7}$$

Here, we address a novel interpretation of the stability radius called *relative resilience*.

It refers to the situation where the processing time increase of uncertain tasks is bounded by an amount of time that is proportional to its nominal duration.

Mathematically, the vector ξ is formed by $\xi_j = \alpha t_j$ for $j\in\widetilde{V}$ and $\xi_j = 0$ for $j\in V\setminus\widetilde{V}$, where $\alpha\geq 0$. We do not present α as a standard norm, but just as rate of increase in the duration of the uncertain tasks that one aims to maximise. The new stability radius and ball are denoted by $\rho_{rr}(\cdot,\cdot)$ and by $B_{rr}(\cdot)$.

Theorem 3. *The relative resilience $\rho_{rr}(s,t)$ of $s\in F(t)$ is calculated as follows*

$$\rho_{rr}(s,t) = \min_{p\in\widetilde{W}} \frac{T - \sum\limits_{j\in V_p} t_j}{\sum\limits_{j\in\widetilde{V}_p} t_j}. \tag{2.8}$$

Proof. Let us denote the right-hand side of (2.8) as φ, and the left-hand as just ρ. To prove the present theorem, on should show that $\rho \geq \varphi$ and $\rho \leq \varphi$.

First start with $\rho \geq \varphi$. Let p be an uncertain workstation, exposed to stand the processing time perturbations, which ratio α does not exceed φ. It is not difficult to see that its *perturbed* load is constructed with the following expression:

$$\sum_{j \in V_p} t_j + \alpha \cdot \sum_{j \in \widetilde{V}_p} t_j. \tag{2.9}$$

Taking into account the fact that $\alpha \leq \varphi$ and the inequality $\varphi \leq \frac{T - \sum\limits_{j \in V_p} t_j}{\sum\limits_{j \in \widetilde{V}_p} t_j}$, which is valid for any $p \in \widetilde{W}$ due to the definition of φ, we obtain

$$(2.9) \leq \sum_{j \in V_p} t_j + \frac{T - \sum\limits_{j \in V_p} t_j}{\sum\limits_{j \in \widetilde{V}_p} t_j} \cdot \sum_{j \in \widetilde{V}_p} t_j = T.$$

The latter demonstrates that the load of the workstation p does not exceed the cycle time, whatever a perturbation ratio less or equal to φ. This proves $\rho \geq \varphi$.

Now let us prove that $\rho \leq \varphi$. To do this, it is sufficient to check that any perturbation ratio $\alpha > \varphi$ causes the considered feasible solution to be unfeasible.

As above, based on the definition of φ, we deduce that there exists an uncertain workstation p^* such that $\varphi = \frac{T - \sum_{j \in V_{p^*}} t_j}{\sum_{j \in \widetilde{V}_{p^*}} t_j}$. Then, we can notice that the *perturbed* load (with respect to the ratio α) of the machine p^* violates the cycle time constraint, since

$$\sum_{j \in V_p^*} t_j^{(\alpha)} = \sum_{j \in V_p^*} t_j + \alpha \cdot \sum_{j \in \widetilde{V}_p^*} t_j > \sum_{j \in V_p^*} t_j + \frac{T - \sum\limits_{j \in V_p^*} t_j}{\sum\limits_{j \in \widetilde{V}_p^*} t_j} \cdot \sum_{j \in \widetilde{V}_p^*} t_j = T$$

that proves $\rho \leq \varphi$. □

Thus, to compute the relative resilience of a given solution, we need to seek for the workstation that provides the minimal ratio between its idle time and the sum of processing times of its uncertain tasks, it can be done in polynomial time.

2.2 MILP formulations

Different interpretations of the norm represent different challenges which may happen in the production process. A decision maker (manager) is able to chose those one that

corresponds to his (her) particular case. In this section we are showing the similarity in mixed integer linear problem formulations for three described metrics: ℓ_1, ℓ_∞ and ℓ_{rr}. At the beginning, let introduce necessary decision variables:

- ρ is the stability radius value to maximise;
- $x_{j,p}$ is a binary variable that is set to one if and only if the task j is allocated to the workstation p;
- a_p is a non-negative variable that is positive if the workstation p has at least one uncertain task, or if the workstation p is uncertain (ℓ_1-metric special variable, not applicable for another metrics);
- $\xi_{j,p}$ is a processing time deviation of the task j on the workstation p:
  - in ℓ_∞-metric it shows a real deviation of processing times;
  - in the relative resilience it models a percentage for which processing times are deviating from their nominal values;
  - variables are not applicable for ℓ_1-metric.

The objective function of the considered problem is the same for all formulations:

$$\text{Maximise } \rho$$

Also, production tasks are indivisible, and any of them has to be assigned to exactly one workstation:

$$\sum_{p \in W} x_{j,p} = 1, \quad \forall j \in V. \tag{2.10}$$

Any feasible line balance must satisfy the precedence constraints:

$$\sum_{p \in W} p x_{i,p} \leq \sum_{p \in W} p x_{j,p}, \quad \forall (i,j) \in A, \tag{2.11}$$

$$\sum_{q=p}^{m} x_{i,q} \leq \sum_{q=p}^{m} x_{j,q}, \quad \forall (i,j) \in A, \ \forall p \in W \setminus \{1\}. \tag{2.12}$$

Here, restrictions (2.11) compare indexes of workstation, which accommodate production tasks i and j. If the task j succeeds the task i, then it cannot be assigned to any workstation with an index smaller than the one of the task i. Constraints (2.12) are more classic one. They verify subset of workstation for the presence of tasks. Since, variables $x_{j,p}$ are binary, the left-hand part of (2.12) is turning to zero not later than the right-hand. It could be essential to know that these constraints are surplus, and it is possible to use only one of them depending on a particular studied problem and an interpretation of precedence constraints.

2.2.1 MILP formulation for ℓ_1-metric

The central idea of the MILP formulation for P_1 consists in considering ρ as the minimum idle time of the workstations that process uncertain tasks (see Theorem 1). The cycle time constraints in this metric are presented by its classic version:

$$\sum_{j \in V} t_j x_{j,p} \leq T, \quad \forall p \in W. \tag{2.13}$$

It is possible because tasks time deviations are modelled with the next expressions:

$$x_{j,p} \leq a_p, \quad \forall j \in \widetilde{V}^1, \ \forall p \in W \setminus \widehat{W}, \tag{2.14}$$

$$a_p = 1, \quad \forall p \in \widehat{W}, \tag{2.15}$$

$$\rho \leq T(2 - a_p) - \sum_{j \in V} t_j x_{j,p}, \quad \forall p \in W. \tag{2.16}$$

According to Theorem 1, the stability radius is calculated on the subset $\widetilde{W}$, which consists of two parts: constraints (2.14) indicate if any uncertain task is assigned to the certain workstation p; equations (2.15) set up a_p for all initially uncertain workstations. Finally, the value of the stability radius is obtained as the minimal idle time on the line (see (2.16)). Indeed, it is easy to see that (2.14)–(2.16) imply that $a_p \in \{0, 1\}$. As a consequence, if p is a certain workstation without uncertain tasks, then $a_p = 0$ and (2.13) together with (2.16) yields $\rho \leq T$, which is always valid.

The complete MILP formulation for P_1 maximises ρ and has the constraints (2.10), (2.11), (2.12), (2.13), (2.14), (2.15), (2.16) and the restrictions on variables:

$$\begin{aligned} \rho &\geq 0, \\ a_p &\geq 0, \qquad \forall p \in W \\ x_{j,p} &\in \{0, 1\}, \qquad \forall j \in V, \ \forall p \in W. \end{aligned}$$

2.2.2 MILP formulation for ℓ_∞-metric

MILP formulation for P_∞ is focused on the fact that the processing time of all uncertain tasks can be increased by ρ without losing the feasibility for the optimal solution. The modelling of cycle time constraints requires to take into account possible deviations $\xi_{j,p}$:

$$\sum_{j \in V} t_j x_{j,p} + \sum_{j \in V} \xi_{j,p} \leq T, \quad \forall p \in \widehat{W}, \tag{2.17}$$

$$\sum_{j \in V} t_j x_{j,p} + \sum_{j \in \widetilde{V}^1} \xi_{j,p} \leq T, \quad \forall p \in W \setminus \widehat{W}. \tag{2.18}$$

Any task assigned to $p \in \widehat{W}$ may increase its processing time, that is shown in (2.17). If the workstation p is certain, then only tasks from $\widetilde{V}^1$ may be deviated, see constraints (2.18).

To obtain the result related to Theorem 2, we have to be sure that all tasks time deviations are equal:

$$\rho = \sum_{p \in W} \xi_{j,p}, \quad \forall j \in V. \tag{2.19}$$

Additionally, we may install an upper bound for $\xi_{j,p}$:

$$\xi_{j,p} \leq T \cdot x_{j,p}, \quad \forall j \in V, \ \forall k \in W. \tag{2.20}$$

Any appropriate value UB can be used instead of T in 2.20 (see [58] for details).

Thus, MILP formulation for P_∞ maximises ρ and has the constraints (2.10), (2.11), (2.12), (2.17), (2.18), (2.19), (2.20) and the restrictions on variables:

$$\begin{aligned} \rho &\geq 0, \\ \xi_{j,p} &\geq 0, \qquad \forall j \in V, \ \forall p \in W, \\ x_{j,p} &\in \{0,1\}, \qquad \forall j \in V, \ \forall p \in W. \end{aligned}$$

2.2.3 MILP formulation for ℓ_{rr}-metric

The last model evolves directly from the previous one. It is recalled that, in practice, uncertain tasks can have different processing time deviations. From this point of view, the formulation for P_∞ seems overprotective and less realistic. The new model is changing this considering relative resilience, *i.e.*, the processing time of all uncertain tasks can be increased by a proportion ρ without loosing the feasibility for the optimal solution.

The following constraints together with (2.10), (2.11), (2.12) are presenting the formulation for P_{rr}:

$$\xi_{j,p} \leq UB \cdot x_{j,p}, \qquad \forall j \in V, \ \forall p \in W, \tag{2.21}$$

$$\rho = \sum_{p \in W} \xi_{j,p}, \qquad \forall j \in V, \tag{2.22}$$

$$\sum_{j \in V} t_j x_{j,k} + \sum_{j \in V} t_j \xi_{j,k} \leq T, \qquad \forall k \in \widehat{W}, \tag{2.23}$$

$$\sum_{j \in V} t_j x_{j,k} + \sum_{j \in \widetilde{V}^1} t_j \xi_{j,k} \leq T, \qquad \forall k \in W \setminus \widehat{W}, \tag{2.24}$$

$$\begin{aligned} \xi_{j,k} &\geq 0, \qquad \forall j \in V, \ \forall k \in W, \\ x_{j,k} &\in \{0,1\}, \qquad \forall j \in V, \ \forall k \in W. \end{aligned}$$

Constraints (2.21) ensure that only for one $p \in W$, $\xi_{j,p}$ is non-zero for any fixed $j \in V$ and state that the variation of any uncertain task is bounded by UB, where $UB = \frac{T - \max_{j \in \widetilde{V}^1} t_j}{\max_{j \in \widetilde{V}^1} t_j}$. Inequalities (2.22) state that the deviation ratio of any task is set to ρ. The perturbed load of each workstation does not exceed the cycle time, as shown in constraints (2.23) and (2.24).

2.3 Enhancements

There is no preference in an assignment of tasks to workstations. Nevertheless, the precedence graph usually may say which task would be in the beginning or in the end of the line. This leads us to the introduction of assignment intervals for all production tasks. An interval $Q_j = [l_j, u_j]$ shows that the task j cannot be allocated to the workstation whose place on the line is either earlier than l_j or later than u_j. Such boundaries may be estimated as follows:

$$l_j = \left\lceil \frac{t_j + \sum\limits_{i \in \mathcal{P}(j)} t_i}{T} \right\rceil, \; u_j = m + 1 - \left\lceil \frac{t_j + \sum\limits_{i \in \mathcal{S}(j)} t_i}{T} \right\rceil, \tag{2.25}$$

where $\mathcal{P}(j)$ (resp. $\mathcal{S}(j)$) is a set of all predecessors (resp. all successors) of j in the graph G. Using this information, models can be enriched by the respective constraints:

$$x_{j,p} = 0, \; \forall j \in V, \; \forall p \notin Q_j. \tag{2.26}$$

2.4 Polynomial time solvable case

In this section, we want to briefly discuss an effectively solvable case of considered problems.

Statement 1. *If the number of workstations is equivalent to the number of production tasks, i.e., $m = n$, then SALBP-R can be solved in polynomial time.*

Indeed, in this case any workstation p includes exactly one production task, and its idle time equals to $(T - t_j)$, where $j \in V_p$. If the workstation is uncertain then $V_p = \widetilde{V}_p$. For any feasible (according to the precedence order) line balance s, the stability radius is calculated with one of the following formulas:

- $\rho_1(s,t) = \min\limits_{p \in \widetilde{W}} \{T - t_j \mid j \in V_p\} = T - \max\limits_{j \in \widetilde{V}} t_j$;

- $\rho_\infty(s,t) = \min\limits_{p\in\widetilde{W}} \left\{ \frac{T-t_j}{|\widetilde{V}_p|} \mid j \in V_p \right\} = T - \max\limits_{j\in\widetilde{V}} t_j$, since $|\widetilde{V}_p| = 1$ for all workstations;
- $\rho_{rr}(s,t) = \min\limits_{p\in\widetilde{W}} \left\{ \frac{T-t_j}{t_j} \mid j \in V_p \right\} = \frac{T}{\max\limits_{j\in\widetilde{V}} t_j} - 1.$

It is not difficult to see, that these formulas also present upper bounds on the stability radius for respective metrics. In particular, the formula for the relative resiliency is used to calculate upper bound UB in the constraint (2.21), while the formula for ℓ_∞ can be applied in (2.20) instead of T.

Allocation of tasks to workstations can be performed by any ranking or enumeration algorithm, which sorts nodes of the precedence graph in order of their appearance.

2.5 Computational results

To examine the MILP model for P_{rr} we used the same instances used in [58] which consist of a set of 25 instances that can be found at `http://alb.mansci.de/`. For these instances we added the number of workstations and the cycle time, set as $m = \left\lceil 1.2 \times \frac{\sum_{j\in V} t_j}{T} \right\rceil$ and $T = 1.5 \times \max_{j\in V} t_j$. After that a random permutation of the tasks and workstations have been generated. In any test, only the first $|\widetilde{V}^1|$ elements are considered as uncertain tasks and, similarly, the first $|\widehat{W}|$ elements from the second permutation as uncertain workstations. All these permutations are given in Appendix B. To distinguish the tests we use the number of uncertain tasks and number of uncertain workstations as varying parameters, such that $|\widetilde{V}^1| \in \{\lceil \frac{n}{4} \rceil, \lceil \frac{n}{2} \rceil, \lceil \frac{3n}{4} \rceil, n\}$, and $|\widehat{W}| \in \{0, \lceil \frac{m}{4} \rceil, \lceil \frac{m}{2} \rceil\}$.

The computational results were carried out on a laptop having Inter Core i5 2.5 GHz and 4 GB RAM. The program code was developed in C++ using the commercial solver GUROBI 7.51. An example of results is given in Table 2.1. It is built as follows. The first four columns indicate respectively the instance's name, number of tasks, number of workstations, and cycle time. The next column is the best value of stability radius found (LB), followed by the upper bound (UB) found by GUROBI. Column 7 report the CPU time in seconds for solving the instance and the last column display the gap. The last row of every table displays the number of instances solved optimally, the average computational time and the average gap over all instances. The solving of each instance is limited by 600 seconds, then if an instance is not solved optimally within the time limit, the number 600 is indicated its CPU. The detailed results of each series are given in Appendix A.

Tables A.1 to A.4 gives the results for series of instances without uncertain stations, i.e., $|\widehat{W}| = 0$ for all of them, while $|\widetilde{V}^1| \in \left\{\lceil \frac{n}{4} \rceil, \lceil \frac{n}{2} \rceil, \lceil \frac{3n}{4} \rceil, n\right\}$. In contrary, Tables A.5 to A.10 show all results where both the number of uncertain tasks and stations are gradually increased with each series.

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.36	0
BOWMAN8	8	4	25.5	0.375	0.375	0.02	0
MANSOOR	11	4	67.5	1.813	1.813	0.04	0
JAESCHKE	9	5	9	0.2	0.2	0.01	0
JACKSON	11	6	10.5	0.3	0.3	0.06	0
MITCHELL	21	7	19.5	0.833	0.833	0.07	0
ROSZIEG	25	8	19.5	0.5	0.5	0.01	0
HESKIA	28	8	162	0.514	0.514	0.03	0
LUTZ1	32	9	2100	0.5	0.5	0.01	0
BUXEY	29	11	37.5	0.5	0.5	0.06	0
SAWYER30	30	11	37.5	0.974	0.974	0.32	0
GUNTHER	35	10	60	0.45	0.45	0.06	0
HAHN	53	7	2662.5	0.793	0.793	0.13	0
KILBRID	45	9	82.5	0.5	0.5	0.04	0
TONGE70	70	18	234	0.5	0.5	0.89	0
WARNECKE	58	24	79.5	0.529	0.529	0.64	0
ARC83	83	17	5536.5	1.018	1.045	600	0.027
LUTZ3	89	18	111	0.5	0.5	0.65	0
BARTHOLD	148	12	574.5	0.5	0.5	1.11	0
MUKHERJE	94	20	256.5	0.5	0.5	0.53	0
ARC111	111	22	8533.5	0.687	0.687	3.68	0
LUTZ2	89	38	15	0.5	0.5	19.44	0
WEE-MAG	75	60	40.5	0.558	0.558	6.51	0
BARTHOL2	148	41	124.5	0.5	0.5	7.92	0
SCHOLL	297	41	2079	0.878	0.949	600	0.08
				#OPT: **23/25**		**49.70**	**0.004**

Table 2.1: Solving P_{rr} with GUROBI for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = 0$

For the sake of illustrating the results presented in these tables, the solutions returned for the instance JACKSON are shown in Figures 2.2 – 2.5. That instance is defined by the precedence graph in Figure 2.1.

Table A.2 includes the results where half of production tasks are uncertain, but all workstations are certain. Figure 2.2 shows the optimal solution obtained by GUROBI for the presented MILP formulation. Uncertain tasks are pictured as grey rectangles. There are tasks 2, 3, 7, 8, 9 and 10. Its length equals to the respective nominal processing time. The optimal value of the stability radius is $\rho = 0.3$. By the definition, any uncertain task may increase its processing time in $(1 + \rho)$ times. For example, the task 3 has $t_3 = 5$, then after the deviation, the new time would be $(1 + \rho) \cdot t_3 = 1.3 \cdot 5 = 6.5$. Figure 2.3 presents task times deviations. Here, the load time of workstations 4 and 6 hits the cycle time, which means that any smallest perturbation bigger than ρ will lead to the process

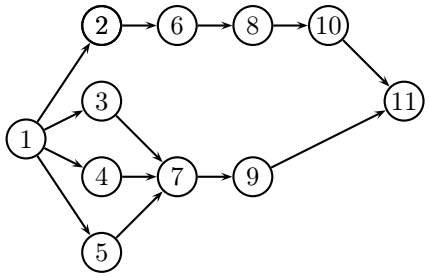

Figure 2.1: Precedence constraints of the instance JACKSON

interruption.

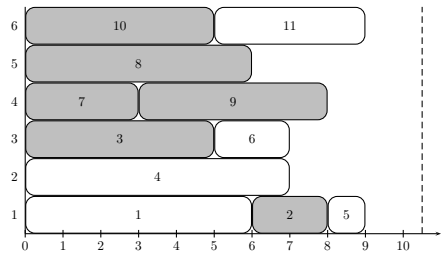

Figure 2.2: Optimal solution for JACKSON, $|\widetilde{V}^1| = 6$ and $|\widehat{W}| = 0$.

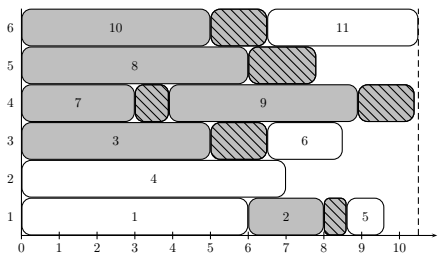

Figure 2.3: Deviation by the stability radius value for the solution on Figure 2.2

The group of results for similar tests but with uncertain workstations is presented in Table A.8. Again, tasks 2, 3, 7, 8, 9 and 10 are uncertain, but also workstations 3, 5 and 6 are uncertain (see for details Appendix B). Then, all the tasks assigned to one of them become uncertain as well. In this situation, the optimal value of the stability radius is $\rho = 0.1667$. The line balance with all deviations is in Figure 2.5. Clearly, we cannot apply any variation bigger than ρ, since the workstation 6 reaches the cycle time.

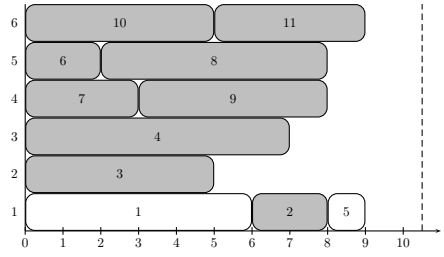

Figure 2.4: Optimal solution for JACKSON, $|\widetilde{V}^1| = 6$ and $|\widehat{W}| = 3$.

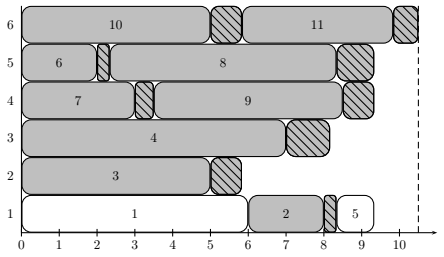

Figure 2.5: Deviation by the stability radius value for the solution on Figure 2.4

This observation shows how important the knowledge of the initial data and how

solution is changing according to it. Moreover, if we compare our results with analogical one for two classic metrics in the paper [58], then we can see the impact of the metric. Table 2.2 presents the values of the stability radius for the instance JACKSON in all three metrics and different values of parameters $\widetilde{V}^1$ and $\widehat{W}$. The maximal total deviation keeps the same value through all tests (value ρ_1), while the maximal uniform deviation (ρ_∞) is reducing from 1.5 to 0.75 when the number of uncertain tasks and workstations are growing up. For the proportional robustness the value also goes down, but it shows the processing time deviation in percents from the initial value. Because of this, we provide the column "$\rho_{rr} \cdot t_{\max}$" which represents the deviation of the longest task. In general case, all three values are different, and it says to decision makers how they can analyse a problem and chose a better combination of parameters according to the available knowledge.

$\lvert\widetilde{V}^1\rvert$	$\lvert\widehat{W}\rvert$	ρ_1	ρ_∞	ρ_{rr}	$\rho_{rr} \cdot t_{\max}$
$\lceil \frac{n}{4} \rceil$	0	1.5	1.5	0.3	2.1
$\lceil \frac{n}{2} \rceil$	0	1.5	1.25	0.3	2.1
n	0	1.5	0.75	0.167	1.167
$\lceil \frac{n}{4} \rceil$	$\lceil \frac{m}{4} \rceil$	1.5	1.5	0.3	2.1
$\lceil \frac{n}{2} \rceil$	$\lceil \frac{m}{2} \rceil$	1.5	0.75	0.167	1.167

Table 2.2: Comparison of results for JACKSON

A summary of Tables A.1–A.10 (placed in Appendix A) is given in Table 2.3.

The complexity for resolution raises together with either the number of uncertain tasks or number of uncertain workstations. Firstly, we can see it in Tables A.1, A.2, A.3 and A.4. The number of optimal solutions in them decreases as follow: 23–18–15–14. The average gap is also growing up from 0.004 to 0.054, 0.071 and 0.162.

Secondarily, series with uncertain stations can be compared in pairs. In Tables A.5 and A.6, the number of optimal solutions falls down from 22 to 18, and the average gap is changing from 0.054 to 0.153. And so on for all rest the tables in Appendix A.

However, all three series with $|\widetilde{V}^1| = \lceil \frac{3n}{4} \rceil$ (Series 3, 9 and 10 in Table 2.3) show that the small amount of uncertain workstations may bring minor improvements to results comparing to those one without them. We can explain this by the fact that for the high number of uncertain tasks, the set of uncertain workstation makes in advance some solutions comparable or even equivalent. It helps to find an optimal solution faster in some instances, but not in general. In the end, all the results fit to the last test with every tasks considered as uncertain which has the lowest number of instances solved optimally. Since the increase of $|\widetilde{V}^1|$ and $|\widehat{W}|$ means that more tasks and workstations are involved in the objective values, it causes the increase of difficulty for a commercial solver to find an optimal line balance.

Series	$\lvert\widetilde{V}^1\rvert$	$\lvert\widehat{W}\rvert$	#OPT	Avg. CPU	Avg. GAP
1	$\lceil\frac{n}{4}\rceil$	0	23	49.70	0.004
2	$\lceil\frac{n}{2}\rceil$	0	18	174.49	0.054
3	$\lceil\frac{3n}{4}\rceil$	0	15	243.95	0.071
4	n	0	14	268.17	0.162
5	$\lceil\frac{n}{4}\rceil$	$\lceil\frac{m}{4}\rceil$	22	77.80	0.054
6	$\lceil\frac{n}{4}\rceil$	$\lceil\frac{m}{2}\rceil$	18	185.25	0.153
7	$\lceil\frac{n}{2}\rceil$	$\lceil\frac{m}{4}\rceil$	18	190.64	0.059
8	$\lceil\frac{n}{2}\rceil$	$\lceil\frac{m}{2}\rceil$	16	217.61	0.093
9	$\lceil\frac{3n}{4}\rceil$	$\lceil\frac{m}{4}\rceil$	16	225.05	0.135
10	$\lceil\frac{3n}{4}\rceil$	$\lceil\frac{m}{2}\rceil$	15	253.94	0.138

Table 2.3: Summary results for all series

2.6 Conclusion and perspectives

The robust balancing for simple paced assembly lines without buffer stock nor parallel workstations was considered. It consists in finding a line configuration with the greatest stability radius subject to restricted number of workstations, fixed cycle time, precedence constraints, and task time variability. The stability radius is evaluated in three metrics: ℓ_1, ℓ_∞ and ℓ_{rr}. For each of them, the corresponding problem was proven to be strongly $\mathcal{NP}$-hard (see [58]). A MILP formulation were proposed for each problem. Numerical results show that the used commercial solver can find an optimal solution in less than 10 minutes for half of the instances.

We analysed the influence of different parameters on the value of the stability radius. Among them there are the number of uncertain tasks and stations, graph density and chosen metric. The results show that the next step for future research should be concerned about development of efficient methods for the problem resolution. Branch-and-bound and dynamic algorithms seem more appropriate, but there are also several meta-heuristics that can be applied. Another attractive idea is investigating a new industrial optimisation problem that can be called as "reverse robust balancing". This problem appears when we seek a simple assembly line configuration whose stability radius has to be greater than a given value enforced by a decision maker. In this situation, it is not always possible to find such a solution and the unique possibility to get around this difficulty is to use parallel workstations with duplicated tasks that require supplementary financial expenses. As a consequence, the aim of the reverse robust balancing problem is to find a line configuration with the desired value of stability radius, while minimising the number of parallel workstations.

Chapter 3

Robust Balancing of Transfer Lines

Contents

We consider a transfer line balancing problem (TLBP) in which each production task must be assigned to one element of the set of stations used for the task execution. Any station is formed of several mechanical multi-spindle heads called blocks. Production tasks assigned to the same block are going to be operated in parallel mode by tools installed to the multi-spindle head. The order of blocks on the station is linear and fixed. An objective is maximising the line robustness under the given number of stations and cycle time constraints. We assume that during the lifespan of the transfer line, the minor product properties as well as material types may change and, consequently, deviate

the processing times of production tasks from initially estimated values. The stability radius is used as an appropriate robustness measure for the line balancing with task time deviations. We elaborate formulas of the stability radius calculation for two traditional metrics and develop respective mixed integer linear programming (MILP) models and a general heuristic approach for the problem resolution.

3.1 Constitution of a transfer line

Transfer lines are widely used in mechanical industry for mass production of a single type of product or several types of product if the differences in their parameters are negligible [28], [39]. These lines also can represent subsections (sub-conveyor) on an assembly line, which produce production parts to be assembled [35]. Designing transfer lines is a very complex problem due to manufacturing and design constraints and to the large number of possible decisions. At the design stage, all the required tasks to manufacture a product are known in advance. Since stations presented in mechanical industry are generally automatic and semi-automatic, here and after, we are using the word *machine* to picture a single device on the line. It is necessary to define the machines and pieces of equipment (tools and spindle heads) to create the corresponding production line. Such a line is characterised by a significant cost and a long exploitation period. Thus, finding a good (and if possible the best) design solution is a crucial problem.

The transfer line allows reducing the number of machines and pieces of equipment (line cost) as well as the occupied area for the considered line, assigning the tasks into blocks. There is at least one block at each machine. All tasks of the same block are executed simultaneously by one spindle head. Each spindle head may have several tools which perform tasks simultaneously. As in previous chapter, we consider the case where there are no intermediate buffers; all blocks (spindle heads) of the same machine are executed sequentially, all the machines are linearly ordered and are activated simultaneously. Thus, at the preliminary design stage, it is necessary to assign each task required to manufacture a product to one block, and each block to one machine. The assignment of blocks to a machine defines at the same time the order of their activation on it. An example of such a line is given in Figure 3.1.

We assume that the working time of each block is equal to the longest processing time among all tasks of this block. The load time of any machine is equal to the sum of block working times (for all blocks assigned to this machine).

In addition to the fact that the balancing of transfer lines is similar to the one for assembly lines, known methods for SALB problems cannot be used directly for the problem at hand because:

- the tasks of each block are executed simultaneously (the block time is equal to the longest processing time and not the sum of processing times);

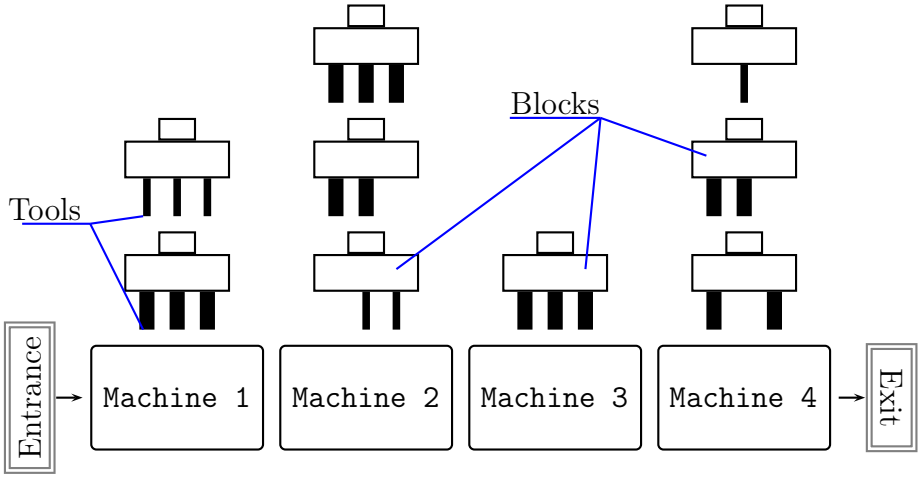

Figure 3.1: Scheme of a transfer line with four machines

- the blocks are not known in advance and must be built during optimisation;
- additional constraints on compatibility of grouping operations in the blocks and at the stations may exist (to obtain realisable design decisions).

The problem considered in the chapter is then an extension of SALB problems. And of course, if each block can process a single task at a time, then the problem is identical to SALBP.

Among the classic formulation, where the objective is to minimise either the number of machines (and blocks) or cycle time, we can find exact optimisation methods ([16], [17], [19], [20]) and heuristic procedures ([5], [18], [24]). Comparison of these methods was presented by Guschinskaya and Dolgui in [34].

The formulations proposed in this thesis deal with robustness of transfer lines. The number of machines and cycle time are given a priori, and we are maximising the robustness that is measured by the stability radius. In the next section, we explain how to calculate its value for balances in two classic metrics.

3.2 Stability radius

The balancing transfer lines consists of an assignment of the set of production tasks $V = \{1, \ldots, n\}$ to the set of machines $W = \{1, \ldots, m\}$ such that some tasks are grouped

into blocks for parallel execution. Any task $j \in V$ is known with its nominal processing time t_j. It is considered that there exists a non-empty set of *uncertain tasks* $\widetilde{V}$ ($\widetilde{V} \subseteq V$) whose processing time may deviate from its nominal value with regard to time without any additional information. Each other task from $V \setminus \widetilde{V}$ is named *certain*. By contrast with the previous chapter, we do not consider uncertain machines, which means that all the machines are certain. This implies that the origin of uncertainty is restricted to tasks only, regardless of the machines that process them.

To evaluate the robustness of a feasible solution, we use the concept of *stability radius* whose formal definition requires some supplementary notations, which can be found in Table 3.1 together with detailed basic notations.

V	is the set of all necessary tasks, *i.e.*, $\{1, 2, \ldots, n\}$;
$\widetilde{V}$	is the set of uncertain tasks, where $\widetilde{V} \subseteq V$;
W	is the set of all available machines, *i.e.*, $\{1, 2, \ldots, m\}$;
G	is a directed acyclic graph (V, A) representing the precedence constraints, where A is the set of arcs;
t_j	is a non-negative nominal processing time of task j;
t	is a vector expressing the nominal processing task times, *i.e.*, $(t_1, t_2, \ldots, t_n)$;
$F(t)$	is the set of all feasible solutions with respect to a given vector t;
Ξ	is the set of vectors, where each of which presents possible non-negative processing time deviations for the uncertain tasks, *i.e.*, $\{\xi \in \mathbb{R}^n_+ \mid \xi_j = 0,\ j \in V \setminus \widetilde{V}\}$[2];
T	is the cycle time;
$r_{\max}$	is the maximal number of tasks, which can be assigned to one block;
$b_{\max}$	is the maximal number of blocks, which can be allocated into one machine;
U	is the set of all available blocks, *i.e.*, $\{1, 2, \ldots, mb_{\max}\}$;
$U(p)$	is the set of blocks of machine p, *i.e.*, $\{(p-1) \cdot b_{\max} + 1, \ldots, p \cdot b_{\max}\}$;
V_k	is the set of all tasks assigned to block k;
$\widetilde{V}_k$	is the set of all uncertain tasks assigned to block k, *i.e.*, $V_k \cap \widetilde{V}$;
τ_k	is the nominal working time of block k, *i.e.*, $\max_{j \in V_k} t_j$;
$\widetilde{U}(p)$	is the set of uncertain blocks of machine p, where each one has at least one uncertain assigned task;
$\widetilde{W}$	is the set of uncertain machines, where each one has at least one uncertain block.

Table 3.1: Basic notations for TLBP

According to these notations, the stability radius of a feasible solution $s \in F(t)$ can be defined as follows [see 66]:

$$\rho(s, t) = \max\{\varepsilon \geq 0 \,|\, \forall \xi \in B(\varepsilon) \ \ (s \in F(t + \xi))\}, \tag{3.1}$$

where $B(\varepsilon) = \{\xi \in \Xi \,|\, \|\xi\| \leq \varepsilon\}$.

In other words, $\rho(s, t)$ is determined as the value of the radius of the greatest closed ball $B(\cdot)$, called *stability ball*, representing the deviations of the uncertain task nominal processing times, for which s remains feasible. Any element ξ of $B(\cdot)$ is evaluated based

on a given norm $\|\cdot\|$ defining the distance between vectors t and $t+\xi$ (or the amplitude of deviations from t).

Two norms ℓ_1 ($\|\cdot\|_1$) and ℓ_∞ ($\|\cdot\|_\infty$) are studied in details, where by definition $\|\xi\|_1 = \sum_{j\in\widetilde{V}} \xi_j$ and $\|\xi\|_\infty = \max_{j\in\widetilde{V}} \xi_j$. As a consequence, the notations $\rho_1(\cdot,\cdot)$, $B_1(\cdot)$ and $\rho_\infty(\cdot,\cdot)$, $B_\infty(\cdot)$ will be used for ℓ_1 and ℓ_∞, respectively.

Let us give an illustrative example of the interpretation of the stability radius in the ℓ_∞- and ℓ_1-norms. The following problem instance is considered: $n = 7$, $m = 1$, $\widetilde{V} = \{2, 5, 6\}$, $t = (5, 4, 2, 3, 2.5, 2, 2.5)$, $T = 11.5$, $r_{\max} = 3$. There is no precedence constraint. A feasible assignment is shown in Figure 3.2: the total load of the machine is less than the cycle time. Working times of blocks are $\tau_1 = 5$, $\tau_2 = 3$ and $\tau_3 = 2.5$ by the definition.

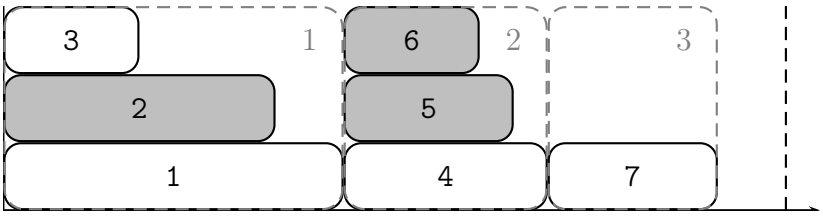

Figure 3.2: Example of assignment

The stability radius for ℓ_1-metric shows the maximal total deviation of all uncertain tasks that keeps solution admissibility. Most dangerous impact for the system would be if such deviation concentrated on only one task. The rectangle shaped by lines in Figure 3.3 represents this amplitude that is equal to 1.5. Indeed, the deviation can be applied to any of uncertain tasks: 2, 5 or 6 without having the machine load exceeding the cycle time. Similarly, the stability radius for ℓ_∞-metric is different and describes the maximal deviation of any uncertain task that does not change solution feasibility. In Figure 3.4 we increase the processing time of tasks 2 and 5 by 1.25 (two shaped rectangles). It is not difficult to see that any processing time increase larger than 1.25 compromises the cycle time.

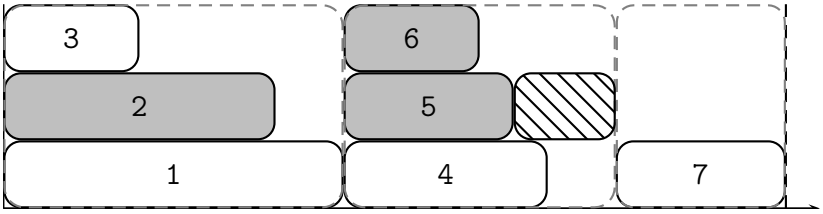

Figure 3.3: Stability radius for ℓ_1-metric

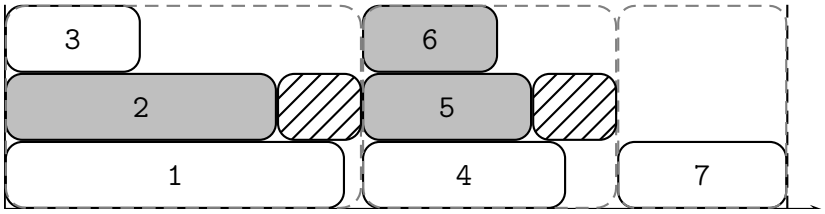

Figure 3.4: Stability radius for ℓ_∞-metric

3.2.1 Save time

Before returning to the stability radius, an important parameter called *save time* should be introduced. By contrast with simple assembly lines, the load time of a machine is the sum of its blocks working times, which are formed by the longest task assigned to the block. The example presented above shows that if the longest task is a certain one, and the block also possesses one or several uncertain tasks, then the processing time of an uncertain task may increase (to some extend) without changing neither its block working time, nor the load time of its machine. The save time is introduced to account for this situation. It is determined only for uncertain blocks and calculated as the difference between the block working time and the processing time of its longest uncertain task, *i.e.*, $\tau_k - \max_{j \in \widetilde{V}_k} t_j$, $k \in \widetilde{U}(p)$, $p \in W$. Consequently, the save time can be zero and non-zero, see Figure 3.5 and Figure 3.6, respectively. Hereafter, the save time of uncertain block k of the machine p is denoted as $\Delta_k^{(p)}$ and the *minimal save time* of the machine $p \in \widetilde{W}$ as $\Delta_{\min}^{(p)} = \min_{k \in \widetilde{U}(p)} \Delta_k^{(p)}$.

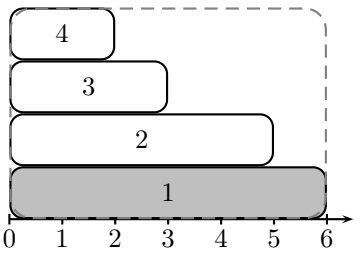

Figure 3.5: Block with uncertain tasks having zero save time

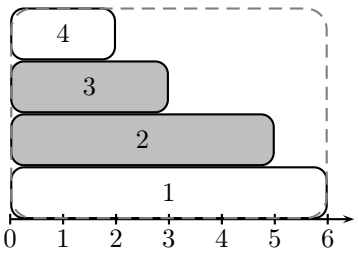

Figure 3.6: Block with uncertain tasks having non-zero save time

3.2.2 Calculation of stability radius for ℓ_1-metric

Theorem 4. *The stability radius ρ_1 for a given feasible solution is calculated as follows*

$$\rho_1 = \min_{p \in \widetilde{W}} \left\{ T - \sum_{k \in U(p)} \tau_k + \Delta_{\min}^{(p)} \right\}. \tag{3.2}$$

Proof. Let us denote the right-hand side of (3.2) as φ_1. To prove the present theorem, one needs to show that $\rho_1 \geq \varphi_1$ and $\rho_1 \leq \varphi_1$.

