STOCHASTIC ASSEMBLY LINE BALANCING PROBLEMS INVOLVING ROBOTS AND RELIABILITY RESTRICTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUHAMMET CEYHAN ŞAHİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2022

Approval of the thesis:

STOCHASTIC ASSEMBLY LINE BALANCING PROBLEMS INVOLVING ROBOTS AND RELIABILITY RESTRICTION

submitted by **MUHAMMET CEYHAN ŞAHİN** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Esra Karasakal
Head of Department, **Industrial Engineering** _____

Assist. Prof. Dr. Mustafa Kemal Tural
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

Assoc. Prof. Dr. Sedef Meral
Industrial Engineering, METU _____

Assoc. Prof. Dr. İsmail Serdar Bakal
Industrial Engineering, METU _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering, METU _____

Assist. Prof. Dr. Derya Dinler
Industrial Engineering, Hacettepe University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: MUHAMMET CEYHAN ŞAHİN

Signature :

ABSTRACT

STOCHASTIC ASSEMBLY LINE BALANCING PROBLEMS INVOLVING ROBOTS AND RELIABILITY RESTRICTION

ŞAHİN, MUHAMMET CEYHAN
M.S., Department of Industrial Engineering
Supervisor: Assist. Prof. Dr. Mustafa Kemal Tural

July 2022, 99 pages

When considering assembly processes in the manufacturing ecosystem, the task times may vary from cycle to cycle, especially in assembly lines where manual operations are abundant. Line stops, defective products, and off-line tasks caused by the uncertainty in assembly processes can be highly costly for companies. Stochastic assembly line balancing problems (SALBPs) consider the task processing times as random variables to deal with uncertainty in real-life assembly operations.

The difficulties faced due to uncertainties in assembly processes can be alleviated by using advanced technological solutions. Manufacturing companies replace human workers with robots in their assembly processes to keep up with the Industry 4.0 technological revolution and get the upper hand over competitors. A popular approach in the manufacturing industry is to design an assembly line with human-robot collaboration. In the first part of this thesis, we investigate a robotic stochastic assembly line balancing problem (RSALBP), with the motivation to observe the effects of robots on the cycle time in stochastic assembly lines where human workers and robots operate in different workstations.

In the literature, robotic assembly line balancing is only studied with deterministic task times. However, assembly line balancing contains stochastic processes in real life. We assume that the processing time of each task follows a normal distribution whose parameters depend on the type of the operator performing the task, with robots having much less (possibly zero) variation in task times than human workers. It is assumed that human workers are fully capable while robots can perform a subset of the tasks. We study type-II RSALBP, which aims to minimize the cycle time for an assembly line with stochastic task times, given a fixed number of workstations and robots, and fixed confidence levels for the workstations. This problem is NP-hard and includes non-linearity. We propose a mixed-integer second-order cone programming formulation and a constraint programming formulation to solve the problem. Instances from the literature are used to test the effectiveness of the proposed formulations. Additionally, the effects of robots on cycle times are evaluated by conducting a computational study with a comprehensive experimental design.

In the second part of this thesis, we consider a type-II stochastic assembly line balancing problem with a reliability restriction (type-II SALBP-R) where all the operators are identical. Given a fixed cycle time, the reliability of an assembly line is defined as the probability that the workload of none of the workstations exceeds the cycle time. In type-II SALBP-R, we aim to minimize the cycle time for an assembly line with stochastic task times, given a fixed number of workstations and a fixed lower bound on reliability. This problem has been investigated in only a few studies in the literature. We propose the first matheuristic for the problem and compare its performance with an existing heuristic from the literature.

Keywords: Assembly lines, Robotic assembly line balancing, Stochastic assembly line balancing, Industry 4.0, Human-robot collaboration, Reliability

ÖZ

ROBOTLARI VE GÜVENİLİRLİK KISITLAMASINI İÇEREN STOKASTİK MONTAJ HATTI DENGELEME PROBLEMLERİ

ŞAHİN, MUHAMMET CEYHAN
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Mustafa Kemal Tural

Temmuz 2022 , 99 sayfa

İmalat ekosistemindeki montaj süreçleri göz önüne alındığında, özellikle manuel işlemlerin yoğunlukta olduğu montaj hatlarında görev süreleri çevrimden çevrime değişebilir. Montaj süreçlerindeki belirsizliğin neden olduğu hat duruşları, arızalı ürünler ve hat dışı işler şirketler için oldukça maliyetli olabilir. Stokastik montaj hattı dengeleme problemleri (SALBP'ler), gerçek hayattaki montaj işlemlerindeki belirsizlikle başa çıkmak için iş sürelerini rastgele değişkenler olarak ele alır.

Montaj süreçlerindeki belirsizlikler nedeniyle yaşanan zorluklar, ileri teknolojik çözümler kullanılarak azaltılabilir. İmalat şirketleri, Endüstri 4.0 teknolojik devrimine ayak uydurmak ve rakiplerine üstünlük sağlamak için montaj süreçlerinde işçileri robotlarla değiştiriyor. İmalat endüstrisinde popüler bir yaklaşım, insan-robot işbirliği ile bir montaj hattı tasarlamaktır. Bu tezin ilk bölümünde, işçilerin ve robotların farklı iş istasyonlarında çalıştığı stokastik montaj hatlarında robotların çevrim süresi üzerindeki etkilerini gözlemlemek amacıyla bir robotik stokastik montaj hattı dengeleme problemini (RSALBP) araştırıyoruz.

Literatürde robotik montaj hattı dengeleme sadece deterministik iş süreleri ile çalışılmaktadır. Ancak montaj hattı dengeleme, gerçek hayatta stokastik süreçleri içerir. Her iş süresinin, robotların iş sürelerinde işçilerden çok daha az (muhtemelen sıfır) varyasyona sahip olduğu, parametreleri görevi gerçekleştiren operatörün türüne bağlı olan bir normal dağılım izlediğini varsayıyoruz. İnsan işçilerin tüm işleri yapma yeteneği olduğu, robotların ise işlerin bir alt kümesini yapabildiği varsayılmaktadır. İş istasyonu sayısı, robot sayısı, ve iş istasyonları için güven seviyeleri sabit kabul edilen, stokastik iş sürelerine sahip bir montaj hattı için çevrim süresini en aza indirmeyi amaçlayan tip-II RSALBP'yi inceliyoruz. Bu problem NP-zordur ve doğrusal olmayan kısıtlar içerir. Problemi çözmek için bir karma tamsayılı ikinci dereceden koni programlama formülasyonu ve bir kısıt programlama formülasyonu öneriyoruz. Önerilen formülasyonların etkinliğini test etmek için literatürden örnekler kullanılmaktadır. Ayrıca kapsamlı bir deneysel tasarım ile hesaplamalı bir çalışma yapılarak robotların çevrim süreleri üzerindeki etkileri değerlendirilmektedir.

Bu tezin ikinci bölümünde, tüm operatörlerin özdeş olduğu, bir güvenilirlik kısıtlaması ile tip-II stokastik montaj hattı dengeleme problemini (tip-II SALBP-R) ele alıyoruz. Sabit bir çevrim süresi verildiğinde, bir montaj hattının güvenilirliği, iş istasyonlarının hiçbirinin iş yükünün çevrim süresini aşmama olasılığı olarak tanımlanır. Tip-II SALBP-R'de, sabit sayıda iş istasyonu ve sabit bir güvenilirlik alt sınırı ile, stokastik iş sürelerine sahip bir montaj hattı için çevrim süresini en aza indirmeyi amaçlıyoruz. Bu problem literatürde çok az çalışmada incelenmiştir. Problem için ilk mat-sezgisel yöntemi öneriyoruz ve performansını literatürdeki mevcut bir sezgisel yöntem ile karşılaştırıyoruz.

Anahtar Kelimeler: Montaj hatları, Robotik montaj hattı dengeleme, Stokastik montaj hattı dengeleme, Endüstri 4.0, İnsan-robot işbirliği, Güvenilirlik

To my grandfather Ali

ACKNOWLEDGMENTS

This endeavor would not have been possible without my supervisor, Mustafa Kemal Tural, who gave me the chance to work with him and always trusted me. He guided me throughout my graduate studies, not only in terms of academic research but also in terms of study discipline, character, and ethics. It has always been a pleasure to work with him.

I want to thank the jury members of the thesis examining committee, Prof. Dr. Meral Azizoğlu, Assoc. Prof. Dr. Sedef Meral, Assoc. Prof. Dr. İsmail Serdar Bakal, and Assist. Prof. Dr. Derya Dinler for their precious feedback.

I am always proud of being a member of the METU-IE department as a student and a teaching assistant, where there are many respectful professors, perceptive colleagues, and outstanding students. The years I spent in this university and the department will always be the most valuable ones in my life.

I owe an acknowledgment to my friends for never stopping believing in the good in me. I would like to thank my friends Gizem Aslaner and Burak Kısaer, who have always been by my side since high school. I want to thank my friends Yağmur Caner and Tuna Berk Kaya, who supported me in every way during my undergraduate and graduate years. I would also like to thank Beril Akkaya and Can Er, with whom we spent precious days in a short time.

I would like to extend my sincere thanks to Dilay Özkan, a part of all my memories in the last three years, for being the light on my darkest days and the cheer on my brightest days.

Lastly, I would like to thank my family for their altruism and for always encouraging me to follow my dreams.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

ALBP	Assembly Line Balancing Problem
SALBP	Stochastic Assembly Line Balancing Problem
SALBP-R	Stochastic Assembly Line Balancing Problem with Reliability Restriction
RALBP	Robotic Assembly Line Balancing Problem
RSALBP	Robotic Stochastic Assembly Line Balancing Problem
NLMIP	Non-Linear Mixed-Integer Programming
MISOCP	Mixed-Integer Second-Order Cone Programming
CP	Constraint Programming
BHA	Bidirectional Heuristic Algorithm
DP	Dynamic Programming
IP	Integer Programming

CHAPTER 1

INTRODUCTION

An assembly line consists of workstations in different arrangements, where each workstation has a series of tasks to be performed repeatedly by one or more operators. Deciding on which tasks should be performed in which workstation is a fundamental problem in assembly line balancing. Assembly line balancing problems (ALBPs) can be classified according to the number of models produced on the line, the nature of task duration, the pattern of flow, and the objective function [1]. Depending on the number of product models produced on the same line, it can be a single-model or mixed-model assembly line. The processing times of the tasks can be constant or may include some randomness. In highly automated lines, it is reasonable to assume that the processing times of the tasks are constant which brings about deterministic ALBPs. However, the task times in assembly operations usually vary in real life, especially when tasks are performed manually. Problems that consider the task processing times as random variables are called stochastic assembly line balancing problems (SALBPs).

For a human worker, the processing times of tasks may vary due to several reasons including lack of experience, distraction, task complexity, or environmental factors [2]. By the nature of the process, task times may differ even when the operator is a robot. For example, a robot may spend different times for searching or finding the part to be assembled from a stack. Moreover, the orientation or condition of the part (e.g., the part can be defective) may also result in variations in task times. Similarly, the semi-assembled product on which the assembly will be done may come in front of the robot with different orientations which may also cause variations in task times. Assembly lines can be investigated in two categories: unpaced or paced assembly

lines [3]. In unpaced lines, the operator is allowed to complete the tasks assigned to the workstation before the semi-assembled product is moved to the next workstation. On the other hand, in paced lines, an operator in a workstation can complete the tasks assigned to that workstation only within a given cycle time. To deal with possible incomplete tasks due to the cycle time violation by workstations in stochastic paced lines, one common approach is assigning tasks to workstations such that the total task time of each workstation is less than or equal to the cycle time with at least a given probability [4]. In this thesis, we study two stochastic assembly line balancing problems, where in each problem the line is assumed to be paced. In the first one, we work on an SALBP with human-robot collaboration considering different cases in our experiments for the variability in the task times. In some experiments, it is assumed that the processing time of each task is a random variable whose parameters depend on the type of the operator performing the task with robots having much less, but non-zero, variation in task times than human workers. In some other experiments, the task times are constant for robots, while they are still random variables for human workers. In the second one, we work on an SALBP involving identical operators where there is a restriction on the probability that the cycle time will be violated in at least one workstation.

The flow pattern, which depends on the arrangement of workstations, is another determinant to classify assembly lines [5]. For example, in a U-shape layout, the flow takes place along a U-shaped line and operators that are located inside this “U" shape can perform tasks on both sides of the line. In a straight-line layout; however, the flow occurs along a straight, one-way line. Another important aspect of the classification of ALBPs in the literature is the objective function. ALBPs are studied with many different objectives such as minimizing cost, maximizing line efficiency, and minimizing idle time subject to some constraint sets accordingly. However, minimizing the number of workstations or the cycle time are the most common objectives. As Scholl [6] presented, in type-I ALBP, the cycle time is given, and the aim is to find the minimum number of workstations, whereas, in type-II ALBP, the objective is to find the minimum cycle time for a given number of workstations.

Assembly lines are essential parts of manufacturing processes. With the Industry 4.0 paradigm, significant changes are observed in the production ecosystem [7]. An as-

sembly company needs to catch up with the speed and quality standards of its supply chain to avoid problems. Holding on in the long term in a supply chain that has assimilated Industry 4.0 depends on catching the trends promptly and adapting quickly. In addition to what Industry 4.0 brings, production volumes need to increase to meet the increasing demand. Also, products are diversified in line with customer needs, which makes the assembly processes more complicated. When we add the increasing competition to all these, an assembly company cannot survive for long with traditional methods. As a reflection, robots are replacing human workers day by day in assembly processes bringing in several benefits, such as cost reduction and quality improvement. Moreover, with robots, variations in processing times of the tasks can be reduced which may possibly result in decreased cycle times. In this study, the cost component of the assembly line is neglected. The analysis in this thesis is made by changing the number of robots used in the assembly line. We expect the decision maker to evaluate the cost in each case and determine the number of robots to be used in the assembly line.

An assembly line is called a robotic assembly line if it consists of workstations where robots perform tasks instead of some or all of the human workers. A robotic assembly line balancing problem (RALBP) studies the traditional assembly line balancing problems with the challenges and improvements brought by robots replacing human workers. RALBP with stochastic task times has not been studied in the literature to the best of our knowledge. To fill in this gap in the literature, we first investigate in this thesis the effects of robots on the cycle time in type-II robotic stochastic assembly line balancing problem (type-II RSALBP), where we assume that the processing time of each task follows a normal distribution whose parameters depend on the type of the operator (human worker or robot) performing the task, where it is possible to set the standard deviations of the task times to zero for robots. To this end, we first propose three mathematical programming formulations (one of which is a transient formulation) to solve type-II RSALBP to optimality. We then set up an experimental design by varying several parameters of the problem including the number of robots, capabilities of them, and the parameters of the normal distributions and test the models on four well-known assembly line balancing problems from the literature.

In the second part of this thesis, we investigate a stochastic assembly line balancing

problem with identical operators, where there is a lower bound on the reliability of the assembly line (type-II SALBP-R). In this problem, we aim to find the lowest possible cycle time such that the probability that the cycle time is not violated in any of the workstations is greater than or equal to the given lower bound on the reliability.

The organization of the remainder of the thesis is as follows. In Chapter 2, we provide a review of the related literature. Chapter 3 starts with the description of type-II RSALBP in Section 3.1. A non-linear mixed-integer programming and a mixed-integer second-order cone programming formulations are presented for type-II RSALBP in Section 3.1.1. In Section 3.1.2, a constraint programming formulation is developed for type-II RSALBP. Section 3.2 presents the computational studies for type-II RSALBP: experimental settings in Section 3.2.1, illustrative examples in Section 3.2.2, and computational results in Section 3.2.3. Type-II SALBP-R is described in Section 4.1 of Chapter 4. A constraint programming formulation for type-II SALBP is given in Section 4.1.1, which is used in the proposed matheuristic algorithm for type-II SALBP-R. This matheuristic is described in Section 4.1.2. Results of the computational experiments for type-II SALBP-R are given in Section 4.2. Conclusion and some future research directions are provided in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

The progressive assembly of products manufactured by operators has been practiced in the industry since the times of Henry Ford. The assembly line balancing problem is a significant manufacturing problem that emerged in the early 1950s, and it was formulated mathematically by Salveson (1955) [8] for the first time. Since then, different types of ALBPs have been studied in the academia and industry. To the best of our knowledge, however, there are no studies directly related to RSALBP, and there is only a few studies on type-II SALBP-R in the literature. Therefore, we review the relevant literature for SALBP and RALBP.

2.1 Review on SALBP

SALBP was studied by Moodie and Young (1965) [9] for the first time. The authors assumed that the task times are normally distributed random variables and developed a heuristic method which assigns tasks to workstations in such a way that the total task time of each workstation, i.e., workload, is less than or equal to a given cycle time with probability at least a given constant α. This given probability α is called the confidence level throughout this thesis. After this study, different types of SALBPs were investigated in the literature.

A heuristic procedure to minimize the sum of incompletion cost and labor cost of a paced assembly line with stochastic task times was developed by Kottas and Lau [10]. Incompletion cost occurs when a task and its followers cannot be completed within the given cycle time and hence have to be processed off the line. Labor cost on the other hand is taken as the number of workstations multiplied by the cycle time. A

bidirectional heuristic algorithm (BHA) was developed by Liu et al. [11] for type-II SALBP-R to minimize the cycle time of a stochastic assembly line with normally distributed task times. The assembly line reliability, which is the probability that none of the workstations' workloads exceeds the cycle time, and the number of workstations are predefined. The algorithm involves a bidirectional assignment procedure to assign tasks to workstations, and a trade and transfer procedure to balance the workloads of the workstations.

In the study of Carraway [12], the objective was to minimize the number of workstations for a given cycle time and a confidence level α. The author proposed a dynamic programming (DP) approach which is an extension of the DP approach of Held et al. [13] that was developed for the deterministic ALBP. Nkasu and Leung [14] proposed a methodology for stochastic assembly line balancing where different probability distributions (exponential, gamma, normal, uniform, and Weibull) and different objective functions (including minimizing the number of workstations, minimizing the balance delay, and minimizing the cycle time) were considered. W. Zhang et al. [15] proposed a multi-objective hybrid evolutionary algorithm to minimize the cycle time and the total processing cost of the line, considering the task times as uniformly distributed random variables.

Baykasoğlu and Özbakır [16] used a multiple-rule-based genetic algorithm to balance U-shaped assembly lines assuming that each task time follows a normal distribution. Özcan et al. [17] studied a stochastic U-shaped mixed-model assembly line balancing and sequencing problem with normally distributed task times and developed a genetic algorithm for its solution. Zacharia and Nearchou [18] used a genetic algorithm to balance single-model straight assembly lines considering fuzzy task times. They employed a multi-objective method to solve the fuzzy SALBP where the minimization of the cycle time is considered as the primary objective and minimization of the smoothness index or balance delay as the secondary ones.

Constraint programming (CP) has been used as a convenient solution approach in several assembly line balancing problems. It was first used by Bockmayr and Pisaruk (2001) [19] to present a hybrid solution approach for ALBPs, combined with an integer programming (IP) formulation. The authors developed a branch-and-cut

algorithm with an IP formulation to solve a deterministic ALBP in which CP is used to prune the search tree. The objective was to minimize the number of workstations, where the workload of any workstation cannot exceed the given cycle time. Although several following studies used CP in deterministic ALBPs, there are relatively less number of studies in the literature which use CP in the context of SALBPs. Pınarbaşı et al. [20] applied CP to an SALBP to minimize the smoothness index of a single-model, straight assembly line. They considered variations not only in task times but also in flow processes between the workstations of the assembly line. Flow process variations include the arrival process variations to a workstation and the departure process variations from a workstation. They proposed an algorithm which integrates CP and queuing theory. In this algorithm, CP was used to implement task assignments to workstations, and queuing theory was used to evaluate the performance of the assembly line under these task assignment combinations. Pınarbaşı and Alakaş [21] studied type-II SALBP with normally distributed task times. They developed four different mathematical formulations which aim to minimize the cycle time of the line with given number of workstations and a confidence level α: a non-linear mixed integer programming (NLMIP) model, a CP model, and linear approximations of both NLMIP and CP models. They computationally compared the objective function values (cycle time) of these solution approaches against that of the BHA proposed by Liu et al. [11] and show that only the CP model outperforms the BHA.

Despite all these studies, SALBP has been studied in the literature considering only human workers as operators, not robots. Moreover, type-II SALBP-R has only been investigated in the study of Liu et al. [11], where a heuristic procedure is developed. This thesis contributes to the literature by proposing the first matheuristic for type-II SALBP-R. We next provide a review of the studies on RALBPs in the literature.

2.2 Review on RALBP

Beginning with the use of robots in assembly processes in the industry, RALBP has found a place in academic studies. This has gained momentum with the increase in the use of robots in manufacturing enterprises after the adoption of Industry 4.0 paradigm. The RALBP was first studied by Rubinovitz et al. (1993) [22]. The

authors proposed a heuristic algorithm to assign a robot to each workstation among a number of different types of robots and to assign tasks to these workstations so that the number of workstations is minimized for a given cycle time, i.e., the authors studied a type-I RALBP. They assumed that each task time is deterministic and depend on the type of the robot that processes the task. After this study, different types of RALBPs were investigated in the literature.

Nicosia et al. [23] studied a problem similar to a RALBP. The authors proposed a DP algorithm to assign a set of machines to workstations and to assign tasks to these workstations in such a way that the workload in each workstation is less than or equal to a given cycle time. It is assumed that each machine has an associated cost and the authors aimed to minimize the total cost of assigned machines to the workstations. Levitin et al. [24] assumed that robots have different capabilities and proposed a genetic algorithm to minimize the cycle time in a robotic assembly line. Nilakantan et al. [25] considered an RALBP where it takes each robot a certain amount of time and energy to process each task. Two objectives are considered; namely, the minimization of the cycle time and energy consumption, assuming a fixed number of workstations which is equal to the number of robots. The authors proposed a particle swarm optimization algorithm for each of the objectives. Minimization of the total energy consumption and makespan in a mixed-model U-shaped RALBP was studied by B. Zhang et al. [26] with a solution approach based on a mathematical model and a dragonfly algorithm. Here, the makespan is the completion time of the last model on the robotic assembly line. Nilakantan et al. [27] developed a mathematical model and proposed a multi-objective co-operative co-evolutionary algorithm to minimize the total operation and standby power consumptions of robots in assembly lines and to maximize the line efficiency simultaneously. In the study of Li et al. [28], robot purchasing costs are considered as well as the setup times, which depend on the sequence of the tasks processed on the line. The objective is minimizing total purchasing cost and cycle time. They proposed an elitist non-dominated sorting genetic algorithm and an improved multi-objective artificial bee colony algorithm to solve the problem.

In most of the studies on robotic assembly line balancing, only robots are considered to perform the tasks in workstations. There have also been studies that consider collaboration between human workers and robots. For a general production environ-

ment, El Zaatari [29] classified collaboration in four different scenarios, based on the way human workers and robots operate together: supportive, simultaneous, sequential, and independent. In the supportive scenario, the human worker and robot work on the same process of the same work piece, while in the sequential scenario, they work on different but consecutive processes of the same work piece. The robot and human worker work on the same work piece but nonsuccessive processes in the simultaneous scenario, whereas they independently work on different work pieces in the independent scenario. In the context of assembly line balancing, Chutima [30] provides a detailed classification of different types of human-robot collaboration. We do not go into such a detailed classification here, but mention that human worker and robot can operate in the same workstation (they may or may not be allowed to work on the same tasks). On the other hand, human worker and robot can operate in parallel or serial workstations.

Weckenborg et al. [31] assume that human workers are fully capable, whereas robots are able to perform a subset of the tasks. Some of the tasks that a robot cannot perform alone can be performed in collaboration with a human worker. When a task is assigned to a workstation where a human worker and a robot both exist, the task can be performed in collaboration or one of the operators (human worker or robot) can perform the task. The processing times may vary depending on the operator(s) performing the tasks: human worker, robot, or both. The authors proposed a hybrid genetic algorithm to assign a fixed number robots to workstations and tasks to operators to minimize the cycle time.

In the study of Li et al. [32], different types of robots with different task processing times and purchasing costs are considered. It is allowed for a human worker and a robot to work in the same workstation. In this case, they cannot work on different tasks simultaneously, but instead, they either work on different tasks one after another or work on the same task. The authors proposed a multi-objective migrating bird optimization algorithm to minimize the total purchasing cost of the robots and the cycle time of the assembly line.

Samouei and Ashayeri [33] defined skill levels (low, medium, high) for human workers. In a workstation, a human worker or a robot may operate alone, or an assisting

robot may help a human worker. When a human worker and a robot are assigned to the same workstation, they work on the same tasks collaboratively. Different costs and task processing times are assumed depending on the operator(s) and the skill levels of the human workers. The authors developed two mathematical models for mixed-model assembly line balancing to assign operators from a limited number of different types of operators, and tasks to workstations. The first model minimizes the total cost for a given cycle time and the second one minimizes a weighted sum of the total cost and the cycle time.

Çil et al. [34] studied a mixed-model assembly line balancing problem considering human-robot collaboration with the objective of minimizing the cycle time. In this study, a human worker and a robot can be assigned to the same workstation, or each can work in a workstation alone. When assigned to the same workstation, a human worker and a robot are allowed to work on different tasks one after another, i.e., they cannot work simultaneously on different tasks or on the same task together. The authors implemented a bee algorithm and an artificial bee colony algorithm to assign different types of operators to the workstations and tasks to operators to minimize the cycle time of the mixed-model assembly line.

In summary, in all these studies, deterministic task times were considered and RALBP with stochastic task times appears to have never been studied. This thesis contributes to the literature by examining a stochastic assembly line balancing problem with human-robot collaboration where human workers and robots operate in different workstations. It is assumed that the variations of the task times are much less (possibly zero) for robots than human workers. The objective is to minimize the line's cycle time for a given confidence level α. We developed two novel mathematical models (and a transition model) to formulate this problem and solve several problem instances from the literature to optimality. We interpreted the effects and contributions of different types of robots on cycle times of assembly lines by conducting a computational study with a comprehensive experimental design, in which we variate the confidence level, robots' capabilities, mean and variances of the task times, and the number of robots for four different problem instances. In the next chapter, we first formally define type-II RSALBP and then present the mathematical programming formulations we developed for the problem.

CHAPTER 3

ROBOTIC STOCHASTIC ASSEMBLY LINE BALANCING

3.1 Problem Definition and Mathematical Formulations for Type-II RSALBP

In the first part of this thesis, we consider a type-II robotic stochastic assembly line balancing problem; namely, type-II RSALBP. In this problem, there are m serially located workstations. In r of these workstations, the operators are to be robots; while human workers are to perform the tasks in the remaining $m - r$ workstations. It is assumed that all robots are identical and their capabilities are limited (i.e., they are able to perform a subset of the tasks). On the other hand, human workers, who are also identical, are fully capable.

Each task has associated with it a set of immediate predecessors and a task time which is assumed to be normally distributed whose parameters (mean and standard deviation) depend on the type of the operator (human worker or robot) performing the task. In our computational experiments, the variations of the task times (i.e., standard deviations) are assumed to be much less (possibly zero) in robots than human workers. A task cannot be processed before all its immediate predecessors are finished. The workload of a workstation is the sum of the processing times of the tasks that are assigned to that workstation. As the task times are random, the workload of a workstation is also a random variable. For this reason, as commonly done in the stochastic assembly line balancing literature, we assume that a confidence level α is given and for each workstation, the probability that the workload exceeds the cycle time should be kept less than or equal to $1 - \alpha$.

In type-II RSALBP, the aims are to allocate given numbers of robots and human workers to workstations and assign tasks to workstations so as to minimize the cycle time

such that for each workstation, the probability that the workload of the workstation is less than or equal to the cycle time is at least α.

The assumptions of type-II RSALBP are given below.

1. Only one type of product or different products with very similar assembly features are assembled.
2. Assembly line is one-sided, paced, and straight.
3. Number of workstations is known and given.
4. Precedence relations of the tasks are known and given.
5. A task can be assigned to only one workstation and cannot be divided.
6. Each workstation can have only one human worker or a robot.
7. Number of robots is known and given.
8. Human workers are identical and capable of doing all tasks.
9. Robots are identical and their capabilities are known.
10. Task times are independent and normally distributed random variables.
11. Means and variances of task times are known and depend on the type of the operator.
12. The breakdown and maintenance of robots are not considered.
13. The failures, costs, and defective products led by incomplete tasks are neglected.

Pınarbaşı and Alakaş [21] studied type-II SALBP which is NP-hard because of its combinatorial nature [6]. The problem studied in our thesis; namely, type-II RSALBP, is also NP-hard, as it generalizes type-II SALBP. Pınarbaşı and Alakaş [21] formulated type-II SALBP as NLMIP and CP models. The NLMIP model was solved on some instances from the literature using BARON solver [35]. Inspired by this study, we first formulated type-II RSALBP as an NLMIP model and used BARON solver

to solve some instances. The results, however, were not promising. Therefore, we developed two other formulations: a mixed-integer second-order cone programming (MISOCP) formulation (see Section 3.1.1), and a CP formulation (see Section 3.1.2). Note that our CP formulation is not based on the formulation of Pınarbaşı and Alakaş [21] who do not consider robots in their study. Detailed comparison of the results obtained from these three formulations is given in Section 3.2.3.1.

The following notation is common in the proposed formulations.

Indices:

i, k: tasks (1,...,n)

j: workstations (1,...,m)

l: operator type (l=1: robot; l=2: human worker)

Parameters:

n: number of tasks

m: number of workstations

r: number of robots

α: confidence level

μ_{il}: mean processing time of task i for an operator of type l

σ_{il}^2: variance of task i for an operator of type l

$P(i)$: set of immediate predecessors of task i

$$Cap(i) = \begin{cases} 1, & \text{if task } i \text{ can be performed by robots} \\ 0, & \text{otherwise} \end{cases}$$

3.1.1 NLMIP and MISOCP Formulations for Type-II RSALBP

The following decision variables are used in both NLMIP and MISOCP formulations.

Decision variables:

ct: cycle time

$$x_{ijl} = \begin{cases} 1, & \text{if task } i \text{ is assigned to workstation } j \text{ with operator type } l \\ 0, & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, & \text{if a robot is assigned to workstation } j \\ 0, & \text{otherwise} \end{cases}$$

In type-II RSALBP, task times are assumed to be independent and normally distributed random variables. As a result, the workload of each workstation, which is the sum of the processing times of the tasks that are assigned to the workstation, is also a normally distributed random variable. In this problem, the workload of each workstation should be less than or equal to the cycle time with at least a given probability α. Let w_j denote the workload of workstation j. We have the following constraint for each workstation.

$$P(w_j \leq ct) \geq \alpha \qquad \forall j\,. \tag{3.1}$$

Inequality (3.1) implies that the constraint $w_j \leq ct$ holds for any j with a probability greater than or equal to α. Let the mean and standard deviation of w_j be μ_j and σ_j, respectively. Inequality (3.1) can be written as

$$P\left(Z \leq \frac{ct - \mu_j}{\sigma_j}\right) \geq \alpha \qquad \forall j\,, \tag{3.2}$$

where Z follows the standard normal distribution.