First start with $\rho_1 \geq \varphi_1$. Let p be an uncertain machine, exposed to stand the processing time deviations $\xi \in B_1(\varphi_1)$. It is not difficult to see that its *perturbed* load cannot be greater than the value of the following expression

$$\sum_{k \in U(p)} \tau_k + \left[\sum_{j \in \widetilde{V}} \xi_j - \Delta_{\min}^{(p)} \right]^+, \tag{3.3}$$

where $[x]^+ = \max\{0, x\}$, $x \in \mathbb{R}$. Moreover, taking into account the fact that $\xi \in B_1(\varphi_1)$ and the inequality $\varphi_1 \leq T - \sum_{k \in U(p)} \tau_k + \Delta_{\min}^{(p)}$ that is valid for any $p \in \widetilde{W}$ due to the definition of φ_1, we obtain

$$(3.3) \leq \sum_{k \in U(p)} \tau_k + \left[\varphi_1 - \Delta_{\min}^{(p)} \right]^+ \leq \sum_{k \in U(p)} \tau_k + \left[T - \sum_{k \in U(p)} \tau_k \right]^+ = T.$$

The latter demonstrates that the load of the machine p does not exceed the cycle time, whatever $\xi \in B_1(\varphi_1)$. This proves $\rho_1 \geq \varphi_1$.

Now, let us show that $\rho_1 \leq \varphi_1$. To do this, it is sufficient to check that for any $\delta > \varphi_1$ there exists a vector of processing time deviations $\xi^* \in B_1(\delta)$, which causes the considered feasible solution to be unfeasible.

As above, based on the definition of φ_1, we deduce that there exists an uncertain machine p^* so that $\varphi_1 = T - \sum_{k \in U(p^*)} \tau_k + \Delta_{\min}^{(p^*)}$. Let k^* be an uncertain block of this machine having the least save time, *i.e.*, $\Delta_{k^*}^{(p^*)} = \Delta_{\min}^{(p^*)}$, and let j^* be a longest uncertain task of this block, *i.e.*, $t_{j^*} = \max_{j \in \widetilde{V}_{k^*}} t_j$. Then, setting $\xi^* \in B_1(\delta)$, where $\xi_j^* = \delta$, if $j = j^*$ and 0 otherwise, we notice that the *perturbed* load of the machine p^* violates the cycle time constraint, since

$$\sum_{k \in U(p^*)} \tau_k + \left[\delta - \Delta_{\min}^{(p^*)} \right]^+ > \sum_{k \in U(p^*)} \tau_k + \left[\varphi_1 - \Delta_{\min}^{(p^*)} \right]^+ = \sum_{k \in U(p^*)} \tau_k + \left[T - \sum_{k \in U(p^*)} \tau_k \right]^+ = T$$

that proves $\rho_1 \leq \varphi_1$. □

In the case of $r_{\max} = 1$, the configuration of the transfer line is transformed into that of the simple assembly line, since at most one task can be allocated to block. Hence, the save time of all uncertain blocks becomes equal to zero, *i.e.*, $\Delta_k^{(p)} = 0$, $k \in \widetilde{U}(p)$, $p \in \widetilde{W}$. As a consequence, Theorem 4 directly yields

Corollary 1 ([58]). *The stability radius ρ_1 for a given configuration of simple assembly line is calculated as* $\min_{p \in \widetilde{W}} \left\{ T - \sum_{k \in U(p)} \tau_k \right\}$.

3.2.3 Calculation of stability radius for ℓ_∞-metric

For any uncertain machine p, let us introduce the following function, which is useful for further statements:

$$\theta(p, q) = \frac{T - \sum_{k \in U(p)} \tau_k + \sum_{i=1}^{q} \Delta_{\pi_i}^{(p)}}{q}, \tag{3.4}$$

where $\Delta_{\pi_1}^{(p)} \leq \Delta_{\pi_2}^{(p)} \leq \ldots \leq \Delta_{\pi_{|\widetilde{U}(p)|}}^{(p)}$ is the non-decreasing order of the save time for all uncertain blocks of the machine $p \in \widetilde{W}$ and $q = 1, \ldots, |\widetilde{U}(p)|$. The following evident expression is valid for any p and $q \in \{1, \ldots, |\widetilde{U}(p)| - 1\}$:

$$\Delta_{\pi_{q+1}}^{(p)} = (q+1) \cdot \theta(p, q+1) - q \cdot \theta(p, q). \tag{3.5}$$

Lemma 1. *Let p be a given uncertain machine having $|\widetilde{U}(p)| \geq 3$ and let there be $q^\star \in \{1, \ldots, |\widetilde{U}(p)| - 1\}$ such that $\theta(p, q^\star + 1) \geq \theta(p, q^\star)$, then*

$$\theta(p, q^\star + \alpha + 1) \geq \theta(p, q^\star + \alpha) \tag{3.6}$$

holds for any $\alpha \in \{0, \ldots, |\widetilde{U}(p)| - q^\star - 1\}$.

Proof. This lemma is proven by recurrence on α. First, it can immediately be checked that if there exists $q^\star \in \{1, \ldots, |\widetilde{U}(p)| - 1\}$ such that $\theta(p, q^\star + 1) \geq \theta(p, q^\star)$, then (3.6) is satisfied for $\alpha = 0$. Now, assume that (3.6) holds up to some integer $\alpha \leq |\widetilde{U}(p) - q^\star - 2|$, and prove that it also holds up to $\alpha + 1$. To do this, let us develop inequality (3.6). Based on expression (3.5), we obtain

$$\theta(p, q^\star + \alpha + 1) - \theta(p, q^\star + \alpha) = \frac{\Delta_{\pi_{q^\star+\alpha+1}}^{(p)} - \theta(p, q^\star + \alpha)}{q^\star + \alpha + 1}.$$

Since the latter expression is non-negative due to (3.6), then we have $\Delta_{\pi_{q^\star+\alpha+1}}^{(p)} \geq \theta(p, q^\star + \alpha)$. Multiplying it by $(q^\star + \alpha)$ and then adding $\Delta_{\pi_{q^\star+\alpha+1}}^{(p)}$, yields $\Delta_{\pi_{q^\star+\alpha+1}}^{(p)} \geq \theta(p, q^\star + \alpha + 1)$. By definition of the permutation π, we have $\Delta_{\pi_{q^\star+\alpha+2}}^{(p)} \geq \Delta_{\pi_{q^\star+\alpha+1}}^{(p)}$, and, as a consequence, we deduce that

$$\Delta_{\pi_{q^\star+\alpha+2}}^{(p)} \geq \theta(p, q^\star + \alpha + 1). \tag{3.7}$$

Now, let us show that $\theta(p, q^\star+\alpha+2) \geq \theta(p, q^\star+\alpha+1)$. By analogy with the previous development, we have

$$\theta(p, q^\star + \alpha + 2) - \theta(p, q^\star + \alpha + 1) = \frac{\Delta^{(p)}_{\pi_{q^\star+\alpha+2}} - \theta(p, q^\star + \alpha + 1)}{q^\star + \alpha + 2},$$

which is non-negative due to (3.7). This proves Lemma 1. □

Lemma 2. *Let $q^{(p)}_{\min}$ denote an index on which the function $\theta(p, q)$ reaches its minimum for a given uncertain machine p, then $\Delta^{(p)}_{\pi_{q^{(p)}_{\min}}} \leq \theta(p, q^{(p)}_{\min})$ and additionally $\theta(p, q^{(p)}_{\min}) \leq \Delta^{(p)}_{\pi_{q^{(p)}_{\min}+1}}$, if $q^{(p)}_{\min} < |\widetilde{U}(p)|$.*

Proof. First we prove that $\Delta^{(p)}_{\pi_{q^{(p)}_{\min}}} \leq \theta(p, q^{(p)}_{\min})$, which is evident if $q^{(p)}_{\min} = 1$. Let $q^{(p)}_{\min} \geq 2$. From the definition of $q^{(p)}_{\min}$, we deduce that $\theta(p, q^{(p)}_{\min}) \leq \theta(p, q^{(p)}_{\min} - 1)$. Then, using the latter inequality and representation (3.5), we obtain

$$\Delta^{(p)}_{\pi_{q^{(p)}_{\min}}} = q^{(p)}_{\min} \cdot \theta(p, q^{(p)}_{\min}) - (q^{(p)}_{\min} - 1) \cdot \theta(p, q^{(p)}_{\min} - 1) \leq$$

$$q^{(p)}_{\min} \cdot \theta(p, q^{(p)}_{\min}) - (q^{(p)}_{\min} - 1) \cdot \theta(p, q^{(p)}_{\min}) = \theta(p, q^{(p)}_{\min}).$$

Now, we prove that $\theta(p, q^{(p)}_{\min}) \leq \Delta^{(p)}_{\pi_{q^{(p)}_{\min}+1}}$, if $q^{(p)}_{\min} < |\widetilde{U}(p)|$. As above, the definition of $q^{(p)}_{\min}$ implies that $\theta(p, q^{(p)}_{\min}) \leq \theta(p, q^{(p)}_{\min} + 1)$, which yields the following

$$\Delta^{(p)}_{\pi_{q^{(p)}_{\min}+1}} = (q^{(p)}_{\min} + 1) \cdot \theta(p, q^{(p)}_{\min} + 1) - q^{(p)}_{\min} \cdot \theta(p, q^{(p)}_{\min}) \geq$$

$$(q^{(p)}_{\min} + 1) \cdot \theta(p, q^{(p)}_{\min}) - q^{(p)}_{\min} \cdot \theta(p, q^{(p)}_{\min}) = \theta(p, q^{(p)}_{\min}).$$

□

Lemma 3. *Any uncertain machine p supports each processing time deviation from $B_\infty(\theta(p, q^{(p)}_{\min}))$.*

Proof. Let p be an uncertain machine of some feasible solution, exposed to stand the processing time deviations from $B_\infty(\theta(p, q^{(p)}_{\min}))$. It is not difficult to see that its *perturbed* load can not be greater than the value of the following expression

$$\sum_{k \in U(p)} \tau_k + \sum_{l \in \widetilde{U}(p)} \left[\theta(p, q^{(p)}_{\min}) - \Delta^{(p)}_l\right]^+, \tag{3.8}$$

where $[x]^+ = \max\{0, x\}$, $x \in \mathbb{R}$. Then, based on Lemma 2 and the definition of the function $\theta(p, q)$, we obtain

$$(3.8) = \sum_{k \in U(p)} \tau_k + \sum_{j=1}^{q^{(p)}_{\min}} \left(\theta(p, q^{(p)}_{\min}) - \Delta^{(p)}_{\pi_j}\right) = \sum_{k \in U(p)} \tau_k + q^{(p)}_{\min} \cdot \theta(p, q^{(p)}_{\min}) - \sum_{j=1}^{q^{(p)}_{\min}} \Delta^{(p)}_{\pi_j} = T$$

that holds according to the introduction 3.4 of $\theta(p, q)$ and proves the present lemma. □

Figure 3.7 shows an illustrative example of a machine with five blocks which have been sorted in non-decreasing order of the save time: $\Delta_1^{(p)} \le \Delta_2^{(p)} \le \Delta_3^{(p)} \le \Delta_4^{(p)} \le \Delta_5^{(p)}$. The cycle time is 21 and the total load of the machines is 19. Using the formula we calculate the value of θ (see Table 3.2). Here, $q_{\min}^{(p)} = 3$ and the minimal value of the function $\theta(p,q)$ is 1.67. Figure 3.8 shows changing of the machines load within the deviation $\Delta_3^{(p)}$. It is not difficult to see, that the only available space for further deviations is 0.5 which should be divided between first three blocks, *i.e.*, the stability radius for this configuration is equal to $\theta(p, q_{\min}^{(p)}) = 1.67$.

q	1	2	3	4	5
Δ_q^p	0.5	1.0	1.5	2.0	2.0
$\theta(p,q)$	2.5	1.75	1.67	1.75	1.8

Table 3.2: Value of θ. Example from the Figure 3.7

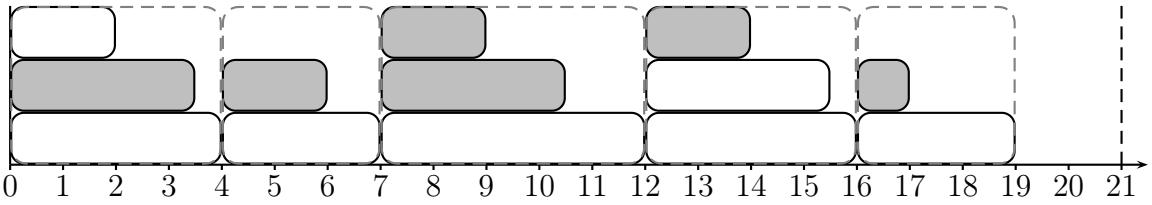

Figure 3.7: A machine with ordered uncertain blocks

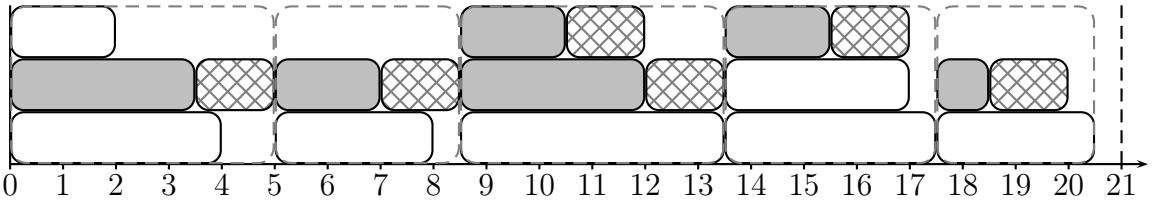

Figure 3.8: Example for ℓ_∞-metric: deviation $\Delta_3^{(p)} = 1.5$

Theorem 5. *The stability radius ρ_∞ for a given feasible solution is calculated as follows*

$$\rho_\infty = \min_{p \in \widetilde{W}} \min_{q=1,\dots,|\widetilde{U}(p)|} \theta(p,q). \tag{3.9}$$

Proof. Let us denote the right-hand side of (3.9) as φ_∞. To prove the present theorem, one needs to show that $\rho_\infty \ge \varphi_\infty$ and $\rho_\infty \le \varphi_\infty$. The inequality $\rho_\infty \ge \varphi_\infty$ is a direct consequence of Lemma 3.

Now, let us show that $\rho_\infty \le \varphi_\infty$. To do this, it is sufficient to check that for any $\delta > \varphi_\infty$ there exists a vector of processing time deviations $\xi^* \in B_\infty(\delta)$, which causes the considered feasible solution to be unfeasible.

As above, based on the definition of φ_∞, we deduce that there exists an uncertain machine p^* and an index q^* such that $q^* \cdot \varphi_\infty = T - \sum_{k \in U(p^*)} \tau_k + \sum_{i=1}^{q^*} \Delta_{\pi_i}^{(p^*)}$. Then,

setting $\xi^* \in B_\infty(\delta)$, where $\xi^*_j = \delta$, for any $j \in \widetilde{V}_k$, $k \in \widetilde{U}(p^*)$, we notice that the *perturbed* load of the machine p^* violates the cycle time constraint, since

$$\sum_{k\in U(p^*)} \tau_k + \sum_{i=1}^{|\widetilde{U}(p^*)|} \left[\delta - \Delta^{(p^*)}_{\pi_i}\right]^+ \geq \sum_{k\in U(p^*)} \tau_k + \sum_{i=1}^{q^*} \left[\delta - \Delta^{(p^*)}_{\pi_i}\right]^+ \geq \sum_{k\in U(p^*)} \tau_k + \left[\sum_{i=1}^{q^*} \left(\delta - \Delta^{(p^*)}_{\pi_i}\right)\right]^+ >$$

$$\sum_{k\in U(p^*)} \tau_k + \left[\sum_{i=1}^{q^*} \left(\varphi_\infty - \Delta^{(p^*)}_{\pi_i}\right)\right]^+ = \sum_{k\in U(p^*)} \tau_k + \left[q^* \cdot \varphi_\infty - \sum_{i=1}^{q^*} \Delta^{(p^*)}_{\pi_i}\right]^+ =$$

$$\sum_{k\in U(p^*)} \tau_k + \left[T - \sum_{k\in U(p^*)} \tau_k\right]^+ = T$$

that proves $\rho_\infty \leq \varphi_\infty$. □

Based on the same arguments developed above for $r_{\max} = 1$ and the ℓ_1-norm case, Theorem 5 implies

Corollary 2 ([66]). *The stability radius ρ_∞ for a given configuration of simple assembly line is calculated as* $\min_{p\in\widetilde{W}} \frac{T - \sum_{k\in U(p)} \tau_k}{|\widetilde{U}(p)|}$.

3.3 MILP formulations

3.3.1 Complexity

The studied problem P is strongly $\mathcal{NP}$-hard. Indeed, it is sufficient to consider an instance of P, where the maximal number of tasks per block is limited by 1, *i.e.*, $r_{\max} = 1$. Then, P reduces to the same problem as the one investigated in [58], which is proven to be $\mathcal{NP}$-hard in the strong sense.

Below, we present two MILP formulations: one for P_1 and another one for P_∞. For both formulations the value of $b_{\max}$ is fixed to $\max\left\{k \;\middle|\; \sum_{i=1}^{k} t_{\pi_i} \leq T\right\}$, where $(\pi_1, \pi_2, \ldots, \pi_n)$ is a permutation of V with respect to the non-decreasing order of their processing times. It is easy to see, that the latter expression can be computed in $O(n \log n)$ time.

3.3.2 MILP formulation for P_1

P_1 is formulated as a mixed integer linear program on the following decision variables: ρ_1 is the stability radius value to maximise; $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the block k; y_k is equal to 1 if the block k is not empty

and 0, otherwise; $\tau_k \geq 0$ determines the working time of the block k; $\Delta_{\min}^{(p)} \geq 0$ represents the minimal value of the save time among all the blocks allocated to machine p; a_p is a non-negative variable that is positive if machine p processes at least one assigned uncertain task; z_k is set to 1 if an uncertain task is allocated to the block k and 0, otherwise. The central idea of the MILP formulation for P_1 consists in maximising ρ_1, the minimum idle time over all the machines that process uncertain tasks (see Theorem 4).

$$\text{Maximise} \quad \rho_1$$

$$\sum_{k \in U} x_{j,k} = 1 \qquad \forall j \in V \tag{3.10}$$

$$\sum_{j \in V} x_{j,k} \leq r_{\max} \qquad \forall k \in U \tag{3.11}$$

$$x_{j,k} \leq y_k \qquad \forall k \in U,\ \forall j \in V \tag{3.12}$$

$$y_k \leq \sum_{j \in V} x_{j,k} \qquad \forall k \in U \tag{3.13}$$

$$y_{k+1} \leq y_k \qquad \forall p \in W,\ \forall k \in U(p) \setminus \{pb_{\max}\} \tag{3.14}$$

$$t_j \cdot x_{j,k} \leq \tau_k \qquad \forall k \in U,\ \forall j \in V \tag{3.15}$$

$$\Delta_{\min}^{(p)} \leq T \cdot (2 - y_k - z_k) + \tau_k - t_j \cdot x_{j,k} \qquad \forall p \in W,\ \forall k \in U(p),\ \forall j \in \widetilde{V} \tag{3.16}$$

$$x_{j,k} \leq z_k \qquad \forall k \in U,\ \forall j \in \widetilde{V} \tag{3.17}$$

$$\sum_{k \in U(p)} \tau_k \leq T \qquad \forall p \in W \tag{3.18}$$

$$\sum_{q=k}^{|U|-1} x_{i,q} \leq \sum_{q=k+1}^{|U|} x_{j,q} \qquad \forall (i,j) \in A,\ \forall k \in U \setminus \{mb_{\max}\} \tag{3.19}$$

$$x_{j,k} \leq a_p \qquad \forall p \in W,\ \forall k \in U(p),\ \forall j \in \widetilde{V} \tag{3.20}$$

$$\rho_1 \leq T \cdot (2 - a_p) - \sum_{k \in U(p)} \tau_k + \Delta_{\min}^{(p)} \qquad \forall p \in W \tag{3.21}$$

$$\rho_1 \geq 0$$

$$\Delta_{\min}^{(p)} \geq 0,\ a_p \geq 0 \qquad \forall p \in W$$

$$\tau_k \geq 0 \qquad \forall k \in U$$

$$x_{j,k} \in \{0,1\} \qquad \forall j \in V,\ \forall k \in U$$

$$z_k \in \{0,1\} \qquad \forall k \in U$$

$$y_k \in \{0,1\} \qquad \forall k \in U$$

Constraints (3.10) ensure that each task is allocated to exactly one block. Inequalities (3.11) show that each block contains at most r_{max} tasks. Any block having at least one assigned task is not empty, otherwise it is empty, as enforced by (3.12) and (3.13). Constraints (3.14) ensures that block $k+1$ has to be empty if block k is empty. The working time of a block cannot be less than the processing time of any task allocated to

it, as provided by (3.15). Constraints (3.16) express the definition of $\Delta_{\min}^{(p)}$. Inequalities (3.17) ensure that the block k is uncertain if there is at least one uncertain task assigned to it, and is certain otherwise. As for constraints (3.18), they state that the load of any machine does not exceed the cycle time. The precedence constraints are expressed by inequalities (3.19). Constraints (3.20) – (3.21) describe the result obtained in Theorem 4. Indeed, it is easy to see that (3.20) – (3.21) implies that $a_p \in \{0, 1\}$. As a consequence, if machine p has no uncertain task, then $a_p = 0$ and (3.18) together with (3.21) yields $\rho_1 \leq T$, which is always valid. Otherwise, when $a_p = 1$, (3.21) is a corollary of (3.2).

3.3.3 MILP formulation for P_∞

P_∞ is formulated as a mixed integer linear program on the following decision variables: ρ_∞ is the stability radius value to maximise; $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the block k; y_k is equal to 1 if the block k is not empty and 0, otherwise; $\xi_{j,k}$ is a deviation time of the task j on the block k; $\tau_k \geq 0$ determines the time of the block k including deviations. The main idea of the MILP formulation for P_∞ is that the processing time of all uncertain tasks can be increased by ρ_∞ without compromising feasibility in any optimal solution.

$$\text{Maximise} \quad \rho_\infty$$

$$\sum_{k \in U} x_{j,k} = 1 \qquad \forall j \in V \tag{3.22}$$

$$\sum_{j \in V} x_{j,k} \leq r_{\max} \qquad \forall k \in U \tag{3.23}$$

$$x_{j,k} \leq y_k \qquad \forall k \in U,\ \forall j \in V \tag{3.24}$$

$$y_k \leq \sum_{j \in V} x_{j,k} \qquad \forall k \in U \tag{3.25}$$

$$y_{k+1} \leq y_k \qquad \forall p \in W,\ \forall k \in U(p) \setminus \{pb_{\max}\} \tag{3.26}$$

$$\xi_{j,k} \leq T \cdot x_{j,k} \qquad \forall k \in U,\ \forall j \in \widetilde{V} \tag{3.27}$$

$$t_j \cdot x_{j,k} + \xi_{j,k} \leq \tau_k \qquad \forall k \in U,\ \forall j \in \widetilde{V} \tag{3.28}$$

$$t_j \cdot x_{j,k} \leq \tau_k \qquad \forall k \in U,\ \forall j \in V \setminus \widetilde{V} \tag{3.29}$$

$$\rho_\infty = \sum_{k \in U} \xi_{j,k} \qquad \forall j \in \widetilde{V} \tag{3.30}$$

$$\sum_{k \in U(p)} \tau_k \leq T \qquad \forall p \in W \tag{3.31}$$

$$\sum_{q=k}^{|U|-1} x_{i,q} \leq \sum_{q=k+1}^{|U|} x_{j,q} \qquad \forall (i,j) \in A,\ \forall k \in U \setminus \{mb_{\max}\} \tag{3.32}$$

$$\rho_\infty \geq 0$$

$$\xi_{j,k} \geq 0 \qquad \forall j \in V,\ \forall k \in U$$

$$x_{j,k} \in \{0,1\} \qquad \forall j \in V,\ \forall k \in U$$

$$y_k \in \{0,1\} \qquad \forall k \in U$$

Constrains (3.22) – (3.26) and (3.31) – (3.32) are respectively equivalent to constraints (3.10), – (3.14), (3.18) and (3.19) from the MILP formulation for P_1. The value of the variables $\xi_{j,k} \geq 0$ is determined in (3.27) – (3.29). Inequality (3.27) says that any deviation of an uncertain task assigned to the block k cannot be greater than the cycle time. These deviations may lead to increase the working time of the block as shown in (3.28). Constraints (3.29) do not involve any $\xi_{j,k}$ variables since they model the working time of blocks based on certain tasks only. Constraints (3.30) enforce all uncertain tasks have the same deviation time, which defines ρ_∞, the problem objective.

3.4 Pre-processing

3.4.1 Assignment intervals and empty blocks

First, an algorithm seeking a lower bound on the number of blocks necessary for assigning a given set of tasks is proposed. It not only uses the ratio between the cardinality of this set and $r_{\max}$, but it also takes the cycle time into account. More precisely, it first forms the blocks of the input set of tasks so as the sum of the working time of the obtained blocks is minimised. Second, it finds a lower bound on the number of machines necessary to allocate all these blocks. Thus, the seeking number of blocks is equal to the index of the first block of the last needful machine. This algorithm is formally presented below, where D is considered as a given set of tasks, which could be V itself or some of its subsets.

Algorithm $\alpha(D)$.

1. Let $(\sigma_1, \sigma_2, \ldots, \sigma_{|D|})$ be a permutation of D with respect to the non-increasing order of their processing times.
2. Form $\omega = \left\lceil \frac{|D|}{r_{\max}} \right\rceil$ blocks $B_1, B_2, \ldots, B_\omega$, where each of which has at exactly $r_{\max}$ tasks except possibly the last one, such that the task σ_j is in the block $B_{\lceil \frac{j}{r_{\max}} \rceil}$, $j \in D$. This permits, inter alia, to determine the working time of each obtained block: $\tau_k = \max_{i \in B_k} t_i$, $k = 1, 2, \ldots, \omega$.
3. The lower bound on the number of machines able to allocate all the obtained blocks is equal to $\lambda = \frac{1}{T} \cdot \sum_{k=1}^{\omega} \tau_k$. From this, the returned lower bound on the number of blocks equals $b_{\max} \cdot \lceil \lambda - 1 \rceil + 1$.

Because of Step 1, Algorithm $\alpha(D)$ can be implemented to run in $O(|D| \log |D|)$ time.

It is recalled that when building a feasible assignment of all given tasks to blocks and blocks to machines, precedence constraints have to be satisfied. Thus, for example, for each task $j \in V$, the set of all its predecessors, denoted as $P(j)$, has to be assigned before j. In the same manner, the set of all successors of j, denoted by $S(j)$, has to be assigned after j. Thus, for any task j, we introduce the interval $Q_j = [l_j, u_j]$, where l_j (resp. u_j) is the lowest (resp. uppermost) index of the block able to accommodate the task j. For the efficient solving of our problem, these intervals should be as tight as possible. In order to reach this goal, we provide below five different rules for reducing Q_j.

Reduction rules.

- Rule 1 is based on the knowledge of the quantity of all predecessors and all successors for a given task in the precedence graph: $l_j^{(1)} = \left\lceil \frac{|P(j)|}{r_{\max}} \right\rceil + 1$ and $u_j^{(1)} = mb_{\max} + 1 - \left(\left\lceil \frac{|S(j)|}{r_{\max}} \right\rceil + 1 \right)$ for any $j \in V$.
- Rule 2 consists in applying Algorithm $\alpha(\cdot)$ on the sets of predecessors and successors of each task: $l_j^{(2)} = \alpha(P(j)) + 1$ and $u_j^{(2)} = mb_{\max} + 1 - (\alpha(S(j)) + 1)$ for any $j \in V$. Here, $\alpha(\cdot)$ is the result of application of Algorithm $\alpha(\cdot)$ on a given set of tasks.
- Rule 3 is similar to Rule 2, but the input set of tasks for Algorithm $\alpha(\cdot)$ includes also the task j itself: $l_j^{(3)} = \alpha(P(j) \cup \{j\})$ and $u_j^{(3)} = mb_{\max} + 1 - \alpha(S(j) \cup \{j\})$.
- Rule 4 analyses the assignment intervals of all direct predecessors and successors for all tasks as follows: $l_j^{(4)} = \max\limits_{(i,j) \in A} l_i + 1$ and $u_j^{(4)} = \min\limits_{(j,i) \in A} u_i - 1$ for any $j \in V$.
- Rule 5 takes also (with respect to Rule 4) into account the cardinality of the sets of direct predecessors $P^*(j)$ and direct successors $S^*(j)$ of j: $l_j^{(5)} = \min\limits_{(i,j) \in A} l_i + \left\lceil \frac{|P^*(j)|}{r_{\max}} \right\rceil$ and $u_j^{(5)} = \max\limits_{(j,i) \in A} u_i - \left\lceil \frac{|S^*(j)|}{r_{\max}} \right\rceil$.

The reduced assignment intervals serve at detecting a set of unused (or a priori empty) blocks, and then to take advantage of this piece of information in order to reduce even further these intervals. Algorithm $\beta(p)$ presents a procedure of seeking unused blocks for a given machine $p \in W$.

Algorithm $\beta(p)$.

1. Compute the set $V(p) = \{j \in V \mid Q_j \cap U(p) \neq \emptyset\}$ of assignable tasks to machine p.
2. Based on $V(p)$, determine the maximal number of blocks $b_{\max}^{(p)}$ for the machine p by determining $\max \left\{ k \mid \sum_{i=1}^{k} t_{\pi_i} \leq T \right\}$, where $(\pi_1, \pi_2, \ldots, \pi_{|V(p)|})$ is a permutation of $V(p)$ with respect to the non-decreasing order of their processing times. Thus, the set $\widehat{U}(p) = \{(p-1)b_{\max} + 1 + b_{\max}^{(p)}, \ldots, pb_{\max}\}$ of unused (or a priori empty) blocks is introduced for the machine p.
3. For each task $j \in V(p)$, update its interval Q_j as follows. If $l_j \in \widehat{U}(p)$, then set $l_j = pb_{\max} + 1$. If $u_j \in \widehat{U}(p)$, then set $u_j = (p-1)b_{\max} + b_{\max}^{(p)}$.

It is not difficult to see that the running time of Algorithm $\beta(\cdot)$ is not greater than $O(n \log n)$, because of Step 2.

The whole process of finding unused blocks and reducing assignment intervals can be summarised in the following approach, where all the tasks assignment intervals are initialised to $[1, mb_{\max}]$.

Approach REDUCTION.

1. For each task $j \in V$, compute $Q_j^{(1)}$, $Q_j^{(2)}$ and $Q_j^{(3)}$ by applying respectively Rules 1, 2 and 3. Update the intervals Q_j as follows : $l_j = \max\{l_j, l_j^{(1)}, l_j^{(2)}, l_j^{(3)}\}$ and $u_j = \min\{u_j, u_j^{(1)}, u_j^{(2)}, u_j^{(3)}\}$.
2. For each task $j \in V$, compute $Q_j^{(4)}$ and $Q_j^{(5)}$ by applying respectively Rules 4 and 5. Update the intervals Q_j as follows : $l_j = \max\{l_j, l_j^{(4)}, l_j^{(5)}\}$ and $u_j = \min\{u_j, u_j^{(4)}, u_j^{(5)}\}$.
3. For each machine $p \in W$, apply Algorithm $\beta(p)$ and update the intervals Q_j with respect to the constructed sets $\widehat{U}(p)$ of unused blocks.
4. If there exists a task $j \in V$ such that Q_j was reduced on Steps 2 or 3, then go to Step 2. Otherwise stop.

Example of calculation

Given $n = 11$ production tasks which have to be assigned to $m = 4$ machines with respect to the cycle time $T = 6$. Any block is able to execute up to $r_{\max} = 2$ tasks in parallel mode. Nominal processing times are presented by the vector $t = (3, 4, 3, 5, 4, 2, 2, 4, 3, 1, 3)$. Figure 3.9 portrays the precedence order.

It is not difficult to see that the maximal number of blocks per machine $b_{\max}$ equals to 3, since $1+2+2 < T < 1+2+2+3$. Thus, the set $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ includes all available blocks. The result of computing assignment intervals Q_j for each task $j \in V$ is shown in Table 3.3. Below, we describe in details the approach REDUCTION for the next three tasks: 5, 6 and 7.

Initially, all assignment intervals are installed as $[1, 12]$. During the first step, we calculate new borders using Rules 1, 2 and 3.

Task 5. Rule 1 requires to use the quantity of all predecessors and all successors and to estimate the number of blocks that they can occupy. It is easy to see that $P(5) = \{1, 2\}$

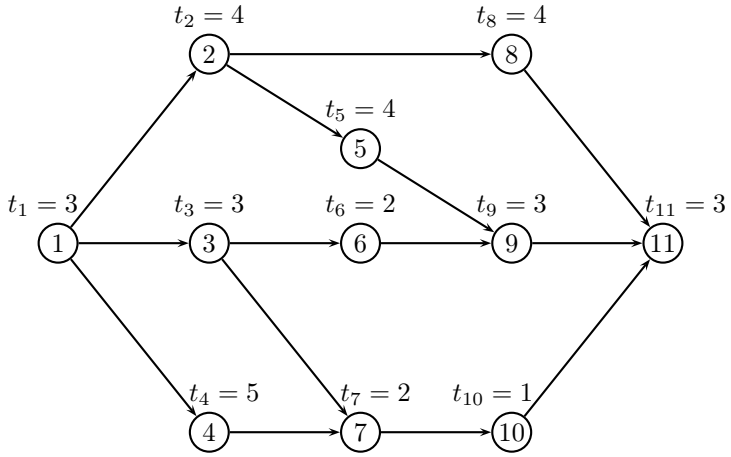

Figure 3.9: Example. Precedence constraints

and $S(5) = \{9, 11\}$. Then, $\left\lceil \frac{|P(5)|}{r_{\max}} \right\rceil = \left\lceil \frac{|S(5)|}{r_{\max}} \right\rceil = 1$, and we obtain first borders $l_5^{(1)} = 2$ and $u_5^{(1)} = 11$. Rule 2 applies Algorithm $\alpha(\cdot)$ for $\{1, 2\}$ and $\{9, 11\}$, separately. From Steps 1 and 2 of this algorithm, the set of tasks $\{1, 2\}$ forms 1 block with $\tau = 4$. From Step 3, we got that the minimal number of blocks needed to allocate all predecessors of the task 5 is equal to 1, *i.e.*, $l_5^{(2)} = 2$. Similarly, the set of all successors $\{9, 11\}$ gives one block with its working time as 4. The result of algorithm $\alpha(\cdot)$ is 1, and $u_5^{(2)} = 11$. Rule 3 enriches the input sets by the task 5 itself. Thus, we apply algorithm $\alpha(\cdot)$ to the set $\{1, 2, 5\}$ and receive 2 blocks such that $B_1 = \{2, 5\}$ and $B_2 = \{1\}$ with $\tau_1 = 4$ and $\tau_2 = 3$, since $t_2 = 4 \geq t_5 = 4 \geq t_1 = 3$. Taking into account that $\frac{\tau_1+\tau_2}{T} = \frac{7}{6}$ is not integer and following Step 3, the minimal number of blocks needed to allocate the tasks $\{1, 2, 5\}$ is equal to $b_{\max} \cdot \left\lfloor \frac{\tau_1+\tau_2}{T} \right\rfloor + 1 = 4$. Same time the set of tasks $\{5, 9, 11\}$ forms 2 blocks such that $B_1 = \{5, 9\}$ and $B_2 = \{11\}$ with $\tau_1 = 4$ and $\tau_2 = 3$, since $t_5 = 4 \geq t_9 = 3 \geq t_{11} = 3$. This means that $\alpha(\{5, 9, 11\})$ is also 4. Consequently, $l_5 = \max\{2, 2, 4\} = 4$ and $u_5 = \min\{11, 11, 9\} = 9$.

Task 6. As above, $P(6) = \{1, 3\}$ and $S(6) = \{9, 11\}$. Then, $\left\lceil \frac{|P(6)|}{r_{\max}} \right\rceil = \left\lceil \frac{|S(6)|}{r_{\max}} \right\rceil = 1$, and Rule 1 brings $l_6^{(1)} = 2$, $u_6^{(1)} = 11$. By applying Algorithm $\alpha(\cdot)$ for $\{1, 3\}$ and $\{9, 11\}$, separately, we again have the result as 1 block for all predecessors (successors, resp.): $l_6^{(2)} = 2$ and $u_6^{(2)} = 11$, according to Rule 2. For the third element we calculate $\alpha(\{1, 3, 6\})$. Thus, we form 2 blocks $B_1 = \{1, 3\}$ and $B_2 = \{6\}$ with $\tau_1 = 3$ and $\tau_2 = 2$. As $\frac{\tau_1+\tau_2}{T} = \frac{5}{6}$ is not integer, the minimal number of blocks needed to allocate the tasks $\{1, 3, 6\}$ is equal to $b_{\max} \cdot \left\lfloor \frac{\tau_1+\tau_2}{T} \right\rfloor + 1 = 1$. In the same way, the set of tasks $\{6, 9, 11\}$ forms the blocks $B_1 = \{9, 11\}$ and $B_2 = \{6\}$, where $\tau_1 = 3$, $\tau_2 = 2$. The value $\frac{\tau_1+\tau_2}{T} = \frac{5}{6}$ again is not integer. This means that $\alpha(\{6, 9, 11\})$ is $b_{\max} \cdot \left\lfloor \frac{\tau_1+\tau_2}{T} \right\rfloor + 1 = 1$. Rule 3 gives us $l_6^{(3)} = 1$ and $u_6^{(3)} = 12$ Hence, the final borders for the task 6: $l_6 = \max\{2, 2, 1\} = 2$ and

$u_6 = \min\{11, 11, 12\} = 11$.

Task 7. For this task, $P(7) = \{1, 3, 4\}$, $S(7) = \{10, 11\}$, and, as a consequence, $l_7^{(1)} = \left\lceil \frac{|P(7)|}{r_{\max}} \right\rceil + 1 = 3$, $u_7^{(1)} = 13 - \left\lceil \frac{|S(7)|}{r_{\max}} \right\rceil - 1 = 11$. Next, $\{1, 3, 4\}$ forms 2 blocks $B_1 = \{1, 4\}$ and $B_2 = \{3\}$ with $\tau_1 = 5$ and $\tau_2 = 3$. Because $\frac{\tau_1+\tau_2}{T} = \frac{8}{6}$ is not integer, then $l_7^{(2)} = \alpha(\{1, 3, 4\}) + 1 = 5$. Successors of the task 7 require only one block, because they are only two, i.e., $u_7^{(2)} = 11$. It is not difficult to see that $l_7^{(3)} = \alpha(\{1, 3, 4, 7\})$ equals to 4. The set of tasks $\{7, 10, 11\}$ generates 2 blocks $B_1 = \{7, 11\}$ and $B_2 = \{10\}$, where $\tau_1 = 3$ and $\tau_2 = 1$. The value $\frac{\tau_1+\tau_2}{T} = \frac{4}{6}$ is not integer, that implies $u_7^{(3)} = 13 - \alpha(\{7, 10, 11\}) = 12$. Finally, the task 7 has borders $l_7 = \max\{3, 5, 4\} = 5$ and $u_7 = \min\{11, 11, 12\} = 11$.

Table 3.3 shows intervals for all tasks obtained during the first step of the approach REDUCTION. These values are used for dynamic improvements on the next steps.

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

Table 3.3: Intervals Q_j for Example 2

Task 5. Rules 4 and 5 deal with direct predecessors $P^*(5) = \{2\}$ and successors $S^*(5) = \{9\}$ of the task. The cardinality for both sets is 1, the borders of the intervals are identical and can be easily calculated as follows: $l_5^{(4,5)} = l_2 + 1 = 3$ and $u_5^{(4,5)} = u_9 - 1 = 10$. All five rules set up the assignment interval $l_5 = \max\{l_5, l_5^{(4)}, l_5^{(5)}\} = 4$ and $u_5 = \min\{u_5, u_5^{(4)}, u_5^{(5)}\} = 9$.

Task 6. This task also has only one predecessor and one successor, and the calculation on Step 2 is identical to the task 5. However, the current value l_6 matches l_3 (the left border of its predecessor) as well as u_6 matches u_9 (the right border of its successor). Because of this, dynamic rules are going to improve the assignment interval of the task 6: $l_6 = l_3 + 1 = 3$, $u_6 = u_9 - 1 = 10$.

Task 7. Two direct predecessors form different manners of calculation, but the same results for Rules 4 and 5: $l_7^{(4)} = \max\{l_3, l_4\} + 1 = 3$ and $l_7^{(5)} = \min\{l_3, l_4\} + \left\lceil \frac{|P^*(7)|}{r_{\max}} \right\rceil = 2 + 1 = 3$. Again, there is only one successor which eventually improves the right border $u_7^{(4,5)} = u_{10} - 1 = 10$. The final assignment interval has limits $l_7 = \max\{l_7, l_7^{(4)}, l_7^{(5)}\} = 5$ and $u_7 = \min\{u_7, u_7^{(4)}, u_7^{(5)}\} = 10$.

Intervals for all production tasks after on this stage are given in Table 3.4. Algorithm $\beta(p)$ is then applied for each machine $p \in W$ to construct the sets $\widehat{U}(p)$ of unused blocks.

Machine 1. It has three blocks with indexes 1, 2 and 3. Using Table 3.4, we can determine the tasks that can be allocated to any of these blocks. There are tasks 1, 2, 3, 4 and 6. Their processing times: 3, 4, 3, 5 and 2, respectively, or 2, 3, 3, 4 and 5, once sorted by non-decreasing order. Then, $b_{\max}^{(1)} = 2$ since $2 + 3 < T < 2 + 3 + 3$, *i.e.*, in any

j	1	2	3	4	5	6	7	8	9	10	11
l_j	1	2	2	2	4	**3**	5	4	5	**6**	8
u_j	5	8	8	9	9	**10**	**10**	11	11	11	12

Table 3.4: Intervals Q_j for Example 2 with dynamic improvements

feasible line balance machine 1 has no more than 2 open blocks. The set of unused blocks is $\widehat{U}(1) = \{3\}$.

Machine 2. As above, the blocks of this machines have indexes 4, 5, and 6. They are present in all assignment intervals, except for the task 11. Sorted processing times $(1, 2, 2, 3, 3, 3, 4, 4, 4, 5)$ lead to $b^{(2)}_{\max} = 3$ that is equal to $b_{\max}$, *i.e.*, there is no unused block on the machine 2, so $\widehat{U}(2) = \emptyset$.