Letting $\Phi(\cdot)$ denote the cumulative distribution function of the standard normal distribution, inequality (3.2) can be rewritten as

$$\frac{ct - \mu_j}{\sigma_j} \geq \Phi^{-1}(\alpha) \qquad \forall j \tag{3.3}$$

which is equivalent to

$$ct \geq \mu_j + \Phi^{-1}(\alpha)\sigma_j \qquad \forall j\,. \tag{3.4}$$

We know that w_j is the sum of the processing times of the tasks that are assigned to workstation j. Task assignments to workstations are represented by binary decision variables x_{ijl}'s. x_{ijl} takes the value of 1 if task i is assigned to workstation j with

operator type l and 0 otherwise. The distribution of the workload of workstation j can be expressed as

$$w_j \sim N\left(\sum_{i=1}^{n}\sum_{l=1}^{2}\mu_{il}x_{ijl}, \sum_{i=1}^{n}\sum_{l=1}^{2}\sigma_{il}^2 x_{ijl}\right) \qquad \forall j\,.$$

Finally, we can rewrite (3.4) using the mean and variance of w_j as

$$ct \geq \sum_{i=1}^{n}\sum_{l=1}^{2}\mu_{il}x_{ijl} + \Phi^{-1}(\alpha)\cdot\sqrt{\sum_{i=1}^{n}\sum_{l=1}^{2}\sigma_{il}^2 x_{ijl}} \qquad \forall j\,. \tag{3.5}$$

Using constraint (3.5), an NLMIP formulation for type-II RSALBP is given next.

$$\text{min} \quad ct \tag{3.6}$$

s.t.

$$ct \geq \sum_{i=1}^{n}\sum_{l=1}^{2}\mu_{il}x_{ijl} + \Phi^{-1}(\alpha)\cdot\sqrt{\sum_{i=1}^{n}\sum_{l=1}^{2}\sigma_{il}^2 x_{ijl}} \qquad \forall j\,, \tag{3.7}$$

$$\sum_{j=1}^{m}\sum_{l=1}^{2}x_{ijl}\cdot j \geq \sum_{j=1}^{m}\sum_{l=1}^{2}x_{kjl}\cdot j \qquad \forall k \in P(i), \forall i\,, \tag{3.8}$$

$$\sum_{j=1}^{m}\sum_{l=1}^{2}x_{ijl} = 1 \qquad \forall i\,, \tag{3.9}$$

$$Cap(i) \geq \sum_{j=1}^{m}x_{ij1} \qquad \forall i\,, \tag{3.10}$$

$$y_j \geq x_{ij1} \qquad \forall i,j\,, \tag{3.11}$$

$$1 - y_j \geq x_{ij2} \qquad \forall i,j\,, \tag{3.12}$$

$$\sum_{j=1}^{m}y_j = r\,, \tag{3.13}$$

$$x_{ijl} \in \{0,1\} \qquad \forall i,j,l\,, \tag{3.14}$$

$$y_j \in \{0,1\} \qquad \forall j\,. \tag{3.15}$$

The objective function (3.6) minimizes the cycle time. Constraint (3.7) ensures that the cycle time cannot be exceeded by any workload with a given confidence level α. Note that this constraint is non-linear. Constraint (3.8) indicates that a task cannot be assigned to an earlier workstation than its immediate predecessor. Constraint (3.9) guarantees that each task is assigned to one workstation. Constraint (3.10) ensures that a task is not assigned a workstation with a robot if the robots are not capable of doing it. Constraints (3.11) and (3.12) relate the x and y decision variables. Constraint (3.13) makes sure that exactly r robots are used. Constraints (3.14) and (3.15) are binary restrictions.

We next propose an MISOCP formulation for type-II RSALBP. A second order cone programming (SOCP) problem is a convex optimization problem which minimizes a linear objective function over second-order cone constraints of the form

$$\|Ax + b\| \leq c^T x + d\,. \tag{3.16}$$

In constraint (3.16), x stands for the vector of decision variables, A is a parameter matrix, b and c represent parameter vectors in appropriate sizes, d indicates a scalar parameter, and $\|\cdot\|$ is the Euclidean norm. Note that linear constraints are SOCP constraints [36]. An MISOCP problem is an SOCP problem where some of the variables are restricted to be integer-valued.

In the NLMIP formulation of type-II RSALBP that we proposed, the only nonlinear constraint is constraint (3.7). Assuming that $\alpha \geq 0.5$ and hence $\phi^{-1}(\alpha) \geq 0$, this constraint can be written as an SOCP constraint as follows.

$$ct \geq \sum_{i=1}^{n}\sum_{l=1}^{2} \mu_{il}x_{ijl} + \Phi^{-1}(\alpha)\cdot \left\| \begin{pmatrix} \vdots \\ \sigma_{il}x_{ijl} \\ \vdots \end{pmatrix}_{i,l} \right\| \qquad \forall j\,. \tag{3.17}$$

Therefore replacing constraint (3.7) in the NLMIP formulation of type-II RSALBP by constraint (3.17), we obtain an MISOCP formulation of the problem. MISOCP problems can be solved using branch-and-bound by solving an SOCP problem at each

node of the branch-and-bound tree. Such solution approaches have been implemented in some commercial solvers which are used to solve several difficult problems in the literature (see e.g., [37, 38]).

3.1.2 CP Formulation for Type-II RSALBP

Some combinatorial optimization problems (e.g., RSALBPs) can be challenging to solve using conventional mathematical programming formulations, especially if the formulations are weak. Constraint programming is an alternative solution approach to formulate and solve combinatorial optimization problems which borrows a wide range of techniques from operations research, computer science, artificial intelligence, and logic. CP has been successfully used to solve different types of combinatorial optimization problems including scheduling problems [39] and ALBPs [40]. Finding solutions to CP problems is possible via commercial solvers like CPLEX CP Optimizer. A CP formulation can contain discrete decision variables, (linear or non-linear) expressions involving these variables, and constraints involving arithmetic and logical operators. A CP formulation for type-II RSALBP is provided next after introducing its decision variables.

Decision variables:

ct: cycle time

x_i: the workstation task i is assigned to

$$y_j = \begin{cases} 1, & \text{if a robot is assigned to workstation } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{if task } i \text{ is assigned to a robot,} \\ 0, & \text{otherwise.} \end{cases}$$

Auxiliary decision variables:

w_j: the cut-off value of the workload of workstation j

$$E_{ijl} = \begin{cases} 1, & \text{if task } i \text{ is assigned to workstation } j \text{ with an operator of type } l, \\ 0, & \text{otherwise.} \end{cases}$$

$$\min \quad ct = \max_{j}\{w_j\} \tag{3.18}$$

s.t.

$$E_{ij1} = (z_i = 1 \wedge x_i = j) \qquad \forall i, j\,, \tag{3.19}$$

$$E_{ij2} = (z_i = 0 \wedge x_i = j) \qquad \forall i, j\,, \tag{3.20}$$

$$w_j = \sum_{i=1}^{n}\sum_{l=1}^{2} \mu_{il} E_{ijl} + \Phi^{-1}(\alpha) \cdot \sqrt{\sum_{i=1}^{n}\sum_{l=1}^{2} \sigma_{il}^2 E_{ijl}} \qquad \forall j\,, \tag{3.21}$$

$$x_i \geq x_k \qquad \forall k \in P(i), \forall i\,, \tag{3.22}$$

$$Cap(i) \geq z_i \qquad \forall i\,, \tag{3.23}$$

$$(z_i = 1 \wedge y_j = 0) \vee (z_i = 0 \wedge y_j = 1) \implies x_i \neq j \qquad \forall i, j\,, \tag{3.24}$$

$$\sum_{j=1}^{m} y_j = r\,, \tag{3.25}$$

$$x_i \in \{1, ..., m\} \qquad \forall i\,, \tag{3.26}$$

$$y_j \in \{0, 1\} \qquad \forall j\,, \tag{3.27}$$

$$z_i \in \{0, 1\} \qquad \forall i\,, \tag{3.28}$$

$$E_{ijl} \in \{0, 1\} \qquad \forall i, j, l\,, \tag{3.29}$$

$$w_j \geq 0 \qquad \forall j\,. \tag{3.30}$$

The objective function (3.18) minimizes the cycle time, which is equal to the largest cut-off value of the workloads of the workstations. Auxiliary variable E_{ijl} is a set of logical expressions indicated by Constraints (3.19) and (3.20). It takes the value of 1 if task i is assigned to workstation j with an operator of type l, and 0 otherwise. Value of E_{ijl} depends on the statement with a conjunction ($\wedge$) operator: E_{ijl} takes the value of 1, if and only if the statements on both sides of the conjunction operator are true. Constraint (3.21), which is a non-linear expression, keeps each of the stochastic workloads less than or equal to the cycle time with a given confidence level α. Constraint (3.22) is to satisfy the precedence relations between tasks. Constraint (3.23) prevents task i to be assigned to a robot, if robot(s) are not capable of performing task i. Constraint (3.24), which is a logical expression, relates the task assignments to robots, and robot assignments to workstations. Here, the value of x_i depends on the statement with a disjunction ($\vee$) and an implication ($\implies$) operators: x_i can take

the value of j, if and only if the statements on both sides of the disjunction are false. Constraint (3.25) makes the number of workstations with a robot equal to the given number of robots. Constraints (3.26)–(3.30) are domain restrictions.

In the next section, we first detail the settings used in our computational experiments and provide some illustrative examples. Then we compare the performances of the three proposed mathematical programming formulations for type-II RSALBP on the instances selected from the literature.

3.2 Computational Study for Type-II RSALBP

3.2.1 Experimental Settings

In our computational experiments, we used a computer with Intel Xeon E2246G, a 3.6 GHz processor, and 16 GB RAM. The proposed MISOCP and CP formulations are coded in ILOG CPLEX Optimization Studio 20.1 [41], while the NLMIP formulation is coded in GAMS IDE 23.9.5. BARON solver, CPLEX Optimizer, and CPLEX CP Optimizer are used to solve the NLMIP, MISOCP, and CP formulations, respectively. We set the CPU time limit to 3600 seconds for all experiments. Four well-known problem instances from the assembly line balancing literature are used: 25-task problem from the study of Nkasu and Leung [14]; Sawyer's 30-task, Gunther et al.'s 35-task, and Kilbridge and Wester's 45-task problems [42]. Deterministic processing times of the tasks (μ_{i2}) and their immediate predecessors ($P(i)$) are given for the 25-task problem in Table 3.2, and for the 30-task, 35-task, and 45-task problems in Tables A.1–A.3 in Appendix A. Precedence diagrams of the problem instances are used directly, and deterministic task processing times in the benchmark problems are used as the mean task processing times (μ_{i2}) for human workers. The standard deviations of the task times for human workers (σ_{i2}) are obtained using the coefficient of variation (cv) as shown in Equation (3.31). In our computational experiments, we used two different cv values.

$$\sigma_{i2} = cv \cdot \mu_{i2} \qquad \forall i\,. \tag{3.31}$$

Mean task processing times of robots (μ_{i1}) are assumed to be a constant multiple of those of the human workers (see Equation (3.32)). This constant is called as the coefficient of mean of robots (cmr). Note that if $cmr < 1$, then the robots work faster than human workers on average. On the other hand when $cmr > 1$, then the robots are slower on average. In our computational experiments, we tried four different cmr values.

$$\mu_{i1} = cmr \cdot \mu_{i2} \qquad \forall i\,. \tag{3.32}$$

The standard deviations of the task times of robots (σ_{i1}) are obtained using Equation (3.33), where cvr denotes the constant of variability for robots. If $cvr < 1$, then the variations in the task times of robots are less than those of the human workers. If cvr is equal to zero, on the other hand, then the task times for robots become deterministic. In our computational experiments, we tried two different cvr values.

$$\sigma_{i1} = cvr \cdot \sigma_{i2} \qquad \forall i\,. \tag{3.33}$$

The capabilities of robots are stated as a percentage of the total number of tasks. According to this percentage, the task capabilities are set by randomly generating a 0-1 vector called the capability vector whose size is equal to the number of tasks. The fraction of ones in this capability vector is equal to the capability of robots. For example, in a 30-task problem where the robots are capable of doing 60% of all tasks, the capability vector consists of 18 ones and 12 zeros. To better compare the results, for each problem instance, 10 different randomly generated capability vectors are used. These randomly generated capabilities of the robot(s) to perform each task for all problem instances can be seen in Tables B.1–B.8 in Appendix B. For instance, for the above example, while all other parameters are the same, 10 different problems are solved for 10 differently generated capability vectors consisting of 18 ones and 12 zeros.

In our computational study, we set an experimental design by changing several parameters. This allows us to understand the effects of different parameters on the

solutions of type-II RSALBP. The parameters used in our experimental design and their corresponding values are shown in Table 3.1.

Table 3.1: Value sets of parameters in experimental design

Parameter	Values
cv	0.2, 0.4
cmr	0.6, 0.8, 1.0, 1.2
cvr	0.0, 0.5
Capability	40%, 60%, 80%, 100%
α	0.90, 0.95
r	1, 2

In the next section, we provide some illustrative examples and in Section 3.2.3, we provide the results of our comprehensive computational experiments.

3.2.2 Illustrative Examples

In this section, we provide some illustrative results on the 25-task problem of Nkasu and Leung [14]. There are 6 workstations in the original problem. Deterministic task times (μ_{i2}) of the tasks and their immediate predecessors ($P(i)$) are given in Table 3.2.

Table 3.2: Deterministic task times and immediate predecessors of the tasks for the 25-task problem

Task	μ_{i2}	$P(i)$	Task	μ_{i2}	$P(i)$
1	6	-	14	5	13
2	6	1	15	4	14
3	5	1	16	12	10, 12, 15
4	8	1	17	10	16
5	9	2	18	5	17
6	5	3	19	15	17
7	4	3	20	10	17
8	5	4	21	5	18
9	6	4	22	6	19, 20, 21
10	10	5	23	10	22
11	5	6, 7	24	5	22, 23
12	6	9	25	8	24
13	2	8, 11			

Our aim is to assign tasks with stochastic processing times to the workstations; and assign r robots to workstations and $m-r$ human workers to the rest while minimizing the cycle time. For human workers, the mean task times are given in Table 3.2. If we take cv as 0.2, the standard deviation of the task time of the first task becomes 1.2, which is calculated according to Equation (3.31). For a robot, on the other hand, when cmr is 0.6, the mean task time for task-1 is calculated as 3.6 using Equation (3.32). Taking cvr as 0.5, the standard deviation of the task time for task 1 is calculated as 0.6 for robots according to Equation (3.33). With a similar approach, all means and standard deviations of the task times for both robots and human workers are calculated using the parameters $n = 25$, $cv = 0.2$, $cmr = 0.6$, and $cvr = 0.5$ and are reported in Table 3.3. Here a dash means that the robots are not capable of doing the corresponding task. Moreover, we took the value of the confidence level α as 0.90 and the capabilities of robots as $\%80$ in our illustrative examples.

Table 3.3: Means and standard deviations of the task times used in illustrative examples

	Robots		Human workers	
Task	μ_{i1}	σ_{i1}	μ_{i2}	σ_{i2}
1	3.6	0.6	6.0	1.2
2	3.6	0.6	6.0	1.2
3	3.0	0.5	5.0	1.0
4	4.8	0.8	8.0	1.6
5	5.4	0.9	9.0	1.8
6	3.0	0.5	5.0	1.0
7	2.4	0.4	4.0	0.8
8	3.0	0.5	5.0	1.0
9	-	-	6.0	1.2
10	6.0	1.0	10.0	2.0
11	-	-	5.0	1.0
12	3.6	0.6	6.0	1.2
13	-	-	2.0	0.4
14	3.0	0.5	5.0	1.0
15	-	-	4.0	0.8
16	7.2	1.2	12.0	2.4
17	6.0	1.0	10.0	2.0
18	3.0	0.5	5.0	1.0
19	9.0	1.5	15.0	3.0
20	-	-	10.0	2.0
21	3.0	0.5	5.0	1.0
22	3.6	0.6	6.0	1.2
23	6.0	1.0	10.0	2.0
24	3.0	0.5	5.0	1.0
25	4.8	0.8	8.0	1.6

As the first example, we consider the case when all the operators are human workers,

i.e., $r = 0$. Figure 3.1 displays the optimal assignments of 25 tasks to 6 workstations which is obtained by solving type-II SALBP. As the task times are random variables, the workload of each workstation is also a random variable. The values shown in black rectangles at the bottom of Figure 3.1 represent the cut-off values (i.e., the $(100\alpha)^{th}$ percentiles) of the workloads that can be exceeded with probability 0.10 $(1 - \alpha)$. The optimal cycle time of type-II SALBP will be equal to the maximum of these cut-off values which will guarantee that the workload of each workstation will be less than or equal to the cycle time with probability at least 0.90 (α). Hence, in this example, the optimum cycle time is found as 34.80 seconds.

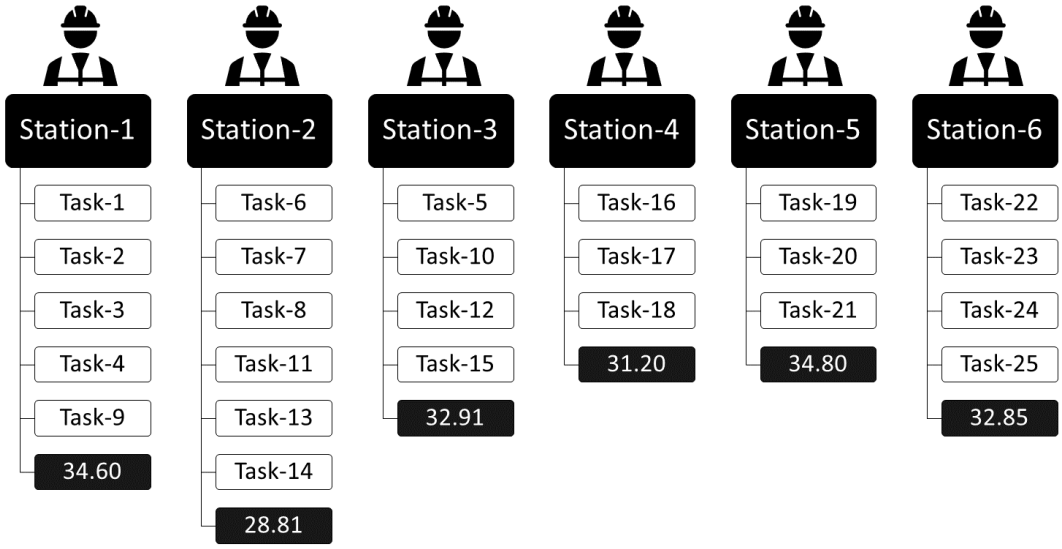

Figure 3.1: Optimal task assignments and cut-off values of the workloads in the illustrative example when $r = 0$

As the second example, we consider the case with two robots ($r = 2$). Figure 3.2 represents the optimal assignments of 25 tasks, 2 robots, and 4 human workers to 6 workstations which was obtained by solving type-II RSALBP. In the optimal solution, the robots are assigned to workstations 2 and 4, and human workers are assigned to the other workstations. The cut-off values of the workloads of the workstations are shown at the bottom of Figure 3.2 in black rectangles. The optimum cycle time which is the maximum of these cut-off values is found as 28.05 seconds. This value was 34.80 when all the operators were human workers. The user can then compare the benefit (i.e., improvement in cycle time) obtained by using these robots and the cost

of the robots to make a purchasing decision.

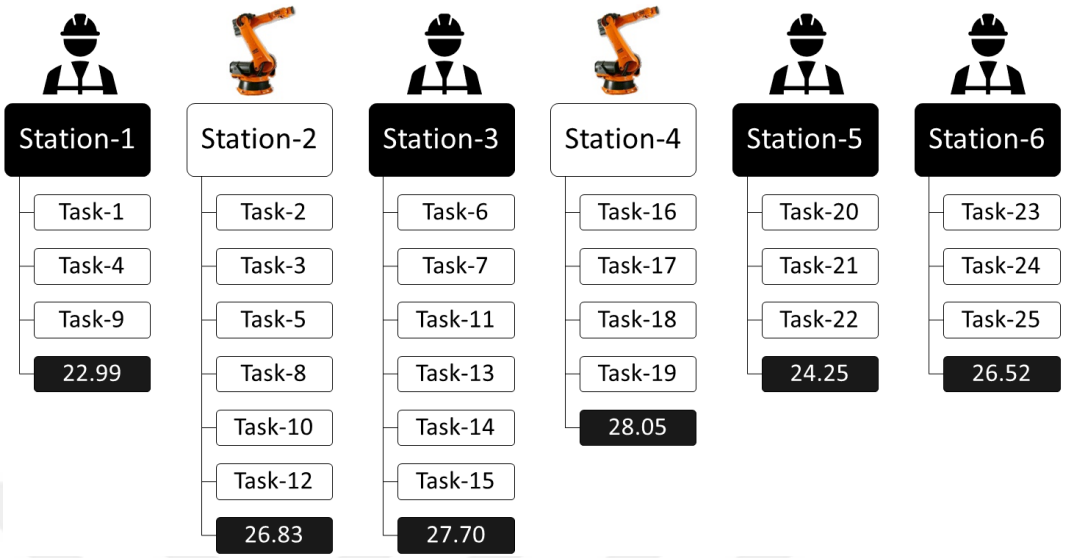

Figure 3.2: Optimal task assignments and cut-off values of the workloads in the illustrative example when $r = 2$

Finally, to show that the optimal solution can be affected by the capabilities of the robots, we made 10 replications by generating different capability vectors assuming again $\%80$ capability for the robots while keeping all other parameters the same. Table 3.4 displays the optimal solutions of the resulting 10 replications where first result corresponds to the solution depicted in Figure 3.2. In this table, we provide the optimal cycle time, the cut-off values of the workloads of the workstations, the number of tasks assigned to each workstation, and the workstations the robots were assigned to. The results show that even though the capabilities of all the robots are the same in these 10 replications, the tasks that the robots are capable of doing may have a significant impact on the optimal solutions. For example, while the optimal cycle time is 27.20 seconds in the fifth replication, it is 29.50 seconds in the seventh one. On the other hand, while the robots are not identical in replications 5, 6, and 8, the optimal cycle time turned out to be 27.20 seconds in all. If the decision maker is to choose among these three types of robots, s/he can choose the cheapest one if the only concerns are cost and the cycle time.

Table 3.4: Optimal cycle times, cut-off values of the workloads, and robot assignments to workstations

		Cut-off values of the workloads (sc)						Nb. of tasks assigned						Robots assigned					
#	ct (sc)	$j=1$	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	28.05	22.99	26.83	27.70	28.05	24.25	26.52	3	6	6	4	3	3	-	R	-	R	-	-
2	27.74	27.14	26.65	24.09	27.74	25.14	26.52	4	7	4	5	2	3	-	R	-	R	-	-
3	28.39	28.25	28.33	28.39	28.33	24.05	22.43	4	4	4	6	2	5	-	-	-	R	-	R
4	29.12	24.83	25.56	27.89	26.00	23.14	29.12	4	3	8	2	3	5	-	-	R	-	-	R
5	27.20	25.33	27.20	26.60	26.00	27.20	26.52	7	5	3	2	5	3	R	-	-	-	R	-
6	27.20	25.37	27.20	27.20	26.00	27.20	26.52	7	5	3	2	5	3	R	-	-	-	R	-
7	29.50	24.83	21.68	28.41	28.68	29.50	26.52	4	6	4	4	4	3	-	R	-	R	-	-
8	27.20	22.99	26.76	26.21	27.11	27.20	26.52	3	5	4	5	5	3	-	-	-	R	R	-
9	29.12	25.68	28.81	23.72	26.00	26.51	29.12	5	6	4	2	3	5	R	-	-	-	-	R
10	27.70	27.26	27.70	27.24	24.05	24.25	26.52	7	6	4	2	3	3	R	-	R	-	-	-

In the next section, we first compare the performances of the three formulations of type-II RSALBP on a subset of selected instances. These experiments show that the performance of the NLMIP formulation is way below the performances of the other formulations in terms of solution time and the solution quality within the given time limit. Following this observation, we then perform the comprehensive experiments only using the MISOCP and CP formulations of type-II RSALBP.

3.2.3 Computational Results

In this section, we provide the results of our computational study which was performed using a comprehensive experimental design by changing several parameters (see Table 3.1 for the list of parameters and their values used in the experiments). Before providing these results, we conducted an initial experiment to evaluate the performances of the three formulations proposed for type-II RSALBP. Pınarbaşı and Alakaş [21] proposed an NLMIP formulation for type-II SALBP and solved some instances using BARON. In our initial experiments, we wanted to evaluate the performance of an extension of this formulation in contrast with the MISOCP and CP formulations.

3.2.3.1 Comparative Results of the Three Formulations

In Tables 3.5 and 3.6, we present the comparative results of the solutions of type-II RSALBP using the proposed mathematical formulations on some selected instances. Each row in these tables correspond to one replication of an instance with some specific parameters. The first column displays the instance number, and the parameter values are shown in the next six columns. Columns 8 to 10 display the optimal objective function values if the instances were solved to optimality within the given 1 hour time limit. Otherwise, the objective function values of the best found solutions are reported. The last three columns include the solution times in CPU seconds.

Table 3.5: Comparative results of the three formulations on 30-task instances with a subset of selected parameters

							ct (sc)			CPU time (sc)		
Ins. #	cvr	cmr	cv	Cap.	r	α	NLMIP	MISOCP	CP	NLMIP	MISOCP	CP
1	0	1	0.2	40%	1	0.90	62.39	62.39	62.39	336.3	13.2	51.9
2	0	1	0.2	40%	1	0.95	64.44	64.44	64.44	1767.2	17.8	30.0
3	0	1	0.2	40%	2	0.90	64.92	64.92	64.92	1329.7	10.8	19.1
4	0	1	0.2	40%	2	0.95	67.17	67.17	67.17	683.1	6.6	14.4
5	0	1	0.2	80%	1	0.90	60.52	60.15	60.15	3600.0	107.6	267.5
6	0	1	0.2	80%	1	0.95	62.00	61.94	61.94	3600.0	81.8	278.5
7	0	1	0.2	80%	2	0.90	58.75	58.75	58.75	3600.0	70.2	328.3
8	0	1	0.2	80%	2	0.95	60.18	60.15	60.15	3600.0	59.1	295.7
9	-	-	0.2	-	0	0.90	61.85	61.84	61.84	1092.7	23.0	59.9
10	-	-	0.2	-	0	0.95	63.79	63.78	63.78	1524.1	27.1	67.8

Table 3.5 shows the comparative results for the 30-task problem instances. CPU times used to obtain the solutions with the NLMIP formulation are much larger than those of the others for all the problem instances. When the capabilities of the robots are 40%, all formulations were able to solve the instances to optimality within an hour. Still, the solution times of the MISOCP and CP formulations are much smaller than those of the NLMIP formulation. The NLMIP formulation cannot solve the instances to optimality when the capabilities of the robots are 80%. For these instances, i.e., instances 5 to 8, the relative gap values reported by BARON at termination were 4.3%,

5.2%, 2.2%, and 3.7%, respectively. The other formulations can successfully solve these instances within at most 6 minutes. When we evaluate the performance of the NLMIP formulation for the instances where no robot usage is allowed (i.e., instances of type-II SALBP), the same conclusions can be drawn in terms of the solution times (see the instances 9 and 10 in Table 3.5).

Table 3.6: Comparative results of the three formulations on 45-task instances with a subset of selected parameters

							ct (sc)			CPU time (sc)		
Ins. #	cvr	cmr	cv	Cap.	r	α	NLMIP	MISOCP	CP	NLMIP	MISOCP	CP
1	0	1	0.2	40%	1	0.90	69.91	69.36	69.10	3600.0	3600.1	1590.9
2	0	1	0.2	40%	1	0.95	73.85	73.10	73.10	3600.0	272.4	15.0
3	0	1	0.2	40%	2	0.90	70.35	69.10	69.10	3600.0	3600.1	15.1
4	0	1	0.2	40%	2	0.95	76.99	73.10	73.10	3600.0	333.8	25.8
5	0	1	0.2	80%	1	0.90	NSF*	68.72	68.40	3600.0	3600.1	3600.1
6	0	1	0.2	80%	1	0.95	122.11	70.60	70.47	3600.0	3602.6	3599.9
7	0	1	0.2	80%	2	0.90	116.18	67.76	67.50	3600.0	3600.3	3600.1
8	0	1	0.2	80%	2	0.95	NSF*	69.08	69.00	3600.0	3600.1	3600.1
9	-	-	0.2	-	0	0.90	70.21	70.31	70.09	3600.0	3600.1	3600.3
10	-	-	0.2	-	0	0.95	73.10	73.10	73.10	3600.0	619.9	12.1

*NSF: No solution found

Table 3.6 shows the comparative results for the 45-task problem instances. Only for instance 9, where $\alpha = 0.90$ and $r = 0$, the NLMIP formulation found a better objective function value than that of the MISOCP formulation within an hour. For other instances, the objective function values of the best solutions found by NLMIP formulation were worse than those of the MISOCP and CP formulations. For instances 5 and 8, the NLMIP formulation was not even able to find a feasible solution within an hour. The results shown in Tables 3.5 and 3.6 clearly show that the NLMIP formulation has the worst performance. Accordingly, we set up a comprehensive experimental design and perform a computational study only using the MISOCP and CP formulations to solve the instances of type-II RSALBP.

We performed a comprehensive computational study to solve the instances generated

from the 25-task, 30-task, 35-task, and 45-task problems by changing several parameters whose values are given in Table 3.1. In addition, 10 different replications of each instance are created by randomly generating vectors of robot capabilities. For these experiments, we provided the average objective function values (average cycle times) of 10 replications of each instance solved by the MISOCP and CP formulations.

3.2.3.2 Detailed Computational Results of 25-task and 30-task Problem Instances

The results of our computational experiments for the 25-task and 30-task problem instances are displayed in Tables 3.8 and 3.10, respectively. In these tables, the first column displays the instance setting number and the next four columns show the parameter values of cvr, cmr, cv, and α. The first two rows show the parameter values of capability and r. The last eight columns show the averages of optimal objective function values obtained by using the MISOCP and CP formulations. The number of workstations m is taken as 6 in all of the 25-task and 30-task problem instances. For all replications of the 25-task and 30-task problem instances, the optimal solution is obtained within the given 1 hour time limit by using both formulations.

For the 25-task problem instances, when no robot is used (i.e., $r = 0$), the optimal cycle times were found as 34.80, 36.16, 39.59, and 42.31 seconds, when the pairs of values of (cv, α) are (0.2, 0.90), (0.2, 0.95), (0.4, 0.90), and (0.4, 0.95), respectively (see Table 3.7). For example, in the case when $cv = 0.2$ and $\alpha = 0.90$, if we have a robot that is 80% capable and that has the properties $cvr = 0$ and $cmr = 0.6$ (see instance with instance setting 1, capability=80%, and $r = 1$), the average cycle time becomes 29.54 seconds resulting in an average saving of 5.26 seconds ($34.80 - 29.54$). Detailed results of the related 10 replications whose average cycle time value is 29.54 seconds can be seen in Table C.1 in Appendix C.

On the other hand, when the robot is not that much capable, the average cycle time may even show an increase with respect to the case with no robots. It can be seen from Table 3.8 that keeping all other parameters the same, as the capability increases, α decreases, cmr decreases, cvr decreases, or cv decreases, the average cycle time decreases. Depending on the capabilities of the robots, a second robot may or may

not provide an additional benefit on top of the first one (in terms of the average cycle time). For the 25-task problem instances, when the robots are 40% or 60% capable, the average cycle time increases if we increase the number of robots from one to two. On the other hand, when the robots are 80% or 100% capable, the addition of a second robot usually decreases the average cycle time.