Machine 3. As for machine 2, almost all tasks can be assigned to its blocks. Thus, $\widehat{U}(3) = \emptyset$.

Machine 4. The set of assignable tasks includes the following tasks: $\{6, 7, 8, 9, 10, 11\}$. Unfortunately, they do not exclude any block from the machine.

The only unused block 3 affects the assignment interval of the tasks 6, which is now $[4, 10]$. The approach REDUCTION requires to get back to Step 2 and to repeat calculations, but it could be noticed that there is no possible changes for any task. Consequently, sets of unused blocks remain the same, and the approach is stopping.

3.4.2 Heuristics

Based on the ideas presented in [32], in this sub-section, we describe a general heuristic methodology applicable for both problems P_1 and P_∞. Then, its multi-start version is provided as well. Assignment intervals found in previous sub-section are used by this heuristic to provide a better solution, which can be introduced for respective MILP models as an initial one.

The heuristic aims to construct a feasible solution having the stability radius greater than a given value ρ. Its main principle is to assign as many tasks as possible to the current block of the current machine. At the beginning, a partially constructed solution contains only one machine having one empty block. The heuristic assigns tasks to this block until no task can be added because of the existing constraints. Then, a new empty block (or a new machine with one empty block) is opened and becomes current. This continues until all tasks are assigned and a feasible solution is obtained or there is no possibility to construct a feasible solution.

The heuristic uses the so-called candidate list $\mathcal{CL}(k, p)$, which contains all tasks assignable to the current block k of the current machine p. This list is built in the

following way: the set of unassigned tasks is analysed and task j is added to $\mathcal{CL}(k,p)$ if all following conditions are satisfied:

1. j has no predecessors or all predecessors of j are already assigned,
2. assigning of j does not violate the cycle time constraint,
3. assigning of j does not exceed the maximal number of tasks per block $r_{\max}$,
4. assigning of j assures that the partially constructed solution has the stability radius greater than ρ,

The formal description of the heuristic as well as the usage of the candidate list is presented below.

Heuristic $\mathcal{H}^1(\rho)$.

1. Mark all tasks as unassigned and open the first machine having one empty block, *i.e.*, set $k = 1$ and $p = 1$.
2. If there exists at least one unassigned task, then construct a candidate list $\mathcal{CL}(k,p)$ for the block k of the machine p. Otherwise, go to Step 6.
3. If $\mathcal{CL}(k,p) = \emptyset$ and $k < b_{\max}$, then open a new empty block on the current machine, *i.e.*, set $k = k + 1$, and go to Step 2. If $\mathcal{CL}(k,p) = \emptyset$, $k = b_{\max}$ and $p < m$, then open a new machine having one empty block, *i.e.*, set $p = p + 1$ and $k = 1$, and go to Step 2. If $\mathcal{CL}(k,p) = \emptyset$, $k = b_{\max}$ and $p = m$, then go to Step 5.
4. If $\mathcal{CL}(k,p)$ is not empty, then choose randomly one task from $\mathcal{CL}(k,p)$, assigned it to the current block k and go to Step 2.
5. Stop, a feasible solution is not found.
6. Stop, a feasible solution is found.

In order to diversify the search strategy, a second heuristic method, denoted by $\mathcal{H}^2(\rho)$, is also devised. Its unique difference from the first one, denoted by $\mathcal{H}^1(\rho)$, consists in analysing the precedence graph (and, as a consequence, composing the list $\mathcal{CL}(k,p)$) in the opposite manner, *i.e.*, $\mathcal{H}^2(\rho)$ constructs feasible solutions starting from the last machine and not from the first one.

According to the definition, assignment intervals determine all blocks able to accommodate respective tasks. We can conclude that this information can be used under

construction of the candidate lists for the heuristic. Thus, a new optional condition have to be introduced:

5. global index of the block k, *i.e.* $(p-1)b_{\max}+k$, belongs to the current interval Q_j.

That leads us to two search strategies, heuristic methods, noted as $\mathcal{H}^3(\rho)$ and $\mathcal{H}^4(\rho)$, respectively. They are similar to the previous heuristics except for the additional requirement used when building the candidate list.

A multi-start heuristic $\mathcal{M}(\rho)$ can naturally be built upon $\mathcal{H}(\rho)$ (and extended to $\mathcal{H}^2(\rho)$, $\mathcal{H}^3(\rho)$ and $\mathcal{H}^4(\rho)$). Each attempt of $\mathcal{H}(\rho)$ consists in trying to construct a new feasible solution, whose stability radius is greater than the best one found so far. The number of attempts without improvement is limited by `mslimit`. The formal description of such version is presented below. The entire procedure includes four consecutive launch of $\mathcal{M}(\rho)$ by one for each strategy.

Multi-start heuristic $\mathcal{M}(\rho)$.

1. Set $k = 0$.
2. Apply Heuristic $\mathcal{H}(\rho)$ for a chosen strategy. If a new feasible solution with the stability radius $\rho^* > \rho$ is found, then memorise this solution and set $\rho = \rho^*$, $k = 0$ and repeat Step 2. Otherwise, set $k = k + 1$.
3. If $k <$ `mslimit`, then go to Step 2. Otherwise, stop and return the best found feasible solution.

3.4.3 Global pre-processing approach

Based on the heuristic strategies and reduction rules, this sub-section provides a global approach of enhancements for both MILP formulations, P_1 and P_∞. One of the main ideas of this approach can be found in the following statements.

Statement 2. *Let (s^0, ρ^0) be the pair made of a feasible solution and its stability radius of for the problem instance $\mathcal{I}$ of P_1. Then, s^0 is also a feasible solution for the problem instance $\mathcal{I}'$, which is obtained from $\mathcal{I}$ by increasing the processing time of exactly one uncertain tasks by ρ^0. Moreover, all assignment intervals and unused blocks of $\mathcal{I}'$ are also valid for $\mathcal{I}$.*

Statement 3. *Let (s^0, ρ^0) be the pair made of a feasible solution and its stability radius of for the problem instance $\mathcal{I}$ of P_∞. Then, s^0 is also a feasible solution for the problem instance $\mathcal{I}'$, which is obtained from $\mathcal{I}$ by increasing the processing time of all uncertain*

tasks by ρ^0. Moreover, all assignment intervals and unused blocks of $\mathcal{I}'$ are also valid for $\mathcal{I}$.

In other words, Statements 2 and 3 indicate that analysing $\mathcal{I}'$ may provide tighter assignment intervals for $\mathcal{I}$. The analysis requires taking into account $|\widetilde{V}|$ instances for P_1, and a single one for P_∞. This difference between the two problems is shown below in the following global pre-processing approach.

Pre-processing for P_1.

1. Let (s, ρ) (resp. (s^*, ρ^*)) be respectively the best feasible solution and its stability radius value after applying the multi-start heuristic $\mathcal{M}(-1)$ (resp. $\mathcal{M}(\rho)$) for any strategy.
2. For each uncertain task $j \in \widetilde{V}$, produce the following instructions:
   - 6.1. Set $t_j = t_j + \rho^*$.
   - 6.2. Apply Approach REDUCTION to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$.
   - 6.3. Set $t_j = t_j - \rho^*$.
3. Consider s^* as a starting feasible solution for the corresponding MILP formulation and add the following optimality-based cuts:

$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j, \tag{3.33}$$

$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p). \tag{3.34}$$

Pre-processing for P_∞.

1. Let (s, ρ) (resp. (s^*, ρ^*)) be respectively the best feasible solution and its stability radius value after applying the multi-start heuristic $\mathcal{M}(-1)$ (resp. $\mathcal{M}(\rho)$) for any strategy.
2. For each uncertain task $j \in \widetilde{V}$, set $t_j = t_j + \rho^*$.
3. Apply Approach REDUCTION to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$. If it concerns P_1, then go to Step 6.3.
4. For each uncertain task $j \in \widetilde{V}$, set $t_j = t_j - \rho^*$.

5. Consider s^* as a starting feasible solution for the corresponding MILP formulation and add the following optimality-based cuts:

$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p).$$

Here, constraints (3.33) model the assignment interval for any task $j \in V$ and constraints (3.34) describe all unused blocks. These constraints do not remove any optimal solution, but exclude some feasible ones.

3.5 Computational results

In order to assess the computational effort induced by our models and algorithms, we have used several series of randomly generated instances which can be found on the Benchmarks Library[3] as a part of the program code written for the GAMS environment[4]. The number of tasks per instance varies as follows: two series with 25 tasks (referred to as S1 and S2), one series with 50 ones (S3). Each series is composed of 50 unique instances. Tasks processing times and precedence constraints are those specified in these files. The Order of Strength (OS) is defined as the ratio of the number of edges in the transitive closure of the precedence graph, to the number of edges in the complete graph. This value is different in all series and presented in Table 3.5 for information. The cycle time T for all instances is set to 70 instead of 100 proposed originally. This value has been set after preliminary tests for ensuring that all the instances are feasible. The number of machines m is set to 5 for S1 and S2, and 10 for S3. A set of uncertain tasks for any instance is determined by an individual random permutation added to the data. We propose two additional parameters to distinguish the tests: the block capacity $r_{\max} \in \{2, 3\}$ and the ratio of uncertain tasks %UT=$\{50\%, 75\%, 100\%\}$. Thus, the set of uncertain tasks is defined as the first $|\widetilde{V}|$ tasks of the random permutation.

Series	n	m	OS
S1	25	5	0.5
S2	25	5	0.15
S3	50	10	0.9

Table 3.5: Main characteristics of Series S1, S2 and S3

[3]http://www.math.nsc.ru/AP/benchmarks/Transfer/tests.html

[4]The General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming and optimization. It consists of a language compiler and a stable of integrated high-performance solvers

The computational tests were performed on an Apple MacBook Pro computer equipped of an Intel Core i5 processor at 2.7 GHz with 2 cores and 8 GByte RAM. Gurobi 7.5 was used as a solver for addressing the mixed integer linear programming models proposed above. First, we have launched an experimentation with basic models only as they are presented in Section 3.3, with a maximum allotted time of 10 minutes per instance. Table 3.6 shows the number of optimal solutions, average gap and CPU time in ℓ_1 and ℓ_∞ metrics for series with $r_{\max} = 2$. It can be seen that the problem is difficult to solve. The best result is obtained for ℓ_∞-metric with 50 percents of uncertain tasks in series S1 – 5 optimal solutions are found, out of 50 ones. In series S2 only feasible solutions were obtained, and the gap provided by the solver is quite large. With S3, no solution at all was found, which is indicated by "**n/d**" in the gap column. Table 3.7 presents results with $r_{\max} = 3$. The situation remains unchanged with Series S3, as no feasible solution is found in the allotted time. With S2, the results are slightly better than with $r_{\max} = 2$, as several instances are solved to optimality. The improvement is more visible with Series S1, as more optimal solutions are found, however only 66% of the instances in S1 are solved optimality. All these observations lead us to conclude that basic MILP formulations are not competitive even for small and medium size instances.

		P_1			P_∞		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	0	2.392	600	5	1.324	581.72
	75	1	2.236	595.69	3	2.473	588.50
	100	1	1.073	597.75	1	4.042	599.68
S2	50	0	2.321	600	0	1.450	600
	75	0	4.391	600	0	2.718	600
	100	0	1.706	600	0	4.291	600
S3	50	0	**n/d**	600	0	**n/d**	600
	75	0	**n/d**	600	0	**n/d**	600
	100	0	**n/d**	600	0	**n/d**	600

Table 3.6: Computational results for basic models with $r_{\max} = 2$

		P_1			P_∞		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	20	0.962	522.63	42	0.128	344.71
	75	30	0.386	478.83	27	0.368	471.16
	100	42	0.104	392.42	37	0.521	436.67
S2	50	2	0.675	592.33	7	0.241	558.59
	75	3	1.564	586.70	2	1.107	586.34
	100	10	0.297	545.89	10	0.906	542.62
S3	50	0	**n/d**	600	0	**n/d**	600
	75	0	**n/d**	600	0	**n/d**	600
	100	0	**n/d**	600	0	**n/d**	600

Table 3.7: Computational results for basic models with $r_{\max} = 3$

Now the enhanced models enriched by new constraints based on assignment intervals of tasks introduced in Section 3.4 are also tested. The efficiency of optimality-based cuts depends on the heuristic's results. Indeed, a bigger stability radius gives tighter intervals. This is the reason why the heuristic should be given enough time to return reasonably good solutions. The maximum number of iterations without improvements of the objective value is chosen as the stopping criterion. It is expressed as a linear function of the number of tasks, n, *i.e.*, $f(n) = \alpha n$, where α is a constant coefficient. After time-based experiments with large-size instances, α has been set to 100. This value is the same for all the instances presented below. Thus, in series S1 and S2 the maximum number of iterations without improvements is 2500, and 5000 in series S3. Tests for enhanced models are presented in Tables 3.8 and 3.10. They include results for all ratios of uncertain tasks (%UT).

The first observation that can be made is that the order of strength (OS) has a major impact on the computational efficiency of our algorithms. For example, in series S1, which has an OS value of 0.5, meaning that the transitive closure of the precedence graph includes half of the edges from the complete graph, 38 instances are solved to optimality in ℓ_1-metric (Table 3.8), whereas a single instance is solved to optimality in series S2 for which OS = 0.15. Moreover, our approach achieves good results for series S3 (for which OS = 0.9), despite the fact that instance size is twice larger.

		P_1			P_∞		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	16	0.186	452.14	33	0.198	253.44
	75	10	0.396	509.20	23	0.337	379.45
	100	12	0.321	480.66	38	0.137	210.42
S2	50	0	0.478	600	1	1.255	599.32
	75	0	0.961	600	1	1.993	590.32
	100	1	0.787	595.69	3	2.048	570.61
S3	50	35	0.197	321.66	45	0.071	108.36
	75	35	0.180	341.41	49	0.031	42.29
	100	36	0.171	302.53	50	0.0	13.03

Table 3.8: Results with the heuristics and the cuts with $r_{\max} = 2$.

The results are similar with the ℓ_∞-metric, even if more optimal solutions are found in that case. This is visible in particular for Series S3 where all the instances for which all the tasks are uncertain are solved to optimality with an average computational time of 13 seconds.

The detailed results for each individual instance with $r_{\max} = 2$ can be found in Appendix C: Tables C.1–C.3. Any table includes information for both considered metrics with the instance name and percentages of uncertain tasks preceding the results, see Table 3.9 for an illustrative example. The first two columns correspond to the best value of the stability radius obtained by the heuristic, with its corresponding CPU time in seconds. They are called LB and CPU under the line "Heuristic", respectively. They are followed

		P_1					
		Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU
S1.0	50	28.8	0.17	30.4	30.4	0.0	52.44
	75	28.8	0.13	30.4	30.4	0.0	58.82
	100	28.8	0.12	30.4	30.4	0.0	83.67
S1.1	50	24.6	0.16	26.7	39.2	0.468	600.00
	75	23.8	0.17	23.8	38.8	0.63	600.00
	100	23.8	0.13	23.8	38.8	0.63	600.00
S1.2	50	27.7	0.12	29.1	29.1	0.0	486.42
	75	26.4	0.13	28.1	31.2	0.11	600.00
	100	27.7	0.11	28.0	36.7	0.311	600.00
S1.3	50	33.1	0.18	33.1	51.4	0.553	600.00
	75	29.3	0.16	29.5	51.4	0.742	600.00
	100	29.3	0.18	29.4	51.4	0.748	600.00
S1.4	50	25.0	0.14	28.5	46.771	0.641	600.00
	75	25.5	0.16	26.1	33.1	0.268	600.00
	100	24.3	0.14	26.3	33.1	0.259	600.00
S1.5	50	31.9	0.19	32.3	40.2	0.245	600.00
	75	29.0	0.15	29.0	47.673	0.644	600.00
	100	29.0	0.14	29.4	41.3	0.405	600.00

Table 3.9: Example of detailed results

by four columns related to the results of GUROBI: LB and UB – the best lower and upper bounds obtained after 10 minutes (or less if the optimal solution is found); GAP – relative difference between lower and upper bounds; CPU – computational time for GUROBI (this value is set to 600.00 if no provable optimal solution was found).

Another advantage of the approach can be observed in the results for enhanced models with the parameter $r_{\max} = 3$. The same series of instances are considered. Table 3.10 presents the number of optimal solutions, the average gap and computational time. We can see that increasing block capacity without changes for another parameters leads to better results. Now any single instance from S1 is solved optimally in both metrics, and the average CPU time does not exceed two minutes. Improvements are achieved with Series S2, as more optimal solutions are found. Betterment is less significant with Series S3. It can be explained with the fact that, because of the order of strength, the stability radius cannot increase, while the solution space of the integer linear program is much larger. The output for all instances is presented in Tables C.4–C.6.

The detailed results shown in Appendix C confirm that the objective value decreases when the number of uncertain tasks increases. This is visible, for example, in Table C.1, with instance "S1.9". This instance is solved optimally for all considered ratios of uncertain tasks in both metrics. While %UT is growing up from 50 to 75 and 100, LB is dropping off from 12.9 – 9.467 – 8.133 in ℓ_∞-metric. Such a result was expected, but in some cases

		P_1			P_∞		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	50	0.0	74.58	50	0.0	12.86
	75	50	0.0	118.66	50	0.0	11.13
	100	50	0.0	76.64	50	0.0	0.31
S2	50	28	0.069	355.92	27	0.113	352.50
	75	15	0.165	473.73	15	0.742	452.46
	100	21	0.120	425.40	45	0.007	89.91
S3	50	37	0.195	299.92	47	0.033	112.34
	75	33	0.389	337.79	50	0.0	34.44
	100	38	0.192	290.31	50	0.0	7.80

Table 3.10: Results with the heuristics and the cuts with $r_{\max} = 3$.

it might be helpful to improve solution methods. We can see one opportunity in Table C.6, with instance "S3.41". We have two optimal solutions (when %UT is 50 and 100) with the same value of objective function and a feasible (but not proven optimal) solution for %UT = 75. Hence it is possible to conclude that the solution found with $\%UT = 75$ is optimal.

3.6 Conclusion

An important industrial problem was considered: optimal design of transfer lines. In contrast to the classical formulations where either a number of machines or a cycle time should be minimised, we consider a robust version of the problem, which returns solution that remain feasible despite processing time variation affecting uncertain tasks. The solution of this problem is also a challenge taking into account the exigency of reducing the time from design to manufacturing.

Since the problem is $\mathcal{NP}$-hard, its solution time can be extremely large. The development of efficient methods and choice of a method adapted to each concrete problem is an important issue. Hybrid approach for the configuration of a transfer line with sequentially activated spindle heads at stations is evaluated. It includes both MILP and heuristic components. The heuristic method uses different restrictions for construction of a feasible solution that is used as a starting point in the MILP formulation. The approach has been tested on three data sets each of which contains 50 randomly generated instances.

The experiments have shown that our approach is suitable for both MILP formulations, but results for P_∞ are notably better than for P_1 from the point of view of number of optimal solutions, average gap and computational time. So far, we can conclude that the stability radius can be successfully applied for modelling of uncertainties in Transfer Line Balancing Problem.

On the basis of our analysis, the following conclusion can be drawn. The number of tasks per instance is an important point for assessing the computational effort, but the density of the precedence constraints (order of strength) has an even more significant impact on instance difficulty. Thus, medium size instances with the higher OS bring significantly better results than small size ones with less constraints. Of course, in the case where there are no precedence constraints, the problem would be very similar to the classic bin packing problem, for which very efficient extended formulations are available. Hence, moderately dense precedence constraints may lead to the most difficult instances for the proposed approaches.

Finally, availability of blocks to accomplish more tasks in parallel accelerates the solution process, that is not difficult to see with results for small size instances.

Further investigations will concern improving the multi-start heuristic method as well as the mixed integer linear programming models in order to enlarge our experiments. Since, TLBP can be considered as a generalisation of SALBP, we would like to apply the presented approach for this classical problem.

Chapter 4

Hybrid solution approaches for TLBP

Contents

Relying on a commercial solver to address transfer line balancing problem (TLBP) with a pure mixed integer linear programming approaches has been shown to be inefficient. Addressing larger problem instances requires developing alternative methods that still exploit commercial solvers, by that also benefit from the problem structure. In particular, many different advanced techniques allow to get the access to the calculation process and to interrupt it, which offers interesting perspectives. We are using this ability to devise a hybrid solution approach combining the computational power of a commercial solver and problem properties.

In the previous chapter, we have described pre-processing procedures which reduce the search space by implementing new constraints to mathematical models before their execution. Computational results show that this implementation has essential influence on the number of optimal solutions, average gap and running time per instance. Tables C.1–C.6 have many examples, where the solver has found a better solution comparing to the proposed one by the heuristic method introduced in Section 3.4.2. The main idea developed here is to take advantage of best feasible solution known so far to strengthen the allocation intervals of the tasks, so that the solver can exploit them and converge

faster. Hence, the corresponding constraints (see Chapter 3, (3.33) and (3.34)) can be updated in order to exclude more feasible solutions that are not optimal.

Below, we are reviewing such features of commercial software as *user callbacks*, discussing two possible realisations and applications for considered metrics.

4.1 User callbacks in modern solvers

Originally, a callback is an executable piece of code that is passed as an argument to another piece of code (function, procedure) that is expected to call back the argument at a given time. This execution may be immediate as in a synchronous callback, or it might happen at a later time as in an asynchronous callback. Following implementations of callbacks are widely used in programming languages: subroutines, lambda expressions, blocks, and function pointers.

The idea of the callback application is natural. Let us consider an algorithm or a program that should be executed. The following three events classically occur:

1. Call a the program;
2. Wait for the result;
3. Continue once the result comes in.

Simple launch of MILP models works exactly like this: once we have an instance data we call a solver for developed models, wait for the solution, analyse the results after termination. For small size instances the second step is almost immediate, but the situation is changing for large sizes. Of course, this schema is no longer acceptable when the result requires an unreasonably long amount of time to complete. There are two options to carry experiments in this case:

- Keep the program design and run it at night, weekend, without any ideas how fast MILP solver will go through the branching tree;
- Design the program to guide the seeking process of a commercial solver.

It should be noted that we do not modify an original code of a solver, but use the respective application programming interface (API) to get access to: the best solution found so far, lower and upper bounds, objective value, number of feasible solutions found, number of iterations, status code for a current MIP node and many other internal features. Below we consider two commercial solvers IBM CPLEX 12.7 and GUROBI 7.51 and discuss some particular realisations of callbacks. Any detailed information can be found in their manual user guides.

4.1.1 IBM CPLEX 12.7

According to API for this popular solver, callbacks allow to monitor closely and to guide the behaviour of the optimiser. In particular, the respective user code will be executed regularly during an optimisation or during a tuning session. To use callbacks with CPLEX, a callback function should have been written beforehand, and have been passed to the solver. There are three types of optimisation callbacks: informational callbacks, query callbacks, and control callbacks.

- Informational callbacks give an access to information about the progress of optimisation without interfering with the search path; also useful to terminate optimisation; compatible with dynamic search; compatible with all modes of parallel optimisation.
- Query callbacks, also known as diagnostic callbacks, give an access to information in the course of optimisation; incompatible with dynamic search; incompatible with deterministic parallel optimisation by default.
- Control callbacks are used to change the search path, for example, by interrupting and resuming optimisation under conditions specified by the user; incompatible with dynamic search; incompatible with deterministic parallel optimisation by default.

4.1.2 GUROBI 7.51

Callbacks in GUROBI are implemented within an abstract class. To implement a callback, a subclass of this class should be created, with a callback method. Then an object of the implemented subclass has to be created and passed to the model before starting its optimisation. The callback method will be called periodically during the seeking process. While IBM CPLEX divides callbacks into three types, GUROBI differentiates them by access points. These points determine which information about the progress of the optimisation is available to the user.

- PRESOLVE: during the execution of the presolve process; give access to the number of removed columns and rows, number of changed constraint senses, variable bounds and coefficients.
- SIMPLEX: during simplex method; give access to the number of simplex iterations, objective value, information about primal and dual infeasibility.
- MIP: during MIP process; give access to the best objective and best objective bound, explored and unexplored node count, number of feasible solutions found, count of cutting planes applied, number of simplex iterations;

- MIPSOL: the method is called whenever a new MIP incumbent is found; give access to the new solution, new objective value, current best objective and best bound, explored node count, number of feasible solutions found.

- MIPNODE: during the exploration of a MIP node; give access to the optimisation status of the current MIP node, best objective and best bound, explored node count, number of feasible solutions found.

Statement 4. *Since callbacks always present a function or a method that is called somewhere during optimisation process, we have to take into account the fact that they are going to consume computer resources (memory and time). It may affect results for some instances in experiments.*

4.2 Cuts generation with callbacks

In the preview of this chapter we have mentioned that using callbacks we want to generate optimality-based cuts for considered models. We consider two major technologies to achieve this: lazy constraints and MIP (MILP) restart.

4.2.1 Lazy constraints

Lazy constraints represent simply one portion of the constraints set, and the model would be incomplete (and possibly would deliver incorrect answers) in their absence. Solver always make sure that lazy constraints are satisfied before producing any solution to a MILP model. In our case,, the following constraints are encoded as lazy constraints, they enforce restrictions on decision variables resulting from allocation intervals:

$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p).$$

As we said above, they reduce the feasible space, but do not rule out any optimal solution that the rest of the model permits. Lazy constraints are collected into the pool without interrupting the solution process. After that, when the solver found a candidate integer-feasible solution, it checks one by one all the lazy constraints from the pool and progressively introduces them into the MILP model if the candidate solution violates them. It is supposed that the user designates constraints as lazy in the strong hope and expectation that they will not need to be applied, thus saving computation time by their absence from the working problem. In practice, it is relatively costly (for a variety of reasons) to apply a lazy constraint after a violation is identified, and so the user should err on the side of caution when deciding whether a constraint should be marked as lazy.

GUROBI requires to use a MIPSOL access point to apply lazy constraints into the models. In IBM CPLEX, they should be realised with control callbacks. Unfortunately, it means incompatibility with dynamic search, that may significantly affect the entire seeking process.[5] Thus, the GUROBI solver is more appropriate for lazy constraints application than IBM CPLEX.

Since we are using callbacks for cuts generation, the implementation differs for P_1 and P_∞.

Lazy constraints for P_1.

1. Wait until the MIP solver provides a new feasible solution s with its stability radius value ρ, then pause the execution.
2. If $\rho \leq \rho^*$, where ρ^* is the best known value at this moment, then go to Step 5.
3. Store the new value $\rho^* = \rho$ and for each uncertain task $j \in \widetilde{V}$, produce the following instructions:
   3.1. Set $t_j = t_j + \rho^*$.
   3.2. Apply Approach REDUCTION to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$.
   3.3. Set $t_j = t_j - \rho^*$.
4. Put the following lazy constraints into the POOL:
$$x_{jk} = 0, \quad \forall j \in V,\ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W,\ \forall k \in \widehat{U}(p).$$
5. Resume the MIP solver and go to Step 1.

Lazy constraints for P_∞.

1. Wait until the MIP solver provides a new feasible solution s with its stability radius value ρ, then pause the execution.
2. If $\rho \leq \rho^*$, where ρ^* is the best known value at this moment, then go to Step 7.
3. Store the new value $\rho^* = \rho$ and for each uncertain task $j \in \widetilde{V}$, set $t_j = t_j + \rho^*$.

[5] It makes sense for all versions up to 12.7. Hopefully, in later versions this problem will be solved.

4. Apply Approach REDUCTION to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$.

5. For each uncertain task $j \in \widetilde{V}$, set $t_j = t_j - \rho^*$.

6. Put the following lazy constraints into the POOL:

$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p).$$

7. Resume the MIP solver and go to Step 1.

The approach REDUCTION is matching the one presented in Chapter 3. Thus, callbacks are close to the global pre-processing method that we use to find an initial solution and restrictions on assignment intervals and sets of unused blocks for TLBP. It is not difficult to see that lazy constraints do not interrupt the execution of the solver. Because of this we do not have to store the best solution s in order to resume the search from the same point.

4.2.2 MIP restart

This approach is based on the ability to help the solver find an initial solution with additional information. These hints consist of pairs of variables and values, known as a MIP start, an advanced start, or a warm start.

A MIP start might come from a different problem that has previously been solved or from extra knowledge on the problem, or, what is more common, from an heuristic solution. In addition, the solver can take one or more MIP starts. A MIP start may be a feasible solution of the problem, but it is not mandatory; it may even be infeasible or incomplete. Both considered MIP solvers are able to deal with partially populated vectors, representing pairs of variables and values. They will attempt to fill in values for missing start values, but it is also possible to leave the start value for a variable undefined, in case if it is essential.

When providing a MIP start as data, the solver processes it before starting branch-and-cut during an optimisation. If one or more of the MIP starts define a solution, then the best of these solutions is installed as the incumbent solution. Having an incumbent from the very beginning of branch-and-cut allows to eliminate portions of the search space and thus may result in smaller branch-and-cut trees.

The main idea is to abort the search when a new best solution found, to apply cuts

to the model and to restart the seeking process with the last obtained solution as the incumbent. There are several advantages in doing so: since we generate optimality-based cuts as new constraints, they cannot remove the last solution from the search space. Thus, it can be guaranteed that the MIP start is presenting the feasible solution even for an enhanced model. Consequently, a solver would be appropriately restarted.

The difference from lazy constraints is clear: we do not create a pool of constraints within a single seeking process, but elaborate on the model using cuts generated while the optimisation is aborted. Informational callbacks are used in IBM CPLEX to achieve this goal. It is also possible with control callbacks, but the incompatibility with dynamic search makes this type less preferable. GUROBI allows to use either MIP or MIPSOL access point for the MIP restart approach. Moreover, the entire restart process is automated, and the user does not have to provide a MIP start for a modified model: the solver will try to build one from the solution of the previous model.

MIP restart for P_1.

1. Wait until the MIP solver provides a new feasible solution s with its stability radius value ρ and an estimated upper bound UB, then pause the execution.
2. If $\rho \leq \rho^*$, where ρ^* is the best knonw value at this moment, then resume the MIP solver and go to Step 1.
3. Store the new solution $s^* = s$, stability radius value $\rho^* = \rho$ and current upper bound $UB^* = UB$. Then, abort the search.
4. For each uncertain task $j \in \widetilde{V}$, produce the following instructions:
   - 3.1. Set $t_j = t_j + \rho^*$.
   - 3.2. Apply Approach REDUCTION to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$.
   - 3.3. Set $t_j = t_j - \rho^*$.
5. Form a new model from the previous one by the following constraints:
$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p).$$
6. Install the solution s^* as the initial one with the upper bound UB^* on the objective function. Then, restart the MIP solver for the enhanced model and go to Step 1.

MIP restart for P_∞.

1. Wait until the MIP solver provides a new feasible solution s with its stability radius value ρ and an estimated upper bound UB, then pause the execution.
2. If $\rho \leq \rho^*$, where ρ^* is the best known value at this moment, then resume the MIP solver and go to Step 1.
3. Store the new solution $s^* = s$, stability radius value $\rho^* = \rho$ and current upper bound $UB^* = UB$. Then, abort the search.
4. For each uncertain task $j \in \widetilde{V}$, set $t_j = t_j + \rho^*$.
5. Apply Approach Reduction to construct the sets of unused blocks $\widehat{U}(p)$, $p \in W$, and reduce intervals Q_j, $j \in V$.
6. For each uncertain task $j \in \widetilde{V}$, set $t_j = t_j - \rho^*$.
7. Form a new model from the previous one by the following constraints:

$$x_{jk} = 0, \quad \forall j \in V, \ \forall k \notin Q_j,$$
$$y_k = 0, \quad \forall p \in W, \ \forall k \in \widehat{U}(p).$$

8. Install the solution s^* as the initial one with the upper bound UB^* on the objective function. Then, restart the MIP solver for the enhanced model and go to Step 1.

It is necessary to explain why the MIP restart approach requires to deal with an upper bound as well as a solution and its stability radius value. The problem lies in the restart process. In certain cases an initial solution is not sufficient to speed up the optimisation up to respective boundaries, and the solver spends unneeded time for estimation. An upper bound from the previous launch allows to reduce it to a simple verification.

4.3 Computational results

These tests are equivalent to the one presented in previous chapter, but instead of all 150 instances we are using only the first 10 ones in each series, *i.e.*, instances "S1.0" to "S1.9", "S2.0" to "S2.9" and "S3.0" to "S3.9". All the parameters and presets are the same: the number of tasks per block varies between 2 and 3; and the ratios of uncertain tasks per instance takes values 50, 75 and 100 (%, percents of the initial number of tasks).

For the final experiment, we have selected GUROBI Optimiser as the MIP solver, since implementing callbacks in IBM CPLEX can be more tedious to achieve and did not bring significant advantage over GUROBI in preliminary tests. However, we do not compare these two solvers in order to find the best one, and may not suggest one before another.

Here, we discuss in detail MIP restart approach when callbacks are used, because it has some significant advantages compared to Lazy constraints. Appendix D has detailed results for both approaches with the parameter $r_{\max} = 2$ (Tables D.1 and D.2). An

		P_1							
		Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU
S1.0	50	28.8	0.17	3	0	30.4	30.4	0.0	107.25
	75	28.8	0.12	3	0	30.4	30.4	0.0	190.49
	100	28.8	0.11	3	0	30.4	30.4	0.0	134.35
S1.1	50	24.6	0.17	3	0	27.3	42.2	0.546	600.00
	75	23.4	0.14	0	0	23.7	42.2	0.781	600.00
	100	23.8	0.13	0	0	23.8	38.8	0.63	600.00
S1.2	50	27.7	0.11	5	0	29.1	52.8	0.814	600.00
	75	26.4	0.10	0	0	28.1	36.7	0.306	600.00
	100	27.7	0.10	0	0	28.0	28.9	0.032	600.00
S1.3	50	33.1	0.18	0	0	33.1	51.4	0.553	600.00
	75	29.3	0.16	0	0	29.5	51.4	0.742	600.00
	100	29.3	0.17	0	0	29.4	51.4	0.748	600.00
S1.4	50	26.0	0.13	6	1	28.5	37.9	0.33	600.00
	75	25.5	0.15	0	0	26.3	38.0	0.445	600.00
	100	24.3	0.14	5	0	26.3	36.8	0.399	600.00
S1.5	50	30.8	0.16	0	0	32.3	41.8	0.294	600.00
	75	29.0	0.14	0	0	29.0	47.673	0.644	600.00
	100	29.0	0.13	0	0	29.4	53.88	0.833	600.00
S1.6	50	27.1	0.13	0	0	28.5	28.6	0.004	600.00
	75	26.5	0.12	0	0	27.9	28.6	0.025	600.00
	100	26.5	0.11	0	0	27.9	28.6	0.025	600.00

Table 4.1: Example of detailed results for the hybrid approach

example is in Table 4.1. For any metric the table is divided in three groups of columns. The first group presents the lower bound and computational time of the heuristic method. The third one includes the results of the GUROBI solver. Columns in-between are related to generated optimality based cuts (OBC in the table): column "X" shows the number of additional reductions in assignment intervals when ρ rises up from the heuristic value to the last one obtained by GUROBI, and "Y" shows the number of block which have been closed after applying the reduction rules on the incumbent.

Lazy constraints and MIP restarts produce nearly identical results, and generate the

same number of cuts for studied models, but there are differences in gap and computation time, as can be seen in Table 4.2. Negative values indicate an improvement provided by lazy constraints over MIP restart. By contrast, positive ones mean that MIP restart works more efficiently. Thus, lazy constraints seem appropriate for small size instances in ℓ_1-metric. In all tests, the average gap is better, and the computational time is smaller than for MIP restart. But the picture is all different when we consider ℓ_∞-metric. Only for %UT= 75 in series S2 lazy constraints approach provides a slightly better gap, for any other, the restart of the MIP process is more accurate and faster. Finally, we can see that MIP restart approach take the lead in medium size instances (series S3) having up to almost 4 minutes ahead of lazy constraints.

		P_1		P_∞	
Series	%UT	ΔGAP_{avg}	ΔCPU_{avg}	ΔGAP_{avg}	ΔCPU_{avg}
S1	50	−0.021	−17.08	+0.347	+129.76
	75	−0.015	+41.67	+0.237	+114.66
	100	−0.029	−36.75	+0.293	+102.76
S2	50	−0.376	+0.01	+0.165	+3.73
	75	−0.781	−6.23	−0.089	+33.39
	100	−0.182	−0.00	+0.170	+58.31
S3	50	+0.21	+73.86	+0.426	+236.10
	75	+0.66	+43.63	+0.235	+219.42
	100	+0.60	+69.21	—	+73.71

Table 4.2: Improvements of Lazy constraints over MIP restart in terms of gap and CPU time

In results for $r_{\max} = 2$ (Table D.1) we see that OBC within callbacks work better for ℓ_∞-metric. It happens because of the nature of calculation. Simultaneous augmentation of all uncertain tasks by an extra value of the stability radius has a significant impact on the whole production line, than the same value for ℓ_1-metric, as it is limited to a single task.

Unfortunately, user callbacks consume time to get an information from execution process, that slightly increases computational time for those instances, which can be solved optimally even without advanced techniques. For example, "S3.0" to "S3.9" in ℓ_∞-metric, where every final result matches equivalent one from previous tests, but rises its CPU by 1 to 2 seconds. In ℓ_1-metric, some instances are not proven optimal (comparing to the previous tests): "S1.2" and "S1.7" ($\%UT = 50$); several groups works significantly longer: "S3.0", "S3.1" and "S3.3" (for all $\%UT$), "S3.6" ($\%UT = 50$ and $\%UT = 75$).

Lower and upper bounds (and gap) do not show a general tendency either for betterment nor degradation. We have at least nine possible representations when bounds rise up, remain or drop-down individually or together. However, most of improvements were obtained for instances of the series S2. This suggest to use callbacks preferably for problems with a low order of strength.

There are also several instances which bring good results for any value of the parameter $\%UT$: "S1.3", "S1.6", "S1.9", "S2.8", "S2.9". Since, all instances are randomly generated, there could be many explanations for this. One of them is a particular graph constitution that responds on reductions.

The results for $r_{\max} = 3$ confirm the conclusion drawn here when $r_{\max} = 2$. In ℓ_1-metric (Table 4.3), the average computational time (CPU) raises for all series of tests and parameters %UT, but not dramatically. Besides this, the number of optimal solutions keeps the same, except for series S2, where it falls down by 1 for all tests. This is an interesting result, because it may help to understand how callbacks affect CPU and entire execution process depending on the size of the problem and order of strength. Series S3 has some positive progress in the average gap, which means the reduction of upper bounds by the cuts generated with callbacks.

		MILP			MILP restart		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	10	0.0	70.01	10	0.0	94.57
	75	10	0.0	102.13	10	0.0	109.73
	100	10	0.0	71.25	10	0.0	74.17
S2	50	6	0.056	326.83	5	0.091	375.93
	75	3	0.131	440.54	2	0.293	503.46
	100	5	0.093	388.93	4	0.172	416.90
S3	50	5	0.374	397.43	5	0.360	408.64
	75	5	0.315	393.45	5	0.315	412.98
	100	5	0.234	405.33	5	0.233	405.85

Table 4.3: Average results in ℓ_1 metric

Table 4.4 compares results for ℓ_∞-metric. Series S1 and S3 do not loose any optimal solution in the hybrid solution approach, while S2 has one less for the parameter %UT= 50. Nevertheless, this series improves average gap and CPU in certain cases, which again suggests to use the hybrid approach for instances with a low order of strength. For some other tests the raising of CPU is even less than for ℓ_1-metric. Detailed information for any single instance is in Table D.3 (Appendix D).

4.4 Conclusion

In this chapter, we have presented a hybrid approach for addressing the stability radius maximisation problem under the ℓ_1 and ℓ_∞ metrics for a transfer line balancing problem under task time uncertainties. It consists in a combination of a commercial solver computational power and properties of the studied problem. To make it possible, the advanced technique of commercial solver was used. Two different approaches of callbacks implementation have been considered.

		MILP			MILP restart		
Series	%UT	# OPT	GAP_{avg}	CPU_{avg}	# OPT	GAP_{avg}	CPU_{avg}
S1	50	10	0.0	12.28	10	0.0	15.37
	75	10	0.0	10.10	10	0.0	10.44
	100	10	0.0	0.23	10	0.0	0.39
S2	50	6	0.111	331.94	5	0.130	325.80
	75	3	0.74	442.96	3	0.708	453.28
	100	9	0.006	79.96	9	0.003	79.80
S3	50	10	0.0	127.33	10	0.0	139.57
	75	10	0.0	46.07	10	0.0	63.91
	100	10	0.0	9.32	10	0.0	9.70

Table 4.4: Average results in ℓ_∞ metric

Computational experiments have shown that the CPU time of this hybrid approach is not competitive. However, it helps us to estimate the limits of our method and pushes to investigate new problem properties which can enrich the set of optimality based cuts generated within callbacks. Besides this, the hybrid approach gave interesting results for instances with a small density of the precedence graph, which also shows an opportunity for future applications.

General Conclusion

In this thesis, we have studied a robust optimisation of production lines at the design stage. For optimisation problems that appear at this stage, usually, decision-makers or designers do not have accurate data but an only estimate. Unfortunately, during the system life cycle, some data may be different from estimated values. Thus, a provided on the design stage solution is challenged. The purpose of our study was to mitigate this situation by developing models and approaches which choose among all feasible solutions the one that will be the most stable in the face of data variations. In particular, we worked on line balancing problems under uncertainties on the task processing times.

We began by general studying of existing production lines, their parameters and attributes. According to them, a line balancing problem was presented with three common objective functions minimising the number of machines (line cost), cycle time (maximising the production rate), line efficiency. Since deterministic versions of these problems do not consider any deviation in input data, which can be extremely unpractical for real-life applications, scientists use special techniques for modelling. In the second part of the beginning, we analysed existing in the literature approaches to model data uncertainty in combinatorial optimisation problems. Our objective was to identify the approaches that best fit the context of the design of production lines. The study conducted in Chapter 1 guided us towards the robust approach based on the stability radius, which shows the maximal value of processing time deviations that do not violate solution feasibility or optimality. Since the stability radius initially was used in the sensitivity analysis for a given solution, we refer to a recent formulation of the line balancing problem, which is maximising the robustness of the system. Further, we considered two types of production lines: simple assembly and transfer.

Robust formulations of the simple assembly line balancing problem were presented in Chapter 2. In this type of production lines stations are synchronised by a common conveyor. Production parts placed on it move from the beginning of the line to the end visiting all stations in order of their installation. The problem is to allocate assembly tasks to stations satisfying all manufacturing constraints. The number of stations is limited and the cycle time is fixed. The production goal is to maximise the robustness measure, i.e., stability radius. We have analysed two classic metric for the stability radius calculation and introduced a new one. Respective theorems and formulas were presented. We proposed combined mixed-integer linear programming models, which are aimed to

help decision-makers easily switch from one model to another according to particular situations and available knowledge.

The proposed MILP models have been programmed with C++ using the commercial solver Gurobi. Tests have been carried on instances from the online benchmark library with additional randomly generated data. Two parameters were used to install the level of uncertainty among tasks and stations. The results are encouraging. Even with the time limit on the execution process, optimal solutions can be found in general for instances up to 90 tasks, and up to 150 tasks in some particular tests. Supplementary experiments showed that results do not depend on the precedence graph density. Thus, presented models can be used for industrial instances under the assumption of the respective computational power.

In Chapter 3, we have studied transfer lines where several tasks can be performed at the same time by the same machining head. Each station (machine) contains one or more machining heads (blocks), the heads of each station are activated sequentially, the stations as before are synchronised by a common conveyor. Thus, the transfer line balancing problem consists in assignment of production tasks to machining heads and distribution of heads to machines satisfying all given constraints. The optimisation criterion is the robustness measured by the stability radius. Only ℓ_1 and ℓ_∞ metrics were considered for its calculation. We discovered the corresponding formulas and proved them within a series of lemmas and theorems. Then, two new mathematical models were presented. Their complexity requires some enhancements for the efficient execution. Based on well-known practices for line balancing problems, we introduced assignment intervals of tasks on transfer lines. Such interval shows the indexes of blocks that can execute the considered task. Five rules were elaborated to calculate these intervals for two different metrics and were applied within the approach REDUCTION.

Heuristic method for transfer lines has been elaborated. Besides the idea, which is similar to classic methods for assembly lines, we had to take into account the construction of blocks. To evade repetitive assignment of tasks, four procedures with different restrictions were constructed. Together they compose a multi-start heuristic algorithm which on every iteration is looking for a line balance with a better stability radius than the best obtained earlier. All presented developments form the global pre-processing approach, which not only generate an initial solution and a lower bound, but also create two groups of constraints for MILP formulations.

All models and pre-processing algorithms have been programmed with C++ and resolved by commercial solver Gurobi. Instance data were imported from the library of randomly generated examples for transfer lines. Since these type of production lines consists of automated stations or machines, we did not take into account the uncertainty of stations, but instead we used the parameter that indicate the maximal capacity of blocks. The results approved the central hypothesis that the order of strength has a significant impact on the resolution of the problem. Thus, even small size instances might not have an optimal solution in installed time limits. Another observation told us that

the block capacity also positively affect the resolution process. But this effect weakens with the size of the problem. In general, the presented method showed ability to solve instances whose size is equivalent to previously reported result in the literature related to transfer lines. Real problems of this type rarely exceed 150 tasks, and there are industrial examples sized to about 30 tasks (even less). In some cases this number can be easily reduced by grouping tasks into macro-tasks. So we can say that our approach are already partially operational.

Chapter 4 discussed a hybrid solution approach for the transfer line balancing problem. It consists in a combined application of a commercial MIP solver and optimality-based cuts generation algorithms. We used callback techniques which allow to manage and to do guide the optimisation process. Two commercial solvers were considered in order to find a proper realisation. Several types of callbacks were studied. We chose two possible approaches for cuts generation with callbacks. They are named by lazy constraints and MIP restart, respectively. Lazy constraints approach works with the pool of cuts that are introduced progressively into the model under optimisation. In this case, callbacks do not abort a solver and resume its work as only optimality-based cuts are generated. ℓ_1 and ℓ_∞ metrics required different implementations because of different interpretation of the stability radius. Similar to the global pre-processing method, we used REDUCTION method to determine assignment intervals and unused blocks on machines.

The hybrid solution approach has been programmed with C++ and resolved by commercial solver GUROBI for final tests. Experimentation was carried on the reduced set of instances comparing to Chapter 3. It was expected, that instances may take a bit more time before resolution, and the results proved this expectation for all realisations. The MIP restart approach has shown its advantage over lazy constraints, opportunities for future research and development.

All obtained results reveal a lot of perspectives for future researches in Robust Optimisation of Production Lines. First of all, we have several options to improve MILP models using advanced techniques of commercial solvers. But even in this case, there is a risk to face strict boundaries that will not allow finding an optimal solution for large size instances at an appropriate time. One of the keys to handling this issue lays in the development of meta-heuristics. These higher-level procedures may provide significantly better results with less computational effort. Among successful application, we mainly can see population-based approaches which maintain and improve multiple candidate solutions, often using population characteristics to guide the search. There are evolutionary computations, genetic algorithms, and particle swarm optimisation. Mentioned heuristic algorithms may generate an initial population for them with some enhancements installing task assignment priorities which are necessary for interactions between generations or iterations.

On another side we can develop exact methods to find an optimal solution of the problem. The literature includes many examples of branch-and-bound, branch-and-cut, branch-and-price algorithms for line balancing problems. The idea for any of them is

similar to the proposed in Chapter 3 heuristic method: the root of the branching tree includes an empty station, nodes represent a particular sub-solution. Starting from the root we create new nodes according to the set of tasks available for assignment, block capacity and cycle time constraints within a station. The main thing we are missing here is domination rules that will prune unpromising sub-trees. That leads us to the next perspective – upper bounds investigation.

An estimation of UB for TLBP seems to be divided into two stages: determination of blocks and their assignment to machines. That makes it impossible to apply some classic formulas for calculation since the first stage, in general, is presenting $\mathcal{NP}$-hard problem. Nevertheless, the having ideas may bring good results, which directly boost development of all exact, heuristic and meta-heuristic algorithms.

Our pre-processing technique for the transfer line balancing problem particularly refers to the well-known method called kernelisation. It is using for designing efficient algorithms that achieve their efficiency by a pre-processing stage in which inputs to the algorithm are replaced by a smaller input (kernel). The result of solving the problem on the kernel should either be the same as on the original input, or it should be easy to transform the output on the kernel to the desired output for the original problem. Simplification of the input helps to decision-makers, because they do not always have surplus information at the design stage. Thus, future development in this field has an undeniable practical meaning.

A number of operations research methods have found their way into constraint programming. This development is entirely natural, since both OR and CP have similar goals. CP's constraint-oriented approach to problem solving poses a prescriptive modelling task very similar to that of OR. CP historically has been less concerned with finding optimal than feasible solutions, but this is a superficial difference. There remains a fundamental difference in the way that CP and OR understand constraints. CP typically sees a constraint as a procedure, or at least as invoking a procedure, that operates on the solution space, normally by reducing variable domains. OR sees a constraint set as a whole cloth; the solution algorithm operates on the entire problem rather than the constraints in it. CP can design specialised algorithms for individual constraints or subsets of constraints, thereby exploiting substructure in the problem that OR methods are likely to miss.

MILP models play the role of a relaxation of the respective constraints programming problem. A solution of the relaxation then contributes to domain reduction or helps guide the search. Additional flexibility lies in the fact that instead of MILP models, it is possible to use their relaxations, Lagrangean relaxations, and dynamic programming models. All these methods formulate specialised relaxations for a wide variety of standard situations and provide tools for relaxing global constraints. Other hybridisation schemes decompose the problem so that CP and OR can attack the parts of the problem to which they are best suited. This combination can bring substantial computational benefits. Many presented applications in the literature report acceleration of the seeking process for Scheduling problems up to 1000 times. It could be a good opportunity to apply these approaches to

studied problems.

We considered two metrics (ℓ_1 and ℓ_∞) to calculate the stability radius in the transfer line balancing problem, and one additional (relative resilience) in the simple assembly line balancing problem. However, its classic definition is based on the concept of norm, which have various expressions in science. For example, there is a big group for measuring the difference between two sequences (see Levenshtein, Hamming, Lee and Jaro–Winkler distances). They are opening opportunities for estimation the stability of reconfigurable manufacturing systems (including production lines), whose popularity is growing up last years.

Another perspective concerns the generalisation of this approach to other types of lines, for example to lines on which all blocks are activated simultaneously or which have two possibilities of activation (serially-sequentially). We must also integrate other constraints, such as those related to machining modes, eligible and optimised (cutting speed optimisation), tasks and blocks inclusion/exclusion constraints. The development of a user-friendly prototype integrating all our programs and algorithms is also on the agenda.

In this thesis, we dealt with the problem of designing production lines in the presence of processing times subject to uncertainties. However, the problem of taking into account the data uncertainty on the design stage is not limited to the consideration of variations of these numerical data. Thus, possible changes can also occur at the level of various constraints like precedence constraints. So far, this aspect is not yet well treated in the literature. This is a track of our future research in this domain.

Bibliography

[1] K. Ağpak and H. Gökçen, A chance-constrained approach to stochastic line balancing problem, *European Journal of Operational Research*, vol. 180, pp. 1098–1115, 2007.

[2] S. Agrawal and M. Tiwari, A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem, *International Journal of Production Research*, vol. 46, no. 6, pp. 1405–1429, 2008, cited By 97.

[3] H. Babazadeh, M. Alavidoost, M. F. Zarandi, and S. Sayyari, An enhanced nsga-ii algorithm for fuzzy bi-objective assembly line balancing problems, *Computers & Industrial Engineering*, vol. 123, pp. 189–208, 2018.

[4] O. Battaïa and A. Dolgui, A taxonomy of line balancing problems and their solution approaches, *Int. J. Production Economics*, vol. 142, pp. 259–277, 2013.

[5] D. Battini, M. Faccio, E. Ferrari, A. Persona, and F. Sgarbossa, Design configuration for a mixed-model assembly system in case of low product demand, *The International Journal of Advanced Manufacturing Technology*, vol. 34, no. 1, pp. 188–200, 2007.

[6] A. Baykasoğlu and L. Özbakir, Stochastic u-line balancing using genetic algorithms, *International Journal of Advanced Manufacturing Technology*, vol. 32, no. 1-2, pp. 139–147, 2007, cited By 51.

[7] I. Belassiria, M. Mazouzi, S. ELfezazi, A. Cherrafi, and Z. ELMaskaoui, An integrated model for assembly line re-balancing problem, *International Journal of Production Research*, vol. 56, no. 16, pp. 5324–5344, 2018.

[8] M. L. Bentaha, Conception combinatoire des lignes de désassemblage sous incertitudes, PhD thesis, Saint-Etienne, EMSE, 2014.

[9] M. L. Bentaha, A. Dolgui, and O. Battaïa, A bibliographic review of production line design and balancing under uncertainty, *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 70–75, 2015, 15th IFAC Symposium onInformation Control Problems inManufacturing.

[10] N. Boysen, M. Fliedner, and A. Scholl, A classification of assembly line balancing problems, *European Journal of Operational Research*, vol. 183, pp. 674–693, 2007.

[11] B. Cakir, F. Altiparmak, and B. Dengiz, Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm, *Computers & Industrial Engineering*, vol. 60, no. 3, pp. 376–384, 2011, Recent Developments in the Analysis of Manufacturing and Support Systems.

[12] J. C. Carter and F. N. Silverman, A cost-effective approach to stochastic line balancing with off-line repairs, *Journal of Operations Management*, vol. 4, no. 2, pp. 145–157, 1984.

[13] A. Chakravarty and A. Shtub, A cost minimization procedure for mixed model production lines with normally distributed task times, *European Journal of Operational Research*, vol. 23, pp. 25–36, 1986.

[14] W.-C. Chiang, T. L. Urban, and C. Luo, Balancing stochastic two-sided assembly lines, *International Journal of Production Research*, vol. 54, no. 20, pp. 6232–6250, 2016.

[15] B. Das, A. Garcia-Diaz, C. A. MacDonald, and K. K. Ghoshal, A computer simulation approach to evaluating bowl versus inverted bowl assembly line arrangement with variable operation times, *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 1, pp. 15–24, 2010.

[16] X. Delorme, A. Dolgui, and M. Kovalyov, Combinatorial design of a minimum cost transfer line, *Omega*, vol. 40, pp. 31–41, 2012.

[17] A. Dolgui, B. Finel, N. Guschinsky, G. Levin, and F. Vernadat, Mip approach to balancing transfer lines with blocks of parallel operations, *IIE Transactions*, vol. 38, no. 10, pp. 869–882, 2006.

[18] A. Dolgui, B. Finel, F. Vernadat, N. Guschinsky, and G. Levin, A heuristic approach for transfer lines balancing, *Journal of Intelligent Manufacturing*, vol. 16, no. 2, pp. 159–172, 2005.

[19] A. Dolgui, N. Guschinsky, Y. Harrath, and G. Levin, Une approche de programmation linéaire pour la conception des lignes de transfert, *Journal Europeen des Systemes Automatises*, vol. 36, Jan. 2002.

[20] A. Dolgui, N. Guschinsky, and G. Levin, On problem of optimal design of transfer lines with parallel and sequential operations, in *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA '99*, vol. 1, Feb. 1999, 329–334 vol.1, ISBN: 0-7803-5670-5.

[21] A. Dolgui and S. Kovalev, Scenario based robust line balancing: Computational complexity, *Discrete Applied Mathematics*, vol. 160, no. 13, pp. 1955–1963, 2012.

[22] E. Erel, I. Sabuncuoglu, and H. Sekerci, Stochastic assembly line balancing using beam search, *International Journal of Production Research*, vol. 43, no. 7, pp. 1411–1426, 2005.

[23] Y. Fang, H. Ming, M. Li, Q. Liu, and D. T. Pham, Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time, *International Journal of Production Research*, vol. 0, no. 0, pp. 1–17, 2019.

[24] B. Finel, A. Dolgui, and F. Vernadat, A random search and backtracking procedure for transfer line balancing, *International Journal of Computer Integrated Manufacturing*, vol. 21, no. 4, pp. 376–387, 2008.

[25] F. G. Galizia, M. Bortolini, and C. Mora, Reconfigurable manufacturing systems: Literature review and research trend, *Journal of Manufacturing Systems*, vol. 49, pp. 93–106, Oct. 2018.

[26] R. Gamberini, A. Grassi, and B. Rimini, A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem, *International Journal of Production Economics*, vol. 102, no. 2, pp. 226–243, 2006.

[27] A. M. Geoffrion and R. Nauss, Parametric and postoptimality analysis in integer linear programming, *Management Science*, vol. 23, no. 5, pp. 453–466, 1977.

[28] M. P. Groover, Fundamentals of Modern Manufacturing: Materials, Processes and Systems, Fourth. John Wiley & Sons, Inc., 2010.

[29] F. Guerriero and J. Miltenburg, The stochastic u-line balancing problem, *Naval Research Logistics*, vol. 50, no. 1, pp. 31–57, 2003.

[30] S. K. Gupta and J. Rosenhead, Robustness in sequential investment decisions, *Management Science*, vol. 15, no. 2, B–18–B–29, Oct. 1968.

[31] E. Gurevsky, O. Battaïa, and A. Dolgui, Balancing of simple assembly lines under variations of task processing times, *Annals of Operations Research*, vol. 201, pp. 265–286, 2012.

[32] ——, Stability measure for a generalized assembly line balancing problem, *Discrete Applied Mathematics*, vol. 161, pp. 377–394, 2013.

[33] E. Gurevsky, Ö. Hazır, O. Battaïa, and A. Dolgui, Robust balancing of straight assembly lines with interval task times, *Journal of the Operational Research Society*, vol. 64, no. 11, pp. 1607–1613, 2013.

[34] O. Guschinskaya and A. Dolgui, Comparision of exact and heuristic methods for a transfer line balancing problem, *Int. J. Production Economics*, vol. 120, pp. 276–286, 2009.

[35] G. Halevi, Process and Operation Planning. Kluwer Academic Publishers, Jan. 2003.

[36] Ö. Hazır and A. Dolgui, Assembly line balancing under uncertainty: Robust optimization models and exact solution method, *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 261–267, 2013.

[37] Ö. Hazır and A. Dolgui, A decomposition based solution algorithm for u-type assembly line balancing with interval data, *Computers & Operations Research*, vol. 59, pp. 126–131, 2015.

[38] C. He, Z. Guan, G. Xu, L. Yue, and S. Ullah, Scenario-based robust dominance criteria for multi-objective automated flexible transfer line balancing problem under uncertainty, *International Journal of Production Research*, vol. 0, no. 0, pp. 1–20, 2019.

[39] K. Hitomi, Manufacturing Systems Engineering: A Unified Approach to Manufacturing Technology, Production Management and Industrial Economics, Second. Taylor & Francis, 1996.

[40] N. V. Hop, A heuristic solution for fuzzy mixed-model line balancing problem, *European Journal of Operational Research*, vol. 168, no. 3, pp. 798–810, 2006, Balancing Assembly and Transfer lines.

[41] M. N. Janardhanan, Z. Li, G. Bocewicz, Z. Banaszak, and P. Nielsen, Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times, *Applied Mathematical Modelling*, vol. 65, pp. 256–270, 2019.

[42] E. P. Kao, A preference order dynamic program for stochastic assembly line balancing, *Management Science*, vol. 22, no. 10, pp. 1097–1104, 1976.

[43] ——, Computational experience with a stochastic assembly line balancing algorithm, *Computers & Operations Research*, vol. 6, no. 2, pp. 79–86, 1979.

[44] Y. Kara, T. Paksoy, and C.-T. Chang, Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing, *European Journal of Operational Research*, vol. 195, pp. 335–347, 2009.

[45] J. F. Kottas and H. S. Lau, A stochastic line balancing procedure, *International Journal of Production Research*, vol. 19, no. 2, pp. 177–193, 1981.

[46] P. Kouvelis and G. Yu, Robust discrete optimization and its applications. Springer Science & Business Media, 1997, vol. 14.

[47] T.-C. Lai, Y. Sotskov, A. Dolgui, and A. Zatsiupa, Stability radii of optimal assembly line bbalance with a fixed workstation set, *Int. J. Production Economics*, vol. 182, pp. 356–371, 2016.

[48] T.-C. Lai, Y. N. Sotskov, and A. Dolgui, The stability radius of an optimal line balance with maximum efficiency for a simple assembly line, *European Journal of Operational Research*, vol. 274, no. 2, pp. 466–481, 2019.

[49] J. Lyu, A single-run optimization algorithm for stochastic assembly line balancing problems, *Journal of Manufacturing Systems*, vol. 16, no. 3, pp. 204–210, 1997, cited By 25.

[50] P. R. McMullen and G. V. Frazier, A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations, *International Journal of Production Economics*, vol. 51, no. 3, pp. 177–190, 1997.

[51] M. C. O. Moreira, J.-F. Cordeau, A. M. Costa, and G. Laporte, Robust assembly line balancing with heterogeneous workers, *Computers & Industrial Engineering*, vol. 88, pp. 254–263, 2015.

[52] M. C. O. Moreira, R. Pastor, A. M. Costa, and C. Miralles, The multi-objective assembly line worker integration and balancing problem of type-2, *Computers & Operations Research*, vol. 82, pp. 114–125, 2017.

[53] U. Özcan, Balancing stochastic two-sided assembly lines: A chanse-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm, *European Journal of Operational Research*, vol. 205, pp. 81–97, 2010.

[54] ——, Balancing stochastic parallel assembly lines, *Computers & Operations Research*, vol. 99, pp. 109–122, 2018.

[55] U. Özcan, T. Kellegöz, and B. Toklu, A genetic algorithm for the stochastic mixed-model u-line balancing and sequencing problem, *International Journal of Production Research*, vol. 49, no. 6, pp. 1605–1626, 2011.

[56] J. Pereira, The robust (minmax regret) assembly line worker assignment and balancing problem, *Computers & Operations Research*, vol. 93, pp. 27–40, 2018.

[57] N. R. Reeve and W. H. Thomas, Balancing stochastic assembly lines, *AIIE Transactions*, vol. 5, no. 3, pp. 223–229, 1973.

[58] A. Rossi, E. Gurevsky, O. Battaïa, and A. Dolgui, Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations, *Discrete Applied Mathematics*, vol. 208, pp. 123–136, 2016.

[59] B. Roy and P. Vincke, Relational systems of preference with one or more pseudo-criteria: Some new concepts and results, *Management Science*, vol. 30, no. 11, pp. 1323–1335, 1984.

[60] B. Roy, Robustness in operational research and decision aiding: A multi-faceted issue, *European Journal of Operational Research*, vol. 200, no. 3, pp. 629–638, 2010.

[61] U. Saif, Z. Guan, L. Zhang, J. Mirza, and Y. Lei, Hybrid pareto artificial bee colony algorithm for assembly line balancing with task time variations, *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 2-3, pp. 255–270, 2017.

[62] S. C. Sarin, E. Erel, and E. M. Dar-El, A methodology for solving single-model, stochastic assembly line balancing problem, *Omega*, vol. 27, no. 5, pp. 525–535, 1999.

[63] A. Scholl and C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, *European Journal of Operational Research*, vol. 168, no. 3, pp. 666–693, 2006.

[64] D. Shin, An efficient heuristic for solving stochastic assembly line balancing problems, *Computers & Industrial Engineering*, vol. 18, no. 3, pp. 285–295, 1990.

[65] Y. Sotskov, A. Dolgui, T.-C. Lai, and A. Zatsiupa, Enumerations and stability analisis of feasible and optimal line balance for simple assembly line, *Computers & Industrial Engineering*, vol. 90, pp. 241–258, 2015.

[66] Y. Sotskov, A. Dolgui, and M. Portmann, Stability analysis of an optimal balance for an assembly line with fixed cycle time, *European Journal of Operational Research*, vol. 168, pp. 783–797, 2006.

[67] L. Tiacci, Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times, *International Journal of Production Economics*, vol. 162, pp. 201–215, 2015.

[68] Y. Tsujimura, M. Gen, and E. Kubota, Solving fuzzy assembly-line balancing problem with genetic algorithms, *Computers & Industrial Engineering*, vol. 29, no. 1, pp. 543–547, 1995, Proceedings of the 17th International Conference on Computers and Industrial Engineering.

[69] T. L. Urban and W.-C. Chiang, Designing energy-efficient serial production lines: The unpaced synchronous line-balancing problem, *European Journal of Operational Research*, vol. 248, no. 3, pp. 789–801, 2016.

[70] L. V. Vanegas and A. W. Labib, Application of new fuzzy-weighted average (nfwa) method to engineering design evaluation, *International Journal of Production Research*, vol. 39, no. 6, pp. 1147–1162, 2001.

[71] C. Yang and J. Gao, Balancing mixed-model assembly lines using adjacent cross-training in a demand variation environment, *Computers & Operations Research*, vol. 65, pp. 139–148, 2016.

[72] J.-S. Yao and K. Wu, Ranking fuzzy numbers based on decomposition principle and signed distance, *Fuzzy Sets and Systems*, vol. 116, no. 2, pp. 275–288, 2000.

[73] P. Zacharia and A. Nearchou, A meta-heuristic algorithm for the fuzzy assembly line balancing type-e problem, *Computers & Operations Research*, vol. 40, no. 12, pp. 3033–3044, 2013.

[74] W. Zhang, W. Xu, G. Liu, and M. Gen, An effective hybrid evolutionary algorithm for stochastic multiobjective assembly line balancing problem, *Journal of Intelligent Manufacturing*, vol. 28, no. 3, pp. 783–790, 2017.

[75] Z. Zhang, Q. Tang, Z. Li, and L. Zhang, Modelling and optimisation of energy-efficient u-shaped robotic assembly line balancing problems, *International Journal of Production Research*, vol. 0, no. 0, pp. 1–18, 2018.

[76] F. Zheng, J. He, F. Chu, and M. Liu, A new distribution-free model for disassembly line balancing problem with stochastic task processing times, *International Journal of Production Research*, vol. 56, no. 24, pp. 7341–7353, 2018.

Appendix A

Simple assembly lines. Detailed computational results

Series without uncertain stations

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.36	0
BOWMAN8	8	4	25.5	0.375	0.375	0.02	0
MANSOOR	11	4	67.5	1.813	1.813	0.04	0
JAESCHKE	9	5	9	0.2	0.2	0.01	0
JACKSON	11	6	10.5	0.3	0.3	0.06	0
MITCHELL	21	7	19.5	0.833	0.833	0.07	0
ROSZIEG	25	8	19.5	0.5	0.5	0.01	0
HESKIA	28	8	162	0.514	0.514	0.03	0
LUTZ1	32	9	2100	0.5	0.5	0.01	0
BUXEY	29	11	37.5	0.5	0.5	0.06	0
SAWYER30	30	11	37.5	0.974	0.974	0.32	0
GUNTHER	35	10	60	0.45	0.45	0.06	0
HAHN	53	7	2662.5	0.793	0.793	0.13	0
KILBRID	45	9	82.5	0.5	0.5	0.04	0
TONGE70	70	18	234	0.5	0.5	0.89	0
WARNECKE	58	24	79.5	0.529	0.529	0.64	0
ARC83	83	17	5536.5	1.018	1.045	600	0.027
LUTZ3	89	18	111	0.5	0.5	0.65	0
BARTHOLD	148	12	574.5	0.5	0.5	1.11	0
MUKHERJE	94	20	256.5	0.5	0.5	0.53	0
ARC111	111	22	8533.5	0.687	0.687	3.68	0
LUTZ2	89	38	15	0.5	0.5	19.44	0
WEE-MAG	75	60	40.5	0.558	0.558	6.51	0
BARTHOL2	148	41	124.5	0.5	0.5	7.92	0
SCHOLL	297	41	2079	0.878	0.949	600	0.08
				#OPT: **23/25**		**49.70**	**0.004**

Table A.1: Results for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = 0$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.01	0
BOWMAN8	8	4	25.5	0.214	0.214	0	0
MANSOOR	11	4	67.5	0.691	0.691	0.03	0
JAESCHKE	9	5	9	0.125	0.125	0	0
JACKSON	11	6	10.5	0.3	0.3	0.02	0
MITCHELL	21	7	19.5	0.313	0.313	0.11	0
ROSZIEG	25	8	19.5	0.313	0.313	0.08	0
HESKIA	28	8	162	0.434	0.434	5.03	0
LUTZ1	32	9	2100	0.5	0.5	0.05	0
BUXEY	29	11	37.5	0.389	0.389	1.22	0
SAWYER30	30	11	37.5	0.464	0.464	3.75	0
GUNTHER	35	10	60	0.341	0.341	3.54	0
HAHN	53	7	2662.5	0.401	0.401	0.08	0
KILBRID	45	9	82.5	0.5	0.5	0.08	0
TONGE70	70	18	234	0.345	0.345	126.34	0
WARNECKE	58	24	79.5	0.347	0.423	600	0.217
ARC83	83	17	5536.5	0.425	0.429	600	0.01
LUTZ3	89	18	111	0.382	0.382	9.24	0
BARTHOLD	148	12	574.5	0.437	0.439	600	0.004
MUKHERJE	94	20	256.5	0.357	0.357	2.93	0
ARC111	111	22	8533.5	0.482	0.488	600	0.013
LUTZ2	89	38	15	0.154	0.286	600	0.862
WEE-MAG	75	60	40.5	0.5	0.5	9.8	0
BARTHOL2	148	41	124.5	0.324	0.383	600	0.181
SCHOLL	297	41	2079	0.404	0.433	600	0.072
				#OPT: **18/25**		**174.49**	**0.054**

Table A.2: Results for $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = 0$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.01	0
BOWMAN8	8	4	25.5	0.206	0.206	0	0
MANSOOR	11	4	67.5	0.5	0.5	0.01	0
JAESCHKE	9	5	9	0	0	0.01	0
JACKSON	11	6	10.5	0.3	0.3	0.01	0
MITCHELL	21	7	19.5	0.269	0.269	0.04	0
ROSZIEG	25	8	19.5	0.219	0.219	0.07	0
HESKIA	28	8	162	0.298	0.298	9.77	0
LUTZ1	32	9	2100	0.403	0.403	0.55	0
BUXEY	29	11	37.5	0.271	0.271	2	0
SAWYER30	30	11	37.5	0.25	0.25	8.87	0
GUNTHER	35	10	60	0.25	0.25	4.12	0
HAHN	53	7	2662.5	0.207	0.207	0.09	0
KILBRID	45	9	82.5	0.41	0.419	600	0.023
TONGE70	70	18	234	0.258	0.266	600	0.032
WARNECKE	58	24	79.5	0.242	0.324	600	0.338
ARC83	83	17	5536.5	0.268	0.278	600	0.037
LUTZ3	89	18	111	0.26	0.26	69.09	0
BARTHOLD	148	12	574.5	0.311	0.313	600	0.004
MUKHERJE	94	20	256.5	0.248	0.248	4.15	0
ARC111	111	22	8533.5	0.308	0.315	600	0.023
LUTZ2	89	38	15	0.125	0.17	600	0.357
WEE-MAG	75	60	40.5	0.3	0.5	600	0.667
BARTHOL2	148	41	124.5	0.231	0.278	600	0.203
SCHOLL	297	41	2079	0.263	0.285	600	0.086
				#OPT: **15/25**		**243.95**	**0.071**

Table A.3: Results for $|\widetilde{V}^1| = \lceil \frac{3n}{4} \rceil$ and $|\widehat{W}| = 0$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.03	0
BOWMAN8	8	4	25.5	0.159	0.159	0.01	0
MANSOOR	11	4	67.5	0.406	0.406	0.04	0
JAESCHKE	9	5	9	0	0	0.01	0
JACKSON	11	6	10.5	0.167	0.167	0.03	0
MITCHELL	21	7	19.5	0.219	0.219	0.07	0
ROSZIEG	25	8	19.5	0.219	0.219	0.19	0
HESKIA	28	8	162	0.256	0.256	7.7	0
LUTZ1	32	9	2100	0.282	0.282	0.51	0
BUXEY	29	11	37.5	0.172	0.172	3.81	0
SAWYER30	30	11	37.5	0.21	0.21	4.6	0
GUNTHER	35	10	60	0.2	0.2	4.97	0
HAHN	53	7	2662.5	0.14	0.14	0.12	0
KILBRID	45	9	82.5	0.331	0.345	600	0.044
TONGE70	70	18	234	0.194	0.195	600	0.007
WARNECKE	58	24	79.5	0.187	0.206	600	0.102
ARC83	83	17	5536.5	0.218	0.229	600	0.051
LUTZ3	89	18	111	0.194	0.194	82.13	0
BARTHOLD	148	12	574.5	0.222	0.224	600	0.006
MUKHERJE	94	20	256.5	0.161	0.166	600	0.035
ARC111	111	22	8533.5	0.24	0.248	600	0.032
LUTZ2	89	38	15	0.071	0.155	600	1.175
WEE-MAG	75	60	40.5	0.157	0.5	600	2.182
BARTHOL2	148	41	124.5	0.164	0.206	600	0.257
SCHOLL	297	41	2079	0.192	0.224	600	0.165
				#OPT: **14/25**		**268.17**	**0.162**

Table A.4: Results for $|\widetilde{V}^1| = n$ and $|\widehat{W}| = 0$

Series with uncertain stations (manual execution)

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.01	0
BOWMAN8	8	4	25.5	0.214	0.214	0	0
MANSOOR	11	4	67.5	1.75	1.75	0.01	0
JAESCHKE	9	5	9	0.125	0.125	0	0
JACKSON	11	6	10.5	0.3	0.3	0.02	0
MITCHELL	21	7	19.5	0.5	0.5	0.07	0
ROSZIEG	25	8	19.5	0.219	0.219	0.13	0
HESKIA	28	8	162	0.514	0.514	0.04	0
LUTZ1	32	9	2100	0.5	0.5	0.02	0
BUXEY	29	11	37.5	0.5	0.5	0.09	0
SAWYER30	30	11	37.5	0.786	0.786	1.45	0
GUNTHER	35	10	60	0.429	0.429	0.14	0
HAHN	53	7	2662.5	0.478	0.478	0.13	0
KILBRID	45	9	82.5	0.5	0.5	0.05	0
TONGE70	70	18	234	0.5	0.5	4.68	0
WARNECKE	58	24	79.5	0.529	0.529	8.21	0
ARC83	83	17	5536.5	0.695	0.712	600	0.025
LUTZ3	89	18	111	0.426	0.426	5.85	0
BARTHOLD	148	12	574.5	0.5	0.5	1.1	0
MUKHERJE	94	20	256.5	0.5	0.5	1.15	0
ARC111	111	22	8533.5	0.687	0.687	74.67	0
LUTZ2	89	38	15	0.25	0.5	600	1
WEE-MAG	75	60	40.5	0.558	0.558	6.78	0
BARTHOL2	148	41	124.5	0.5	0.5	40.48	0
SCHOLL	297	41	2079	0.711	0.949	600	0.335
				#OPT: **22/25**		**77.80**	**0.054**

Table A.5: Results for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.03	0
BOWMAN8	8	4	25.5	0.214	0.214	0	0
MANSOOR	11	4	67.5	0.985	0.985	0.01	0
JAESCHKE	9	5	9	0	0	0.01	0
JACKSON	11	6	10.5	0.167	0.167	0.07	0
MITCHELL	21	7	19.5	0.3	0.3	0.06	0
ROSZIEG	25	8	19.5	0.219	0.219	0.07	0
HESKIA	28	8	162	0.514	0.514	0.04	0
LUTZ1	32	9	2100	0.5	0.5	0.11	0
BUXEY	29	11	37.5	0.5	0.5	0.36	0
SAWYER30	30	11	37.5	0.442	0.442	13.62	0
GUNTHER	35	10	60	0.364	0.364	2.54	0
HAHN	53	7	2662.5	0.467	0.467	0.12	0
KILBRID	45	9	82.5	0.5	0.5	0.11	0
TONGE70	70	18	234	0.376	0.376	388.17	0
WARNECKE	58	24	79.5	0.395	0.517	600	0.31
ARC83	83	17	5536.5	0.438	0.475	600	0.085
LUTZ3	89	18	111	0.321	0.321	9.74	0
BARTHOLD	148	12	574.5	0.5	0.5	1.87	0
MUKHERJE	94	20	256.5	0.425	0.433	600	0.019
ARC111	111	22	8533.5	0.604	0.632	600	0.046
LUTZ2	89	38	15	0.167	0.448	600	1.688
WEE-MAG	75	60	40.5	0.558	0.558	14.25	0
BARTHOL2	148	41	124.5	0.368	0.496	600	0.348
SCHOLL	297	41	2079	0.35	0.814	600	1.324
				#OPT: **18/25**		**185.25**	**0.153**

Table A.6: Results for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.01	0
BOWMAN8	8	4	25.5	0.214	0.214	0	0
MANSOOR	11	4	67.5	0.607	0.607	0.01	0
JAESCHKE	9	5	9	0.125	0.125	0	0
JACKSON	11	6	10.5	0.3	0.3	0.01	0
MITCHELL	21	7	19.5	0.313	0.313	0.11	0
ROSZIEG	25	8	19.5	0.219	0.219	0.34	0
HESKIA	28	8	162	0.429	0.429	2.72	0
LUTZ1	32	9	2100	0.5	0.5	0.19	0
BUXEY	29	11	37.5	0.342	0.342	3.44	0
SAWYER30	30	11	37.5	0.442	0.442	3.18	0
GUNTHER	35	10	60	0.341	0.341	2.09	0
HAHN	53	7	2662.5	0.307	0.307	0.13	0
KILBRID	45	9	82.5	0.5	0.5	0.46	0
TONGE70	70	18	234	0.34	0.34	515.32	0
WARNECKE	58	24	79.5	0.347	0.387	600	0.114
ARC83	83	17	5536.5	0.333	0.341	600	0.024
LUTZ3	89	18	111	0.306	0.306	9.89	0
BARTHOLD	148	12	574.5	0.436	0.439	600	0.006
MUKHERJE	94	20	256.5	0.357	0.357	16.56	0
ARC111	111	22	8533.5	0.442	0.45	600	0.019
LUTZ2	89	38	15	0.154	0.273	600	0.773
WEE-MAG	75	60	40.5	0.5	0.5	11.59	0
BARTHOL2	148	41	124.5	0.297	0.383	600	0.291
SCHOLL	297	41	2079	0.347	0.433	600	0.251
				#OPT: **18/25**		**190.64**	**0.059**

Table A.7: Results for $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.02	0
BOWMAN8	8	4	25.5	0.214	0.214	0.01	0
MANSOOR	11	4	67.5	0.534	0.534	0.07	0
JAESCHKE	9	5	9	0	0	0.01	0
JACKSON	11	6	10.5	0.167	0.167	0.04	0
MITCHELL	21	7	19.5	0.219	0.219	0.06	0
ROSZIEG	25	8	19.5	0.219	0.219	0.16	0
HESKIA	28	8	162	0.409	0.409	0.38	0
LUTZ1	32	9	2100	0.483	0.483	0.31	0
BUXEY	29	11	37.5	0.293	0.293	1.74	0
SAWYER30	30	11	37.5	0.339	0.339	3.01	0
GUNTHER	35	10	60	0.304	0.304	3.11	0
HAHN	53	7	2662.5	0.307	0.307	0.13	0
KILBRID	45	9	82.5	0.5	0.5	1.91	0
TONGE70	70	18	234	0.272	0.292	600	0.074
WARNECKE	58	24	79.5	0.314	0.37	600	0.177
ARC83	83	17	5536.5	0.302	0.316	600	0.047
LUTZ3	89	18	111	0.233	0.233	10.95	0
BARTHOLD	148	12	574.5	0.426	0.436	600	0.025
MUKHERJE	94	20	256.5	0.343	0.357	600	0.04
ARC111	111	22	8533.5	0.416	0.446	600	0.071
LUTZ2	89	38	15	0.154	0.277	600	0.803
WEE-MAG	75	60	40.5	0.5	0.5	18.37	0
BARTHOL2	148	41	124.5	0.258	0.383	600	0.487
SCHOLL	297	41	2079	0.265	0.422	600	0.589
				#OPT: **16/25**		**217.61**	**0.093**

Table A.8: Results for $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.03	0
BOWMAN8	8	4	25.5	0.206	0.206	0	0
MANSOOR	11	4	67.5	0.5	0.5	0.01	0
JAESCHKE	9	5	9	0	0	0.02	0
JACKSON	11	6	10.5	0.3	0.3	0.02	0
MITCHELL	21	7	19.5	0.269	0.269	0.09	0
ROSZIEG	25	8	19.5	0.219	0.219	0.07	0
HESKIA	28	8	162	0.296	0.296	3.9	0
LUTZ1	32	9	2100	0.36	0.36	0.7	0
BUXEY	29	11	37.5	0.25	0.25	1.86	0
SAWYER30	30	11	37.5	0.25	0.25	9.2	0
GUNTHER	35	10	60	0.25	0.25	3.28	0
HAHN	53	7	2662.5	0.207	0.207	0.13	0
KILBRID	45	9	82.5	0.394	0.394	180.8	0
TONGE70	70	18	234	0.25	0.267	600	0.068
WARNECKE	58	24	79.5	0.242	0.309	600	0.276
ARC83	83	17	5536.5	0.261	0.262	600	0.004
LUTZ3	89	18	111	0.233	0.233	10.02	0
BARTHOLD	148	12	574.5	0.312	0.313	600	0.003
MUKHERJE	94	20	256.5	0.248	0.248	15.99	0
ARC111	111	22	8533.5	0.302	0.309	600	0.023
LUTZ2	89	38	15	0.071	0.214	600	1.997
WEE-MAG	75	60	40.5	0.3	0.5	600	0.667
BARTHOL2	148	41	124.5	0.233	0.278	600	0.195
SCHOLL	297	41	2079	0.249	0.285	600	0.144
				#OPT: **16/25**		**225.05**	**0.135**

Table A.9: Results for $|\widetilde{V}^1| = \lceil \frac{3n}{4} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$

Instance	n	m	T	LB	UB	CPU	GAP
MERTENS	7	4	9	0	0	0.01	0
BOWMAN8	8	4	25.5	0.206	0.206	0	0
MANSOOR	11	4	67.5	0.5	0.5	0	0
JAESCHKE	9	5	9	0	0	0.01	0
JACKSON	11	6	10.5	0.167	0.167	0.03	0
MITCHELL	21	7	19.5	0.219	0.219	0.04	0
ROSZIEG	25	8	19.5	0.219	0.219	0.13	0
HESKIA	28	8	162	0.29	0.29	1.77	0
LUTZ1	32	9	2100	0.314	0.314	0.85	0
BUXEY	29	11	37.5	0.25	0.25	3.12	0
SAWYER30	30	11	37.5	0.25	0.25	3.71	0
GUNTHER	35	10	60	0.25	0.25	2.35	0
HAHN	53	7	2662.5	0.207	0.207	0.11	0
KILBRID	45	9	82.5	0.385	0.39	600	0.011
TONGE70	70	18	234	0.238	0.238	319.91	0
WARNECKE	58	24	79.5	0.239	0.308	600	0.288
ARC83	83	17	5536.5	0.256	0.262	600	0.025
LUTZ3	89	18	111	0.207	0.207	16.52	0
BARTHOLD	148	12	574.5	0.309	0.313	600	0.013
MUKHERJE	94	20	256.5	0.239	0.246	600	0.031
ARC111	111	22	8533.5	0.299	0.309	600	0.034
LUTZ2	89	38	15	0.071	0.194	600	1.714
WEE-MAG	75	60	40.5	0.3	0.5	600	0.667
BARTHOL2	148	41	124.5	0.209	0.278	600	0.332
SCHOLL	297	41	2079	0.212	0.285	600	0.348
				#OPT: **15/25**		**253.94**	**0.138**