Table 3.7: Optimal cycle times of 25-task problem instances for MISOCP and CP formulations when $r = 0$

				Cut-off values of the workloads (sc)						Nb. of tasks assigned					
Ins. #	cv	α	ct (sc)	$j = 1$	2	3	4	5	6	1	2	3	4	5	6
1	0.2	0.90	34.80	34.80	34.80	32.99	31.20	34.80	32.85	4	4	7	3	3	4
2	0.2	0.95	36.16	35.62	35.12	28.47	32.40	36.16	33.94	5	4	6	3	3	4
3	0.4	0.90	39.59	35.82	31.62	39.15	35.41	39.59	36.69	5	6	4	3	3	4
4	0.4	0.95	42.31	40.24	38.73	33.75	37.79	42.31	38.87	5	6	4	3	3	4

The robots may be beneficial even when they work at the same speed with the human workers or even when they are slower. Consider the replications of a problem instance with $cvr = 0.0$, $cmr = 1.2$, $cv = 0.4$, $\alpha = 0.95$, and capability=80%. If no robot is used, the optimal cycle time is 42.31 seconds when $cv = 0.4$ and $\alpha = 0.95$. On the other hand, when one robot with 80% capability is used, which is on average 20% slower than the human worker (as $cmr = 1.2$), the average cycle time over (10 replications) becomes 39.46 seconds (see instance setting 16 with capability=80% and $r = 1$). Detailed results of the related 10 replications whose average cycle time value is 39.46 seconds can be seen in Table C.2 in Appendix C.This saving in cycle time is due to the robot being very robust in performing the tasks in comparison with the human worker. Indeed, for this instance the standard deviations of the task times for the robot are all zero. This observation shows that robots can be beneficial even when they are slower if they can reduce or eliminate uncertainty in task times.

Table 3.8: Average optimal cycle times of 10 replications for MISOCP and CP formulations on 25-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins. #	cvr	cmr	cv	α	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$
1	0.0	0.6	0.2	0.90	34.18	39.62	31.70	32.56	29.54	26.62	29.40	26.40
2	0.0	0.6	0.2	0.95	35.37	40.87	32.83	33.50	30.57	27.08	29.94	26.40
3	0.0	0.6	0.4	0.90	38.41	43.93	35.63	36.14	32.37	29.22	31.53	27.23
4	0.0	0.6	0.4	0.95	40.82	46.38	37.89	38.24	33.74	30.67	33.35	28.20
5	0.0	0.8	0.2	0.90	34.36	39.62	33.11	33.63	31.74	29.77	31.30	29.18
6	0.0	0.8	0.2	0.95	35.43	40.87	33.89	34.48	32.46	30.57	32.24	29.60
7	0.0	0.8	0.4	0.90	38.41	43.93	36.00	37.07	34.53	32.70	34.40	31.51
8	0.0	0.8	0.4	0.95	40.82	46.38	38.09	38.96	36.31	33.83	35.85	32.80
9	0.0	1.0	0.2	0.90	34.84	39.66	34.02	34.99	33.24	32.95	32.85	32.39
10	0.0	1.0	0.2	0.95	36.13	40.87	35.26	35.99	34.37	33.87	33.94	33.35
11	0.0	1.0	0.4	0.90	38.86	43.93	37.62	38.30	36.33	35.58	35.78	34.24
12	0.0	1.0	0.4	0.95	41.13	46.38	39.24	40.04	38.03	36.76	37.80	36.00
13	0.0	1.2	0.2	0.90	35.16	39.95	34.85	36.31	34.80	34.80	34.80	34.80
14	0.0	1.2	0.2	0.95	36.34	41.15	36.14	37.29	36.09	36.09	36.00	36.00
15	0.0	1.2	0.4	0.90	39.43	44.06	38.94	39.79	37.95	37.89	36.69	36.00
16	0.0	1.2	0.4	0.95	41.99	46.38	40.71	41.62	39.46	39.03	38.87	37.80
17	0.5	0.6	0.2	0.90	34.18	39.62	31.90	32.78	30.21	28.12	30.13	27.20
18	0.5	0.6	0.2	0.95	35.37	40.87	32.96	33.88	31.24	28.92	30.93	27.93
19	0.5	0.6	0.4	0.90	38.41	43.93	35.79	36.62	33.96	30.97	33.39	30.05
20	0.5	0.6	0.4	0.95	40.82	46.38	38.07	38.70	35.96	32.66	35.03	32.05
21	0.5	0.8	0.2	0.90	34.49	39.62	33.42	34.18	32.61	31.36	32.39	30.45
22	0.5	0.8	0.2	0.95	35.68	40.87	34.42	35.15	33.51	32.12	33.35	31.25
23	0.5	0.8	0.4	0.90	38.72	43.93	37.03	37.74	35.72	34.41	35.45	33.84
24	0.5	0.8	0.4	0.95	41.13	46.38	39.08	39.91	37.35	36.33	36.86	35.26
25	0.5	1.0	0.2	0.90	34.84	39.82	34.47	35.46	33.81	33.77	32.85	32.40
26	0.5	1.0	0.2	0.95	36.13	41.02	35.60	36.54	34.78	34.72	33.94	33.35
27	0.5	1.0	0.4	0.90	39.43	43.95	38.45	39.28	37.24	36.89	36.69	35.78
28	0.5	1.0	0.4	0.95	41.90	46.38	40.65	41.55	39.35	38.58	38.87	37.80
29	0.5	1.2	0.2	0.90	35.20	40.19	34.85	36.82	34.80	35.14	34.80	34.80
30	0.5	1.2	0.2	0.95	36.50	41.33	36.19	37.99	36.16	36.27	36.16	36.16
31	0.5	1.2	0.4	0.90	39.84	44.22	39.60	40.72	39.59	39.59	39.59	39.59
32	0.5	1.2	0.4	0.95	42.35	46.48	42.29	42.95	42.25	42.25	42.16	42.16

Results obtained using the proposed formulations for the 30-task problem instances are given in Table 3.10. When no robot is used (i.e., $r = 0$), the optimal cycle times were found as 61.84, 63.78, 68.69, and 72.77 seconds when the pairs of values of (cv, α) are (0.2, 0.90), (0.2, 0.95), (0.4, 0.90), and (0.4, 0.95), respectively (see Table

3.9). For example, adding a robot with 60% capability, which has the properties of $cvr = 0.5$ and $cmr = 0.8$ to the assembly line in place of a human worker in the case where $cv = 0.4$ and $\alpha = 0.95$ can save 4.44 seconds (72.77-68.33) on average (see instance setting 24 with capability=60% and $r = 1$). Detailed results of the related 10 replications whose average cycle time value is 68.33 seconds can be seen in Table C.3 in Appendix C. In cases where the robots are operating slower than the human workers (i.e., $cmr = 1.2$), the average cycle time usually increases by replacing a human worker with a robot. On the other hand, in the case where $cv = 0.4$ and $cvr = 0.0$, the average cycle time decreases if the robot is capable of performing more than 40% of the tasks even when the robots operate slower than human workers (i.e., $cmr = 1.2$) (see instances settings 15 and 16 in Table 3.10). This shows us that using robots can be very beneficial in assembly lines (in terms of average cycle times) when the variability of task processing times are high, even if the robots perform the tasks slower than the human workers.

Table 3.9: Optimal cycle times of 30-task problem instances for MISOCP and CP formulations when $r = 0$

				Cut-off values of the workloads (sc)						Nb. of tasks assigned					
Ins. #	cv	α	ct (sc)	$j = 1$	2	3	4	5	6	1	2	3	4	5	6
1	0.2	0.90	61.84	61.61	61.78	61.33	61.84	60.09	59.67	6	5	4	5	4	6
2	0.2	0.95	63.78	63.48	63.70	63.41	63.78	62.38	61.56	6	5	4	5	4	6
3	0.4	0.90	68.69	68.22	68.55	68.68	68.69	68.18	66.34	6	5	4	5	4	6
4	0.4	0.95	72.77	72.62	72.55	72.59	72.00	72.77	70.12	6	5	5	4	4	6

For all 30-task problem instances where robots are 40% capable, the average cycle time increases as the number of robots increases from 1 to 2. On the other hand, if the robots are more skilled (i.e., 60%, 80%, or 100% capable), increasing the number of robots from 1 to 2 usually results in a decrease in average cycle time. Exceptions for this generally occur when the robots are operating with some uncertainty in task processing times and are slower than human workers (i.e., when $cvr = 0.5$ and $cmr = 1.2$). For example, for instance settings 29, 30, 31, and 32, the average cycle time increases with an increase in the number of robots from 1 to 2 irrespective of the capabilities of the robots.

Table 3.10: Average optimal cycle times of 10 replications for MISOCP and CP formulations on 30-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins. #	cvr	cmr	cv	α	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$
1	0.0	0.6	0.2	0.90	58.22	63.21	54.93	53.54	53.90	47.89	53.65	47.40
2	0.0	0.6	0.2	0.95	59.98	65.27	56.48	55.13	55.23	48.88	54.88	48.60
3	0.0	0.6	0.4	0.90	64.26	70.60	60.21	59.03	58.39	51.09	58.04	50.40
4	0.0	0.6	0.4	0.95	68.14	74.73	63.35	62.64	61.00	52.54	60.60	52.11
5	0.0	0.8	0.2	0.90	59.54	64.13	57.80	55.82	57.60	54.39	57.58	53.93
6	0.0	0.8	0.2	0.95	61.37	66.07	59.37	57.60	59.11	55.56	58.96	55.09
7	0.0	0.8	0.4	0.90	65.72	71.01	63.10	61.74	62.75	58.07	62.49	57.60
8	0.0	0.8	0.4	0.95	69.36	75.01	66.82	65.11	65.74	60.21	65.58	59.24
9	0.0	1.0	0.2	0.90	61.24	65.07	60.46	59.86	60.11	58.91	60.09	58.75
10	0.0	1.0	0.2	0.95	63.00	66.79	62.00	60.98	61.67	60.13	61.56	60.00
11	0.0	1.0	0.4	0.90	67.33	71.90	65.93	64.24	65.64	63.08	65.53	62.84
12	0.0	1.0	0.4	0.95	70.76	75.83	69.00	67.25	68.73	65.56	68.30	65.00
13	0.0	1.2	0.2	0.90	62.63	66.27	62.14	63.44	62.06	62.40	62.02	62.18
14	0.0	1.2	0.2	0.95	64.48	68.17	64.06	64.73	63.96	64.04	63.78	63.78
15	0.0	1.2	0.4	0.90	68.86	72.90	68.15	68.18	67.87	67.24	67.79	66.71
16	0.0	1.2	0.4	0.95	72.53	76.61	71.58	70.97	71.07	69.78	70.80	69.60
17	0.5	0.6	0.2	0.90	58.38	63.39	55.46	54.21	55.14	49.97	54.88	49.58
18	0.5	0.6	0.2	0.95	60.36	65.43	57.48	56.12	56.91	51.51	56.44	51.19
19	0.5	0.6	0.4	0.90	65.28	70.69	62.36	60.77	61.19	55.35	60.96	54.84
20	0.5	0.6	0.4	0.95	69.12	74.75	66.11	64.33	64.70	58.32	64.44	57.83
21	0.5	0.8	0.2	0.90	60.11	64.42	58.59	56.58	58.38	55.62	58.14	55.51
22	0.5	0.8	0.2	0.95	62.06	66.37	60.37	58.22	60.13	57.46	60.05	57.20
23	0.5	0.8	0.4	0.90	66.56	71.57	64.81	63.05	64.46	61.58	64.35	61.17
24	0.5	0.8	0.4	0.95	70.38	75.74	68.33	67.01	68.08	64.61	68.07	64.19
25	0.5	1.0	0.2	0.90	61.96	65.63	61.22	61.32	60.98	60.41	60.96	60.09
26	0.5	1.0	0.2	0.95	63.84	67.64	62.92	62.92	62.59	62.02	62.46	61.84
27	0.5	1.0	0.4	0.90	68.38	72.46	67.37	66.79	67.08	66.01	67.00	65.76
28	0.5	1.0	0.4	0.95	72.01	76.30	71.25	70.20	70.86	69.18	70.59	68.92
29	0.5	1.2	0.2	0.90	62.97	66.68	62.65	64.48	62.41	63.69	62.33	63.63
30	0.5	1.2	0.2	0.95	64.88	68.56	64.60	66.18	64.40	65.35	64.39	65.20
31	0.5	1.2	0.4	0.90	69.66	73.59	69.20	70.53	69.12	69.34	69.05	69.22
32	0.5	1.2	0.4	0.95	73.48	77.53	73.08	73.80	72.94	73.07	72.77	72.78

Similarly, it can be seen from instance setting 25 in Table 3.10 that when robots are working at the same speed with human workers (i.e., when $cmr = 1.0$), but with some uncertainty in task times ($cvr = 0.5$), increasing the number of robots with 60% capability from 1 to 2 may increase the average cycle time. A careful decision

on whether to use a robot in an assembly line has to be made by considering the robots' speed, capabilities, and level of uncertainty of the task processing times.

3.2.3.3 Detailed Computational Results of 35-task and 45-task Problem Instances

Tables 3.12 and 3.14 show the averages of the optimal or overall best found objective function values (for instances not solved to optimality within an hour) for the 35-task and 45-task problem instances, respectively. The number of workstations m is taken as 9 in all of the instances. The cycle time values in the tables are the averages of 10 overall best objective function values obtained from the replications generated by using random robot capabilities. All of the rows and columns of Tables 3.12 and 3.14 are similar to those of Tables 3.8 and 3.10 except the last two rows and columns. In the last two rows of Tables 3.12 and 3.14, the number of times each formulation (MISOCP and CP) finds the overall best objective function value are given out of a total of 320 replications for a fixed capability and r. On the other hand, in the last two columns of Tables 3.12 and 3.14, the number of times each formulation (MISOCP and CP) finds the overall best objective function value are given out of a total of 80 replications for fixed cvr, cmr, cv, and α values. Note that for a fixed instance setting and for fixed values of capability and r, if the relative difference between the best objective function values found by the MISOCP and CP formulations in an hour is less than 10^{-4}, then both formulations are assumed to have found the overall best solution. When we compare the formulations in terms of the number of times the overall best solution is found, it can be seen that the CP formulation performs better than the MISOCP formulation for both 35-task and 45-task problem instances.

Table 3.11: Optimal cycle times of 35-task problem instances for MISOCP and CP formulations when $r = 0$

				Cut-off values of the workloads (sc)									Nb. of tasks assigned								
Ins. #	cv	α	ct (sc)	$j=1$	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
1	0.2	0.90	63.99	62.49	63.55	63.95	62.69	63.99	62.78	63.36	62.42	60.36	4	3	5	2	4	4	4	4	5
2	0.2	0.95	66.54	64.90	66.26	66.49	65.44	66.54	64.98	65.73	65.38	63.30	4	3	5	2	4	4	4	4	5
3	0.4	0.90	73.02	73.02	71.26	72.72	72.38	72.99	70.56	71.71	72.85	70.72	5	4	3	2	4	4	4	4	5
4	0.4	0.95	78.51	77.84	78.51	77.87	77.97	77.15	77.63	77.90	75.59	74.57	5	3	2	5	3	5	4	4	4

For the 35-task problem instances, when no robot is used (i.e., $r = 0$), the optimal cycle times were found as 63.99, 66.54, 73.02, and 78.51 seconds, when the pairs of values of (cv, α) are (0.2, 0.90), (0.2, 0.95), (0.4, 0.90), and (0.4, 0.95), respectively (see Table 3.11). Adding, for example, a robot with 100% capability in place of a human worker, which has the properties of $cvr = 0.0$ and $cmr = 0.8$ to the assembly line in the case where $cv = 0.4$ and $\alpha = 0.95$ can reduce the average cycle time by 6.82 (78.51-71.69) seconds (see instance setting 8 with capability 100% and $r = 1$). Detailed results of the related instance whose cycle time value is 71.69 seconds can be seen in Table C.4 in Appendix C. When Table 3.12 is examined in detail, it can be seen that replacing one of the human workers with a robot always decreases the average cycle except when $cmr = 1.2$ (i.e., when the robot is slower). Still, when the robot is slower, the average cycle time may decrease by replacing a human worker with a robot depending on the robots' skills.