Table A.10: Results for $|\widetilde{V}^1| = \lceil \frac{3n}{4} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$

Appendix B

Simple assembly lines. Detailed instances

Data added to instance `ARC83`

Random task permutation
39 12 18 41 8 23 56 69 30 59 16 26 7 2 57 14 47 27 11 37 81 36 1 68 73 71 61 3 72 64 19 34 45 65 6 43 35 29 74 62 75 66 33 20 21 58 9 24 50 82 22 78 13 25 83 5 54 4 38 60 49 52 40 17 55 79 53 10 32 48 46 28 76 44 67 77 51 63 31 15 80 42 70
Random workstation permutation
13 5 4 3 16 9 1 6 17 12 7 14 8 15 10 2 11

Data added to instance `ARC111`

Random task permutation
39 20 95 64 85 33 75 49 52 13 107 1 61 98 81 94 18 70 26 63 56 38 35 44 16 45 89 77 19 73 23 43 74 32 67 51 96 58 76 5 109 28 4 7 71 102 101 12 34 30 9 10 17 99 100 3 25 104 111 57 72 84 65 108 80 82 91 14 27 90 92 41 11 60 50 93 54 22 103 68 88 8 59 47 97 2 29 79 48 86 83 66 24 6 46 62 69 78 87 31 15 55 36 53 40 110 21 106 37 42 105
Random workstation permutation
4 8 1 20 7 21 19 2 15 13 17 11 16 3 10 5 12 18 22 9 6 14

Data added to instance `BARTHOL2`

Random task permutation
39 77 70 121 74 75 136 145 96 33 60 16 58 12 40 107 115 98 8 120 131 18 129 37 126 111 106 76 6 36 108 7 38 92 48 141 122 97 34 143 66 44 41 54 123 127 104 4 114 57 128 21 137 24 67 42 2 130 135 68 47 73 19 146 109 35 113 134 147 83 100 85 87 95 28 55 22 81 63 139 26 9 13 88 103 64 31 56 59 25 3 51 14 72 29 71 1 89 65 125 82 53 52 45 148 50 78

43 11 142 46 5 15 117 93 23 20 138 91 119 132 90 110 61 84 112 140 32 17 133 49 79 124 116 102 27 99 101 118 30 10 62 94 80 105 86 69 144
Random workstation permutation
34 19 25 32 29 4 37 12 1 13 21 30 11 35 23 28 5 8 17 10 22 40 36 7 14 31 24 2 16 41 27 6 20 9 18 39 38 33 15 3 26

Data added to instance `BARTHOLD`

Random task permutation
39 77 70 121 74 75 136 145 96 33 60 16 58 12 40 107 115 98 8 120 131 18 129 37 126 111 106 76 6 36 108 7 38 92 48 141 122 97 34 143 66 44 41 54 123 127 104 4 114 57 128 21 137 24 67 42 2 130 135 68 47 73 19 146 109 35 113 134 147 83 100 85 87 95 28 55 22 81 63 139 26 9 13 88 103 64 31 56 59 25 3 51 14 72 29 71 1 89 65 125 82 53 52 45 148 50 78 43 11 142 46 5 15 117 93 23 20 138 91 119 132 90 110 61 84 112 140 32 17 133 49 79 124 116 102 27 99 101 118 30 10 62 94 80 105 86 69 144
Random workstation permutation
12 7 4 1 8 6 9 3 2 5 11 10

Data added to instance `BOWMAN8`

Random task permutation
7 6 5 3 8 2 4 1
Random workstation permutation
4 3 2 1

Data added to instance `BUXEY`

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

Data added to instance `GUNTHER`

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

Data added to instance `HAHN`

Random task permutation
5 41 50 53 20 22 21 6 29 9 35 11 13 8 49 10 38 34 32 25 46 36 23 30 19 18 7 26 24 37 51 16 39 43 28 12 17 52 14 15 2 1 44 45 4 47 40 3 33 48 42 31 27
Random workstation permutation
1 2 4 5 7 3 6

Data added to instance `HESKIA`

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

Data added to instance `JACKSON`

Random task permutation
8 9 10 2 7 3 1 5 6 11 4
Random workstation permutation
5 3 6 4 1 2

Data added to instance `JAESCHKE`

Random task permutation
6 4 9 8 5 1 2 7 3
Random workstation permutation
3 2 1 5 4

Data added to instance `KILBRID`

Random task permutation
16 8 22 45 31 32 23 19 18 37 21 44 5 14 15 4 30 38 12 3 6 43 1 33 25 13 28 42 7 39 2 10 24 35 36 27 29 34 40 20 41 17 11 9 26
Random workstation permutation
9 8 1 2 5 7 3 4 6

Data added to instance `LUTZ1`

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

Data added to instance `LUTZ2`

Random task permutation
3 85 71 18 30 46 62 2 11 53 58 14 78 12 22 52 82 20 68 27 28 37 73 87 89 50 13 25 61 17 70 86 24 47 54 29 5 66 1 31 56 69 79 4 21 75 74 64 32 41 38 9 59 63 10 48 49 7 19 39 65 15 51 60 43 55 76 33 57 8 23 84 35 26 44 6 83 40 45 80 16 88 36 42 67 72 77 81 34
Random workstation permutation
23 8 38 6 20 19 29 7 15 24 13 39 2 34 22 4 17 35 31 33 26 36 25 16 1 3 14 32 27 5 28 30 10 21 9 18 37 12 11

Data added to instance `LUTZ3`

Random task permutation
27 37 22 75 29 32 56 5 78 41 47 82 7 59 4 15 80 25 28 57 34 16 55 61 54 35 42 71 44 73 50 31 63 76 51 72 40 20 81 83 33 21 85 36 3 43 48 79 19 46 77 70 64 12 6 18 67 23 45 2 30 69 9 11 10 17 53 52 86 89 8 38 74 65 24 39 14 26 49 13 87 62 88 84 58 66 68 60 1
Random workstation permutation
8 16 18 6 5 17 9 10 11 7 1 14 15 3 13 2 4 12

Data added to instance `MANSOOR`

Random task permutation
7 9 4 2 11 1 10 6 3 8 5
Random workstation permutation
3 4 1 2

Data added to instance `MERTENS`

Random task permutation
3 7 6 4 5 2 1
Random workstation permutation
3 4 2 1

Data added to instance `MITCHELL`

Random task permutation
14 10 16 5 8 7 17 2 3 15 19 18 12 21 4 11 13 9 1 20 6
Random workstation permutation
6 3 7 4 5 1 2

Data added to instance `MUKHERJE`

Random task permutation
79 33 54 7 88 18 40 30 25 65 3 5 17 87 73 45 52 8 53 68 83 64 80 32 57 92 42 59 72 48 61 50 24 82 9 35 63 75 23 85 49 67 70 81 58 77 21 89 36 22 4 90 47 12 51 2 56 10 78 14 69 19 27 76 6 66 43 60 39 28 46 38 71 93 44 94 1 37 26 84 13 16 74 31 41 62 86 34 15 55 11 20 29 91
Random workstation permutation
12 4 17 10 3 1 6 15 13 7 18 5 11 20 14 9 19 2 16 8

Data added to instance `ROSZIEG`

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

Data added to instance `SAWYER30`

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

Data added to instance `SCHOLL`

Random task permutation
121 53 239 163 222 272 207 175 15 271 285 250 74 168 228 214 263 196 256 30 211 147 144 227 201 106 46 141 52 184 177 146 219 240 205 249 125 283 171 216 68 223 234 10 29 145 274 48 209 19 108 194 111 182 183 160 77 269 200 112 85 41 237 2 25 257 124 235 164 149 270 24 136 9 169 203 291 210 220 76 57 23 99 88 42 225 64 120 162 16 122 155 62 34 289 116 148 60 245 102 248 32 114 246 231 142 199 193 192 258 118 190 189 1 130 84 265 153 204 259 101 104 218 134 113 82 47 20 81 282 224 58 33 296 91 229 131 281 186 27 179 253 65 35 247 51 254 4 129 242 73 208 161 295 137 212 241 105 266 45 151 275 103 8 31 264 152 233 119 260 244 92 202 133 37 284 59 66 273 174 195 159 49 70 187 252 166 165 191 226 79 36 67 185 294 83 7 55 251 293 297 39 78 96 97 69 197 279 157 206 90 178 213 110 287 56 180 93 11 107 44 126 286 109 80 17 139 150 292 167 172 230 94 143 238 5 40 232 135 18 138 13 75 236 127 117 156 217 3 95 288 12 278 262 215 22 132 86 173 38 261 176 71 140 170 181 54 123 21 61 6 277 87 115 154 290 26 280 89 243 98 128 72 188 28 255 63 198 158 268 221 43 100 14 276 50 267
Random workstation permutation
27 14 28 4 35 34 11 30 9 20 13 29 1 36 32 22 17 41 12 10 21 2 15 33 40 18 24 5 8 19 37 6 38 31 39 26 16 23 3 25 7

Data added to instance `TONGE70`

Random task permutation
49 16 13 57 11 9 63 24 59 14 21 69 36 55 37 7 15 39 33 46 52 67 5 3 19 45 28 25 4 48 22 31 56 61 10 23 12 66 34 54 50 38 62 65 30 40 43 47 68 51 58 42 44 70 20 26 27 17 41 8 64 32 1 60 18 29 2 53 35 6
Random workstation permutation
10 4 2 16 8 5 18 1 6 17 7 14 3 15 13 9 12 11

Data added to instance `WARNECKE`

Random task permutation
35 29 42 51 15 21 38 33 45 41 40 34 14 47 57 1 49 3 22 10 27 16 26 28 19 37 8 36 43 54 12 9 4 24 30 6 50 58 44 13 55 11 7 17 46 2 32 18 53 31 48 52 25 20 39 23 5 56
Random workstation permutation
5 8 23 21 17 18 13 4 19 6 15 3 14 1 9 16 11 20 24 12 10 22 7 2

Data added to instance `WEE-MAG`

Random task permutation
40 10 47 50 59 43 33 49 17 4 56 41 21 55 11 37 51 74 71 28 18 25 20 52 2 53 65 63 35 38 26 6 67 66 75 69 3 64 60 68 44 24 14 54 42 61 12 19 9 16 46 7 22 45 27 39 15 31 23 57 34 29 72 73 32 1 5 8 13 30 36 70 58 62 48
Random workstation permutation
18 1 8 34 5 20 10 12 17 31 33 15 30 38 36 4 22 14 25 2 23 24 43 35 27 7 13 39 40 28 6 11 45 44 21 16 9 41 32 42 37 26 29 3 19

Appendix C

Transfer lines. Detailed computational results

Table C.1: Series 1. Example of table for metrics comparison; $r_{\max} = 2$.

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S1.0	50	28.8	0.17	30.4	30.4	0.0	52.44	19.4	0.41	21.15	21.15	0.0	15.57
	75	28.8	0.13	30.4	30.4	0.0	58.82	11.867	0.35	13.2	13.2	0.0	23.97
	100	28.8	0.12	30.4	30.4	0.0	83.67	9.5	0.48	10.133	10.133	0.0	6.74
S1.1	50	24.6	0.16	26.7	39.2	0.468	600.00	16.0	0.48	19.3	19.3	0.0	12.07
	75	23.8	0.17	23.8	38.8	0.63	600.00	10.3	0.44	11.55	11.55	0.0	193.60
	100	23.8	0.13	23.8	38.8	0.63	600.00	7.8	0.47	7.933	7.933	0.0	458.60
S1.2	50	27.7	0.12	29.1	29.1	0.0	486.42	16.1	0.43	17.55	21.369	0.218	600.00
	75	26.4	0.13	28.1	31.2	0.11	600.00	12.1	0.57	13.0	13.0	0.0	524.50
	100	27.7	0.11	28.0	36.7	0.311	600.00	9.2	0.35	9.333	9.333	0.0	8.09
S1.3	50	33.1	0.18	33.1	51.4	0.553	600.00	20.6	0.44	22.05	24.9	0.129	600.00
	75	29.3	0.16	29.5	51.4	0.742	600.00	12.85	0.58	14.0	39.075	1.791	600.00
	100	29.3	0.18	29.4	51.4	0.748	600.00	9.4	0.73	9.833	18.0	0.831	600.00
S1.4	50	25.0	0.14	28.5	46.771	0.641	600.00	17.65	0.43	18.6	18.6	0.0	133.00
	75	25.5	0.16	26.1	33.1	0.268	600.00	11.15	0.59	12.35	12.35	0.0	54.03
	100	24.3	0.14	26.3	33.1	0.259	600.00	8.5	0.47	8.767	8.767	0.0	18.85
S1.5	50	31.9	0.19	32.3	40.2	0.245	600.00	20.15	0.37	20.65	20.65	0.0	22.11
	75	29.0	0.15	29.0	47.673	0.644	600.00	14.45	0.64	14.45	16.35	0.131	600.00
	100	29.0	0.14	29.4	41.3	0.405	600.00	9.567	0.49	9.833	9.833	0.0	283.20
S1.6	50	27.0	0.16	28.5	35.4	0.242	600.00	17.7	0.48	18.9	20.8	0.101	600.00
	75	26.5	0.13	26.6	36.1	0.357	600.00	12.8	0.52	13.1	14.3	0.092	600.00
	100	26.5	0.12	27.9	28.6	0.025	600.00	8.833	0.41	9.3	9.3	0.0	13.30
S1.7	50	32.7	0.13	38.5	38.5	0.0	243.34	20.5	0.85	21.1	32.8	0.555	600.00
	75	29.1	0.15	29.6	41.0	0.385	600.00	13.65	0.53	14.3	20.3	0.42	600.00
	100	29.0	0.12	29.0	43.9	0.514	600.00	9.7	0.56	9.867	9.867	0.0	570.00
S1.8	50	32.4	0.17	32.4	32.4	0.0	138.04	18.5	0.47	20.6	20.6	0.0	48.04
	75	26.9	0.14	28.0	35.1	0.254	600.00	12.15	0.54	13.7	13.7	0.0	476.50
	100	26.7	0.14	27.9	35.1	0.258	600.00	8.9	0.51	9.633	9.633	0.0	33.22
S1.9	50	23.7	0.08	24.4	24.4	0.0	64.73	12.9	0.23	12.9	12.9	0.0	3.90
	75	24.4	0.08	24.4	24.4	0.0	120.12	9.467	0.25	9.467	9.467	0.0	2.11
	100	24.4	0.07	24.4	24.4	0.0	30.68	8.133	0.21	8.133	8.133	0.0	0.05
S1.10	50	25.4	0.12	26.7	27.6	0.034	600.00	18.3	0.37	20.2	20.2	0.0	31.47
	75	25.0	0.13	25.7	25.7	0.0	141.71	11.367	0.32	11.367	11.367	0.0	8.13

Table C.1 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	24.9	0.10	25.2	25.2	0.0	103.97	8.3	0.32	8.4	8.4	0.0	3.28
S1.11	50	23.6	0.10	25.6	25.6	0.0	75.83	12.95	0.23	12.95	12.95	0.0	4.32
	75	23.0	0.10	25.4	25.4	0.0	75.04	8.533	0.35	8.633	8.633	0.0	1.55
	100	23.0	0.09	25.4	25.4	0.0	44.97	7.833	0.27	8.467	8.467	0.0	0.22
S1.12	50	27.1	0.18	27.1	36.5	0.347	600.00	18.5	0.57	18.5	18.5	0.0	94.97
	75	25.1	0.19	25.9	45.1	0.741	600.00	10.75	0.44	12.05	18.25	0.515	600.00
	100	25.1	0.15	25.9	36.5	0.409	600.00	8.367	0.55	8.867	8.867	0.0	80.00
S1.13	50	25.5	0.09	25.5	25.5	0.0	65.58	15.3	0.28	17.3	17.3	0.0	2.64
	75	25.5	0.09	25.5	25.5	0.0	82.56	10.5	0.28	11.95	11.95	0.0	5.12
	100	25.5	0.09	25.5	25.5	0.0	66.93	8.5	0.28	8.5	8.5	0.0	0.31
S1.14	50	30.7	0.12	32.1	32.1	0.0	225.65	19.0	0.50	20.3	20.3	0.0	101.20
	75	28.7	0.11	30.1	30.3	0.007	600.00	11.45	0.37	12.2	12.2	0.0	22.43
	100	28.7	0.10	30.1	30.1	0.0	171.88	9.567	0.37	10.033	10.033	0.0	4.90
S1.15	50	25.1	0.15	25.3	28.9	0.142	600.00	16.05	0.39	16.05	16.05	0.0	29.85
	75	24.3	0.14	24.7	41.0	0.66	600.00	11.15	0.54	11.55	11.55	0.0	45.24
	100	24.3	0.13	24.7	33.8	0.368	600.00	8.1	0.54	8.367	8.367	0.0	68.15
S1.16	50	28.4	0.14	29.8	39.3	0.319	600.00	19.35	0.55	21.1	46.93	1.224	600.00
	75	27.6	0.18	27.6	39.3	0.424	600.00	12.55	0.57	13.15	19.65	0.494	600.00
	100	27.6	0.17	27.6	39.3	0.424	600.00	9.033	0.55	9.333	11.5	0.232	600.00
S1.17	50	32.0	0.14	32.0	32.0	0.0	140.43	18.05	0.44	18.4	18.4	0.0	78.17
	75	26.9	0.16	27.1	36.2	0.336	600.00	12.9	0.40	14.0	14.7	0.05	600.00
	100	26.9	0.15	28.1	36.2	0.288	600.00	8.933	0.40	9.367	9.367	0.0	16.43
S1.18	50	30.0	0.18	30.7	50.6	0.648	600.00	20.15	0.49	20.6	20.6	0.0	23.30
	75	30.4	0.20	30.7	50.6	0.648	600.00	13.85	0.56	13.85	17.4	0.256	600.00
	100	30.4	0.18	30.7	35.3	0.15	600.00	10.0	0.79	10.233	10.233	0.0	86.67
S1.19	50	30.8	0.14	31.4	37.8	0.204	600.00	19.0	0.50	19.85	19.85	0.0	31.92
	75	29.4	0.15	29.4	42.1	0.432	600.00	12.0	0.42	13.35	14.6	0.094	600.00
	100	29.1	0.11	29.4	36.9	0.255	600.00	9.733	0.40	9.8	9.8	0.0	14.20
S1.20	50	32.3	0.16	32.8	43.003	0.311	600.00	19.4	0.85	19.7	50.1	1.543	600.00
	75	25.9	0.18	26.3	44.4	0.688	600.00	11.2	0.64	11.45	39.964	2.49	600.00
	100	25.9	0.17	27.4	49.068	0.791	600.00	8.6	0.77	9.133	14.65	0.604	600.00
S1.21	50	31.9	0.17	33.0	40.2	0.218	600.00	20.1	0.68	20.75	37.6	0.812	600.00
	75	30.3	0.16	31.2	38.8	0.244	600.00	14.8	0.71	15.15	20.1	0.327	600.00
	100	30.3	0.15	30.8	38.8	0.26	600.00	10.233	0.63	10.433	11.1	0.064	600.00

Table C.1 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S1.22	50	30.6	0.14	33.8	35.4	0.047	600.00	18.5	0.50	19.9	19.9	0.0	116.80
	75	24.5	0.15	25.3	41.5	0.64	600.00	11.65	0.47	12.05	16.55	0.373	600.00
	100	24.5	0.14	25.3	38.6	0.526	600.00	8.167	0.56	8.467	8.467	0.0	427.70
S1.23	50	32.8	0.14	33.4	41.7	0.249	600.00	18.4	0.46	19.45	24.15	0.242	600.00
	75	27.4	0.15	27.4	54.7	0.996	600.00	12.55	0.62	12.7	20.25	0.594	600.00
	100	27.4	0.15	27.4	48.268	0.762	600.00	9.133	0.59	9.233	12.767	0.383	600.00
S1.24	50	33.3	0.18	34.0	38.2	0.124	600.00	20.5	0.44	20.6	20.6	0.0	33.35
	75	26.3	0.20	26.5	49.062	0.851	600.00	11.25	0.52	12.7	20.0	0.575	600.00
	100	26.3	0.17	27.3	43.8	0.604	600.00	9.067	0.56	9.233	10.367	0.123	600.00
S1.25	50	30.9	0.11	31.9	35.1	0.1	600.00	20.3	0.37	21.05	21.05	0.0	208.00
	75	29.0	0.14	29.0	44.484	0.534	600.00	12.85	0.43	14.15	14.15	0.0	245.30
	100	29.0	0.13	29.3	40.5	0.382	600.00	9.667	0.40	10.0	10.0	0.0	144.40
S1.26	50	20.8	0.08	21.2	21.2	0.0	16.71	11.3	0.21	13.8	13.8	0.0	2.62
	75	20.7	0.08	21.2	21.2	0.0	27.14	7.5	0.28	7.5	7.5	0.0	9.34
	100	20.3	0.09	21.2	21.2	0.0	30.04	6.767	0.22	7.067	7.067	0.0	1.07
S1.27	50	30.2	0.14	30.8	40.6	0.318	600.00	19.3	0.45	20.2	22.5	0.114	600.00
	75	26.7	0.12	29.2	36.1	0.236	600.00	11.3	0.53	12.8	15.05	0.176	600.00
	100	26.7	0.11	29.2	31.6	0.082	600.00	8.733	0.42	9.733	9.733	0.0	14.19
S1.28	50	32.4	0.20	32.4	50.1	0.546	600.00	18.5	0.63	20.15	30.7	0.524	600.00
	75	26.4	0.17	27.6	50.1	0.815	600.00	12.3	0.64	13.2	21.75	0.648	600.00
	100	26.4	0.17	27.6	50.1	0.815	600.00	8.8	0.52	9.2	17.65	0.918	600.00
S1.29	50	35.0	0.15	38.0	38.8	0.021	600.00	19.5	0.56	19.9	21.2	0.065	600.00
	75	29.3	0.17	31.3	50.2	0.604	600.00	13.15	0.60	14.05	19.4	0.381	600.00
	100	28.9	0.16	29.7	38.8	0.306	600.00	9.633	0.65	10.633	10.633	0.0	84.29
S1.30	50	34.1	0.16	35.0	35.0	0.0	149.85	19.5	0.45	19.5	19.5	0.0	178.50
	75	26.3	0.14	26.7	36.3	0.36	600.00	12.05	0.51	12.25	16.95	0.384	600.00
	100	26.3	0.13	26.7	35.0	0.311	600.00	8.633	0.43	8.9	8.9	0.0	32.47
S1.31	50	30.7	0.18	32.9	40.4	0.228	600.00	20.3	0.47	20.7	21.7	0.048	600.00
	75	28.2	0.12	29.0	42.081	0.451	600.00	11.867	0.60	12.55	12.55	0.0	301.10
	100	28.2	0.11	29.0	37.412	0.29	600.00	9.367	0.44	9.667	9.667	0.0	11.61
S1.32	50	33.4	0.17	33.4	49.286	0.476	600.00	18.7	0.57	19.4	49.436	1.548	600.00
	75	27.4	0.16	29.4	37.6	0.279	600.00	12.05	0.48	13.1	30.7	1.344	600.00
	100	27.4	0.15	29.4	43.7	0.486	600.00	9.2	0.66	9.8	10.233	0.044	600.00
S1.33	50	30.9	0.13	30.9	30.9	0.0	202.80	17.9	0.42	19.6	19.6	0.0	33.74

Table C.1 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	28.1	0.14	28.1	28.1	0.0	325.15	12.05	0.66	13.75	13.75	0.0	102.80
	100	28.1	0.13	28.1	28.1	0.0	117.56	9.3	0.65	9.367	9.367	0.0	4.85
S1.34	50	31.4	0.24	31.7	40.5	0.278	600.00	19.0	0.54	20.25	20.25	0.0	272.20
	75	25.9	0.20	27.1	57.2	1.111	600.00	11.9	0.58	12.7	21.45	0.689	600.00
	100	25.9	0.17	27.5	43.9	0.596	600.00	8.667	0.62	9.167	18.0	0.964	600.00
S1.35	50	27.5	0.12	30.3	30.3	0.0	96.22	19.5	0.30	19.55	19.55	0.0	23.88
	75	27.3	0.12	28.1	28.1	0.0	112.62	12.3	0.30	13.45	13.45	0.0	7.78
	100	27.3	0.12	28.1	28.1	0.0	196.83	9.1	0.36	9.367	9.367	0.0	1.75
S1.36	50	31.3	0.11	31.3	32.8	0.048	600.00	15.65	0.40	16.4	16.4	0.0	82.29
	75	23.7	0.13	24.0	38.4	0.6	600.00	11.3	0.39	11.4	11.4	0.0	61.17
	100	23.7	0.12	25.4	27.2	0.071	600.00	7.9	0.40	8.467	8.467	0.0	7.67
S1.37	50	25.2	0.11	26.2	26.2	0.0	78.69	16.7	0.28	17.6	17.6	0.0	11.56
	75	23.0	0.11	23.4	25.2	0.077	600.00	10.65	0.33	10.85	10.85	0.0	18.30
	100	23.0	0.10	23.4	23.4	0.0	87.46	7.8	0.33	7.8	7.8	0.0	6.81
S1.38	50	24.7	0.15	25.0	25.0	0.0	62.42	13.35	0.41	13.85	13.85	0.0	13.05
	75	24.7	0.14	25.0	25.0	0.0	57.55	10.7	0.44	12.2	12.2	0.0	8.77
	100	24.7	0.13	25.0	25.0	0.0	72.81	8.333	0.28	8.333	8.333	0.0	0.81
S1.39	50	28.2	0.14	28.2	37.5	0.33	600.00	18.2	0.72	18.2	18.2	0.0	28.16
	75	28.2	0.13	28.2	33.8	0.199	600.00	12.9	0.41	12.9	12.9	0.0	143.80
	100	28.1	0.11	28.1	37.5	0.335	600.00	9.367	0.51	9.367	9.367	0.0	9.71
S1.40	50	27.1	0.15	27.1	36.1	0.332	600.00	16.8	0.42	18.5	18.5	0.0	36.53
	75	24.8	0.16	25.6	37.4	0.461	600.00	11.95	0.57	12.1	14.15	0.169	600.00
	100	24.8	0.15	26.0	36.1	0.388	600.00	8.367	0.49	8.8	8.8	0.0	16.00
S1.41	50	26.8	0.12	28.8	28.8	0.0	107.36	18.05	0.34	19.65	19.65	0.0	12.80
	75	25.6	0.11	27.1	27.1	0.0	458.98	11.5	0.52	12.9	12.9	0.0	15.18
	100	25.6	0.10	27.1	27.1	0.0	225.76	8.667	0.40	9.033	9.033	0.0	5.97
S1.42	50	30.7	0.17	30.7	42.0	0.368	600.00	18.9	0.56	19.65	50.1	1.55	600.00
	75	25.7	0.18	25.9	39.7	0.533	600.00	11.45	0.60	12.4	21.0	0.694	600.00
	100	25.8	0.16	26.8	49.294	0.839	600.00	8.4	0.54	8.9	17.9	1.011	600.00
S1.43	50	32.8	0.20	32.8	40.8	0.244	600.00	19.95	0.76	20.2	39.8	0.97	600.00
	75	27.0	0.19	28.5	49.198	0.726	600.00	12.65	1.08	14.25	37.6	1.639	600.00
	100	27.0	0.18	28.5	45.0	0.579	600.00	8.7	0.65	9.5	19.9	1.095	600.00
S1.44	50	34.0	0.21	34.0	40.1	0.179	600.00	18.95	0.49	20.6	20.75	0.007	600.00
	75	28.4	0.21	28.5	45.894	0.61	600.00	13.0	0.65	13.4	20.5	0.53	600.00

Table C.1 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	28.4	0.20	28.5	40.1	0.407	600.00	9.1	0.56	9.5	15.2	0.6	600.00
S1.45	50	28.3	0.14	28.3	29.4	0.039	600.00	17.15	0.59	18.2	18.2	0.0	117.60
	75	22.1	0.12	22.8	36.8	0.614	600.00	9.9	0.42	9.9	16.95	0.712	600.00
	100	22.1	0.11	22.8	36.4	0.596	600.00	7.367	0.45	7.6	7.6	0.0	517.40
S1.46	50	31.1	0.16	31.1	34.3	0.103	600.00	18.25	0.35	18.25	18.25	0.0	19.94
	75	26.9	0.15	26.9	33.0	0.227	600.00	11.85	0.57	12.9	12.9	0.0	391.50
	100	26.9	0.14	26.9	33.0	0.227	600.00	8.967	0.49	9.067	9.067	0.0	14.14
S1.47	50	25.6	0.15	27.2	40.4	0.485	600.00	16.1	0.54	16.25	20.2	0.243	600.00
	75	24.0	0.14	25.0	40.4	0.616	600.00	10.05	0.51	11.0	21.45	0.95	600.00
	100	22.7	0.11	23.8	40.4	0.697	600.00	7.567	0.38	8.233	8.233	0.0	240.00
S1.48	50	28.7	0.14	29.7	32.2	0.084	600.00	16.85	0.44	19.3	19.3	0.0	177.00
	75	27.5	0.17	27.7	33.1	0.195	600.00	12.2	0.39	13.7	13.7	0.0	109.20
	100	27.5	0.16	27.7	33.1	0.195	600.00	9.167	0.43	9.467	9.467	0.0	59.85
S1.49	50	30.1	0.20	30.1	39.4	0.309	600.00	16.7	0.46	18.1	18.1	0.0	469.80
	75	25.6	0.13	25.9	39.0	0.506	600.00	11.65	0.43	12.1	16.25	0.343	600.00
	100	25.4	0.14	26.4	39.1	0.481	600.00	8.167	0.48	8.8	8.8	0.0	53.24

Table C.2: **Series 2. Example of table for metrics comparison; $r_{\max} = 2$.**

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S2.0	50	35.8	0.19	35.8	50.2	0.402	600.00	21.4	0.83	22.7	50.2	1.211	600.00
	75	31.3	0.18	32.5	50.2	0.545	600.00	14.3	0.94	15.65	50.2	2.208	600.00
	100	30.0	0.16	31.3	54.267	0.734	600.00	9.967	0.96	10.567	34.766	2.29	600.00
S2.1	50	31.3	0.24	33.5	56.8	0.696	600.00	17.75	0.80	20.0	50.2	1.51	600.00
	75	25.1	0.21	26.9	51.2	0.903	600.00	11.45	0.83	11.75	45.408	2.864	600.00
	100	25.4	0.18	26.2	50.9	0.943	600.00	8.467	1.06	8.867	23.05	1.6	600.00
S2.2	50	33.1	0.18	34.7	45.4	0.308	600.00	20.65	0.76	22.3	51.4	1.305	600.00
	75	28.5	0.17	30.2	43.8	0.45	600.00	12.95	0.76	14.0	36.7	1.621	600.00
	100	28.6	0.17	29.7	43.8	0.475	600.00	9.433	0.79	9.967	18.35	0.841	600.00
S2.3	50	35.6	0.19	38.3	59.5	0.554	600.00	22.15	1.02	22.4	51.4	1.295	600.00
	75	31.2	0.31	33.6	60.0	0.786	600.00	13.75	0.95	15.2	51.4	2.382	600.00
	100	30.0	0.17	30.3	60.0	0.98	600.00	9.9	1.29	10.2	45.686	3.479	600.00
S2.4	50	31.9	0.25	32.5	59.7	0.837	600.00	18.7	0.92	20.65	50.5	1.446	600.00
	75	27.5	0.32	29.0	52.7	0.817	600.00	11.75	0.91	13.2	50.1	2.795	600.00
	100	26.0	0.17	26.9	56.413	1.097	600.00	8.567	1.48	9.1	45.375	3.986	600.00
S2.5	50	35.5	0.21	37.3	51.8	0.389	600.00	21.2	0.79	22.7	51.8	1.282	600.00
	75	33.4	0.18	34.1	60.0	0.76	600.00	14.3	0.84	15.25	50.354	2.302	600.00
	100	30.5	0.18	31.6	58.5	0.851	600.00	9.767	0.78	10.333	37.225	2.602	600.00
S2.6	50	28.0	0.22	28.6	35.2	0.231	600.00	17.05	0.56	21.35	21.35	0.0	565.30
	75	26.0	0.15	28.5	35.4	0.242	600.00	12.55	0.50	14.3	14.3	0.0	115.50
	100	26.0	0.14	28.6	28.6	0.0	384.09	8.833	0.53	9.533	9.533	0.0	14.37
S2.7	50	38.5	0.19	38.5	55.8	0.449	600.00	21.25	0.73	23.0	51.8	1.252	600.00
	75	30.6	0.19	31.2	55.66	0.784	600.00	14.15	1.17	15.45	50.6	2.275	600.00
	100	30.4	0.19	31.1	52.2	0.678	600.00	10.233	1.00	10.567	38.538	2.647	600.00
S2.8	50	34.5	0.20	36.8	51.4	0.397	600.00	20.2	0.63	20.8	42.8	1.058	600.00
	75	27.5	0.16	29.1	51.4	0.766	600.00	12.8	0.62	13.75	41.3	2.004	600.00
	100	27.5	0.19	28.8	51.4	0.785	600.00	9.167	1.07	9.7	22.6	1.33	600.00
S2.9	50	34.5	0.18	35.2	52.1	0.48	600.00	20.65	0.57	21.3	50.7	1.38	600.00
	75	30.4	0.23	31.2	52.1	0.67	600.00	14.05	0.85	15.05	50.347	2.345	600.00
	100	30.8	0.18	30.8	48.073	0.561	600.00	10.167	0.70	10.267	20.0	0.948	600.00
S2.10	50	34.6	0.20	36.6	57.0	0.557	600.00	20.1	0.69	22.7	51.6	1.273	600.00

Table C.2 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	28.6	0.19	28.8	57.0	0.979	600.00	13.0	0.79	14.3	50.2	2.51	600.00
	100	28.5	0.16	30.0	57.0	0.9	600.00	9.5	0.91	9.967	45.57	3.572	600.00
S2.11	50	34.5	0.22	36.3	53.6	0.477	600.00	20.95	0.68	22.45	51.7	1.303	600.00
	75	29.3	0.19	30.1	56.4	0.874	600.00	13.85	0.82	14.8	50.9	2.439	600.00
	100	28.8	0.17	30.1	53.2	0.767	600.00	9.767	1.08	10.033	32.758	2.265	600.00
S2.12	50	32.4	0.18	33.5	52.7	0.573	600.00	19.2	0.85	20.4	50.9	1.495	600.00
	75	25.9	0.23	27.0	52.7	0.952	600.00	11.75	1.28	13.25	50.9	2.842	600.00
	100	25.9	0.19	27.2	52.7	0.938	600.00	8.533	0.92	8.967	44.165	3.925	600.00
S2.13	50	34.2	0.15	34.2	41.6	0.216	600.00	19.25	0.53	21.1	48.941	1.319	600.00
	75	26.2	0.17	26.7	51.3	0.921	600.00	12.75	0.66	13.8	40.6	1.942	600.00
	100	26.6	0.15	27.6	44.4	0.609	600.00	8.8	0.80	9.467	20.3	1.144	600.00
S2.14	50	36.0	0.33	36.8	59.5	0.617	600.00	21.05	0.80	21.55	50.1	1.325	600.00
	75	30.4	0.17	31.4	57.7	0.838	600.00	14.35	1.21	14.9	50.1	2.362	600.00
	100	30.4	0.15	31.7	57.7	0.82	600.00	10.167	0.79	10.667	40.087	2.758	600.00
S2.15	50	31.8	0.16	33.7	50.7	0.504	600.00	18.7	0.87	19.85	50.5	1.544	600.00
	75	25.0	0.26	26.5	50.7	0.913	600.00	11.95	1.04	12.55	38.1	2.036	600.00
	100	25.0	0.15	25.8	49.867	0.933	600.00	8.267	0.78	9.0	19.05	1.117	600.00
S2.16	50	33.8	0.15	33.8	58.7	0.737	600.00	20.75	0.97	21.6	50.9	1.356	600.00
	75	27.9	0.15	28.6	108.661	2.799	600.00	13.0	0.81	13.95	22.25	0.595	600.00
	100	27.6	0.14	28.5	52.752	0.851	600.00	9.3	0.76	9.533	22.25	1.334	600.00
S2.17	50	34.6	0.23	36.1	50.85	0.409	600.00	19.55	0.90	20.1	50.4	1.507	600.00
	75	28.0	0.17	30.6	60.0	0.961	600.00	12.85	1.01	13.95	24.35	0.746	600.00
	100	29.4	0.19	29.6	55.815	0.886	600.00	9.5	0.96	9.933	24.35	1.451	600.00
S2.18	50	35.8	0.17	38.4	50.6	0.318	600.00	20.45	0.98	22.35	50.6	1.264	600.00
	75	30.4	0.23	30.7	50.6	0.648	600.00	14.65	1.09	15.1	50.6	2.351	600.00
	100	30.3	0.20	30.9	50.6	0.638	600.00	10.1	0.91	10.567	23.35	1.21	600.00
S2.19	50	36.6	0.14	37.8	55.3	0.463	600.00	21.0	0.74	22.3	52.0	1.332	600.00
	75	30.1	0.20	30.7	55.8	0.818	600.00	13.9	0.91	14.65	51.2	2.495	600.00
	100	30.0	0.19	30.7	49.242	0.604	600.00	9.833	0.68	10.567	22.1	1.091	600.00
S2.20	50	32.5	0.17	32.5	50.8	0.563	600.00	19.8	0.89	20.25	50.1	1.474	600.00
	75	27.2	0.22	27.2	53.9	0.982	600.00	12.4	0.72	14.35	50.1	2.491	600.00
	100	26.8	0.20	27.6	57.2	1.072	600.00	8.933	0.77	9.433	38.342	3.065	600.00
S2.21	50	34.3	0.16	34.4	53.6	0.558	600.00	20.8	0.72	23.15	51.2	1.212	600.00
	75	29.9	0.18	31.3	41.3	0.319	600.00	13.85	0.63	14.5	40.881	1.819	600.00

Table C.2 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	29.9	0.17	31.2	41.3	0.324	600.00	10.033	0.81	10.567	18.387	0.74	600.00
S2.22	50	32.4	0.21	33.8	55.7	0.648	600.00	18.6	0.70	19.95	50.4	1.526	600.00
	75	25.5	0.17	26.5	58.2	1.196	600.00	12.2	0.91	13.65	50.4	2.692	600.00
	100	25.5	0.24	26.6	58.2	1.188	600.00	8.733	1.20	9.0	44.502	3.945	600.00
S2.23	50	34.3	0.23	35.8	52.7	0.472	600.00	19.15	0.55	20.5	49.622	1.421	600.00
	75	28.3	0.18	28.9	52.7	0.824	600.00	13.45	0.80	14.25	23.2	0.628	600.00
	100	27.4	0.14	29.1	52.7	0.811	600.00	9.1	0.68	9.767	20.25	1.073	600.00
S2.24	50	35.9	0.22	36.2	57.3	0.583	600.00	20.1	0.99	21.2	50.8	1.396	600.00
	75	27.1	0.26	28.1	57.3	1.039	600.00	12.75	0.81	13.95	50.8	2.642	600.00
	100	27.6	0.20	27.7	57.3	1.069	600.00	9.0	1.02	9.467	46.027	3.862	600.00
S2.25	50	37.6	0.23	39.8	51.8	0.302	600.00	21.7	0.76	22.0	51.1	1.323	600.00
	75	29.4	0.18	30.8	53.1	0.724	600.00	14.1	0.80	14.75	51.1	2.464	600.00
	100	29.4	0.17	30.8	55.5	0.802	600.00	9.833	0.78	10.267	31.184	2.037	600.00
S2.26	50	33.0	0.23	35.2	40.9	0.162	600.00	17.6	0.57	19.3	23.2	0.202	600.00
	75	25.3	0.20	30.2	50.952	0.687	600.00	11.1	0.68	12.0	21.162	0.764	600.00
	100	24.0	0.16	25.2	51.7	1.052	600.00	8.067	0.89	8.667	19.2	1.215	600.00
S2.27	50	32.0	0.20	33.8	44.7	0.322	600.00	19.6	0.59	21.85	28.7	0.314	600.00
	75	27.0	0.15	29.3	39.1	0.334	600.00	12.45	0.77	14.05	19.55	0.391	600.00
	100	26.6	0.18	29.9	39.1	0.308	600.00	8.933	0.63	9.967	9.967	0.0	219.20
S2.28	50	34.3	0.18	35.8	50.1	0.399	600.00	19.9	1.02	20.8	50.1	1.409	600.00
	75	27.3	0.25	28.3	116.561	3.119	600.00	12.75	0.78	14.2	50.1	2.528	600.00
	100	27.3	0.22	28.1	60.0	1.135	600.00	9.133	0.81	9.533	29.221	2.065	600.00
S2.29	50	34.4	0.19	34.4	51.8	0.506	600.00	20.05	0.82	21.3	50.2	1.357	600.00
	75	30.5	0.21	31.8	52.9	0.664	600.00	13.8	0.91	14.5	50.2	2.462	600.00
	100	29.8	0.21	31.1	55.7	0.791	600.00	9.867	0.85	10.667	41.505	2.891	600.00
S2.30	50	34.1	0.22	34.9	59.8	0.713	600.00	19.9	0.92	21.0	50.3	1.395	600.00
	75	27.5	0.22	28.7	50.3	0.753	600.00	12.3	0.98	14.2	50.3	2.542	600.00
	100	27.8	0.30	28.5	50.7	0.779	600.00	9.033	0.96	9.5	47.145	3.963	600.00
S2.31	50	33.7	0.23	36.3	43.0	0.185	600.00	18.45	0.72	21.3	37.8	0.775	600.00
	75	27.3	0.18	30.1	35.7	0.186	600.00	12.2	0.79	14.2	17.85	0.257	600.00
	100	27.3	0.17	29.7	36.3	0.222	600.00	8.867	0.59	10.067	10.067	0.0	96.30
S2.32	50	33.8	0.22	35.9	51.5	0.435	600.00	19.05	0.67	20.45	50.6	1.474	600.00
	75	28.5	0.17	29.4	60.0	1.041	600.00	13.35	1.06	14.8	49.663	2.356	600.00
	100	27.8	0.18	29.9	50.6	0.692	600.00	9.2	0.95	9.8	33.872	2.456	600.00

Table C.2 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S2.33	50	35.3	0.20	38.1	60.0	0.575	600.00	21.0	1.12	21.7	50.1	1.309	600.00
	75	30.4	0.27	31.1	58.712	0.888	600.00	14.4	1.31	15.8	50.1	2.171	600.00
	100	30.1	0.23	31.2	56.753	0.819	600.00	10.033	1.14	10.5	41.559	2.958	600.00
S2.34	50	32.4	0.21	34.9	53.1	0.521	600.00	18.75	0.85	20.75	45.6	1.198	600.00
	75	27.4	0.16	27.6	59.8	1.167	600.00	12.2	0.81	13.7	23.6	0.723	600.00
	100	26.7	0.18	27.1	59.8	1.207	600.00	8.667	0.70	9.3	22.8	1.452	600.00
S2.35	50	33.7	0.13	36.8	42.3	0.149	600.00	21.85	0.86	23.05	24.3	0.054	600.00
	75	31.0	0.19	33.7	48.6	0.442	600.00	14.0	0.54	15.25	24.8	0.626	600.00
	100	30.2	0.16	32.3	42.2	0.307	600.00	10.067	0.73	10.767	11.9	0.105	600.00
S2.36	50	30.6	0.17	30.8	51.9	0.685	600.00	18.5	0.46	19.9	50.4	1.533	600.00
	75	27.3	0.23	28.4	51.9	0.827	600.00	12.65	0.68	13.45	24.0	0.784	600.00
	100	27.2	0.19	28.1	43.0	0.53	600.00	8.9	0.97	9.467	16.5	0.743	600.00
S2.37	50	32.3	0.25	32.3	51.4	0.591	600.00	18.1	0.72	21.25	50.8	1.391	600.00
	75	26.6	0.23	27.3	56.4	1.066	600.00	12.6	0.83	13.6	40.992	2.014	600.00
	100	26.1	0.16	26.7	56.4	1.112	600.00	8.767	1.29	9.233	21.25	1.301	600.00
S2.38	50	35.6	0.19	37.7	53.9	0.43	600.00	20.3	0.68	21.8	50.5	1.317	600.00
	75	29.3	0.21	31.4	53.9	0.717	600.00	13.8	0.87	15.65	50.5	2.227	600.00
	100	29.3	0.23	31.3	50.3	0.607	600.00	9.833	0.88	10.4	43.901	3.221	600.00
S2.39	50	36.3	0.26	36.9	51.921	0.407	600.00	21.2	0.83	22.6	50.1	1.217	600.00
	75	30.3	0.34	31.5	172.412	4.473	600.00	13.75	1.13	14.9	50.1	2.362	600.00
	100	30.6	0.27	31.1	52.9	0.701	600.00	10.167	1.44	10.5	49.128	3.679	600.00
S2.30	50	34.1	0.22	34.9	59.8	0.713	600.00	19.9	0.92	21.0	50.3	1.395	600.00
	75	27.5	0.22	28.7	50.3	0.753	600.00	12.3	0.98	14.2	50.3	2.542	600.00
	100	27.8	0.30	28.5	50.7	0.779	600.00	9.033	0.96	9.5	47.145	3.963	600.00
S2.31	50	33.7	0.23	36.3	43.0	0.185	600.00	18.45	0.72	21.3	37.8	0.775	600.00
	75	27.3	0.18	30.1	35.7	0.186	600.00	12.2	0.79	14.2	17.85	0.257	600.00
	100	27.3	0.17	29.7	36.3	0.222	600.00	8.867	0.59	10.067	10.067	0.0	96.30
S2.32	50	33.8	0.22	35.9	51.5	0.435	600.00	19.05	0.67	20.45	50.6	1.474	600.00
	75	28.5	0.17	29.4	60.0	1.041	600.00	13.35	1.06	14.8	49.663	2.356	600.00
	100	27.8	0.18	29.9	50.6	0.692	600.00	9.2	0.95	9.8	33.872	2.456	600.00
S2.33	50	35.3	0.20	38.1	60.0	0.575	600.00	21.0	1.12	21.7	50.1	1.309	600.00
	75	30.4	0.27	31.1	58.712	0.888	600.00	14.4	1.31	15.8	50.1	2.171	600.00
	100	30.1	0.23	31.2	56.753	0.819	600.00	10.033	1.14	10.5	41.559	2.958	600.00
S2.34	50	32.4	0.21	34.9	53.1	0.521	600.00	18.75	0.85	20.75	45.6	1.198	600.00

Table C.2 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	27.4	0.16	27.6	59.8	1.167	600.00	12.2	0.81	13.7	23.6	0.723	600.00
	100	26.7	0.18	27.1	59.8	1.207	600.00	8.667	0.70	9.3	22.8	1.452	600.00
S2.35	50	33.7	0.13	36.8	42.3	0.149	600.00	21.85	0.86	23.05	24.3	0.054	600.00
	75	31.0	0.19	33.7	48.6	0.442	600.00	14.0	0.54	15.25	24.8	0.626	600.00
	100	30.2	0.16	32.3	42.2	0.307	600.00	10.067	0.73	10.767	11.9	0.105	600.00
S2.36	50	30.6	0.17	30.8	51.9	0.685	600.00	18.5	0.46	19.9	50.4	1.533	600.00
	75	27.3	0.23	28.4	51.9	0.827	600.00	12.65	0.68	13.45	24.0	0.784	600.00
	100	27.2	0.19	28.1	43.0	0.53	600.00	8.9	0.97	9.467	16.5	0.743	600.00
S2.37	50	32.3	0.25	32.3	51.4	0.591	600.00	18.1	0.72	21.25	50.8	1.391	600.00
	75	26.6	0.23	27.3	56.4	1.066	600.00	12.6	0.83	13.6	40.992	2.014	600.00
	100	26.1	0.16	26.7	56.4	1.112	600.00	8.767	1.29	9.233	21.25	1.301	600.00
S2.38	50	35.6	0.19	37.7	53.9	0.43	600.00	20.3	0.68	21.8	50.5	1.317	600.00
	75	29.3	0.21	31.4	53.9	0.717	600.00	13.8	0.87	15.65	50.5	2.227	600.00
	100	29.3	0.23	31.3	50.3	0.607	600.00	9.833	0.88	10.4	43.901	3.221	600.00
S2.39	50	36.3	0.26	36.9	51.921	0.407	600.00	21.2	0.83	22.6	50.1	1.217	600.00
	75	30.3	0.34	31.5	172.412	4.473	600.00	13.75	1.13	14.9	50.1	2.362	600.00
	100	30.6	0.27	31.1	52.9	0.701	600.00	10.167	1.44	10.5	49.128	3.679	600.00

Table C.3: **Series 3. Example of table for metrics comparison; $r_{\max} = 2$.**

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S3.0	50	15.7	0.40	15.7	15.7	0.0	167.88	7.2	0.77	7.2	7.2	0.0	8.74
	75	15.5	0.40	15.5	15.5	0.0	378.43	4.05	1.01	4.05	4.05	0.0	12.48
	100	15.5	0.40	15.5	15.5	0.0	178.52	3.925	0.98	3.925	3.925	0.0	5.92
S3.1	50	16.4	0.42	17.6	17.6	0.0	187.87	7.2	0.86	7.2	7.2	0.0	23.23
	75	16.4	0.41	17.6	17.6	0.0	151.98	4.6	0.91	4.6	4.6	0.0	16.20
	100	16.4	0.41	17.6	17.6	0.0	119.50	4.1	0.89	4.4	4.4	0.0	3.86
S3.2	50	17.8	0.40	17.8	23.408	0.315	600.00	9.0	0.92	9.0	9.0	0.0	18.85
	75	17.8	0.41	17.9	18.0	0.006	600.00	6.65	0.94	6.65	6.65	0.0	14.90
	100	17.8	0.42	17.8	28.145	0.581	600.00	4.5	0.98	4.5	4.5	0.0	3.27
S3.3	50	11.6	0.45	11.6	11.6	0.0	476.83	5.45	0.83	5.45	5.45	0.0	34.79
	75	11.6	0.45	11.6	11.6	0.0	254.68	3.067	1.00	3.067	3.067	0.0	34.77
	100	11.6	0.42	11.6	11.6	0.0	303.28	2.9	1.04	2.9	2.9	0.0	21.13
S3.4	50	14.0	0.38	14.0	14.0	0.0	39.96	4.95	0.80	4.95	4.95	0.0	14.09
	75	14.0	0.38	14.0	14.0	0.0	59.43	4.15	0.85	4.15	4.15	0.0	2.59
	100	14.0	0.37	14.0	14.0	0.0	23.59	3.575	0.75	3.575	3.575	0.0	0.49
S3.5	50	14.1	0.47	14.1	31.102	1.206	600.00	7.5	1.01	7.5	7.5	0.0	89.53
	75	14.1	0.48	14.1	28.9	1.05	600.00	4.7	1.16	4.7	4.7	0.0	34.34
	100	14.1	0.47	14.1	21.9	0.553	600.00	3.525	1.16	3.525	3.525	0.0	46.04
S3.6	50	19.4	0.42	20.7	20.7	0.0	309.95	11.2	0.88	11.2	11.2	0.0	65.78
	75	18.3	0.45	20.7	20.7	0.0	470.85	7.05	1.03	7.4	7.4	0.0	91.80
	100	18.3	0.44	18.3	23.5	0.284	600.00	6.1	1.02	6.9	6.9	0.0	1.66
S3.7	50	20.0	0.45	20.7	22.3	0.077	600.00	9.133	0.99	9.7	9.7	0.0	83.70
	75	20.0	0.46	20.7	28.06	0.356	600.00	7.05	1.21	7.05	7.05	0.0	78.46
	100	20.0	0.44	20.7	20.7	0.0	391.54	5.975	1.11	5.975	5.975	0.0	4.63
S3.8	50	24.1	0.71	24.1	60.0	1.49	600.00	12.95	1.18	12.95	19.35	0.494	600.00
	75	23.5	0.52	23.5	60.0	1.553	600.00	8.667	1.72	8.667	22.25	1.567	600.00
	100	23.2	0.49	23.2	60.0	1.586	600.00	7.733	1.43	8.167	8.167	0.0	28.84
S3.9	50	4.4	0.41	4.4	4.4	0.0	91.04	2.2	0.60	2.55	2.55	0.0	3.73
	75	4.4	0.42	4.4	4.4	0.0	38.81	1.1	0.71	1.2	1.2	0.0	14.89
	100	4.4	0.42	4.4	4.4	0.0	35.95	1.1	0.80	1.1	1.1	0.0	22.22
S3.10	50	11.0	0.45	11.0	11.0	0.0	337.35	4.4	0.82	4.4	4.4	0.0	54.90

Table C.3 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	11.0	0.46	11.0	11.0	0.0	125.44	3.667	0.96	3.667	3.667	0.0	10.76
	100	11.0	0.44	11.0	11.0	0.0	247.46	2.75	1.07	2.75	2.75	0.0	31.03
S3.11	50	13.0	0.48	13.0	13.0	0.0	250.34	6.6	0.92	6.6	22.017	2.336	600.00
	75	13.0	0.49	13.0	19.8	0.523	600.00	4.625	1.10	4.95	4.95	0.0	38.54
	100	13.0	0.47	13.0	13.0	0.0	251.12	3.25	1.06	3.25	3.25	0.0	22.30
S3.12	50	7.3	0.38	9.8	9.8	0.0	15.14	4.333	0.65	4.7	4.7	0.0	6.33
	75	7.3	0.39	9.8	9.8	0.0	19.01	3.1	0.72	3.25	3.25	0.0	0.45
	100	7.3	0.38	9.8	9.8	0.0	13.90	1.825	0.77	2.45	2.45	0.0	4.63
S3.13	50	11.2	0.42	11.6	11.6	0.0	282.46	4.5	0.81	4.5	4.5	0.0	19.55
	75	11.2	0.43	11.6	11.6	0.0	326.40	3.733	0.89	3.85	3.85	0.0	11.54
	100	11.2	0.42	11.6	11.6	0.0	273.06	2.8	0.90	2.9	2.9	0.0	2.51
S3.14	50	18.4	0.47	18.4	32.6	0.772	600.00	9.2	1.00	9.2	9.2	0.0	220.50
	75	18.4	0.45	18.4	22.3	0.212	600.00	7.7	1.10	7.7	7.7	0.0	10.08
	100	18.4	0.45	18.4	21.9	0.19	600.00	6.133	1.05	6.133	6.133	0.0	2.08
S3.15	50	12.9	0.42	12.9	20.3	0.574	600.00	8.867	0.88	10.5	10.5	0.0	57.67
	75	12.9	0.42	12.9	19.967	0.548	600.00	5.2	0.97	5.2	5.2	0.0	31.77
	100	12.9	0.41	12.9	20.3	0.574	600.00	3.325	1.00	3.325	3.325	0.0	5.99
S3.16	50	14.1	0.41	14.1	14.1	0.0	274.24	7.05	0.86	7.05	7.05	0.0	40.90
	75	12.2	0.43	12.2	12.2	0.0	211.72	6.1	0.92	6.1	6.1	0.0	4.56
	100	12.2	0.42	12.2	12.2	0.0	268.78	3.05	0.94	3.05	3.05	0.0	18.22
S3.17	50	12.2	0.61	13.7	13.7	0.0	138.25	6.75	0.92	6.75	6.75	0.0	68.03
	75	9.8	0.46	13.7	13.7	0.0	198.66	3.267	1.01	4.733	4.733	0.0	28.75
	100	9.8	0.47	13.7	13.7	0.0	131.26	2.45	0.98	3.55	3.55	0.0	14.25
S3.18	50	19.9	0.42	20.9	20.9	0.0	275.58	8.933	1.02	10.5	10.5	0.0	55.57
	75	19.6	0.45	20.9	21.049	0.007	600.00	6.633	1.12	6.633	6.633	0.0	5.36
	100	19.6	0.44	20.9	20.9	0.0	441.58	5.6	1.02	6.175	6.175	0.0	1.84
S3.19	50	13.7	0.42	13.7	13.7	0.0	467.59	7.867	0.87	7.867	7.867	0.0	123.50
	75	13.7	0.44	13.7	13.7	0.0	344.19	4.567	0.99	4.567	4.567	0.0	28.02
	100	13.7	0.42	13.7	13.7	0.0	121.68	3.425	1.02	3.425	3.425	0.0	16.23
S3.20	50	6.6	0.43	6.6	6.6	0.0	69.82	3.25	0.75	3.25	3.25	0.0	8.20
	75	6.6	0.43	6.6	6.6	0.0	19.81	2.933	0.84	2.933	2.933	0.0	3.59
	100	6.6	0.42	6.6	6.6	0.0	42.88	1.65	0.87	1.65	1.65	0.0	3.80
S3.21	50	15.6	0.39	15.6	15.6	0.0	237.19	6.6	0.79	6.85	6.85	0.0	54.52
	75	13.3	0.40	13.3	13.3	0.0	183.58	4.15	0.84	4.15	4.15	0.0	5.41

Table C.3 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	13.3	0.41	13.3	13.3	0.0	125.35	3.9	0.78	3.9	3.9	0.0	0.50
S3.22	50	12.6	0.43	12.6	12.6	0.0	332.59	5.133	0.84	5.133	5.133	0.0	42.42
	75	12.6	0.43	12.6	12.6	0.0	402.21	4.2	0.86	4.2	4.2	0.0	34.30
	100	12.6	0.42	12.6	12.6	0.0	401.15	3.15	0.91	3.15	3.15	0.0	1.62
S3.23	50	10.2	0.39	10.2	10.2	0.0	129.70	4.35	0.82	4.35	4.35	0.0	45.42
	75	10.2	0.40	10.2	10.2	0.0	99.99	3.05	0.86	3.05	3.05	0.0	10.28
	100	10.2	0.40	10.2	10.2	0.0	104.84	2.55	0.87	2.55	2.55	0.0	12.52
S3.24	50	10.9	0.41	12.3	14.729	0.197	600.00	4.8	0.74	4.8	4.8	0.0	22.50
	75	10.9	0.42	12.3	12.3	0.0	268.61	3.15	0.83	3.15	3.15	0.0	36.83
	100	10.9	0.41	12.3	12.3	0.0	432.27	2.725	0.94	3.075	3.075	0.0	44.78
S3.25	50	17.6	0.55	17.6	24.6	0.398	600.00	7.667	0.97	9.0	9.0	0.0	96.96
	75	16.9	0.46	17.6	17.6	0.0	595.50	6.533	1.05	7.367	7.367	0.0	28.93
	100	16.9	0.44	17.6	17.6	0.0	259.44	4.225	1.07	5.175	5.175	0.0	8.78
S3.26	50	14.0	0.45	14.0	14.0	0.0	336.79	6.55	0.94	6.55	6.55	0.0	74.53
	75	14.0	0.46	14.0	14.0	0.0	387.50	4.667	1.02	4.667	4.667	0.0	73.53
	100	14.0	0.44	14.0	14.0	0.0	370.48	4.667	1.06	4.667	4.667	0.0	1.29
S3.27	50	11.7	0.46	11.7	11.7	0.0	369.65	9.3	0.88	9.3	9.3	0.0	35.75
	75	11.7	0.48	11.7	11.7	0.0	538.43	4.05	1.02	4.05	4.05	0.0	42.81
	100	11.7	0.46	11.7	20.5	0.752	600.00	2.925	1.09	2.925	2.925	0.0	41.51
S3.28	50	7.1	0.35	7.1	7.1	0.0	37.96	3.35	0.61	3.35	3.35	0.0	4.73
	75	7.1	0.35	7.1	7.1	0.0	30.45	1.775	0.71	1.775	1.775	0.0	4.33
	100	7.1	0.36	7.1	7.1	0.0	26.96	1.775	0.76	1.775	1.775	0.0	0.55
S3.29	50	11.2	0.45	11.2	11.2	0.0	145.68	5.6	0.91	5.6	5.6	0.0	54.55
	75	11.2	0.44	11.2	11.2	0.0	312.84	3.4	1.00	3.4	3.4	0.0	24.12
	100	11.2	0.44	11.2	11.2	0.0	277.14	2.35	0.99	2.35	2.35	0.0	12.73
S3.30	50	18.4	0.45	18.4	24.3	0.321	600.00	8.933	0.97	10.1	10.5	0.04	600.00
	75	18.4	0.46	18.4	34.841	0.894	600.00	8.0	1.01	8.7	8.7	0.0	239.10
	100	18.4	0.45	20.7	22.2	0.072	600.00	6.133	1.04	6.967	6.967	0.0	34.40
S3.31	50	18.4	0.68	18.4	46.511	1.528	600.00	7.667	0.98	8.1	8.1	0.0	326.00
	75	15.7	0.48	15.7	43.4	1.764	600.00	5.267	1.09	6.4	6.4	0.0	31.18
	100	15.7	0.46	15.7	33.4	1.127	600.00	5.133	1.10	5.133	5.133	0.0	1.95
S3.32	50	14.0	0.50	14.0	14.0	0.0	229.27	6.9	0.99	6.9	6.9	0.0	27.76
	75	14.0	0.51	14.0	14.0	0.0	315.71	4.667	1.17	4.667	4.667	0.0	8.06
	100	14.0	0.48	14.0	14.0	0.0	286.88	3.5	1.16	3.5	3.5	0.0	1.96

Table C.3 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S3.33	50	13.8	0.48	13.8	13.8	0.0	247.90	6.367	0.94	6.367	6.367	0.0	101.10
	75	12.7	0.58	12.7	12.7	0.0	244.42	3.767	1.26	3.767	3.767	0.0	34.16
	100	12.7	0.57	12.7	12.7	0.0	182.59	3.175	1.15	3.175	3.175	0.0	44.10
S3.34	50	5.7	0.39	6.2	6.2	0.0	69.09	2.467	0.63	2.467	2.467	0.0	3.00
	75	5.7	0.39	6.2	6.2	0.0	46.49	2.0	0.73	2.0	2.0	0.0	0.44
	100	5.7	0.39	6.2	6.2	0.0	52.51	1.425	0.78	1.55	1.55	0.0	1.45
S3.35	50	9.5	0.41	9.5	9.5	0.0	169.18	6.033	0.78	6.067	6.067	0.0	27.84
	75	9.5	0.42	9.5	9.5	0.0	179.51	4.0	0.86	4.0	4.0	0.0	8.88
	100	9.5	0.42	9.5	9.5	0.0	142.92	2.375	0.93	2.375	2.375	0.0	8.27
S3.36	50	8.3	0.41	8.3	8.3	0.0	35.72	4.15	0.75	4.15	4.15	0.0	83.48
	75	8.3	0.42	8.3	8.3	0.0	88.06	3.833	0.83	3.833	3.833	0.0	8.64
	100	8.3	0.42	8.3	8.3	0.0	87.39	2.075	0.86	2.075	2.075	0.0	3.65
S3.37	50	18.3	0.49	18.3	35.746	0.953	600.00	11.1	1.07	11.1	17.5	0.577	600.00
	75	18.3	0.50	18.3	41.439	1.264	600.00	7.667	1.34	7.767	7.767	0.0	152.70
	100	18.3	0.47	18.3	43.0	1.35	600.00	6.1	1.10	6.1	6.1	0.0	24.59
S3.38	50	17.4	0.48	17.4	27.861	0.601	600.00	8.7	1.10	8.7	9.7	0.115	600.00
	75	17.4	0.48	17.4	17.7	0.017	600.00	7.1	1.13	7.833	7.833	0.0	16.13
	100	17.4	0.46	17.4	17.7	0.017	600.00	4.95	1.19	4.95	4.95	0.0	2.08
S3.39	50	18.0	0.46	18.0	32.0	0.778	600.00	8.8	1.13	8.8	8.8	0.0	130.40
	75	18.0	0.45	18.0	30.6	0.7	600.00	6.833	1.13	6.833	6.833	0.0	62.77
	100	18.0	0.44	18.0	32.0	0.778	600.00	6.0	1.09	6.0	6.0	0.0	2.87
S3.40	50	15.7	0.44	15.7	15.7	0.0	253.86	7.85	0.82	7.85	7.85	0.0	83.86
	75	15.7	0.44	15.7	16.9	0.076	600.00	4.675	0.97	4.675	4.675	0.0	20.45
	100	15.7	0.43	15.7	20.4	0.299	600.00	4.675	0.92	4.675	4.675	0.0	1.28
S3.41	50	7.6	0.42	7.6	9.8	0.289	600.00	3.0	0.80	3.0	3.0	0.0	34.40
	75	7.6	0.43	7.6	7.6	0.0	363.55	2.8	0.88	2.9	2.9	0.0	6.66
	100	7.6	0.42	7.6	7.6	0.0	254.77	1.9	0.96	1.9	1.9	0.0	6.94
S3.42	50	13.9	0.42	13.9	13.9	0.0	262.24	8.1	0.82	8.1	8.1	0.0	13.11
	75	13.9	0.42	13.9	13.9	0.0	148.31	5.567	1.13	5.8	5.8	0.0	3.13
	100	13.9	0.39	13.9	13.9	0.0	166.30	4.633	0.95	4.633	4.633	0.0	0.94
S3.43	50	8.0	0.41	8.7	8.7	0.0	96.35	3.8	0.74	4.0	4.0	0.0	52.51
	75	8.0	0.40	8.7	8.7	0.0	115.69	2.433	0.81	2.433	2.433	0.0	5.63
	100	8.0	0.40	8.7	8.7	0.0	78.33	2.0	0.88	2.175	2.175	0.0	25.90
S3.44	50	8.6	0.41	9.6	9.6	0.0	224.32	3.533	0.70	3.533	3.533	0.0	18.61

Table C.3 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	8.6	0.40	9.6	9.6	0.0	305.42	2.65	0.82	2.65	2.65	0.0	57.86
	100	8.6	0.41	9.6	9.6	0.0	195.38	2.15	0.83	2.4	2.4	0.0	14.65
S3.45	50	10.6	0.45	11.0	17.0	0.545	600.00	8.3	0.79	8.7	8.7	0.0	6.70
	75	10.6	0.43	10.6	11.1	0.047	600.00	3.533	0.90	3.533	3.533	0.0	21.81
	100	10.6	0.41	10.6	16.748	0.58	600.00	2.65	0.95	2.75	2.75	0.0	66.23
S3.46	50	8.6	0.40	8.9	8.9	0.0	67.86	3.85	0.74	4.133	4.133	0.0	21.11
	75	8.6	0.41	8.9	8.9	0.0	73.96	2.967	0.87	2.967	2.967	0.0	12.01
	100	8.6	0.40	8.9	8.9	0.0	97.75	2.15	0.90	2.225	2.225	0.0	6.46
S3.47	50	8.5	0.36	8.5	8.5	0.0	152.63	2.833	0.84	2.833	2.833	0.0	24.58
	75	8.5	0.36	8.5	8.5	0.0	114.37	2.175	0.76	2.175	2.175	0.0	11.75
	100	8.5	0.36	8.5	8.5	0.0	18.03	2.125	0.73	2.125	2.125	0.0	6.43
S3.48	50	15.8	0.41	15.8	15.8	0.0	201.59	6.9	0.91	8.6	8.6	0.0	30.10
	75	15.8	0.39	15.8	15.8	0.0	577.73	4.45	0.97	4.45	4.45	0.0	72.95
	100	15.8	0.39	15.8	15.8	0.0	294.04	3.95	0.93	3.95	3.95	0.0	1.32
S3.49	50	7.3	0.40	7.3	7.3	0.0	99.08	4.55	0.67	4.8	4.8	0.0	3.86
	75	6.2	0.41	7.1	7.1	0.0	78.34	2.85	0.82	2.85	2.85	0.0	5.17
	100	6.2	0.40	7.1	7.1	0.0	27.71	1.55	0.84	1.775	1.775	0.0	10.30

Table C.4: **Series 1. Example of table for metrics comparison; $r_{\max} = 3$.**

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S1.0	50	34.7	0.11	34.7	34.7	0.0	71.10	22.3	0.28	22.3	22.3	0.0	1.64
	75	34.7	0.11	34.7	34.7	0.0	136.86	19.8	0.38	19.8	19.8	0.0	0.72
	100	34.7	0.09	34.7	34.7	0.0	70.31	17.35	0.22	17.35	17.35	0.0	0.02
S1.1	50	36.0	0.12	38.6	38.6	0.0	37.53	21.35	0.41	22.25	22.25	0.0	3.44
	75	35.7	0.12	37.0	37.0	0.0	63.84	18.35	0.45	19.4	19.4	0.0	4.52
	100	35.8	0.16	37.0	37.0	0.0	47.42	17.85	0.35	18.5	18.5	0.0	0.38
S1.2	50	36.5	0.12	38.0	38.0	0.0	78.83	29.3	0.36	31.5	31.5	0.0	17.13
	75	36.5	0.12	36.7	36.7	0.0	84.54	20.95	0.43	21.1	21.1	0.0	3.16
	100	36.5	0.11	36.7	36.7	0.0	20.08	18.25	0.31	18.35	18.35	0.0	0.04
S1.3	50	41.2	0.13	41.2	41.2	0.0	82.63	36.0	0.39	36.0	36.0	0.0	30.33
	75	40.5	0.18	41.1	41.1	0.0	93.14	22.4	0.62	22.4	22.4	0.0	14.90
	100	40.4	0.17	41.1	41.1	0.0	62.03	20.2	0.54	20.55	20.55	0.0	0.66
S1.4	50	36.8	0.12	37.2	37.2	0.0	81.76	32.9	0.47	33.1	33.1	0.0	2.04
	75	36.0	0.10	36.0	36.0	0.0	23.62	18.75	0.35	19.45	19.45	0.0	1.66
	100	34.8	0.10	35.3	35.3	0.0	52.91	17.4	0.30	17.65	17.65	0.0	0.10
S1.5	50	39.7	0.12	40.2	40.2	0.0	75.61	32.7	0.35	38.9	38.9	0.0	2.62
	75	39.7	0.14	40.2	40.2	0.0	90.99	20.45	0.40	20.45	20.45	0.0	2.31
	100	39.7	0.13	40.2	40.2	0.0	42.73	19.85	0.38	20.1	20.1	0.0	0.34
S1.6	50	37.8	0.10	37.8	37.8	0.0	51.34	28.6	0.32	30.3	30.3	0.0	10.25
	75	36.1	0.10	36.1	36.1	0.0	64.46	18.05	0.33	18.05	18.05	0.0	1.15
	100	36.1	0.11	36.1	36.1	0.0	101.34	18.05	0.22	18.05	18.05	0.0	0.02
S1.7	50	41.0	0.15	41.0	41.0	0.0	57.56	36.8	0.39	38.3	38.3	0.0	35.88
	75	40.0	0.17	40.2	40.2	0.0	146.08	20.3	0.54	20.7	20.7	0.0	68.57
	100	40.0	0.17	40.2	40.2	0.0	127.31	19.75	0.46	20.1	20.1	0.0	0.54
S1.8	50	35.5	0.17	39.1	39.1	0.0	110.75	22.7	0.45	28.0	28.0	0.0	15.86
	75	35.1	0.12	35.1	35.1	0.0	229.21	21.05	0.52	21.05	21.05	0.0	1.14
	100	35.1	0.12	35.1	35.1	0.0	78.84	17.55	0.35	17.55	17.55	0.0	0.07
S1.9	50	23.7	0.09	24.4	24.4	0.0	52.96	12.9	0.21	12.9	12.9	0.0	3.64
	75	23.7	0.09	24.4	24.4	0.0	88.56	10.467	0.26	10.467	10.467	0.0	2.82
	100	23.7	0.09	24.4	24.4	0.0	109.48	7.9	0.22	8.133	8.133	0.0	0.08
S1.10	50	30.3	0.11	30.3	30.3	0.0	61.64	25.7	0.29	28.8	28.8	0.0	2.16

Table C.4 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	30.3	0.10	30.3	30.3	0.0	62.86	17.8	0.29	17.8	17.8	0.0	2.91
	100	30.3	0.10	30.3	30.3	0.0	99.47	10.1	0.29	10.1	10.1	0.0	0.17
S1.11	50	26.6	0.09	26.8	26.8	0.0	65.38	13.3	0.25	13.3	13.3	0.0	3.16
	75	26.6	0.09	26.8	26.8	0.0	25.69	10.833	0.28	10.833	10.833	0.0	2.04
	100	26.6	0.09	26.8	26.8	0.0	5.61	8.867	0.23	8.933	8.933	0.0	0.11
S1.12	50	36.9	0.12	37.3	37.3	0.0	88.21	22.55	0.36	22.55	22.55	0.0	4.32
	75	36.5	0.13	36.5	36.5	0.0	49.38	19.2	0.53	19.2	19.2	0.0	2.96
	100	36.2	0.12	36.2	36.2	0.0	104.43	18.1	0.40	18.1	18.1	0.0	0.54
S1.13	50	24.3	0.09	25.7	25.7	0.0	71.39	17.3	0.25	17.3	17.3	0.0	3.77
	75	24.3	0.10	25.7	25.7	0.0	123.98	10.2	0.29	12.15	12.15	0.0	4.26
	100	24.3	0.09	25.7	25.7	0.0	17.37	8.1	0.24	8.567	8.567	0.0	0.51
S1.14	50	40.0	0.12	40.4	40.4	0.0	15.71	23.85	0.34	24.05	24.05	0.0	2.64
	75	36.4	0.15	39.3	39.3	0.0	39.11	20.25	0.33	20.5	20.5	0.0	0.59
	100	35.9	0.11	39.3	39.3	0.0	65.07	17.95	0.29	19.65	19.65	0.0	0.05
S1.15	50	36.5	0.11	36.9	36.9	0.0	72.41	19.85	0.34	19.85	19.85	0.0	2.43
	75	36.5	0.12	36.5	36.5	0.0	102.09	19.85	0.34	19.85	19.85	0.0	0.74
	100	36.5	0.11	36.5	36.5	0.0	101.43	18.25	0.35	18.25	18.25	0.0	0.34
S1.16	50	38.9	0.13	38.9	38.9	0.0	83.19	37.0	0.39	37.0	37.0	0.0	8.27
	75	38.9	0.15	38.9	38.9	0.0	133.79	19.65	0.45	19.65	19.65	0.0	21.59
	100	38.9	0.19	38.9	38.9	0.0	77.90	19.45	0.49	19.45	19.45	0.0	0.24
S1.17	50	36.2	0.11	36.2	36.2	0.0	24.17	36.1	0.33	36.1	36.1	0.0	2.79
	75	36.2	0.11	36.2	36.2	0.0	61.61	18.1	0.36	18.1	18.1	0.0	2.45
	100	36.2	0.11	36.2	36.2	0.0	81.04	18.1	0.29	18.1	18.1	0.0	0.05
S1.18	50	39.8	0.16	39.8	39.8	0.0	89.78	21.85	0.39	21.85	21.85	0.0	40.58
	75	39.2	0.18	39.8	39.8	0.0	113.97	20.55	0.73	20.75	20.75	0.0	4.02
	100	39.8	0.17	39.8	39.8	0.0	52.81	19.9	0.54	19.9	19.9	0.0	0.18
S1.19	50	40.6	0.14	41.3	41.3	0.0	65.12	21.6	0.32	21.6	21.6	0.0	2.82
	75	40.2	0.12	40.6	40.6	0.0	110.34	20.7	0.38	20.7	20.7	0.0	4.42
	100	40.2	0.12	40.6	40.6	0.0	63.09	20.1	0.38	20.3	20.3	0.0	0.10
S1.20	50	37.1	0.17	38.4	38.4	0.0	273.42	35.8	0.44	36.0	36.0	0.0	58.56
	75	37.1	0.17	37.3	37.3	0.0	431.61	19.5	0.63	20.2	20.2	0.0	179.25
	100	37.1	0.16	37.3	37.3	0.0	214.84	18.65	0.59	18.65	18.65	0.0	1.51
S1.21	50	39.1	0.13	40.2	40.2	0.0	36.81	37.6	0.38	38.0	38.0	0.0	45.79
	75	38.8	0.12	38.8	38.8	0.0	92.88	20.1	0.48	20.1	20.1	0.0	5.94

Table C.4 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	38.8	0.12	38.8	38.8	0.0	29.85	19.4	0.42	19.4	19.4	0.0	0.06
S1.22	50	38.6	0.13	38.6	38.6	0.0	83.08	34.3	0.40	34.3	34.3	0.0	7.24
	75	38.1	0.14	38.1	38.1	0.0	126.72	19.3	0.43	19.3	19.3	0.0	1.16
	100	37.8	0.19	37.8	37.8	0.0	123.24	18.7	0.48	18.9	18.9	0.0	0.27
S1.23	50	38.4	0.14	39.1	39.1	0.0	109.19	32.4	0.34	32.4	32.4	0.0	2.50
	75	38.3	0.14	38.4	38.4	0.0	296.27	21.2	0.39	21.35	21.35	0.0	5.68
	100	38.0	0.14	38.3	38.3	0.0	182.83	19.05	0.55	19.15	19.15	0.0	0.78
S1.24	50	39.5	0.12	39.6	39.6	0.0	92.18	34.8	0.43	36.7	36.7	0.0	9.79
	75	39.1	0.14	39.6	39.6	0.0	53.78	20.0	0.42	20.4	20.4	0.0	4.88
	100	38.5	0.12	38.5	38.5	0.0	38.69	19.25	0.46	19.25	19.25	0.0	0.14
S1.25	50	40.5	0.13	40.5	40.5	0.0	77.55	38.3	0.49	38.3	38.3	0.0	2.64
	75	39.8	0.11	40.5	40.5	0.0	106.26	21.35	0.34	21.35	21.35	0.0	1.98
	100	39.8	0.12	40.3	40.3	0.0	60.94	19.9	0.36	20.15	20.15	0.0	0.31
S1.26	50	21.2	0.09	21.3	21.3	0.0	35.65	13.7	0.26	15.85	15.85	0.0	2.25
	75	21.2	0.10	21.3	21.3	0.0	37.60	11.35	0.26	13.25	13.25	0.0	1.44
	100	21.2	0.09	21.3	21.3	0.0	44.14	7.067	0.25	7.1	7.1	0.0	0.27
S1.27	50	39.1	0.12	40.3	40.3	0.0	55.75	34.6	0.41	36.2	36.2	0.0	21.22
	75	37.5	0.13	38.5	38.5	0.0	66.89	21.35	0.50	21.4	21.4	0.0	10.35
	100	37.4	0.15	38.1	38.1	0.0	66.76	18.75	0.52	19.05	19.05	0.0	0.18
S1.28	50	40.7	0.15	40.7	40.7	0.0	107.29	35.8	0.52	37.2	37.2	0.0	15.15
	75	38.0	0.18	38.1	38.1	0.0	100.09	20.35	0.55	20.45	20.45	0.0	18.29
	100	37.8	0.19	38.0	38.0	0.0	112.85	18.9	0.69	19.0	19.0	0.0	0.39
S1.29	50	40.3	0.15	40.8	40.8	0.0	75.29	38.1	0.45	38.6	38.6	0.0	29.33
	75	39.3	0.13	39.3	39.3	0.0	100.16	21.95	0.56	21.95	21.95	0.0	12.21
	100	38.8	0.13	38.8	38.8	0.0	26.74	19.4	0.39	19.4	19.4	0.0	0.09
S1.30	50	39.6	0.11	40.4	40.4	0.0	46.77	33.3	0.34	33.3	33.3	0.0	10.18
	75	36.3	0.12	36.3	36.3	0.0	295.99	19.1	0.39	20.0	20.0	0.0	7.68
	100	36.3	0.12	36.3	36.3	0.0	69.12	18.15	0.34	18.15	18.15	0.0	0.07
S1.31	50	40.4	0.17	40.4	40.4	0.0	73.75	35.6	0.35	35.7	35.7	0.0	3.84
	75	37.8	0.18	38.4	38.4	0.0	129.38	20.2	0.40	20.2	20.2	0.0	0.66
	100	37.3	0.14	37.8	37.8	0.0	82.36	18.9	0.47	18.9	18.9	0.0	0.06
S1.32	50	40.2	0.14	40.2	40.2	0.0	149.98	32.8	0.43	34.3	34.3	0.0	34.83
	75	37.5	0.13	37.5	37.5	0.0	492.48	20.45	0.51	20.45	20.45	0.0	11.69
	100	37.5	0.12	37.5	37.5	0.0	164.66	18.75	0.45	18.75	18.75	0.0	0.93

Table C.4 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S1.33	50	34.3	0.12	34.3	34.3	0.0	49.74	25.1	0.33	28.1	28.1	0.0	4.09
	75	34.1	0.11	34.1	34.1	0.0	59.42	20.2	0.64	20.45	20.45	0.0	1.78
	100	34.1	0.11	34.1	34.1	0.0	92.88	17.05	0.30	17.05	17.05	0.0	0.05
S1.34	50	40.5	0.17	40.5	40.5	0.0	69.93	36.1	0.43	37.1	37.1	0.0	3.06
	75	38.2	0.13	39.0	39.0	0.0	184.79	20.25	0.58	21.4	21.4	0.0	10.02
	100	38.1	0.16	38.3	38.3	0.0	133.86	18.85	0.52	19.15	19.15	0.0	1.73
S1.35	50	35.3	0.10	35.3	35.3	0.0	31.96	19.55	0.24	22.8	22.8	0.0	16.83
	75	34.9	0.10	35.3	35.3	0.0	95.33	15.75	0.27	15.75	15.75	0.0	2.03
	100	34.9	0.10	35.3	35.3	0.0	35.79	11.9	0.28	11.9	11.9	0.0	0.18
S1.36	50	33.6	0.10	38.9	38.9	0.0	46.97	20.4	0.28	28.5	28.5	0.0	17.10
	75	33.6	0.10	34.3	34.3	0.0	257.33	19.2	0.33	19.2	19.2	0.0	1.84
	100	33.6	0.10	34.3	34.3	0.0	75.43	16.8	0.23	17.15	17.15	0.0	0.03
S1.37	50	29.4	0.10	31.6	31.6	0.0	63.26	18.9	0.26	20.9	20.9	0.0	17.07
	75	29.4	0.10	31.6	31.6	0.0	66.24	14.7	0.30	14.7	14.7	0.0	1.43
	100	29.4	0.10	31.6	31.6	0.0	47.51	9.8	0.28	11.333	11.333	0.0	0.26
S1.38	50	30.8	0.09	30.8	30.8	0.0	45.35	12.0	0.27	15.8	15.8	0.0	21.95
	75	30.5	0.10	30.8	30.8	0.0	36.05	12.0	0.30	15.4	15.4	0.0	23.73
	100	30.5	0.11	30.8	30.8	0.0	39.73	10.167	0.26	10.267	10.267	0.0	0.16
S1.39	50	36.8	0.12	36.8	36.8	0.0	64.88	18.75	0.35	18.75	18.75	0.0	19.43
	75	36.8	0.12	36.8	36.8	0.0	99.74	18.75	0.39	18.75	18.75	0.0	1.53
	100	36.8	0.12	36.8	36.8	0.0	61.44	18.4	0.33	18.4	18.4	0.0	0.10
S1.40	50	36.9	0.12	36.9	36.9	0.0	62.41	19.0	0.44	33.0	33.0	0.0	13.41
	75	36.1	0.13	36.9	36.9	0.0	117.54	18.05	0.53	18.55	18.55	0.0	13.11
	100	36.1	0.12	36.9	36.9	0.0	98.39	18.05	0.34	18.45	18.45	0.0	0.06
S1.41	50	36.1	0.10	36.1	36.1	0.0	67.47	19.1	0.30	22.35	22.35	0.0	3.52
	75	35.9	0.11	36.1	36.1	0.0	72.19	18.05	0.37	18.05	18.05	0.0	7.26
	100	35.9	0.10	36.1	36.1	0.0	65.59	17.95	0.27	18.05	18.05	0.0	0.02
S1.42	50	39.1	0.14	39.4	39.4	0.0	72.02	35.8	0.41	36.4	36.4	0.0	54.55
	75	38.7	0.14	39.4	39.4	0.0	71.31	20.35	0.66	20.35	20.35	0.0	12.24
	100	38.7	0.18	39.3	39.3	0.0	92.61	19.35	0.62	19.65	19.65	0.0	0.30
S1.43	50	39.9	0.15	40.8	40.8	0.0	72.03	37.0	0.51	39.8	39.8	0.0	7.04
	75	39.8	0.21	39.8	39.8	0.0	103.10	20.65	0.71	20.65	20.65	0.0	19.71
	100	39.1	0.18	39.8	39.8	0.0	88.87	19.75	0.79	19.9	19.9	0.0	1.14
S1.44	50	39.3	0.14	40.1	40.1	0.0	85.09	35.5	0.43	35.5	35.5	0.0	10.36