Table 3.12: Average cycle times of 10 replications for MISOCP and CP formulations on 35-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability		Best Sol.	
Ins. #	cvr	cmr	cv	α	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	MISOCP	CP
1	0.0	0.6	0.2	0.90	59.94	60.11	58.32	56.05	58.12	54.00	58.10	52.80	80	80
2	0.0	0.6	0.2	0.95	62.09	62.54	60.42	57.99	59.94	55.48	59.51	54.60	80	80
3	0.0	0.6	0.4	0.90	68.36	69.09	66.30	64.29	65.80	60.93	65.67	58.69	79	80
4	0.0	0.6	0.4	0.95	73.50	73.95	70.85	68.96	70.27	64.27	69.00	61.63	80	80
5	0.0	0.8	0.2	0.90	62.31	61.53	62.09	58.71	61.47	58.29	60.80	58.26	80	80
6	0.0	0.8	0.2	0.95	64.73	63.93	63.84	60.45	63.17	59.92	62.70	59.51	80	80
7	0.0	0.8	0.4	0.90	69.93	69.61	68.43	65.93	67.95	64.90	67.80	64.25	80	80
8	0.0	0.8	0.4	0.95	74.62	74.43	72.99	70.75	72.25	68.72	71.69	67.20	80	80
9	0.0	1.0	0.2	0.90	63.30	63.54	62.81	62.28	62.70	61.81	62.69	61.66	80	80
10	0.0	1.0	0.2	0.95	65.52	64.93	65.03	63.48	64.94	63.17	64.90	63.00	80	80
11	0.0	1.0	0.4	0.90	71.25	70.92	70.98	68.35	70.88	68.01	70.56	67.82	80	80
12	0.0	1.0	0.4	0.95	76.35	75.62	75.80	72.79	74.96	71.87	74.63	71.08	80	80
13	0.0	1.2	0.2	0.90	64.51	65.46	64.17	64.35	64.04	64.08	63.99	63.99	80	80
14	0.0	1.2	0.2	0.95	66.88	67.85	66.41	66.42	66.03	65.92	66.00	65.58	80	80
15	0.0	1.2	0.4	0.90	72.88	73.26	72.42	71.64	72.18	70.87	72.00	70.80	70	80
16	0.0	1.2	0.4	0.95	76.92	76.94	76.58	75.54	76.44	74.94	76.44	74.69	80	80
17	0.5	0.6	0.2	0.90	61.13	60.47	59.24	56.81	58.93	55.74	58.90	55.09	80	80
18	0.5	0.6	0.2	0.95	63.29	62.93	61.63	59.31	61.37	58.01	61.28	57.15	80	80
19	0.5	0.6	0.4	0.90	69.32	69.52	67.86	65.62	67.47	63.72	66.89	62.53	80	80
20	0.5	0.6	0.4	0.95	74.42	74.43	72.71	70.45	72.10	68.99	71.52	68.35	80	80
21	0.5	0.8	0.2	0.90	62.42	62.24	62.18	59.99	62.18	59.58	62.18	59.41	80	80
22	0.5	0.8	0.2	0.95	64.91	64.61	64.68	62.14	64.61	61.60	64.50	61.43	80	80
23	0.5	0.8	0.4	0.90	71.21	70.62	70.81	68.05	70.35	67.38	69.98	66.96	79	80
24	0.5	0.8	0.4	0.95	76.32	75.54	75.65	72.70	74.92	71.80	74.63	71.08	80	80
25	0.5	1.0	0.2	0.90	63.87	64.48	63.30	63.09	63.02	62.53	63.02	62.45	80	80
26	0.5	1.0	0.2	0.95	66.29	66.97	65.64	65.38	65.45	64.89	65.44	64.72	80	80
27	0.5	1.0	0.4	0.90	72.59	72.97	72.06	71.17	71.85	70.69	71.71	70.36	70	80
28	0.5	1.0	0.4	0.95	77.40	77.69	76.76	75.68	76.47	75.08	76.45	74.74	69	80
29	0.5	1.2	0.2	0.90	64.71	66.45	64.41	65.13	64.17	64.69	63.99	64.57	80	80
30	0.5	1.2	0.2	0.95	67.36	68.72	67.00	67.46	66.63	67.10	66.54	67.05	80	80
31	0.5	1.2	0.4	0.90	73.29	74.81	73.03	73.34	73.02	73.02	73.02	73.02	80	80
32	0.5	1.2	0.4	0.95	78.53	79.52	78.14	78.23	77.88	77.85	77.87	77.84	80	80
			Best Sol.	MISOCP	319	320	320	319	319	320	320	290		
				CP	320	320	320	320	320	320	320	320		

It can be seen from Table 3.12 that in 35-task problem instances, when the mean task processing times of robots are less than or equal to those of the human workers, i.e., when $cmr \leq 1.0$, the average cycle time usually decreases with an increase in the number of robots from 1 to 2. However, even if the robots are operating at least as fast as the human workers, i.e., when $cmr \leq 1.0$, when their capability is 40%, the average cycle time may show an increase as the number of robots increases from

1 to 2. When the robots are operating slower than the human workers, i.e., when $cmr = 1.2$, the average cycle time generally increases as we increase the number of robots from 1 to 2 if the robots are not very capable (i.e., for capabilities 40% or 60%). On the other hand, when the robots are very skilled (capability=80% or 100%), the average cycle time may decrease with the increase in the number of robots from 1 to 2.

Table 3.13: Optimal cycle times of 45-task problem instances for MISOCP and CP formulations when $r = 0$

				Cut-off values of the workloads (sc)									Nb. of tasks assigned								
Ins. #	cv	α	ct (sc)	$j = 1$	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
1	0.2	0.90	70.15	70.14	70.15	70.04	70.15	69.93	69.10	70.15	70.12	70.00	8	6	3	6	4	1	6	3	8
2	0.2	0.95	73.10	71.09	73.09	73.10	72.41	71.98	71.88	73.01	73.07	72.19	6	8	1	4	3	3	7	6	7
3	0.4	0.90	83.20	76.39	76.92	82.68	81.65	83.20	82.40	78.54	77.95	67.56	8	7	5	6	1	4	3	5	6
4	0.4	0.95	91.19	89.76	91.19	50.27	88.61	83.71	86.07	82.95	88.00	90.73	8	1	3	4	3	4	4	9	9

Table 3.14 shows the averages of the optimal or overall best found objective function values obtained for the 45-task problem instances found by the MISOCP and CP formulations. If no robot is used (i.e., $r = 0$), the optimal cycle times were found as 70.15, 73.10, 83.20, and 91.19 seconds, when the pairs of values of (cv, α) are (0.2, 0.90), (0.2, 0.95), (0.4, 0.90), and (0.4, 0.95), respectively (see Table 3.13). Average cycle time is usually decreased when one of the human workers is replaced by a robot whose capability is greater than 40%. The effect of adding a second robot on top of the first one for the 45-task problem instances is similar to our earlier observations that we made for other problem instances. As an extreme example, when two robots are used instead of human workers, the average cycle time decreases by 26.69 (91.19-64.50) seconds when the robots are 100% capable and their operating characteristics are $cvr = 0.0$ and $cmr = 0.6$ and when $cv = 0.4$ and $\alpha = 0.95$ (see instance setting 4 with capability=100% and $r = 2$). Detailed results of the related instance whose cycle time value is 64.50 seconds can be seen in Table C.5 in Appendix C.

Table 3.14: Average cycle times of 10 replications for MISOCP and CP formulations on 45-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability		Best Sol.	
Ins. #	cvr	cmr	cv	α	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	$ct(r=1)$	$ct(r=2)$	MISOCP	CP
1	0.0	0.6	0.2	0.90	69.10	69.10	68.54	68.03	65.15	61.47	63.44	58.20	46	80
2	0.0	0.6	0.2	0.95	73.10	73.10	72.30	71.75	67.41	63.51	64.86	59.40	46	80
3	0.0	0.6	0.4	0.90	83.20	83.20	81.82	81.18	73.36	68.67	69.12	62.40	53	80
4	0.0	0.6	0.4	0.95	91.19	91.19	89.34	88.61	78.05	72.67	72.40	64.50	44	80
5	0.0	0.8	0.2	0.90	69.10	69.10	68.84	68.57	67.25	65.27	66.40	63.57	44	80
6	0.0	0.8	0.2	0.95	73.10	73.10	72.63	72.30	69.67	67.35	68.19	64.85	44	80
7	0.0	0.8	0.4	0.90	83.20	83.20	82.16	81.77	75.92	72.96	72.80	68.80	53	80
8	0.0	0.8	0.4	0.95	91.19	91.19	89.73	89.27	80.87	77.26	76.45	71.20	45	80
9	0.0	1.0	0.2	0.90	69.10	69.10	69.04	68.92	68.65	67.89	68.41	67.26	36	80
10	0.0	1.0	0.2	0.95	73.10	73.10	72.84	72.69	71.26	70.23	70.47	69.00	45	80
11	0.0	1.0	0.4	0.90	83.20	83.20	82.41	82.18	77.63	76.07	75.25	73.00	45	80
12	0.0	1.0	0.4	0.95	91.19	91.19	89.99	89.71	82.72	80.60	79.10	76.00	45	80
13	0.0	1.2	0.2	0.90	70.58	71.01	70.44	70.83	69.98	70.30	69.75	70.11	2	80
14	0.0	1.2	0.2	0.95	73.10	73.11	72.99	72.99	72.24	72.27	71.88	71.84	44	80
15	0.0	1.2	0.4	0.90	83.20	83.20	82.57	82.51	78.79	78.34	76.88	76.20	44	80
16	0.0	1.2	0.4	0.95	91.19	91.19	90.17	90.06	84.05	83.01	80.97	79.29	44	80
17	0.5	0.6	0.2	0.90	69.10	69.10	68.68	68.24	66.12	63.06	64.80	60.44	44	80
18	0.5	0.6	0.2	0.95	73.10	73.10	72.49	72.04	68.83	65.63	66.97	62.44	44	80
19	0.5	0.6	0.4	0.90	83.20	83.20	82.13	81.61	75.65	72.06	72.44	67.32	44	80
20	0.5	0.6	0.4	0.95	91.19	91.19	89.75	89.21	81.10	77.21	76.76	71.20	44	80
21	0.5	0.8	0.2	0.90	69.10	69.10	68.94	68.73	68.00	66.52	67.57	65.46	44	80
22	0.5	0.8	0.2	0.95	73.10	73.10	72.76	72.52	70.73	69.06	69.69	67.41	44	80
23	0.5	0.8	0.4	0.90	83.20	83.20	82.41	82.12	77.64	75.64	75.17	72.41	45	80
24	0.5	0.8	0.4	0.95	91.19	91.19	90.04	89.72	83.15	80.84	79.75	76.35	44	80
25	0.5	1.0	0.2	0.90	69.62	69.13	69.54	69.07	69.35	68.92	69.26	68.87	5	79
26	0.5	1.0	0.2	0.95	73.10	73.10	72.93	72.88	71.98	71.55	71.49	70.90	44	80
27	0.5	1.0	0.4	0.90	83.20	83.20	82.59	82.50	78.98	78.27	77.17	76.16	45	80
28	0.5	1.0	0.4	0.95	91.19	91.19	90.23	90.10	84.49	83.48	81.61	80.17	44	80
29	0.5	1.2	0.2	0.90	71.04	72.05	71.04	71.97	71.00	71.93	70.99	71.93	5	80
30	0.5	1.2	0.2	0.95	73.13	73.98	73.10	73.91	73.10	73.85	73.10	73.81	2	80
31	0.5	1.2	0.4	0.90	83.20	83.20	82.89	82.89	81.03	81.03	80.10	80.10	80	80
32	0.5	1.2	0.4	0.95	91.19	91.19	90.48	90.48	86.22	86.22	84.10	84.10	80	80
			Best Sol.	MISOCP	279	280	253	259	98	108	20	41		
				CP	319	320	320	320	320	320	320	320		

3.2.3.4 Comparative Performance Measures of MISOCP and CP Formulations

In Table 3.15, a summary of the performances of the MISOCP and CP formulations is provided. For each problem and formulation, the CPU times, relative gaps, % calculated gaps, percent deviations from the overall best solutions, percentage of proven optimal solutions, and percentage of optimal solutions are given in the table. For

each problem size and each formulation, the minimum, average, and the maximum of 2560 values are reported in the table. Note that 2560 is the number of different combinations of the parameter values given in Table 3.1 multiplied with the number of replications 10 (i.e., $2560 = 2 \times 4 \times 2 \times 4 \times 2 \times 2 \times 10$).

Table 3.15: Comparative performance measures of MISOCP and CP formulations on all instances

		CPU time		%Gap	%Calculated gap		%Deviation		%Proven Optimal		% Optimal	
Problem	Att.	MISOCP	CP	MISOCP	MISOCP	CP	MISOCP	CP	MISOCP	CP	MISOCP	CP
	Avg.	7.5	19.3	0.0%	0.0%	0.0%	0.0%	0.0%				
n=25 m=6	max	27.7	70.2	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%	100.0%	100.0%
	min	1.6	9.4	0.0%	0.0%	0.0%	0.0%	0.0%				
	Avg.	89.8	429.9	0.0%	0.0%	0.0%	0.0%	0.0%				
n=30 m=6	max	1260.0	3555.9	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%	100.0%	100.0%
	min	5.3	7.3	0.0%	0.0%	0.0%	0.0%	0.0%				
	Avg.	759.0	751.2	0.2%	0.1%	0.1%	0.0%	0.0%				
n=35 m=9	max	3600.3	3600.9	17.6%	6.5%	6.5%	1.0%	0.0%	94.7%	92.0%	96.0%	96.5%
	min	25.9	15.3	0.0%	0.0%	0.0%	0.0%	0.0%				
	Avg.	2014.0	1730.7	4.4%	4.2%	4.0%	0.2%	0.0%				
n=45 m=9	max	3756.9	3600.6	30.4%	22.3%	21.3%	1.1%	0.0%	48.7%	52.7%	50.6%	52.7%
	min	2.1	5.0	0.0%	0.0%	0.0%	0.0%	0.0%				

When the CPU times are examined in Table 3.15, it can be concluded that the MISOCP formulation performs better in small-size problems, i.e., in 25-task and 30-task problems. On the other hand, the CP formulation outperforms the MISOCP formulation in larger size problems, i.e., in 35-task and 45-task problems. Average CPU time results of 10 replications for both proposed formulations on all problem instances can be seen in Tables D.1–D.8 in Appendix D, where a table for each formulation and benchmark problem is provided separately.

The absolute gap is the difference between the objective function value of the best found integer solution and the best lower bound. On the other hand, the relative gap is the ratio of the absolute gap to the objective function value of the best found integer solution. If an optimal solution cannot be found within the given time limit, CPLEX returns a positive relative gap value as an output for the MISOCP formulations. In Table 3.15, we provided the relative gap values as one of the performance measures. Since CPLEX does not provide the gap values for the CP formulations, the relative

gap values are provided only for the MISOCP formulations. In the %Calculated Gap columns, the relative gaps are calculated with respect to the optimal objective function values if at least one of the MISOCP or CP formulations finds the optimal solution. Otherwise, the gaps are calculated with respect to the best lower bound given by CPLEX for the MISOCP formulation. For example, it is possible that the MISOCP formulation finds the optimal solution, but cannot prove it in the given time limit. In this case, if the CP finds the optimal solution, then the %Calculated Gap value for the MISOCP formulation will be taken as zero. Table 3.15 shows that the relative gap values are similar for both formulations.

The deviation is the difference between the objective function value of the best solution found by one of the formulations, and the objective function value of the overall best solution. For example, if the objective function values of the best solutions found by the MISOCP and CP formulations are a and b, respectively, then the deviations of the MISOCP and CP solutions will be $a - min\{a, b\}$ and $b - min\{a, b\}$, respectively. On the other hand, the percent deviation is the ratio of the deviation to the objective function value of the overall best solution. Percent deviations from the overall best objective function values found for each problem instance are provided under the %Deviation column in Table 3.15. It can be seen from the table that the CP formulation was able to find the overall best solutions for almost all problem instances which is consistent with the results provided in the previous tables (3.8–3.14).