Table C.4 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	38.3	0.23	38.4	38.4	0.0	88.47	19.65	0.49	19.95	19.95	0.0	29.48
	100	38.3	0.23	38.4	38.4	0.0	105.05	19.2	0.75	19.2	19.2	0.0	0.60
S1.45	50	34.3	0.12	34.3	34.3	0.0	40.88	34.1	0.31	34.1	34.1	0.0	2.05
	75	34.3	0.13	34.3	34.3	0.0	55.15	17.25	0.41	17.4	17.4	0.0	6.87
	100	34.3	0.12	34.3	34.3	0.0	50.70	17.15	0.36	17.15	17.15	0.0	0.25
S1.46	50	36.5	0.11	36.5	36.5	0.0	59.30	21.25	0.41	30.9	30.9	0.0	4.00
	75	34.8	0.11	34.8	34.8	0.0	61.92	17.4	0.34	17.4	17.4	0.0	1.43
	100	34.8	0.10	34.8	34.8	0.0	40.69	17.4	0.29	17.4	17.4	0.0	0.06
S1.47	50	35.8	0.12	36.6	36.6	0.0	60.26	32.6	0.35	32.6	32.6	0.0	3.15
	75	35.6	0.12	36.2	36.2	0.0	89.10	18.9	0.52	19.95	19.95	0.0	10.71
	100	35.6	0.11	36.2	36.2	0.0	82.24	17.8	0.33	18.1	18.1	0.0	0.33
S1.48	50	37.3	0.14	37.3	37.3	0.0	66.65	34.1	0.32	34.1	34.1	0.0	3.17
	75	37.3	0.12	37.3	37.3	0.0	65.71	19.85	0.38	20.15	20.15	0.0	2.02
	100	37.3	0.12	37.3	37.3	0.0	45.27	18.65	0.35	18.65	18.65	0.0	0.10
S1.49	50	37.4	0.12	37.4	37.4	0.0	217.06	22.5	0.32	27.0	27.0	0.0	3.27
	75	35.9	0.12	36.7	36.7	0.0	235.36	20.45	0.49	20.45	20.45	0.0	3.15
	100	35.9	0.12	36.2	36.2	0.0	78.36	17.95	0.34	18.1	18.1	0.0	0.48

Table C.5: **Series 2. Example of table for metrics comparison; $r_{\max} = 3$.**

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S2.0	50	43.3	0.15	45.1	50.2	0.113	600.00	41.0	0.70	41.0	49.934	0.218	600.00
	75	41.1	0.18	41.7	50.2	0.204	600.00	23.05	0.74	23.15	50.2	1.168	600.00
	100	40.7	0.19	41.5	41.5	0.0	566.87	20.25	0.85	20.75	20.75	0.0	5.19
S2.1	50	41.3	0.25	42.9	47.9	0.117	600.00	36.0	0.70	37.3	50.2	0.346	600.00
	75	37.9	0.28	38.6	47.9	0.241	600.00	21.3	0.80	21.75	39.3	0.807	600.00
	100	37.5	0.24	38.3	43.0	0.123	600.00	18.8	1.06	19.15	19.15	0.0	31.41
S2.2	50	41.9	0.20	42.2	42.2	0.0	68.33	38.0	0.46	40.8	40.8	0.0	17.08
	75	40.3	0.20	40.8	40.8	0.0	72.82	22.1	0.83	22.45	22.45	0.0	85.15
	100	40.3	0.18	40.7	40.7	0.0	93.76	20.05	0.62	20.35	20.35	0.0	1.57
S2.3	50	43.6	0.26	44.6	51.7	0.159	600.00	40.4	0.92	41.5	51.4	0.239	600.00
	75	41.5	0.20	42.2	51.7	0.225	600.00	22.05	0.73	23.2	51.4	1.216	600.00
	100	40.9	0.17	41.3	51.4	0.245	600.00	20.45	0.76	20.65	20.65	0.0	117.76
S2.4	50	42.6	0.20	45.0	52.7	0.171	600.00	36.8	0.71	38.8	50.5	0.302	600.00
	75	38.8	0.17	40.1	50.1	0.249	600.00	21.5	1.16	22.35	50.1	1.242	600.00
	100	38.0	0.17	38.8	50.1	0.291	600.00	18.95	0.88	19.4	20.55	0.059	600.00
S2.5	50	42.7	0.16	46.0	46.0	0.0	195.54	38.9	0.61	41.1	41.1	0.0	496.42
	75	41.1	0.20	41.3	46.6	0.128	600.00	22.9	0.62	23.25	49.411	1.125	600.00
	100	40.7	0.17	41.1	41.8	0.017	600.00	20.35	0.80	20.55	20.55	0.0	16.89
S2.6	50	35.4	0.11	35.4	35.4	0.0	58.21	35.4	0.42	35.4	35.4	0.0	1.19
	75	35.4	0.10	35.4	35.4	0.0	18.40	17.7	0.40	17.7	17.7	0.0	10.89
	100	35.4	0.10	35.4	35.4	0.0	34.19	17.7	0.33	17.7	17.7	0.0	0.04
S2.7	50	45.8	0.16	46.4	46.4	0.0	389.08	39.4	0.65	41.5	41.5	0.0	359.57
	75	41.4	0.19	41.7	52.5	0.259	600.00	22.7	0.90	22.9	50.6	1.21	600.00
	100	41.3	0.15	41.7	52.2	0.252	600.00	20.6	0.96	20.9	20.9	0.0	22.44
S2.8	50	41.5	0.18	41.5	41.5	0.0	84.08	36.9	0.89	40.4	40.4	0.0	11.49
	75	40.4	0.25	40.8	40.8	0.0	114.16	21.05	0.74	21.5	35.1	0.633	600.00
	100	40.5	0.17	40.8	40.8	0.0	73.44	19.95	0.79	20.4	20.4	0.0	2.28
S2.9	50	43.0	0.14	43.7	43.7	0.0	73.09	38.3	0.57	40.0	40.0	0.0	33.67
	75	41.0	0.15	41.3	41.6	0.007	600.00	22.45	0.68	22.8	22.8	0.0	133.57
	100	41.1	0.17	41.3	41.3	0.0	120.99	20.6	0.63	20.65	20.65	0.0	2.06
S2.10	50	43.0	0.26	44.7	53.0	0.186	600.00	39.6	0.65	40.3	51.6	0.28	600.00

Table C.5 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	39.9	0.17	40.0	57.0	0.425	600.00	21.2	0.75	21.8	50.2	1.303	600.00
	100	39.9	0.17	40.2	57.0	0.418	600.00	20.05	1.05	20.65	20.65	0.0	101.49
S2.11	50	41.8	0.15	44.9	44.9	0.0	76.69	35.7	0.52	38.8	51.7	0.332	600.00
	75	40.4	0.20	41.2	50.9	0.235	600.00	21.25	0.64	22.8	50.9	1.232	600.00
	100	40.1	0.19	40.6	40.6	0.0	478.00	20.05	0.93	20.3	20.3	0.0	4.56
S2.12	50	41.9	0.27	43.5	51.3	0.179	600.00	37.3	0.80	39.4	50.9	0.292	600.00
	75	38.4	0.21	39.4	52.7	0.338	600.00	21.35	1.06	21.6	50.9	1.356	600.00
	100	38.2	0.19	39.0	52.2	0.338	600.00	19.05	0.81	19.7	19.9	0.01	600.00
S2.13	50	39.1	0.14	39.9	39.9	0.0	150.28	36.0	0.60	37.6	37.6	0.0	22.15
	75	39.0	0.21	39.2	39.2	0.0	571.60	20.6	0.57	21.35	21.35	0.0	313.96
	100	38.3	0.13	39.2	39.2	0.0	151.30	19.05	0.61	19.6	19.6	0.0	2.68
S2.14	50	43.1	0.22	43.6	50.1	0.149	600.00	39.9	0.78	41.3	42.3	0.024	600.00
	75	40.3	0.20	41.3	50.1	0.213	600.00	22.65	0.71	23.55	50.1	1.127	600.00
	100	40.5	0.17	41.1	50.1	0.219	600.00	20.25	0.81	20.55	20.55	0.0	28.36
S2.15	50	39.7	0.17	40.3	40.3	0.0	179.42	35.4	0.54	36.6	36.6	0.0	206.63
	75	37.7	0.19	38.1	38.1	0.0	82.33	20.8	0.66	21.2	38.1	0.797	600.00
	100	37.7	0.23	38.1	38.1	0.0	70.73	18.9	0.75	19.05	19.05	0.0	0.39
S2.16	50	44.6	0.20	44.6	44.6	0.0	216.96	38.1	0.81	39.9	39.9	0.0	480.93
	75	39.9	0.22	40.5	44.5	0.099	600.00	21.3	0.70	21.55	24.7	0.146	600.00
	100	39.7	0.17	40.2	44.5	0.107	600.00	19.65	0.81	20.1	20.1	0.0	35.62
S2.17	50	40.5	0.18	40.7	40.7	0.0	299.29	36.1	0.49	38.1	38.1	0.0	43.59
	75	39.9	0.23	40.3	40.7	0.01	600.00	20.6	0.59	21.65	21.65	0.0	251.91
	100	39.9	0.15	40.3	40.8	0.012	600.00	19.9	0.59	20.15	20.15	0.0	6.05
S2.18	50	40.9	0.20	41.2	41.2	0.0	138.30	40.3	0.72	40.6	40.6	0.0	117.36
	75	40.4	0.24	40.9	50.6	0.237	600.00	21.8	0.82	22.45	50.6	1.254	600.00
	100	40.4	0.20	41.1	41.1	0.0	222.75	20.2	0.81	20.55	20.55	0.0	5.83
S2.19	50	47.4	0.15	47.6	47.6	0.0	96.71	39.4	0.54	40.4	40.4	0.0	244.56
	75	40.7	0.17	41.6	41.6	0.0	469.39	22.15	0.71	22.35	40.8	0.826	600.00
	100	40.6	0.14	41.4	41.4	0.0	96.75	20.25	0.52	20.7	20.7	0.0	1.84
S2.20	50	46.7	0.19	46.8	46.8	0.0	488.97	37.7	0.64	39.2	39.2	0.0	271.73
	75	39.0	0.20	39.0	51.1	0.31	600.00	21.6	0.79	22.05	23.3	0.057	600.00
	100	38.8	0.20	39.2	57.6	0.469	600.00	19.4	0.72	19.7	19.7	0.0	87.44
S2.21	50	43.6	0.22	43.9	52.0	0.185	600.00	38.9	0.70	39.6	51.2	0.293	600.00
	75	40.9	0.18	41.3	41.3	0.0	33.57	21.95	0.77	22.45	22.45	0.0	464.21

Table C.5 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	40.2	0.21	41.2	41.2	0.0	98.96	20.2	0.90	20.6	20.6	0.0	0.69
S2.22	50	41.6	0.20	41.6	50.4	0.212	600.00	36.2	0.84	38.5	50.4	0.309	600.00
	75	37.8	0.21	38.4	50.4	0.313	600.00	20.75	0.87	21.25	50.4	1.372	600.00
	100	37.8	0.20	37.8	50.4	0.333	600.00	18.9	1.07	19.2	19.2	0.0	383.12
S2.23	50	39.5	0.17	40.5	40.5	0.0	98.93	34.7	0.66	34.7	34.7	0.0	35.08
	75	39.0	0.14	40.5	40.5	0.0	206.69	21.35	0.58	21.75	21.75	0.0	23.56
	100	39.0	0.13	40.4	40.4	0.0	234.02	19.5	0.61	20.2	20.2	0.0	2.96
S2.24	50	44.7	0.24	44.7	56.1	0.255	600.00	37.2	0.90	39.5	50.8	0.286	600.00
	75	39.5	0.17	40.2	57.3	0.425	600.00	22.15	1.50	22.6	50.8	1.248	600.00
	100	39.3	0.17	40.0	53.0	0.325	600.00	19.7	0.93	20.1	20.4	0.015	600.00
S2.25	50	44.2	0.15	44.2	45.0	0.018	600.00	39.3	0.57	41.8	41.8	0.0	233.19
	75	40.7	0.16	42.0	42.6	0.014	600.00	22.0	0.75	22.45	51.1	1.276	600.00
	100	40.6	0.14	40.9	52.13	0.275	600.00	20.3	0.63	20.85	20.85	0.0	13.93
S2.26	50	40.9	0.14	40.9	40.9	0.0	87.05	24.55	0.42	24.55	24.55	0.0	6.81
	75	38.4	0.18	38.4	40.9	0.065	600.00	20.8	0.67	21.7	21.7	0.0	7.20
	100	36.9	0.14	38.4	40.9	0.065	600.00	18.6	0.82	19.2	19.2	0.0	6.02
S2.27	50	41.7	0.19	44.1	44.1	0.0	88.93	36.4	0.65	38.5	38.5	0.0	4.09
	75	38.9	0.17	39.1	39.1	0.0	39.13	21.15	0.67	21.9	21.9	0.0	17.58
	100	38.1	0.20	39.1	39.1	0.0	49.74	19.05	0.78	19.55	19.55	0.0	0.15
S2.28	50	46.1	0.21	46.1	50.1	0.087	600.00	37.2	0.81	39.3	50.1	0.275	600.00
	75	38.4	0.20	39.0	56.1	0.438	600.00	21.3	0.89	21.7	50.1	1.309	600.00
	100	38.5	0.19	39.4	45.1	0.145	600.00	19.25	0.77	19.7	19.7	0.0	28.09
S2.29	50	42.5	0.16	42.6	50.2	0.178	600.00	37.1	0.59	39.8	50.2	0.261	600.00
	75	40.0	0.16	41.1	50.5	0.229	600.00	21.25	0.66	21.65	50.2	1.319	600.00
	100	40.2	0.21	40.7	40.7	0.0	488.71	19.7	0.83	20.35	20.35	0.0	10.44
S2.30	50	43.1	0.23	43.5	50.3	0.156	600.00	37.2	0.59	39.7	50.3	0.267	600.00
	75	39.6	0.19	39.8	50.3	0.264	600.00	21.8	0.91	22.55	50.3	1.231	600.00
	100	39.5	0.24	39.6	50.3	0.27	600.00	19.75	1.08	19.9	20.4	0.025	600.00
S2.31	50	43.0	0.18	43.0	43.0	0.0	90.73	37.8	0.45	37.8	37.8	0.0	4.65
	75	36.3	0.14	36.3	36.3	0.0	101.27	18.75	0.46	18.75	18.75	0.0	2.88
	100	36.3	0.14	36.3	36.3	0.0	60.67	18.15	0.47	18.15	18.15	0.0	0.07
S2.32	50	40.6	0.20	42.2	42.2	0.0	157.25	37.6	0.75	40.0	48.045	0.201	600.00
	75	39.7	0.18	40.3	43.7	0.084	600.00	20.45	0.95	21.0	50.3	1.395	600.00
	100	39.4	0.19	40.3	42.2	0.047	600.00	19.7	0.96	20.2	20.2	0.0	14.38

Table C.5 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S2.33	50	46.5	0.19	48.2	48.2	0.0	534.18	38.8	0.72	40.8	42.6	0.044	600.00
	75	40.3	0.23	40.8	50.1	0.228	600.00	22.5	0.99	22.6	50.1	1.217	600.00
	100	40.2	0.25	40.8	50.1	0.228	600.00	20.05	0.97	20.4	20.4	0.0	18.32
S2.34	50	41.0	0.19	42.8	42.8	0.0	195.20	36.1	0.58	38.9	38.9	0.0	31.98
	75	38.4	0.22	39.3	47.2	0.201	600.00	21.2	0.91	21.8	21.8	0.0	95.00
	100	38.4	0.28	39.5	45.6	0.154	600.00	19.05	1.07	19.75	19.75	0.0	35.81
S2.35	50	42.3	0.21	42.3	42.3	0.0	102.36	24.8	0.42	24.8	24.8	0.0	3.58
	75	41.4	0.14	42.0	42.0	0.0	177.96	22.75	0.75	23.05	23.05	0.0	15.04
	100	40.4	0.16	42.0	42.0	0.0	87.43	20.2	0.69	21.0	21.0	0.0	1.01
S2.36	50	40.4	0.19	40.9	40.9	0.0	98.91	35.6	0.58	37.1	37.1	0.0	21.81
	75	38.4	0.16	38.8	38.8	0.0	91.26	21.8	0.51	22.35	22.35	0.0	15.66
	100	37.3	0.15	38.5	38.5	0.0	78.39	19.2	0.52	19.25	19.25	0.0	0.75
S2.37	50	40.1	0.18	41.3	41.3	0.0	113.60	37.3	0.76	39.4	39.4	0.0	393.66
	75	38.4	0.25	39.2	40.9	0.043	600.00	20.35	0.67	21.7	33.1	0.525	600.00
	100	38.4	0.21	39.1	40.3	0.031	600.00	19.2	0.66	19.55	19.55	0.0	31.44
S2.38	50	48.4	0.17	48.4	50.5	0.043	600.00	39.1	0.94	40.7	50.5	0.241	600.00
	75	40.5	0.20	41.0	50.9	0.241	600.00	22.65	1.20	23.1	50.5	1.186	600.00
	100	40.5	0.17	40.7	50.3	0.236	600.00	20.15	1.00	20.45	20.45	0.0	29.18
S2.39	50	48.4	0.22	48.4	51.4	0.062	600.00	39.9	1.04	41.1	50.1	0.219	600.00
	75	40.2	0.22	40.9	51.4	0.257	600.00	22.45	1.35	22.5	50.1	1.227	600.00
	100	40.2	0.22	40.9	50.2	0.227	600.00	20.05	1.02	20.45	20.45	0.0	200.80
S2.40	50	39.1	0.22	39.1	50.1	0.281	600.00	35.9	0.58	38.3	38.3	0.0	426.06
	75	38.5	0.20	38.5	38.5	0.0	327.88	19.75	0.72	20.0	40.1	1.005	600.00
	100	38.5	0.16	38.5	40.1	0.042	600.00	19.25	0.69	19.25	19.25	0.0	7.85
S2.41	50	39.4	0.14	39.7	39.7	0.0	74.30	32.8	0.57	33.0	33.0	0.0	5.96
	75	39.1	0.16	39.1	39.1	0.0	149.05	20.05	0.50	20.35	20.35	0.0	15.14
	100	38.9	0.14	39.1	39.1	0.0	180.26	19.4	0.53	19.55	19.55	0.0	3.75
S2.42	50	42.9	0.17	46.6	50.2	0.077	600.00	36.8	0.75	38.6	50.1	0.298	600.00
	75	38.8	0.27	39.2	50.2	0.281	600.00	20.65	0.87	21.5	50.1	1.33	600.00
	100	38.4	0.24	39.7	39.7	0.0	303.68	19.2	0.85	19.85	19.85	0.0	34.00
S2.43	50	42.2	0.18	43.2	51.8	0.199	600.00	37.0	0.98	39.9	50.9	0.276	600.00
	75	39.7	0.19	39.7	53.7	0.353	600.00	21.2	0.76	21.7	50.9	1.346	600.00
	100	39.6	0.21	40.4	46.3	0.146	600.00	19.6	1.07	20.2	20.2	0.0	102.44
S2.44	50	43.0	0.17	43.0	43.0	0.0	139.48	38.1	0.58	39.4	39.4	0.0	78.74

Table C.5 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	38.7	0.15	39.3	50.1	0.275	600.00	22.15	0.59	22.4	22.4	0.0	170.67
	100	38.6	0.17	39.6	40.1	0.013	600.00	19.15	0.70	19.8	19.8	0.0	4.74
S2.45	50	40.2	0.16	42.1	42.1	0.0	181.23	34.5	0.49	35.2	35.2	0.0	12.70
	75	35.6	0.15	36.4	46.8	0.286	600.00	19.5	0.97	19.75	23.4	0.185	600.00
	100	35.7	0.15	36.4	36.4	0.0	229.65	17.8	0.61	18.2	18.2	0.0	6.62
S2.46	50	42.0	0.20	42.0	51.5	0.226	600.00	39.1	0.93	40.4	50.1	0.24	600.00
	75	40.0	0.23	40.0	50.4	0.26	600.00	21.0	0.72	21.55	50.1	1.325	600.00
	100	39.7	0.20	40.3	50.1	0.243	600.00	19.75	0.82	20.25	20.25	0.0	48.32
S2.47	50	40.8	0.15	40.8	40.8	0.0	118.74	35.4	0.52	35.8	35.8	0.0	259.97
	75	36.6	0.17	37.5	37.5	0.0	230.54	19.95	0.66	20.95	32.3	0.542	600.00
	100	36.2	0.18	36.8	36.8	0.0	149.38	18.3	0.56	18.4	18.4	0.0	4.08
S2.48	50	42.4	0.15	42.6	52.5	0.232	600.00	38.1	0.68	39.4	44.375	0.126	600.00
	75	39.8	0.20	40.3	51.9	0.288	600.00	21.65	0.95	22.55	50.5	1.239	600.00
	100	39.5	0.15	39.8	51.9	0.304	600.00	19.7	0.60	20.15	20.15	0.0	32.68
S2.49	50	43.6	0.28	43.8	50.5	0.153	600.00	38.4	1.18	39.4	50.5	0.282	600.00
	75	39.2	0.28	39.4	60.0	0.523	600.00	21.1	1.38	21.5	50.5	1.349	600.00
	100	38.6	0.18	38.7	54.5	0.408	600.00	19.3	1.46	19.6	24.75	0.263	600.00

Table C.6: **Series 3. Example of table for metrics comparison;** $r_{\max} = 3$**.**

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S3.0	50	15.7	0.42	15.7	15.7	0.0	195.49	8.433	0.83	8.433	8.433	0.0	17.26
	75	15.5	0.41	15.5	15.5	0.0	256.36	5.1	0.94	5.1	5.1	0.0	5.86
	100	15.5	0.41	15.5	15.7	0.013	600.00	3.925	1.00	3.925	3.925	0.0	21.22
S3.1	50	16.4	0.44	18.4	18.4	0.0	238.13	7.2	0.87	7.2	7.2	0.0	104.59
	75	16.4	0.43	18.4	18.4	0.0	215.41	4.6	0.94	4.6	4.6	0.0	18.92
	100	16.4	0.44	18.4	18.4	0.0	318.67	4.6	0.95	4.6	4.6	0.0	1.51
S3.2	50	17.9	0.43	17.9	47.2	1.637	600.00	9.0	0.89	9.0	9.0	0.0	106.30
	75	17.9	0.44	17.9	34.6	0.933	600.00	6.8	0.96	6.967	6.967	0.0	21.24
	100	17.9	0.43	17.9	20.4	0.14	600.00	5.233	1.05	5.233	5.233	0.0	5.00
S3.3	50	12.0	0.46	12.2	16.3	0.336	600.00	5.7	0.94	5.7	5.7	0.0	24.86
	75	12.0	0.47	12.0	16.3	0.358	600.00	4.0	0.94	4.075	4.075	0.0	6.49
	100	12.0	0.46	12.0	12.0	0.0	457.53	3.0	1.04	3.0	3.0	0.0	48.30
S3.4	50	14.0	0.41	14.0	14.0	0.0	122.51	4.95	0.83	4.95	4.95	0.0	34.94
	75	14.0	0.40	14.0	14.0	0.0	61.16	4.667	0.93	4.75	4.75	0.0	1.16
	100	14.0	0.40	14.0	14.0	0.0	22.95	3.575	0.85	3.575	3.575	0.0	0.49
S3.5	50	16.7	0.49	16.7	30.002	0.797	600.00	8.65	1.02	8.65	8.65	0.0	63.27
	75	16.7	0.49	16.7	26.3	0.575	600.00	7.767	1.17	8.6	8.6	0.0	5.34
	100	16.7	0.49	16.7	19.4	0.162	600.00	5.567	1.12	5.733	5.733	0.0	1.76
S3.6	50	19.4	0.44	20.7	20.7	0.0	371.40	11.2	0.92	11.2	11.2	0.0	500.87
	75	19.4	0.43	20.7	20.7	0.0	366.84	7.05	1.05	8.1	8.1	0.0	68.53
	100	19.4	0.43	19.4	36.0	0.856	600.00	6.467	1.03	6.9	6.9	0.0	2.59
S3.7	50	22.1	0.46	22.1	22.3	0.009	600.00	9.133	1.05	9.9	9.9	0.0	108.38
	75	22.1	0.48	22.1	23.9	0.081	600.00	7.367	1.26	7.367	7.367	0.0	74.47
	100	22.1	0.48	22.1	22.1	0.0	219.44	7.367	0.95	7.367	7.367	0.0	0.37
S3.8	50	27.2	0.54	27.2	53.2	0.956	600.00	15.7	1.19	18.25	18.25	0.0	309.56
	75	27.2	0.50	27.2	60.0	1.206	600.00	14.7	1.30	15.65	15.65	0.0	248.78
	100	27.2	0.49	27.2	58.9	1.165	600.00	9.067	1.35	9.267	9.267	0.0	3.48
S3.9	50	4.4	0.42	4.4	4.4	0.0	46.74	2.2	0.60	2.55	2.55	0.0	3.32
	75	4.4	0.42	4.4	4.4	0.0	34.69	1.1	0.71	1.2	1.2	0.0	11.88
	100	4.4	0.42	4.4	4.4	0.0	34.73	1.1	0.80	1.1	1.1	0.0	8.49
S3.10	50	12.3	0.49	12.3	12.3	0.0	190.69	4.4	0.89	4.8	4.8	0.0	64.64

Table C.6 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	12.3	0.45	12.3	12.3	0.0	248.06	3.667	1.02	3.667	3.667	0.0	75.39
	100	12.3	0.45	12.3	12.3	0.0	145.70	3.25	1.03	3.25	3.25	0.0	7.06
S3.11	50	14.2	0.44	15.6	15.6	0.0	117.12	6.6	0.88	6.6	6.6	0.0	35.42
	75	14.2	0.46	15.4	15.4	0.0	204.72	4.733	0.99	4.95	4.95	0.0	34.93
	100	14.2	0.47	15.4	15.4	0.0	398.81	3.9	0.98	3.9	3.9	0.0	1.48
S3.12	50	7.3	0.41	9.8	9.8	0.0	14.51	4.333	0.69	4.7	4.7	0.0	4.81
	75	7.3	0.39	9.8	9.8	0.0	69.60	3.1	0.70	3.25	3.25	0.0	0.45
	100	7.3	0.39	9.8	9.8	0.0	14.22	1.825	0.78	2.45	2.45	0.0	4.52
S3.13	50	9.2	0.45	12.9	12.9	0.0	176.84	6.0	0.84	6.0	6.0	0.0	13.21
	75	9.2	0.44	12.9	12.9	0.0	184.25	3.533	0.92	4.025	4.025	0.0	19.32
	100	9.2	0.44	12.9	12.9	0.0	216.36	2.3	0.91	3.225	3.225	0.0	8.82
S3.14	50	21.9	0.46	21.9	22.3	0.018	600.00	10.4	1.01	10.4	10.4	0.0	394.77
	75	21.2	0.45	21.2	22.3	0.052	600.00	7.767	1.14	7.767	7.767	0.0	14.82
	100	21.2	0.42	21.2	21.9	0.033	600.00	7.067	0.82	7.067	7.067	0.0	0.33
S3.15	50	13.3	0.43	13.3	21.562	0.621	600.00	8.75	0.84	10.7	10.7	0.0	87.56
	75	12.9	0.43	12.9	21.4	0.659	600.00	5.1	0.94	5.2	5.2	0.0	10.30
	100	12.9	0.44	12.9	20.3	0.574	600.00	3.325	1.00	3.325	3.325	0.0	4.66
S3.16	50	15.0	0.41	15.0	15.0	0.0	217.57	7.05	0.90	7.05	7.05	0.0	144.28
	75	12.2	0.43	12.2	12.2	0.0	207.77	5.9	1.02	6.1	6.1	0.0	12.29
	100	12.2	0.43	12.2	12.2	0.0	201.60	3.833	0.95	3.833	3.833	0.0	2.45
S3.17	50	12.9	0.45	14.8	14.8	0.0	136.76	7.767	0.77	8.6	8.6	0.0	21.51
	75	12.1	0.45	13.9	13.9	0.0	121.68	4.033	0.94	5.3	5.3	0.0	4.18
	100	12.1	0.44	13.9	13.9	0.0	97.60	3.025	0.98	3.7	3.7	0.0	8.14
S3.18	50	19.9	0.48	21.0	21.0	0.0	471.70	8.933	1.04	10.55	10.55	0.0	47.25
	75	19.9	0.50	19.9	24.8	0.246	600.00	6.633	1.15	6.7	6.7	0.0	5.21
	100	19.9	0.49	21.0	21.0	0.0	387.82	5.6	1.10	6.7	6.7	0.0	2.09
S3.19	50	19.1	0.43	19.1	19.1	0.0	359.42	8.467	0.85	8.467	8.467	0.0	53.93
	75	19.1	0.43	19.1	19.6	0.026	600.00	6.367	1.01	6.367	6.367	0.0	17.23
	100	19.1	0.42	19.1	19.1	0.0	227.47	6.367	0.68	6.367	6.367	0.0	0.14
S3.20	50	6.6	0.42	6.6	6.6	0.0	65.48	3.3	0.75	3.3	3.3	0.0	4.99
	75	6.6	0.44	6.6	6.6	0.0	153.66	3.25	0.91	3.25	3.25	0.0	2.25
	100	6.6	0.43	6.6	6.6	0.0	85.44	1.65	0.91	1.65	1.65	0.0	3.50
S3.21	50	17.9	0.40	17.9	17.9	0.0	82.08	6.6	0.80	6.85	6.85	0.0	76.00
	75	14.8	0.40	15.2	15.2	0.0	113.52	4.15	0.84	4.15	4.15	0.0	5.63

Table C.6 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	100	14.8	0.39	15.2	15.2	0.0	123.84	3.9	0.79	3.9	3.9	0.0	0.48
S3.22	50	13.3	0.45	13.3	13.3	0.0	403.35	6.15	0.80	7.5	7.5	0.0	26.50
	75	13.3	0.46	13.3	16.3	0.226	600.00	4.433	0.88	4.433	4.433	0.0	26.92
	100	13.3	0.43	13.3	13.3	0.0	306.54	3.85	0.95	3.85	3.85	0.0	1.19
S3.23	50	10.2	0.39	10.2	10.2	0.0	134.83	4.667	0.81	4.8	4.8	0.0	62.68
	75	10.2	0.39	10.2	10.2	0.0	115.49	3.333	0.86	3.5	3.5	0.0	7.88
	100	10.2	0.41	10.2	10.2	0.0	73.50	2.55	0.86	2.55	2.55	0.0	5.81
S3.24	50	14.9	0.41	14.9	14.9	0.0	178.50	7.45	0.80	7.45	7.45	0.0	9.09
	75	14.9	0.40	14.9	14.9	0.0	98.42	4.967	0.92	5.0	5.0	0.0	9.54
	100	14.9	0.41	14.9	14.9	0.0	163.10	3.725	0.93	3.725	3.725	0.0	0.85
S3.25	50	19.6	0.46	22.1	22.1	0.0	308.36	12.3	0.93	12.3	12.3	0.0	94.08
	75	19.6	0.46	21.3	21.3	0.0	522.37	8.433	1.01	8.433	8.433	0.0	63.76
	100	19.6	0.44	21.3	21.3	0.0	482.58	6.533	1.34	7.1	7.1	0.0	3.47
S3.26	50	16.8	0.43	20.1	20.1	0.0	228.31	7.4	0.95	8.067	8.067	0.0	97.94
	75	16.8	0.44	20.1	20.1	0.0	547.33	5.6	1.07	6.7	6.7	0.0	19.55
	100	16.8	0.46	20.1	20.1	0.0	97.60	5.6	1.03	6.7	6.7	0.0	1.20
S3.27	50	16.2	0.45	16.2	16.2	0.0	348.53	8.633	0.90	9.3	9.3	0.0	7.81
	75	11.7	0.46	11.7	14.4	0.231	600.00	4.05	1.01	4.05	4.05	0.0	42.09
	100	11.7	0.47	11.7	24.2	1.068	600.00	2.925	1.08	2.925	2.925	0.0	37.39
S3.28	50	7.1	0.39	7.1	7.1	0.0	36.71	3.35	0.65	3.35	3.35	0.0	5.55
	75	7.1	0.39	7.1	7.1	0.0	26.01	1.775	0.72	1.775	1.775	0.0	8.02
	100	7.1	0.38	7.1	7.1	0.0	28.95	1.775	0.77	1.775	1.775	0.0	0.55
S3.29	50	11.2	0.43	11.2	11.2	0.0	144.30	5.6	0.91	5.6	5.6	0.0	187.13
	75	11.2	0.45	11.2	11.2	0.0	216.38	3.4	1.03	3.4	3.4	0.0	37.78
	100	11.2	0.45	11.2	11.2	0.0	272.18	2.35	1.00	2.35	2.35	0.0	29.36
S3.30	50	20.9	0.45	20.9	60.0	1.871	600.00	8.75	1.00	10.1	10.5	0.04	600.00
	75	20.9	0.45	20.9	196.119	8.384	600.00	8.133	1.10	9.2	9.2	0.0	44.73
	100	20.9	0.46	20.9	56.0	1.679	600.00	6.967	1.12	7.0	7.0	0.0	1.43
S3.31	50	16.1	0.48	16.1	29.705	0.845	600.00	7.667	1.04	8.1	8.1	0.0	283.05
	75	15.7	0.48	15.7	46.074	1.935	600.00	5.267	1.07	7.667	7.667	0.0	22.56
	100	15.7	0.48	15.7	30.037	0.913	600.00	5.133	1.11	5.133	5.133	0.0	3.82
S3.32	50	14.0	0.46	15.4	15.4	0.0	225.16	6.9	1.02	6.9	6.9	0.0	53.72
	75	14.0	0.47	14.6	14.6	0.0	289.58	4.667	1.11	4.667	4.667	0.0	8.39
	100	14.0	0.47	14.6	14.6	0.0	194.72	3.5	1.15	3.65	3.65	0.0	2.62

Table C.6 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
S3.33	50	13.8	0.46	13.8	13.8	0.0	335.22	4.4	0.92	6.367	6.367	0.0	233.91
	75	11.1	0.47	12.7	12.7	0.0	531.19	3.767	1.01	3.767	3.767	0.0	32.75
	100	11.1	0.46	12.7	12.7	0.0	383.69	2.775	1.12	3.175	3.175	0.0	51.20
S3.34	50	6.2	0.39	6.2	6.2	0.0	39.95	2.467	0.62	2.467	2.467	0.0	2.03
	75	6.2	0.40	6.2	6.2	0.0	51.11	2.0	0.73	2.0	2.0	0.0	0.55
	100	6.2	0.40	6.2	6.2	0.0	54.95	1.55	0.82	1.55	1.55	0.0	0.79
S3.35	50	9.5	0.41	9.5	9.5	0.0	166.60	5.9	0.76	6.067	6.067	0.0	24.71
	75	9.5	0.41	9.5	9.5	0.0	121.23	4.0	0.88	4.0	4.0	0.0	7.88
	100	9.5	0.42	9.5	9.5	0.0	134.86	2.375	0.91	2.375	2.375	0.0	9.80
S3.36	50	8.3	0.41	8.3	8.3	0.0	61.53	4.15	0.77	5.9	5.9	0.0	14.12
	75	8.3	0.42	8.3	8.3	0.0	75.35	3.9	0.90	4.05	4.05	0.0	6.75
	100	8.3	0.42	8.3	8.3	0.0	62.58	2.075	0.88	2.075	2.075	0.0	3.64
S3.37	50	20.9	0.47	20.9	45.664	1.185	600.00	11.85	1.05	12.0	17.5	0.458	600.00
	75	20.9	0.49	20.9	55.537	1.657	600.00	7.767	1.23	8.65	8.65	0.0	436.09
	100	20.9	0.47	20.9	56.4	1.699	600.00	6.967	1.21	7.767	7.767	0.0	14.39
S3.38	50	17.7	0.48	17.7	21.9	0.237	600.00	9.7	1.08	12.15	12.15	0.0	175.95
	75	17.7	0.50	17.7	17.7	0.0	463.70	8.1	1.19	8.1	8.1	0.0	45.21
	100	17.7	0.50	17.7	17.7	0.0	445.88	5.9	1.23	5.9	5.9	0.0	1.59
S3.39	50	18.0	0.50	18.0	35.535	0.974	600.00	8.8	1.09	8.8	18.731	1.128	600.00
	75	18.0	0.49	18.0	41.1	1.283	600.00	6.95	1.22	6.95	6.95	0.0	66.64
	100	18.0	0.48	18.0	41.1	1.283	600.00	6.0	1.16	6.0	6.0	0.0	2.60
S3.40	50	15.7	0.43	15.7	19.8	0.261	600.00	7.85	0.83	7.85	7.85	0.0	49.83
	75	15.7	0.44	15.7	21.4	0.363	600.00	4.675	1.05	4.675	4.675	0.0	35.04
	100	15.7	0.44	15.7	15.7	0.0	273.84	4.675	1.01	4.675	4.675	0.0	1.22
S3.41	50	7.6	0.41	7.6	7.6	0.0	492.94	3.0	0.81	3.0	3.0	0.0	50.85
	75	7.6	0.41	7.6	17.0	1.237	600.00	2.325	0.85	2.9	2.9	0.0	36.13
	100	7.6	0.43	7.6	7.6	0.0	277.74	1.9	0.98	1.9	1.9	0.0	19.34
S3.42	50	16.3	0.44	17.4	17.4	0.0	224.45	7.4	0.92	8.75	8.75	0.0	43.37
	75	16.3	0.44	17.4	17.4	0.0	106.61	5.433	1.05	5.8	5.8	0.0	6.67
	100	16.3	0.44	17.4	17.4	0.0	177.92	4.667	1.07	5.8	5.8	0.0	1.45
S3.43	50	9.3	0.40	9.3	9.3	0.0	84.68	4.2	0.76	4.65	4.65	0.0	19.54
	75	9.3	0.41	9.3	9.3	0.0	25.18	3.033	0.89	3.15	3.15	0.0	9.02
	100	9.3	0.40	9.3	9.3	0.0	56.59	2.325	0.89	2.325	2.325	0.0	6.42
S3.44	50	8.6	0.40	9.6	9.6	0.0	184.20	3.533	0.71	3.533	3.533	0.0	20.26

Table C.6 – continued from previous page

		P_1						P_∞					
		Heuristic		Solver				Heuristic		Solver			
Inst	%UT	LB	CPU	LB	UB	GAP	CPU	LB	CPU	LB	UB	GAP	CPU
	75	8.6	0.41	9.6	9.6	0.0	193.94	2.65	0.80	2.65	2.65	0.0	46.92
	100	8.6	0.41	9.6	9.6	0.0	213.93	2.15	0.81	2.4	2.4	0.0	14.22
S3.45	50	14.9	0.43	16.3	16.3	0.0	251.62	7.767	0.81	8.8	8.8	0.0	18.81
	75	14.9	0.40	16.3	16.3	0.0	363.95	5.667	0.97	5.667	5.667	0.0	2.16
	100	14.9	0.40	16.3	16.3	0.0	177.39	4.1	0.99	5.1	5.1	0.0	1.07
S3.46	50	8.9	0.41	8.9	8.9	0.0	74.76	4.133	0.76	4.133	4.133	0.0	42.59
	75	8.9	0.56	8.9	8.9	0.0	71.04	2.967	0.88	2.967	2.967	0.0	18.41
	100	8.9	0.41	8.9	8.9	0.0	70.18	2.225	0.91	2.225	2.225	0.0	6.09
S3.47	50	8.5	0.40	8.5	8.5	0.0	178.16	3.6	0.68	3.6	3.6	0.0	20.01
	75	8.5	0.41	8.5	8.5	0.0	204.22	2.4	0.75	2.5	2.5	0.0	5.95
	100	8.5	0.40	8.5	8.5	0.0	71.98	2.125	0.80	2.125	2.125	0.0	19.97
S3.48	50	15.4	0.44	17.4	17.4	0.0	191.53	6.9	0.92	8.65	8.65	0.0	47.54
	75	15.4	0.41	17.1	17.1	0.0	289.86	5.8	0.98	6.1	6.1	0.0	3.53
	100	15.4	0.42	17.1	17.1	0.0	310.61	4.45	1.01	4.75	4.75	0.0	1.84
S3.49	50	7.3	0.40	7.3	7.3	0.0	94.79	4.55	0.67	4.8	4.8	0.0	4.03
	75	6.4	0.40	7.1	7.1	0.0	137.96	2.85	0.80	2.85	2.85	0.0	6.18
	100	6.4	0.42	7.1	7.1	0.0	30.94	1.6	0.82	1.775	1.775	0.0	9.96

Appendix D

Hybrid approach with callbacks. Application for TLBP

Table D.1: MIP restart. First 10 instances from S1, S2 and S3; $r_{\max} = 2$.

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
S1.0	50	28.8	0.17	3	0	30.4	30.4	0.0	107.25	17.95	0.43	12	0	21.15	21.15	0.0	58.78
	75	28.8	0.12	3	0	30.4	30.4	0.0	190.49	11.867	0.34	4	0	13.2	13.2	0.0	13.63
	100	28.8	0.11	3	0	30.4	30.4	0.0	134.35	9.4	0.45	0	0	10.133	10.133	0.0	9.11
S1.1	50	24.6	0.17	3	0	27.3	42.2	0.546	600.00	16.0	0.42	14	0	19.3	19.3	0.0	27.92
	75	23.4	0.14	0	0	23.7	42.2	0.781	600.00	10.4	0.46	9	0	11.55	11.55	0.0	450.71
	100	23.8	0.13	0	0	23.8	38.8	0.63	600.00	7.8	0.47	7	0	7.933	7.933	0.0	248.87
S1.2	50	27.7	0.11	5	0	29.1	52.8	0.814	600.00	16.1	0.42	5	0	17.55	21.35	0.217	600.00
	75	26.4	0.10	0	0	28.1	36.7	0.306	600.00	11.8	0.41	4	0	13.0	13.0	0.0	486.83
	100	27.7	0.10	0	0	28.0	28.9	0.032	600.00	8.8	0.33	0	0	9.333	9.333	0.0	24.39
S1.3	50	33.1	0.18	0	0	33.1	51.4	0.553	600.00	20.6	0.43	11	0	22.05	23.35	0.059	600.00
	75	29.3	0.16	0	0	29.5	51.4	0.742	600.00	12.85	0.59	12	0	14.85	35.0	1.357	600.00
	100	29.3	0.17	0	0	29.4	51.4	0.748	600.00	9.4	0.72	4	0	9.833	20.45	1.08	600.00
S1.4	50	26.0	0.13	6	1	28.5	37.9	0.33	600.00	17.65	0.43	0	0	18.6	18.6	0.0	177.65
	75	25.5	0.15	0	0	26.3	38.0	0.445	600.00	11.15	0.58	4	0	12.35	12.35	0.0	67.81
	100	24.3	0.14	5	0	26.3	36.8	0.399	600.00	8.5	0.47	0	0	8.767	8.767	0.0	72.13
S1.5	50	30.8	0.16	0	0	32.3	41.8	0.294	600.00	20.15	0.42	3	0	20.65	20.65	0.0	64.58
	75	29.0	0.14	0	0	29.0	47.673	0.644	600.00	14.45	0.63	0	0	14.45	16.35	0.131	600.00
	100	29.0	0.13	0	0	29.4	53.88	0.833	600.00	9.567	0.49	0	0	9.833	16.35	0.663	600.00
S1.6	50	27.1	0.13	0	0	28.5	28.6	0.004	600.00	17.7	0.53	0	0	18.9	18.9	0.0	229.05
	75	26.5	0.12	0	0	27.9	28.6	0.025	600.00	12.8	0.51	0	0	13.1	13.1	0.0	577.02
	100	26.5	0.11	0	0	27.9	28.6	0.025	600.00	8.833	0.41	7	0	9.3	9.3	0.0	21.92
S1.7	50	33.4	0.15	3	0	38.3	40.6	0.06	600.00	20.55	0.54	0	0	21.1	40.6	0.924	600.00
	75	29.1	0.14	0	0	29.6	52.5	0.774	600.00	13.65	0.54	10	0	14.6	20.65	0.414	600.00
	100	29.0	0.12	0	0	29.0	43.9	0.514	600.00	9.7	0.48	0	0	9.867	10.367	0.051	600.00
S1.8	50	32.4	0.17	0	0	32.4	33.9	0.046	600.00	17.3	0.45	43	1	20.6	20.6	0.0	84.89
	75	26.9	0.13	0	0	28.0	38.8	0.386	600.00	12.15	0.54	19	0	13.7	14.55	0.062	600.00
	100	26.7	0.13	0	0	27.1	35.1	0.295	600.00	8.9	0.51	7	0	9.633	9.633	0.0	147.34
S1.9	50	23.7	0.08	0	0	24.4	24.4	0.0	46.45	12.9	0.22	0	0	12.9	12.9	0.0	4.14
	75	24.4	0.07	0	0	24.4	24.4	0.0	121.11	9.467	0.24	0	0	9.467	9.467	0.0	2.11
	100	24.4	0.07	0	0	24.4	24.4	0.0	31.01	8.133	0.20	0	0	8.133	8.133	0.0	0.05
S2.0	50	35.8	0.17	0	0	35.8	50.2	0.402	600.00	21.4	0.85	0	0	21.65	50.2	1.319	600.00
	75	31.9	0.19	0	0	33.1	100.4	2.033	600.00	14.1	0.76	4	0	15.4	50.2	2.26	600.00

Table D.1 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
	100	30.0	0.15	4	0	30.8	57.464	0.866	600.00	9.933	0.84	8	0	10.567	32.525	2.078	600.00
S2.1	50	32.2	0.27	2	0	33.4	60.0	0.796	600.00	18.4	0.72	0	0	19.8	50.2	1.535	600.00
	75	25.1	0.20	0	0	26.8	51.2	0.91	600.00	11.45	0.82	0	0	11.7	46.1	2.94	600.00
	100	25.4	0.17	0	0	26.6	50.9	0.914	600.00	8.467	1.06	0	0	9.0	23.95	1.661	600.00
S2.2	50	33.9	0.14	0	0	34.5	73.637	1.134	600.00	20.3	0.57	5	0	22.2	47.2	1.126	600.00
	75	28.5	0.17	10	0	30.2	53.177	0.761	600.00	12.9	0.63	6	0	14.15	49.224	2.479	600.00
	100	28.6	0.16	0	0	29.8	53.337	0.79	600.00	9.433	0.88	0	0	10.067	20.5	1.036	600.00
S2.3	50	34.6	0.17	2	0	36.5	51.4	0.408	600.00	20.9	0.78	2	0	22.4	51.4	1.295	600.00
	75	31.2	0.30	0	0	33.6	120.71	2.593	600.00	13.85	0.84	0	0	15.05	51.4	2.415	600.00
	100	30.0	0.16	0	0	30.6	58.6	0.915	600.00	9.633	1.01	0	0	10.267	45.645	3.446	600.00
S2.4	50	32.6	0.20	4	0	34.9	113.014	2.238	600.00	19.45	1.01	0	0	20.65	50.5	1.446	600.00
	75	26.3	0.24	0	0	27.8	104.763	2.768	600.00	11.8	1.12	0	0	13.1	50.1	2.824	600.00
	100	26.0	0.16	0	0	27.4	59.9	1.186	600.00	8.567	1.49	0	0	9.067	41.855	3.616	600.00
S2.5	50	35.3	0.22	0	0	37.3	51.8	0.389	600.00	20.3	0.66	9	0	22.8	51.8	1.272	600.00
	75	33.4	0.17	0	0	34.1	58.2	0.707	600.00	13.75	0.77	0	0	14.85	50.334	2.39	600.00
	100	30.5	0.17	0	0	31.6	56.059	0.774	600.00	9.8	0.83	6	0	10.4	35.76	2.438	600.00
S2.6	50	26.0	0.16	2	0	28.5	35.2	0.235	600.00	17.95	0.39	3	0	21.35	21.35	0.0	562.61
	75	26.0	0.14	2	0	28.5	54.6	0.916	600.00	12.65	0.72	0	0	14.3	14.3	0.0	266.02
	100	26.0	0.14	2	0	28.6	35.4	0.238	600.00	8.667	0.49	0	0	9.533	9.533	0.0	16.86
S2.7	50	38.5	0.22	0	0	38.5	55.8	0.449	600.00	21.4	0.62	6	0	23.0	51.8	1.252	600.00
	75	30.8	0.16	6	0	31.7	110.0	2.47	600.00	14.1	1.02	0	0	15.35	50.6	2.296	600.00
	100	30.4	0.18	6	0	31.9	78.662	1.466	600.00	9.967	0.80	0	0	10.667	24.2	1.269	600.00
S2.8	50	35.0	0.17	8	0	36.0	86.293	1.397	600.00	19.95	0.58	5	0	21.05	41.3	0.962	600.00
	75	28.2	0.21	0	0	29.1	51.4	0.766	600.00	12.45	0.81	1	0	13.75	47.05	2.422	600.00
	100	27.5	0.19	3	0	29.5	52.3	0.773	600.00	9.133	0.92	2	0	9.7	20.75	1.139	600.00
S2.9	50	35.2	0.15	0	0	35.2	52.1	0.48	600.00	20.9	0.63	0	0	22.3	37.012	0.66	600.00
	75	30.4	0.23	2	0	30.8	53.1	0.724	600.00	13.95	0.78	0	0	15.15	47.067	2.107	600.00
	100	30.8	0.17	0	0	30.8	52.433	0.702	600.00	10.167	0.70	0	0	10.333	19.8	0.916	600.00
S3.0	50	15.7	0.39	0	0	15.7	15.7	0.0	290.92	7.2	0.77	0	0	7.2	7.2	0.0	5.84
	75	15.5	0.40	0	0	15.5	15.5	0.0	382.10	4.05	0.95	0	0	4.05	4.05	0.0	12.58
	100	15.5	0.39	0	0	15.5	15.5	0.0	182.84	3.925	0.98	0	0	3.925	3.925	0.0	5.96
S3.1	50	16.4	0.40	0	0	17.6	17.6	0.0	284.10	7.2	0.85	0	0	7.2	7.2	0.0	35.03
	75	16.4	0.40	2	0	17.6	17.6	0.0	269.24	4.6	0.91	0	0	4.6	4.6	0.0	16.32
	100	16.4	0.40	2	0	17.6	17.6	0.0	271.38	4.1	0.89	0	0	4.4	4.4	0.0	5.77

Table D.1 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
S3.2	50	17.8	0.39	0	0	17.8	24.6	0.382	600.00	9.0	0.93	0	0	9.0	9.0	0.0	19.03
	75	17.8	0.40	0	0	17.9	18.0	0.006	600.00	6.65	0.94	0	0	6.65	6.65	0.0	15.10
	100	17.8	0.40	0	0	17.8	29.2	0.64	600.00	4.5	0.98	0	0	4.5	4.5	0.0	3.27
S3.3	50	11.6	0.44	0	0	11.6	11.6	0.0	482.85	5.45	0.83	0	0	5.45	5.45	0.0	35.52
	75	11.6	0.43	0	0	11.6	11.6	0.0	258.50	3.067	1.01	0	0	3.067	3.067	0.0	35.47
	100	11.6	0.41	0	0	11.6	11.6	0.0	309.10	2.9	1.04	0	0	2.9	2.9	0.0	21.32
S3.4	50	14.0	0.37	0	0	14.0	14.0	0.0	40.52	4.95	0.80	0	0	4.95	4.95	0.0	14.20
	75	14.0	0.37	0	0	14.0	14.0	0.0	60.03	4.15	0.83	0	0	4.15	4.15	0.0	2.62
	100	14.0	0.36	0	0	14.0	14.0	0.0	24.32	3.575	0.74	0	0	3.575	3.575	0.0	0.49
S3.5	50	14.1	0.46	0	0	14.1	31.102	1.206	600.00	7.5	1.00	0	0	7.5	7.5	0.0	91.47
	75	14.1	0.47	0	0	14.1	29.857	1.118	600.00	4.7	1.15	0	0	4.7	4.7	0.0	34.69
	100	14.1	0.46	0	0	14.1	24.1	0.709	600.00	3.525	1.17	0	0	3.525	3.525	0.0	46.70
S3.6	50	19.4	0.41	15	0	20.7	20.7	0.0	319.66	11.2	0.88	0	0	11.2	11.2	0.0	68.50
	75	18.3	0.43	15	0	20.7	20.7	0.0	539.90	7.05	1.02	19	0	7.4	7.4	0.0	178.19
	100	18.3	0.43	0	0	18.3	23.5	0.284	600.00	6.1	1.02	69	1	6.9	6.9	0.0	2.58
S3.7	50	20.0	0.43	0	0	20.7	34.7	0.676	600.00	9.133	0.98	24	0	9.7	9.7	0.0	190.82
	75	20.0	0.44	0	0	20.7	23.9	0.155	600.00	7.05	1.20	0	0	7.05	7.05	0.0	79.66
	100	20.0	0.43	0	0	20.7	20.7	0.0	378.47	5.975	1.11	0	0	5.975	5.975	0.0	4.61
S3.8	50	23.8	0.58	0	0	23.8	60.0	1.521	600.00	12.95	1.78	0	0	12.95	24.8	0.915	600.00
	75	23.5	0.52	0	0	23.5	60.0	1.553	600.00	8.667	1.43	0	0	8.667	21.1	1.435	600.00
	100	23.2	0.47	0	0	23.2	60.0	1.586	600.00	7.767	1.33	1	0	8.167	8.167	0.0	29.44
S3.9	50	4.4	0.39	0	0	4.4	4.4	0.0	92.32	2.2	0.61	0	0	2.55	2.55	0.0	5.75
	75	4.4	0.40	0	0	4.4	4.4	0.0	39.96	1.1	0.70	0	0	1.2	1.2	0.0	22.82
	100	4.4	0.40	0	0	4.4	4.4	0.0	36.54	1.1	0.79	0	0	1.1	1.1	0.0	22.86

Table D.2: Lazy constraints. First 10 instances from S1, S2 and S3; $r_{\max} = 2$.

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
S1.0	50	28.8	0.16	3	0	30.4	30.4	0.0	101.24	19.3	0.42	7	0	21.15	21.15	0.0	14.28
	75	28.8	0.12	3	0	30.4	32.9	0.082	600.00	11.867	0.34	4	0	13.2	14.545	0.102	600.00
	100	28.8	0.11	3	0	30.4	30.4	0.0	71.43	9.4	0.44	0	0	10.133	10.133	0.0	9.25
S1.1	50	24.6	0.16	3	0	26.7	35.5	0.33	600.00	16.0	0.41	14	0	19.3	19.3	0.0	14.93
	75	23.4	0.13	0	0	23.5	38.8	0.651	600.00	10.4	0.46	4	0	10.95	17.821	0.627	600.00
	100	23.4	0.12	0	0	23.5	38.8	0.651	600.00	7.8	0.47	1	0	7.867	16.945	1.154	600.00
S1.2	50	27.7	0.11	0	0	28.0	40.727	0.455	600.00	16.1	0.42	5	0	17.55	43.066	1.454	600.00
	75	26.4	0.10	0	0	26.4	36.7	0.39	600.00	11.8	0.41	4	0	13.0	13.0	0.0	388.21
	100	27.7	0.10	0	0	28.0	36.7	0.311	600.00	8.8	0.33	0	0	9.333	9.333	0.0	244.25
S1.3	50	33.1	0.18	0	0	33.1	51.4	0.553	600.00	20.6	0.44	11	0	22.05	40.613	0.842	600.00
	75	29.3	0.15	0	0	29.3	51.4	0.754	600.00	12.85	0.57	12	0	14.0	37.486	1.678	600.00
	100	29.3	0.17	0	0	29.3	51.4	0.754	600.00	9.4	0.72	4	0	9.833	20.25	1.059	600.00
S1.4	50	26.0	0.13	6	1	28.5	28.5	0.0	418.40	17.65	0.42	0	0	18.6	26.978	0.45	600.00
	75	25.5	0.14	0	0	26.3	27.6	0.049	600.00	11.15	0.58	3	0	12.0	16.55	0.379	600.00
	100	24.3	0.13	5	0	26.3	31.9	0.213	600.00	8.5	0.47	0	0	8.767	14.804	0.689	600.00
S1.5	50	30.8	0.16	0	0	32.3	40.2	0.245	600.00	20.15	0.40	3	0	20.6	24.45	0.187	600.00
	75	29.0	0.14	0	0	29.0	50.256	0.733	600.00	14.45	0.64	0	0	14.45	20.45	0.415	600.00
	100	29.0	0.13	0	0	29.5	40.8	0.383	600.00	9.567	0.49	0	0	9.8	18.41	0.879	600.00
S1.6	50	27.1	0.13	0	0	28.5	35.4	0.242	600.00	17.7	0.52	0	0	17.7	29.218	0.651	600.00
	75	26.5	0.12	0	0	27.6	34.346	0.244	600.00	12.8	0.51	0	0	13.1	13.1	0.0	532.77
	100	26.5	0.11	0	0	27.9	27.9	0.0	267.35	8.833	0.40	7	0	9.3	9.3	0.0	11.80
S1.7	50	33.4	0.15	0	0	34.7	48.858	0.408	600.00	20.55	0.53	0	0	21.1	43.9	1.081	600.00
	75	29.1	0.13	0	0	29.6	51.8	0.75	600.00	13.65	0.53	6	0	14.3	22.65	0.584	600.00
	100	29.0	0.11	0	0	29.0	44.6	0.538	600.00	9.7	0.47	0	0	9.867	19.15	0.941	600.00
S1.8	50	32.4	0.17	0	0	32.4	39.1	0.207	600.00	17.3	0.45	43	1	20.6	20.6	0.0	60.66
	75	26.7	0.11	0	0	28.0	36.491	0.303	600.00	12.7	0.50	0	0	12.85	19.85	0.545	600.00
	100	26.7	0.13	0	0	27.7	37.053	0.338	600.00	8.9	0.51	7	0	9.633	9.633	0.0	85.90
S1.9	50	23.7	0.07	0	0	24.4	24.4	0.0	63.28	12.9	0.22	0	0	12.9	12.9	0.0	54.72
	75	24.4	0.07	0	0	24.4	24.4	0.0	128.40	9.467	0.24	0	0	9.467	9.467	0.0	23.70
	100	24.4	0.07	0	0	24.4	24.4	0.0	59.12	8.133	0.20	0	0	8.133	8.133	0.0	0.18
S2.0	50	35.8	0.17	0	0	35.8	50.2	0.402	600.00	21.4	0.83	1	0	22.7	50.2	1.211	600.00

Table D.2 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
	75	31.9	0.18	0	0	33.1	60.0	0.813	600.00	14.3	0.85	4	0	15.3	50.2	2.281	600.00
	100	30.0	0.14	8	0	32.1	57.083	0.778	600.00	9.933	0.84	8	0	10.467	38.629	2.691	600.00
S2.1	50	32.2	0.26	2	0	33.0	54.998	0.667	600.00	18.4	0.74	0	0	19.9	50.2	1.523	600.00
	75	25.1	0.19	0	0	25.8	51.2	0.984	600.00	11.45	0.82	0	0	11.7	44.777	2.827	600.00
	100	25.4	0.16	0	0	26.6	49.307	0.854	600.00	8.467	1.04	0	0	8.7	23.95	1.753	600.00
S2.2	50	33.9	0.13	0	0	34.5	43.8	0.27	600.00	20.1	0.53	5	0	22.1	50.877	1.302	600.00
	75	28.5	0.16	0	0	28.9	43.8	0.516	600.00	12.9	0.66	6	0	14.15	36.7	1.594	600.00
	100	28.6	0.15	0	0	29.4	45.4	0.544	600.00	9.433	0.86	0	0	9.933	20.755	1.089	600.00
S2.3	50	35.9	0.20	1	0	36.1	51.7	0.432	600.00	20.75	0.90	2	0	22.4	51.4	1.295	600.00
	75	31.2	0.28	0	0	33.1	56.974	0.721	600.00	13.85	0.82	0	0	14.95	50.838	2.401	600.00
	100	30.0	0.15	0	0	30.1	58.089	0.93	600.00	9.633	0.99	0	0	10.267	38.243	2.725	600.00
S2.4	50	32.6	0.19	4	0	34.9	52.7	0.51	600.00	19.45	0.99	0	0	20.0	50.5	1.525	600.00
	75	26.3	0.23	0	0	28.9	52.7	0.824	600.00	11.8	1.12	0	0	13.0	50.1	2.854	600.00
	100	26.0	0.15	0	0	26.8	56.07	1.092	600.00	8.567	1.45	0	0	9.033	43.186	3.781	600.00
S2.5	50	35.3	0.21	0	0	37.3	55.621	0.491	600.00	20.3	0.65	8	0	22.55	51.8	1.297	600.00
	75	33.4	0.16	0	0	34.1	56.638	0.661	600.00	13.75	0.78	0	0	14.75	49.695	2.369	600.00
	100	30.5	0.16	0	0	31.6	56.284	0.781	600.00	9.8	0.82	6	0	10.467	24.45	1.336	600.00
S2.6	50	26.0	0.15	2	0	28.6	32.4	0.133	600.00	17.95	0.39	3	0	21.35	39.747	0.862	600.00
	75	26.0	0.13	2	0	28.6	28.6	0.0	537.73	12.65	0.71	0	0	14.3	17.7	0.238	600.00
	100	26.0	0.13	2	0	28.5	32.4	0.137	600.00	8.667	0.49	0	0	9.533	16.784	0.761	600.00
S2.7	50	38.5	0.21	0	0	38.5	55.8	0.449	600.00	21.4	0.63	6	0	23.0	51.8	1.252	600.00
	75	30.8	0.15	6	0	32.0	59.6	0.863	600.00	14.1	1.01	0	0	15.35	50.6	2.296	600.00
	100	30.4	0.17	6	0	31.9	52.2	0.636	600.00	9.967	0.79	0	0	10.5	40.422	2.85	600.00
S2.8	50	35.0	0.16	8	0	36.0	47.951	0.332	600.00	19.95	0.57	5	0	21.05	41.3	0.962	600.00
	75	28.2	0.20	0	0	29.1	51.255	0.761	600.00	12.45	0.80	1	0	13.75	51.0	2.709	600.00
	100	27.5	0.18	3	0	29.1	46.314	0.592	600.00	9.133	0.89	2	0	9.567	20.65	1.159	600.00
S2.9	50	35.2	0.14	0	0	35.2	52.1	0.48	600.00	20.9	0.64	0	0	22.2	50.7	1.284	600.00
	75	30.4	0.21	2	0	30.8	52.1	0.692	600.00	13.95	0.77	0	0	14.8	39.629	1.678	600.00
	100	30.8	0.17	0	0	30.8	45.0	0.461	600.00	10.167	0.71	0	0	10.167	21.85	1.149	600.00
S3.0	50	15.7	0.37	0	0	15.7	33.08	1.107	600.00	7.2	0.75	0	0	7.2	7.2	0.0	88.48
	75	15.5	0.37	0	0	15.5	29.042	0.874	600.00	4.05	0.94	0	0	4.05	4.05	0.0	99.68
	100	15.5	0.37	0	0	15.5	36.2	1.335	600.00	3.925	0.96	0	0	3.925	3.925	0.0	85.19
S3.1	50	16.4	0.38	0	0	17.6	17.6	0.0	193.30	7.2	0.84	0	0	7.2	7.25	0.007	600.00
	75	16.4	0.39	2	0	17.6	17.6	0.0	185.01	4.6	0.90	0	0	4.6	4.6	0.0	33.68

Table D.2 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
	100	16.4	0.39	2	0	17.6	17.6	0.0	302.20	4.1	0.88	0	0	4.4	4.4	0.0	33.55
S3.2	50	17.8	0.37	0	0	17.9	22.8	0.274	600.00	9.0	0.91	0	0	9.0	9.0	0.0	132.27
	75	17.8	0.38	0	0	17.9	44.629	1.493	600.00	6.65	0.95	0	0	6.65	6.65	0.0	96.68
	100	17.8	0.38	0	0	17.9	41.84	1.337	600.00	4.5	0.97	0	0	4.5	4.5	0.0	35.74
S3.3	50	11.6	0.42	0	0	11.6	20.1	0.733	600.00	5.45	0.81	0	0	5.45	5.45	0.0	79.78
	75	11.6	0.42	0	0	11.6	25.964	1.238	600.00	3.067	0.98	0	0	3.067	3.067	0.0	518.36
	100	11.6	0.40	0	0	11.6	11.6	0.0	305.00	2.9	1.03	0	0	2.9	2.9	0.0	205.46
S3.4	50	14.0	0.36	0	0	14.0	14.0	0.0	291.41	4.95	0.78	0	0	4.95	4.95	0.0	114.61
	75	14.0	0.35	0	0	14.0	14.0	0.0	425.86	4.15	0.82	0	0	4.15	4.15	0.0	17.44
	100	14.0	0.35	0	0	14.0	14.0	0.0	84.50	3.575	0.73	0	0	3.575	3.575	0.0	3.65
S3.5	50	14.1	0.44	0	0	14.1	36.088	1.559	600.00	7.5	0.98	0	0	7.5	17.845	1.379	600.00
	75	14.1	0.44	0	0	14.1	35.7	1.532	600.00	4.7	1.13	0	0	4.7	5.267	0.121	600.00
	100	14.1	0.45	0	0	14.1	28.9	1.05	600.00	3.525	1.16	0	0	3.525	3.525	0.0	384.87
S3.6	50	19.4	0.41	15	0	20.7	36.321	0.755	600.00	11.2	0.86	0	0	11.2	19.05	0.701	600.00
	75	18.3	0.41	15	0	20.7	20.7	0.0	339.23	7.05	1.09	0	0	7.05	9.3	0.319	600.00
	100	18.3	0.41	15	0	20.7	40.2	0.942	600.00	6.1	1.00	69	1	6.9	6.9	0.0	33.88
S3.7	50	20.0	0.41	0	0	20.7	20.7	0.0	502.17	9.133	0.96	24	0	9.7	11.567	0.192	600.00
	75	20.0	0.42	0	0	20.7	20.7	0.0	382.14	7.05	1.18	0	0	7.05	7.067	0.002	600.00
	100	20.0	0.42	0	0	20.7	20.7	0.0	548.60	5.975	1.10	0	0	5.975	5.975	0.0	54.90
S3.8	50	23.8	0.57	16	1	24.2	60.0	1.479	600.00	12.95	1.74	0	0	12.95	50.4	2.892	600.00
	75	23.5	0.52	16	1	24.2	128.428	4.307	600.00	8.667	1.42	0	0	8.667	37.623	3.341	600.00
	100	23.2	0.45	0	0	23.2	130.109	4.608	600.00	7.767	1.30	1	0	8.167	8.167	0.0	15.46
S3.9	50	4.4	0.38	0	0	4.4	4.4	0.0	62.02	2.2	0.58	0	0	2.55	2.55	0.0	12.05
	75	4.4	0.39	0	0	4.4	4.4	0.0	53.86	1.1	0.69	0	0	1.2	1.2	0.0	25.74
	100	4.4	0.39	0	0	4.4	4.4	0.0	54.45	1.1	0.78	0	0	1.1	1.1	0.0	27.44

Table D.3: MIP restart. First 10 instances from S1, S2 and S3; $r_{\max} = 3$.

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
S1.0	50	34.7	0.11	0	0	34.7	34.7	0.0	77.09	22.3	0.28	0	0	22.3	22.3	0.0	1.64
	75	34.7	0.10	0	0	34.7	34.7	0.0	91.87	19.8	0.37	0	0	19.8	19.8	0.0	0.71
	100	34.7	0.09	0	0	34.7	34.7	0.0	69.19	17.35	0.22	0	0	17.35	17.35	0.0	0.02
S1.1	50	36.0	0.12	2	0	38.6	38.6	0.0	102.37	21.35	0.39	0	0	22.25	22.25	0.0	4.41
	75	35.8	0.12	2	0	37.0	37.0	0.0	63.21	18.35	0.50	6	0	19.4	19.4	0.0	8.19
	100	35.8	0.14	2	0	37.0	37.0	0.0	52.00	17.85	0.35	6	0	18.5	18.5	0.0	0.27
S1.2	50	36.5	0.11	4	0	38.0	38.0	0.0	258.22	29.3	0.37	4	0	31.5	31.5	0.0	12.26
	75	36.5	0.11	0	0	36.7	36.7	0.0	62.85	20.55	0.41	12	1	21.1	21.1	0.0	8.43
	100	36.5	0.10	0	0	36.7	36.7	0.0	54.27	18.25	0.30	0	0	18.35	18.35	0.0	0.04
S1.3	50	41.2	0.13	0	0	41.2	41.2	0.0	89.92	36.0	0.39	0	0	36.0	36.0	0.0	30.06
	75	40.5	0.17	13	0	41.1	41.1	0.0	199.34	22.4	0.63	0	0	22.4	22.4	0.0	14.94
	100	40.2	0.14	7	0	41.1	41.1	0.0	83.82	20.2	0.61	0	0	20.55	20.55	0.0	2.40
S1.4	50	36.8	0.11	3	0	37.2	37.2	0.0	35.35	32.9	0.48	4	0	33.1	33.1	0.0	2.11
	75	36.0	0.10	0	0	36.0	36.0	0.0	26.55	18.75	0.35	11	0	19.45	19.45	0.0	5.82
	100	34.8	0.10	0	0	35.3	35.3	0.0	76.29	17.4	0.30	0	0	17.65	17.65	0.0	0.15
S1.5	50	40.2	0.14	0	0	40.2	40.2	0.0	123.58	37.2	0.40	3	0	38.9	38.9	0.0	4.36
	75	39.7	0.13	1	0	40.2	40.2	0.0	331.02	20.45	0.40	0	0	20.45	20.45	0.0	1.35
	100	39.7	0.12	1	0	40.2	40.2	0.0	67.85	19.85	0.38	0	0	20.1	20.1	0.0	0.43
S1.6	50	37.8	0.10	0	0	37.8	37.8	0.0	63.24	28.6	0.32	0	0	30.3	30.3	0.0	16.19
	75	36.1	0.09	0	0	36.1	36.1	0.0	64.56	18.05	0.33	0	0	18.05	18.05	0.0	1.15
	100	36.1	0.09	0	0	36.1	36.1	0.0	100.27	18.05	0.22	0	0	18.05	18.05	0.0	0.02
S1.7	50	41.0	0.15	0	0	41.0	41.0	0.0	81.93	36.3	0.55	2	0	38.3	38.3	0.0	61.78
	75	40.0	0.16	0	0	40.2	40.2	0.0	70.12	20.3	0.54	8	0	20.7	20.7	0.0	59.90
	100	40.0	0.15	0	0	40.2	40.2	0.0	90.35	19.75	0.45	0	0	20.1	20.1	0.0	0.41
S1.8	50	35.5	0.17	18	1	39.1	39.1	0.0	71.06	22.7	0.44	4	0	28.0	28.0	0.0	17.21
	75	35.1	0.11	0	0	35.1	35.1	0.0	80.81	21.05	0.52	0	0	21.05	21.05	0.0	1.14
	100	35.1	0.10	0	0	35.1	35.1	0.0	84.67	17.55	0.35	0	0	17.55	17.55	0.0	0.07
S1.9	50	23.7	0.08	0	0	24.4	24.4	0.0	42.97	12.9	0.21	0	0	12.9	12.9	0.0	3.67
	75	23.7	0.08	0	0	24.4	24.4	0.0	106.97	10.467	0.26	0	0	10.467	10.467	0.0	2.81
	100	23.7	0.08	0	0	24.4	24.4	0.0	63.00	7.9	0.22	0	0	8.133	8.133	0.0	0.11
S2.0	50	43.3	0.15	0	0	44.7	50.2	0.123	600.00	41.0	0.71	0	0	41.0	49.934	0.218	600.00
	75	41.1	0.17	4	0	41.8	50.2	0.201	600.00	23.05	0.75	0	0	23.15	50.2	1.168	600.00

Table D.3 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
	100	40.7	0.19	4	0	41.4	58.1	0.403	600.00	20.25	0.87	0	0	20.75	20.75	0.0	16.82
S2.1	50	40.9	0.20	0	0	42.5	56.8	0.336	600.00	35.6	0.65	0	0	37.3	50.2	0.346	600.00
	75	37.9	0.27	0	0	38.6	47.9	0.241	600.00	21.3	0.80	0	0	21.75	43.0	0.977	600.00
	100	37.5	0.24	0	0	38.3	43.0	0.123	600.00	18.8	1.26	0	0	19.15	19.15	0.0	30.57
S2.2	50	41.9	0.20	0	0	42.2	42.2	0.0	232.48	38.0	0.47	8	0	40.8	40.8	0.0	26.73
	75	39.8	0.17	16	0	40.8	43.8	0.074	600.00	22.25	0.90	0	0	22.45	22.45	0.0	99.11
	100	40.3	0.18	0	0	40.7	40.7	0.0	164.66	20.05	0.63	0	0	20.35	20.35	0.0	5.66
S2.3	50	43.6	0.26	5	0	44.6	51.7	0.159	600.00	40.4	0.92	5	1	41.5	51.4	0.239	600.00
	75	41.5	0.20	0	0	41.8	51.4	0.23	600.00	22.05	0.74	0	0	23.2	51.4	1.216	600.00
	100	40.9	0.16	1	0	41.2	51.4	0.248	600.00	20.45	0.77	0	0	20.65	20.65	0.0	86.27
S2.4	50	42.6	0.20	1	0	45.8	50.5	0.103	600.00	36.8	0.71	0	0	38.6	50.5	0.308	600.00
	75	38.8	0.17	5	0	40.0	56.2	0.405	600.00	21.5	1.16	0	0	22.35	50.1	1.242	600.00
	100	38.0	0.17	5	0	38.8	55.9	0.441	600.00	18.95	0.89	0	0	19.4	19.9	0.026	600.00
S2.5	50	42.7	0.16	2	0	46.0	46.0	0.0	304.06	38.9	0.62	6	0	41.1	48.9	0.19	600.00
	75	41.1	0.20	3	0	41.3	49.309	0.194	600.00	22.9	0.64	0	0	23.25	51.4	1.211	600.00
	100	40.7	0.17	0	0	40.9	51.4	0.257	600.00	20.35	0.80	0	0	20.55	20.55	0.0	21.59
S2.6	50	35.4	0.11	0	0	35.4	35.4	0.0	58.89	35.4	0.43	0	0	35.4	35.4	0.0	1.18
	75	35.4	0.10	0	0	35.4	35.4	0.0	18.50	17.7	0.39	0	0	17.7	17.7	0.0	11.34
	100	35.4	0.10	0	0	35.4	35.4	0.0	34.79	17.7	0.33	0	0	17.7	17.7	0.0	0.04
S2.7	50	45.8	0.16	0	0	46.4	55.0	0.185	600.00	39.4	0.66	3	0	41.5	41.5	0.0	88.47
	75	41.4	0.19	0	0	41.7	101.2	1.427	600.00	22.7	0.91	0	0	22.9	50.6	1.21	600.00
	100	41.3	0.15	0	0	41.8	52.2	0.249	600.00	20.6	0.95	0	0	20.9	20.9	0.0	29.49
S2.8	50	41.5	0.18	0	0	41.5	41.5	0.0	84.94	36.9	0.90	0	0	40.4	40.4	0.0	68.49
	75	40.4	0.25	0	0	40.8	40.8	0.0	216.07	21.05	0.77	4	0	21.5	22.6	0.051	600.00
	100	40.5	0.16	0	0	40.8	40.8	0.0	137.51	19.95	0.79	0	0	20.4	20.4	0.0	4.48
S2.9	50	43.0	0.14	0	0	43.7	43.7	0.0	78.91	38.3	0.58	15	0	40.0	40.0	0.0	72.97
	75	41.0	0.15	0	0	41.2	47.5	0.153	600.00	22.45	0.68	0	0	22.8	22.8	0.0	222.24
	100	41.1	0.17	0	0	41.3	41.3	0.0	232.02	20.6	0.63	0	0	20.65	20.65	0.0	3.02
S3.0	50	15.7	0.41	0	0	15.7	15.7	0.0	308.11	8.433	0.84	0	0	8.433	8.433	0.0	17.13
	75	15.5	0.40	0	0	15.5	15.5	0.0	258.53	5.1	0.94	0	0	5.1	5.1	0.0	5.81
	100	15.5	0.39	0	0	15.5	15.7	0.013	600.00	3.925	1.02	0	0	3.925	3.925	0.0	21.23
S3.1	50	16.4	0.40	0	0	18.4	18.4	0.0	236.25	7.2	0.87	0	0	7.2	7.2	0.0	59.02
	75	16.4	0.41	0	0	18.4	18.4	0.0	233.69	4.6	0.95	0	0	4.6	4.6	0.0	18.77
	100	16.4	0.42	0	0	18.4	18.4	0.0	323.33	4.6	0.97	0	0	4.6	4.6	0.0	1.51

Table D.3 – continued from previous page

		P_1								P_∞							
		Heuristic		OBC		Solver				Heuristic		OBC		Solver			
Inst	%UT	LB	CPU	X	Y	LB	UB	GAP	CPU	LB	CPU	X	Y	LB	UB	GAP	CPU
S3.2	50	17.9	0.41	0	0	17.9	47.2	1.637	600.00	9.0	0.90	0	0	9.0	9.0	0.0	205.42
	75	17.9	0.42	0	0	17.9	34.6	0.933	600.00	6.8	0.95	0	0	6.967	6.967	0.0	27.48
	100	17.9	0.41	0	0	17.9	20.4	0.14	600.00	5.233	1.05	0	0	5.233	5.233	0.0	4.96
S3.3	50	12.2	0.44	0	0	12.2	14.7	0.205	600.00	5.7	0.94	0	0	5.7	5.7	0.0	24.69
	75	12.0	0.44	0	0	12.0	16.3	0.358	600.00	4.0	0.94	0	0	4.075	4.075	0.0	6.87
	100	12.0	0.44	0	0	12.0	12.0	0.0	456.94	3.0	1.05	0	0	3.0	3.0	0.0	48.06
S3.4	50	14.0	0.39	0	0	14.0	14.0	0.0	121.35	4.95	0.82	0	0	4.95	4.95	0.0	34.74
	75	14.0	0.38	0	0	14.0	14.0	0.0	61.17	4.667	0.93	0	0	4.75	4.75	0.0	1.15
	100	14.0	0.39	0	0	14.0	14.0	0.0	23.25	3.575	0.85	0	0	3.575	3.575	0.0	0.50
S3.5	50	16.7	0.47	0	0	16.7	30.018	0.797	600.00	8.65	1.03	0	0	8.65	8.65	0.0	62.74
	75	16.7	0.48	0	0	16.7	26.3	0.575	600.00	7.767	1.15	0	0	8.6	8.6	0.0	16.39
	100	16.7	0.47	0	0	16.7	19.4	0.162	600.00	5.567	1.12	0	0	5.733	5.733	0.0	2.40
S3.6	50	19.4	0.40	0	0	20.7	20.7	0.0	373.28	11.2	0.91	0	0	11.2	11.2	0.0	496.05
	75	19.4	0.41	0	0	20.7	20.7	0.0	540.95	7.05	1.04	43	1	8.1	8.1	0.0	87.74
	100	19.4	0.41	0	0	19.4	35.958	0.854	600.00	6.467	1.03	64	1	6.9	6.9	0.0	3.30
S3.7	50	22.1	0.45	0	0	22.1	22.3	0.009	600.00	9.133	1.05	0	0	9.9	9.9	0.0	105.24
	75	22.1	0.45	0	0	22.1	23.9	0.081	600.00	7.367	1.25	0	0	7.367	7.367	0.0	75.13
	100	22.1	0.47	0	0	22.1	22.1	0.0	219.03	7.367	0.94	0	0	7.367	7.367	0.0	0.36
S3.8	50	27.2	0.49	0	0	27.2	53.2	0.956	600.00	15.7	1.19	70	0	18.25	18.25	0.0	385.96
	75	27.2	0.48	0	0	27.2	60.0	1.206	600.00	14.7	1.28	60	1	15.65	15.65	0.0	383.18
	100	27.2	0.46	0	0	27.2	58.9	1.165	600.00	9.067	1.33	0	0	9.267	9.267	0.0	6.15
S3.9	50	4.4	0.39	0	0	4.4	4.4	0.0	46.83	2.2	0.59	0	0	2.55	2.55	0.0	4.73
	75	4.4	0.40	0	0	4.4	4.4	0.0	34.65	1.1	0.70	0	0	1.2	1.2	0.0	16.60
	100	4.4	0.40	0	0	4.4	4.4	0.0	34.76	1.1	0.79	0	0	1.1	1.1	0.0	8.55

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Équilibrage robuste de lignes de production : modèles de programmation linéaire en variables mixtes et règles de pré-traitement

Mots clés : optimisation robuste, lignes de production, rayon de stabilité, approche hybride, règles de réduction, PLNE

Résumé : Ce travail porte sur l'optimisation robuste des lignes de production au stade de la conception. La conception de telles lignes peut être interprétée comme un problème d'optimisation consistant à rechercher une configuration optimisant des objectifs individuels et à respecter les contraintes technologiques et économiques. Nous considérons deux types de lignes de production : l'assemblage et le transfert. Le premier peut être représenté comme un ensemble de stations ordonnées linéairement où les tâches sont exécutées de manière séquentielle. Le second type de ligne est constitué de machines de transfert comprenant plusieurs têtes multibroches. Toutes les tâches d'une même tête sont exécutées simultanément, tandis que les outils d'une machine fonctionnent en mode séquentiel.

Nous décrivons différentes approches permettant de modéliser l'incertitude des données dans les problèmes d'équilibrage de ligne. Notre objectif est d'identifier les approches les mieux adaptées au contexte de la conception. En particulier, l'attention se concentre sur l'approche robuste. Nous proposons un nouveau critère d'optimisation basé sur le rayon de stabilité d'une solution réalisable. Ensuite, des formulations robustes sont présentées pour la conception des lignes d'assemblage et de transfert lorsque le temps de traitement des tâches est sujet à des incertitudes. Nous développons également des méthodes heuristiques dont les résultats sont utilisés pour renforcer les modèles mathématiques. Enfin, une nouvelle méthode de résolution hybride est élaborée pour résoudre différentes variantes des problèmes de maximisation du rayon de stabilité.

Title: Robust Balancing of Production Lines: MILP Models and Pre-processing Rules

Keywords: robust optimisation, production lines, stability radius, hybrid approach, reduction rules, MILP

Abstract: This work deals with a robust optimisation of production lines at the design stage. The design of such lines can be interpreted as an optimisation problem that consists in finding a configuration optimising individual objectives and respecting technological and economic constraints. We consider two types of production lines: assembly and transfer lines. The first one can be represented as a set of linearly ordered stations where the tasks are executed sequentially. The second one is composed of transfer machines, including several multi-spindle heads. All tasks within a single head are executed simultaneously, while tools on a machine work in a sequential mode.

We describe different approaches for modelling the uncertainty of data in line balancing problems. Our objective is to identify the approaches that best fit the context of the design. In particular, the attention concentrates on the robust approach. We propose a new optimisation criterion based on the stability radius of a feasible solution. Then, robust formulations are presented for the design of the assembly and transfer lines under variations of task processing times. We also develop heuristic methods whose results are used to improve mathematical models. Finally, a new hybrid resolution method is elaborated to solve different variants of the stability radius maximisation.