Percentage of proven optimal solutions is the ratio of the number of times the optimal solution is found and proven to be optimal by CPLEX within the given time limit to 2560, and are given under %Proven optimal column in Table 3.15 for both formulations. It can be seen from the table that, for small-size problems (i.e., for 25-task and 30-task problems), both formulations always find proven optimal solutions in 1 hour. On the other hand, for 35-task problems, the MISOCP and CP formulations find a proven optimal solution in 94.7% and 92.0% of the cases, respectively. On the other hand, for the largest size problems, the CP formulation finds more proven optimal solutions.

In some cases, both formulations return the same solution, but only one of them is a proven optimal solution. In this case, both formulations have the optimal solution, but

one formulation cannot prove it in the given time limit. For such cases, we counted both solutions as optimal and presented the percentage of times the optimal solutions are found in the last column of Table 3.15. The table shows that the optimal solutions are found by using both formulations for all 25-task and 30-task problems within the given time limit. For both formulations, more than 95% and 50% of the solutions are optimal for the 35-task and 45-task problem instances, respectively. The MISOCP formulation solves a replication of an instance of the 45-task problem with a relative gap of 22.3% in an hour (maximum relative gap value in the table). When we solve the same instance with a 24 hour CPU time limit, a solution with 12.1% relative gap is obtained when the time limit is reached. There is a trade-off between the solution time and the returned relative gap. As assembly line design problems are strategic level problems, use of very large time limits can be justified.

CHAPTER 4

STOCHASTIC ASSEMBLY LINE BALANCING WITH RELIABILITY RESTRICTION

This chapter describes a type-II stochastic assembly line balancing problem given a fixed lower bound on the whole assembly line reliability (type-II SALBP-R). The objective is to minimize the cycle time given normally distributed task processing times and a fixed number of workstations. In this problem, all operators are assumed to be identical. First, we give a constraint programming (CP) formulation to solve type-II stochastic assembly line balancing problem (type-II SALBP) in Section 4.1.1. This formulation is then used to develop a matheuristic algorithm for solving type-II SALBP-R, which is provided in Section 4.1.2. The proposed matheuristic finds a solution such that the workloads of all workstations do not exceed the cycle time with probability equal to the predefined assembly line reliability.

The algorithm starts with solving the CP formulation of type-II SALBP with a given set of confidence levels for workstations, satisfying the given assembly line reliability. After a solution is obtained, the algorithm compares the workstations' given confidence levels and their realized probabilities of not exceeding the cycle time. According to this comparison, the algorithm defines a new set of confidence levels that also satisfies the predefined assembly line reliability. With the new set of confidence levels, a solution from the CP formulation of type-II SALBP is obtained again. Note that the algorithm updates the set of confidence levels in a way that the previous solution satisfies the new set of confidence levels (see Sections 4.1.2 and 4.2.1). After some iterations, the algorithm stops accepting new solutions to the problem when the realized assembly line reliability becomes equal to the predefined lower bound on the assembly line reliability or the number of iterations limit is reached. As a final

improvement, the cycle time obtained from the last solution is reduced to a minimum possible cycle time that satisfies the assembly line reliability without changing the assignments.

The proposed algorithm is applied to Sawyer's 30-task and Kilbridge and Wester's 45-task problems [42] to compare the performance of the algorithm with the Bidirectional Heuristic Algorithm (BHA) of Liu et al. [11]. Computational results show that the proposed matheuristic can be a good alternative of BHA.

4.1 Problem Definition and Solution Approach for Type-II SALBP-R

In the literature, type-II SALBP is mostly studied considering the confidence level for each workstation so that the workload of each workstation cannot exceed the cycle time with at least a given probability α. However, the probability of not exceeding the cycle time in all the workstations (i.e., the assembly line reliability) can be quite different from the defined confidence levels. Assuming that the confidence level of each workstation is α, the assembly line reliability can be α^m in the worst case, which might be much smaller than α.

Consider an assembly line with normally distributed task processing times where $m = 6$ and predefined confidence level α is 0.900 for all workstations. The assembly line reliability can then be $0.900^6 = 0.531$ in the worst case. For an assembly line where the number of workstations is greater (e.g., $m = 9$), the assembly line reliability can be even smaller ($0.900^9 = 0.387$) in the worst case.

A similar concept to assembly line reliability is first discussed by Reeve and Thomas [43]. The authors assumed that if the workload of a workstation exceeds the cycle time, where the task times are normally distributed random variables, the line is stopped, or the assembled product becomes defective. With this motivation, they aimed to minimize the probability of exceeding the cycle time in any of the workstations, given an initial balance and cycle time by reassigning the tasks to the workstations with the trade and transfer method. The same problem was later studied by Suresh et al. [44] by using a genetic algorithm.

For solving type-II SALBP-R, a heuristic; namely the BHA, is proposed by Liu et al. [11]. Given the number of workstations and an assembly line reliability, the authors' objective was to minimize the cycle time by initially assigning tasks to workstations bidirectionally (i.e., tasks are first assigned to workstation 1 considering the precedence relations, then to workstation m, then to workstation 2, then to workstation m-1 etc.) and reassigning the tasks to the workstations by a trade and transfer procedure. Based on this assembly line reliability idea, and to ensure a more reliable assembly line balance in today's competitive manufacturing environment, we investigated the type-II SALBP with reliability restriction. The following notation is used in the description of the problem.

Indices:

i, k: tasks (1,...,n)

j: workstations (1,...,m)

Parameters:

n: number of tasks

m: number of workstations

α_j: confidence level of workstation j

μ_i: mean processing time of task i

σ_i: standard deviation of the processing time of task i

$P(i)$: set of immediate predecessors of task i

R: assembly line reliability

In type-II SALBP-R, task processing times are assumed to be normally distributed random variables with known mean μ_i and standard deviation σ_i. The workload of workstation j is also a normally distributed random variable with known mean

$$\sum_{i=1}^{n} \mu_i(x_i = j) \qquad \forall j\,,$$

and standard deviation

$$\sqrt{\sum_{i=1}^{n} \sigma_i^2(x_i = j)} \qquad \forall j\,.$$

Given a cycle time ct, the confidence level α_j is the probability that the workload of workstation j does not exceed the cycle time, and it can be represented as

$$\alpha_j = \Phi\left(\frac{ct - \sum_{i=1}^{n} \mu_i(x_i = j)}{\sqrt{\sum_{i=1}^{n} \sigma_i^2(x_i = j)}}\right) \qquad \forall j\,. \tag{4.1}$$

On the other hand, the probability that the workload of any workstation j in the whole assembly line not exceeding the cycle time is the assembly line reliability, which is the product of the realized confidence levels of all workstations in the assembly line. In this problem, the reliability of the assembly line, where the number of workstations is m, has to be at least the predefined lower bound R as follows:

$$\prod_{j=1}^{m} \alpha_j \geq R\,. \tag{4.2}$$

In this problem, our aim is to assign tasks to workstations so as to minimize the cycle time such that the reliability of the whole assembly line is at least given probability R (see Equation (4.2)).

The assumptions of type-II SALBP-R are given below.

1. Only one type of product or different products with very similar assembly features are assembled.
2. Assembly line is one-sided, paced, and straight.
3. Number of workstations is known and given.
4. Precedence relations of the tasks are known and given.
5. A task can be assigned to only one workstation and cannot be divided.
6. Each workstation can have only one operator.
7. Operators are identical and capable of doing all tasks.

8. Task times are independent and normally distributed random variables.

9. Means and variances of task times are known.

10. The failures, costs, and defective products led by incomplete tasks are neglected.

4.1.1 CP Formulation for Type-II SALBP

For type-II SALBP-R, we propose a matheuristic which solves a CP formulation of type-II SALBP with fixed confidence levels as subproblems. We next provide the details of this CP formulation.

Decision variables:
ct: cycle time
x_i: the workstation task i is assigned to
Auxiliary decision variables:
w_j: cut-off value of the workload of workstation j

$$\text{min} \quad ct = \max_j \{w_j\} \tag{4.3}$$

s.t.

$$w_j = \sum_{i=1}^{n} \mu_i(x_i = j) + \Phi^{-1}(\alpha_j) \cdot \sqrt{\sum_{i=1}^{n} \sigma_i^2(x_i = j)} \qquad \forall j\,, \tag{4.4}$$

$$x_i \geq x_k \qquad \forall k \in P(i), \forall i\,, \tag{4.5}$$

$$x_i \in \{1, ..., m\} \qquad \forall i\,. \tag{4.6}$$

The objective function (4.3) minimizes the cycle time. Constraint (4.4) keeps the workload of workstation j less than or equal to the cycle time with a given confidence level α_j for all j. Constraint (4.5) is to satisfy the precedence relations between tasks. Constraint (4.6) defines the domain restrictions.

4.1.2 A Matheuristic Algorithm for Type-II SALBP-R

The pseudocode of the matheuristic proposed for solving type-II SALBP-R is provided in Algorithm 1.

Algorithm 1: A matheuristic algorithm for type-II SALBP-R

Initialization: Let $u = 0$ and define confidence level of each workstation, such that $\prod_{j=1}^{m} \alpha_j^u = R$.

Solve the following CP with given confidence level α_j^us:

$$\min ct^u = \max_j \{w_j^u\}$$

s.t.

$$w_j^u = \sum_{i=1}^{n} (x_i^u = j)\mu_i + \Phi^{-1}(\alpha_j^u)\sqrt{\sum_{i=1}^{n}(x_i^u = j)\sigma_i^2} \quad \forall j \ ,$$

$$x_i^u \geq x_k^u \quad \forall i, \quad \forall k \in P(i) \ ,$$

$$x_i^u \in \{1, ..., m\} \quad \forall i \ ,$$

Return $\alpha_j^{u*} = \Phi\Big(\big(ct^{u*} - \sum_{i=1}^{n}(x_i^{u*} = j)\mu_i\big)/\sqrt{\sum_{i=1}^{n}(x_i^{u*} = j)\sigma_i^2}\Big) \quad \forall j$.

while $\big(\prod_{j=1}^{m} \alpha_j^{u*} > R + \delta\big) \wedge \big(u < l\big)$ **do**

  for $j = 1$ *to* m **do**

    if $\alpha_j^{u*} = \alpha_j^u$ **then**

      $\alpha_j^{u+1} = \alpha_j^{u*} - \varepsilon^u$

    else

      $\alpha_j^{u+1} = \alpha_j^{u*}$

    end

  end

  Find an ε^u such that $\prod_{j=1}^{m} \alpha_j^{u+1} = R$.

  $u = u + 1$

  Solve CP with new α_j^us.

  Return α_j^{u*}s and ct^{u*}.

end

while $\prod_{j=1}^{m} \alpha_j^{u*} > R$ **do**

  $ct^{u*} = ct^{u*} - 0.01$,

  $\alpha_j^{u*} = \Phi\Big(\big(ct^{u*} - \sum_{i=1}^{n}(x_i^{u*} = j)\mu_i\big)/\sqrt{\sum_{i=1}^{n}(x_i^{u*} = j)\sigma_i^2}\Big) \quad \forall j$.

  Return α_j^{u*}s and ct^{u*}.

end

The algorithm starts with defining a confidence level for each workstation in a way that their product is equal to the lower bound on the assembly line reliability. Given these confidence levels (i.e., α_j values), a CP for type-II SALBP is solved. Then, the cycle time and the task assignments obtained from the CP solution is used to calculate the realized confidence levels α_j^*'s. Note that the realized confidence levels are greater than or equal to their corresponding predefined confidence levels. If we define α_j^u's as the given confidence levels, and α_j^{u*}s as the realized ones, we have that $\alpha_j^{u*} \geq \alpha_j^u$ for each workstation j and iteration u. Hence, $\prod_{j=1}^{m} \alpha_j^{u*} \geq \prod_{j=1}^{m} \alpha_j^u$ holds in every iteration.

After the initialization procedure, the algorithm checks whether each workstation's realized and predefined confidence levels are equal. If they are equal, the predefined confidence level of the corresponding workstation in the next iteration (α_j^{u+1}) will be equivalent to $\alpha_j^{u*} - \varepsilon^u$. For the workstations that do not have equal realized and predefined confidence levels, the predefined confidence levels in the next iteration (α_j^{u+1}) will be taken as their realized confidence levels in the current iteration (α_j^{u*}). The algorithm computes ε^u in each iteration in such a way that the equality $\prod_{j=1}^{m} \alpha_j^{u+1} = R$ holds true. The procedure applied after the initialization is carried out until the iteration limit is reached ($u = l$) or the realized reliability of the solution becomes less than or equal to a predefined assembly line reliability limit which is taken as a real number that is δ bigger than R, where δ is a very small number, i.e., if $\prod_{j=1}^{m} \alpha_j^{u*} \not> R+\delta$, then the procedure stops. Note that there might be some rounding errors due to the decimal place differences between the parameters used and the solutions returned by CPLEX. Here, δ is added to the predefined assembly line reliability R to specify a stopping condition on the realized reliability $\prod_{j=1}^{m} \alpha_j^{u*}$, in order to avoid such possible rounding errors.

As a final improvement, value of cycle time obtained from the last iteration's CP solution is decreased gradually, until the reliability of assignments with the minimum possible cycle time becomes equal to the predefined assembly line reliability.

Next, our computational experiments using the proposed matheuristic algorithm are presented in detail in Section 4.2 by first providing an illustrative example in Section 4.2.1, and then giving the detailed computational results in Section 4.2.2.

4.2 Computational Study for Type-II SALBP-R

In our computational experiments, we used a computer with Intel Xeon E2246G, a 3.6 GHz processor, and 16 GB RAM. The algorithm is coded in ILOG CPLEX Optimization Studio 20.1 [41], and CPLEX CP Optimizer is used for solving the CP formulation of type-II SALBP in each iteration. We set the CPU time limit to 3600 seconds for all CP solutions in each iteration. Two well-known problem instances from the assembly line balancing literature are used: Sawyer's 30-task and Kilbridge and Wester's 45-task problems [42]. Deterministic task times of the tasks and their immediate predecessors are given for 30-task and 45-task problems in Tables A.1 and A.3 in Appendix A, respectively. Precedence diagrams of the problem instances are used directly, and deterministic task processing times in the benchmark problems are used as the mean task processing times (μ_i). The standard deviations of the task times (σ_i) are obtained using the coefficient of variation (cv) as shown in Equation (4.7). In our computational experiments, we used five different cv values.

$$\sigma_i = cv \cdot \mu_i \qquad \forall i\,. \tag{4.7}$$

In our computational study, we set an experimental design by changing values of cv and predefined assembly line reliability R. This allows us to understand the effects of different parameters on the solutions of type-II SALBP-R.

4.2.1 Illustrative Example

In this section, we provide some illustrative results on the 30-task problem of Sawyer [42]. There are 6 workstations in the original problem. Our aim is to assign tasks with stochastic processing times to the workstations to minimize the cycle time while satisfying the predefined assembly line reliability restriction. If we take cv as 0.3, the standard deviation of the processing time of the first task ($i = 1$) becomes 2.4, which is calculated according to Equation (4.7). Deterministic task times (μ_i) and standard deviations (σ_i) calculated for $cv = 0.3$, and the immediate predecessors ($P(i)$) of the tasks are given in Table 4.1. Moreover, we took the value of the assembly line

reliability R as 0.950 in our illustrative example.

Table 4.1: Means, standard deviations, and immediate predecessors of the tasks used in illustrative examples

Task #	μ_i	σ_i	$P(i)$	Task #	μ_i	σ_i	P(i)
1	8	2.4	-	16	10	3.0	3
2	7	2.1	-	17	2	0.6	3
3	19	5.7	-	18	10	3.0	17
4	10	3.0	1	19	18	5.4	18
5	2	0.6	1	20	16	4.8	14, 16
6	6	1.8	5	21	21	6.3	20
7	14	4.2	4, 6	22	14	4.2	15, 21
8	10	3.0	7	23	16	4.8	22
9	1	0.3	8	24	7	2.1	10, 20
10	4	1.2	-	25	17	5.1	24
11	14	4.2	2	26	9	2.7	9, 25
12	15	4.5	2	27	25	7.5	23, 26
13	5	1.5	12	28	7	2.1	27
14	12	3.6	13	29	14	4.2	27
15	9	2.7	14	30	2	0.6	29

As the illustrative example, we consider the case when all the values of predefined confidence levels (α_j^0) are equal to 0.9915 for the initial problem, i.e., $\alpha_j^0 = R^{1/m} \quad \forall j$, where $R = 0.950$ and $m = 6$. As the task times are random variables, the workload of each workstation is also a random variable. In Table 4.2, the first column shows the iteration number (u), the second column displays the optimal cycle time values, the next six columns represent the number of tasks assigned to each workstation, and the last six columns demonstrate the cut-off values (i.e., the $(100\alpha_j^u)^{th}$ percentiles) of the workloads that can be exceeded with probability $1 - \alpha_j^u$.

Table 4.2: Optimal cycle times, task assignments to workstations, and cut-off values of the workloads of 30-task problem, when $R = 0.950$ and $cv = 0.3$

		Number of tasks assigned						Cut-off values of the workloads					
Ite. #	ct (sc)	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$u=0$	74.51	7	5	4	4	4	6	74.51	74.21	74.11	74.46	73.20	71.61
$u=1$	74.47	5	4	5	6	4	6	73.24	74.25	74.00	74.03	74.46	74.47
$u=2$	74.29	4	6	3	7	4	6	74.29	74.26	73.32	74.19	73.99	72.08
$u=3$	74.25	5	7	3	5	4	6	73.71	73.97	74.22	73.61	74.25	74.23
$u=4$	74.16	4	7	6	3	4	6	74.07	74.16	73.91	73.68	73.74	73.21
$u=5$	74.15	6	5	6	3	4	6	73.99	73.59	74.14	74.13	74.11	74.15
$u=6$	74.15	6	5	6	3	4	6	74.15	74.14	73.62	73.63	73.84	73.56
$u=7$	74.11	7	4	3	6	4	6	73.96	74.07	73.86	73.85	74.11	74.08
$u=8$	74.10	7	4	3	6	4	6	74.09	74.10	74.04	74.07	73.84	73.48
$u=9$	74.08	7	4	3	6	4	6	74.00	73.99	73.82	73.88	74.06	74.08
$u=10$	74.07	7	4	3	6	4	6	74.03	74.04	74.04	74.07	73.93	73.68
$u=11$	74.06	7	4	3	6	4	6	73.98	73.97	73.92	73.98	74.06	74.00

The optimal cycle time of type-II SALBP will be equal to the maximum of these cut-off values which will guarantee that the workload of workstation j will be less than or equal to the cycle time with probability at least α_j^u, or with probability equal to α_j^{u*}. Hence, in the first example iteration ($u = 0$), the optimum cycle time is found as 74.51 seconds, which is the cut-off value of the first workstation's ($j = 1$) workload. In Table 4.3, the first column displays the iteration number, the next six columns represent the predefined α_j^u values for the workloads, and the last six columns show the realized α_j^{u*} values obtained from the optimal solutions by solving type-II SALBP in each iteration.

Table 4.3: Predefined and realized confidence levels for the 30-task problem, when $R = 0.950$ and $cv = 0.3$

	Given α_j^u values						Realized α_j^{u*} values					
Ite. #	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$u=0$	0.9915	0.9915	0.9915	0.9915	0.9915	0.9915	0.9915	0.9923	0.9925	0.9916	0.9942	0.9971
$u=1$	0.9815	0.9923	0.9925	0.9916	0.9942	0.9971	0.9873	0.9928	0.9936	0.9929	0.9942	0.9971
$u=2$	0.9873	0.9928	0.9936	0.9929	0.9899	0.9928	0.9873	0.9928	0.9952	0.9931	0.9908	0.9968
$u=3$	0.9839	0.9894	0.9952	0.9931	0.9908	0.9968	0.9863	0.9905	0.9952	0.9944	0.9908	0.9968
$u=4$	0.9863	0.9905	0.9936	0.9944	0.9892	0.9952	0.9866	0.9905	0.9944	0.9951	0.9903	0.9966
$u=5$	0.9866	0.9863	0.9944	0.9951	0.9903	0.9966	0.9873	0.9886	0.9944	0.9951	0.9903	0.9966
$u=6$	0.9873	0.9886	0.9935	0.9943	0.9895	0.9958	0.9873	0.9886	0.9947	0.9951	0.9904	0.9966
$u=7$	0.9855	0.9868	0.9947	0.9951	0.9904	0.9966	0.9863	0.9870	0.9951	0.9956	0.9904	0.9966
$u=8$	0.9863	0.9870	0.9951	0.9956	0.9895	0.9957	0.9863	0.9870	0.9951	0.9956	0.9903	0.9965
$u=9$	0.9859	0.9866	0.9947	0.9952	0.9903	0.9965	0.9861	0.9869	0.9950	0.9955	0.9903	0.9965
$u=10$	0.9861	0.9869	0.9950	0.9955	0.9897	0.9960	0.9861	0.9869	0.9950	0.9955	0.9902	0.9965
$u=11$	0.9858	0.9866	0.9948	0.9953	0.9902	0.9965	0.9860	0.9868	0.9950	0.9955	0.9902	0.9965

Figure 4.1 demonstrates the comparison of predefined confidence level α_j^0s (represented with the dashed lines) and realized confidence level α_j^{0*}s (represented with the continuous lines) for each workstation j that are obtained by solving type-II SALBP (see iteration 0 in Table 4.3). The algorithm starts with setting the parameters α_j^0s to 0.9915, then using these values, a CP for type-II SALBP is solved. Values of α_j^{0*} are calculated by using the optimal solution values as in Equation 4.1. For the workstations which have the same α_j^{0*} and α_j^0 values, the algorithm defines the α_j^1 values as $\alpha_j^{0*} - \varepsilon^0$. On the other hand, for the workstations which have different α_j^{0*} and α_j^0 values, α_j^1 values are determined to be equal to their corresponding α_j^{0*}s. In this iteration of the illustrative example ($u = 0$), the algorithm sets α_1^1 to $\alpha_1^{0*} - \varepsilon^0$, and the rest of the α_j^1s are set to α_j^{0*} values.

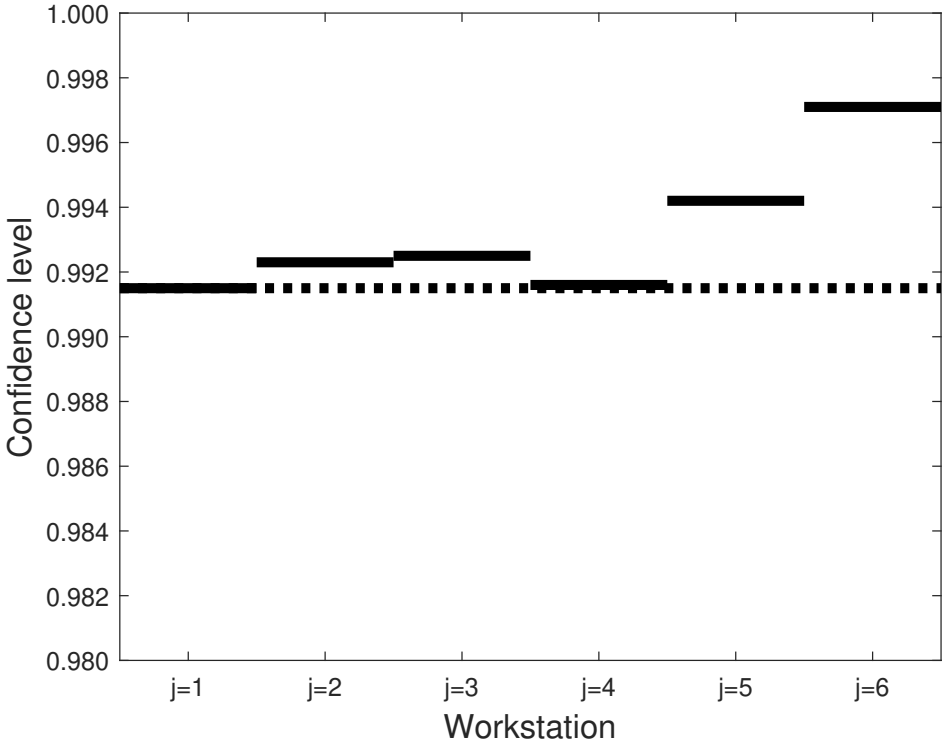

Figure 4.1: Predefined α_j^0 and realized α_j^{0*} values when $cv = 0.3$, $R = 0.950$, and $u = 0$

Algorithm finds an ε^0 satisfying the condition $\prod_{j=1}^{m} \alpha_j^1 = R$. In this iteration, ε^0 is found as 0.0100, and it can be seen from Table 4.3 that $\alpha_1^1 = 0.9815$, which comes from $\alpha_1^1 = \alpha_1^{0*} - \varepsilon^0$, or equivalently, $0.9815 = 0.9915 - 0.0100$.

After finding all α_j^1 values, a CP for type-II SALBP is solved. Values of α_j^{1*} are calculated by using the optimal solution values as in Equation 4.1. Figure 4.2 demonstrates the comparison of predefined confidence level α_j^1s (represented with the dashed lines) and realized confidence level α_j^{1*}s (represented with the continuous lines) in iteration 1 for each workstation j that are obtained by solving type-II SALBP (see iteration 1 in Table 4.3). For the workstations which satisfy the equality $\alpha_j^{1*} = \alpha_j^1$, the algorithm defines the α_j^2 values as $\alpha_j^{1*} - \varepsilon^1$. On the other hand, for the workstations which have different α_j^{1*} and α_j^1 values, α_j^2 values are determined to be equal to their corresponding α_j^{1*}s. In this iteration of the illustrative example ($u = 1$), the algorithm sets α_5^2 and α_6^2 values to $\alpha_5^{1*} - \varepsilon^1$ and $\alpha_6^{1*} - \varepsilon^1$, respectively. The rest of the α_j^2 values are set

to α_j^{1*} values.

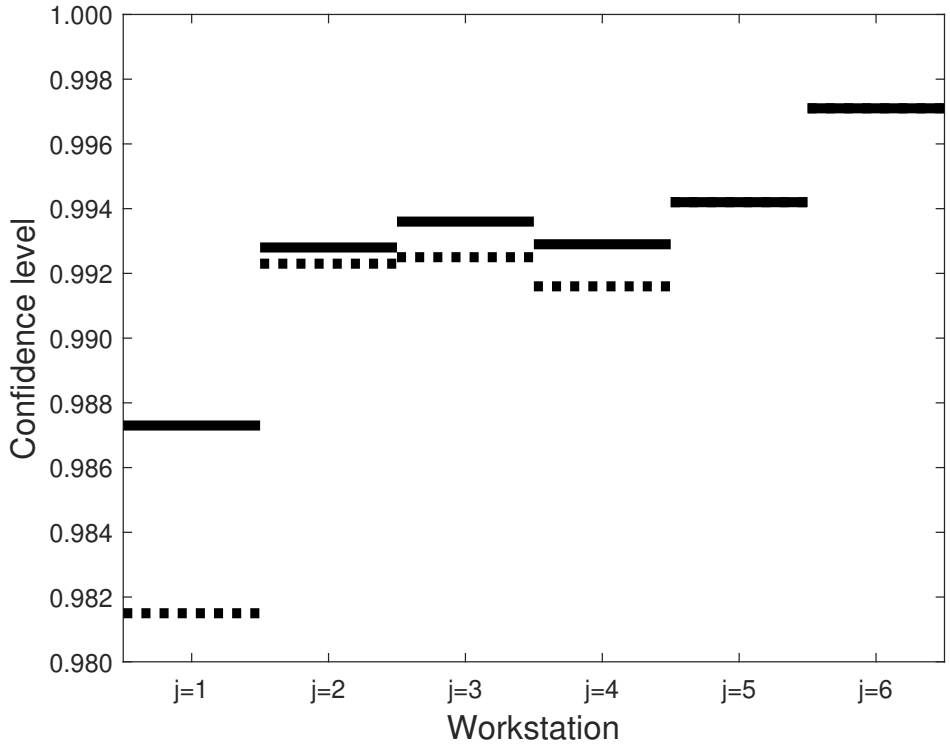

Figure 4.2: Predefined α_j^1 and realized α_j^{1*} values when $cv = 0.3$, $R = 0.950$, and $u = 1$

Algorithm finds an ε^1 satisfying the condition $\prod_{j=1}^{m} \alpha_j^2 = R$. In this iteration, ε^1 is found as 0.0043, and it can be seen from Table 4.3 that $\alpha_5^{1*} = 0.9942$ and $\alpha_6^{1*} = 0.9971$, while $\alpha_5^2 = 0.9899$ and $\alpha_6^2 = 0.9928$, which come from $\alpha_j^2 = \alpha_j^{1*} - \varepsilon^1$ for $j = 5$ and $j = 6$. Algorithm continues to make iterations with the same logic that it does in iteration 1 until the condition $\prod_{j=1}^{m} \alpha_j^{u*} > R + \delta$ does not hold, or until the iteration limit is reached ($u = l$). In this illustrative example, the value of l is chosen to be 50 and the algorithm terminates at $u = 11$, since $\prod_{j=1}^{m} \alpha_j^{11*} \not> 0.950 + \delta$. As a final step, ct^{11} (74.06 seconds) is decreased without changing the task assignments, while the assembly line reliability condition still holds, i.e., $\prod_{j=1}^{m} \alpha_j^{11*} > 0.950$. As a result, keeping the final optimal task assignments, the final cycle time becomes 74.00 seconds, and the algorithm ensures that this solution still satisfies the assembly line reliability condition.

Table 4.4: Optimal cycle times, task assignments to workstations, and cut-off values of the workloads of 30-task problem, when $R = 0.950$ and $cv = 0.3$

			Number of tasks assigned						Cut-off values of the workloads					
Rep. #	Ite. #	ct (sc)	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
	$u=0$	74.51	7	5	4	4	4	6	74.51	74.21	74.11	74.46	73.20	71.61
	$u=1$	74.47	5	4	5	6	4	6	73.24	74.25	74.00	74.03	74.46	74.47
	$u=2$	74.29	4	6	3	7	4	6	74.29	74.26	73.32	74.19	73.99	72.08
	$u=3$	74.25	5	7	3	5	4	6	73.71	73.97	74.22	73.61	74.25	74.23
	$u=4$	74.16	4	7	6	3	4	6	74.07	74.16	73.91	73.68	73.74	73.21
	$u=5$	74.15	6	5	6	3	4	6	73.99	73.59	74.14	74.13	74.11	74.15
1	$u=6$	74.15	6	5	6	3	4	6	74.15	74.14	73.62	73.63	73.84	73.56
	$u=7$	74.11	7	4	3	6	4	6	73.96	74.07	73.86	73.85	74.11	74.08
	$u=8$	74.10	7	4	3	6	4	6	74.09	74.10	74.04	74.07	73.84	73.48
	$u=9$	74.08	7	4	3	6	4	6	74.00	73.99	73.82	73.88	74.06	74.08
	$u=10$	74.07	7	4	3	6	4	6	74.03	74.04	74.04	74.07	73.93	73.68
	$u=11$	74.06	7	4	3	6	4	6	73.98	73.97	73.92	73.98	74.06	74.00
		74.00	7	4	3	6	4	6						
	$u=0$	75.10	5	7	4	5	3	6	75.06	73.81	74.24	74.59	75.10	75.09
	$u=1$	74.93	6	5	4	6	3	6	74.19	73.32	74.22	74.92	74.93	74.09
	$u=2$	74.81	6	5	4	6	3	6	74.71	73.63	74.81	73.38	74.39	74.47
	$u=3$	74.66	6	5	4	6	3	6	74.27	74.60	74.38	74.38	74.66	73.53
2	$u=4$	74.53	6	5	4	6	3	6	74.39	73.93	74.47	73.91	74.53	74.47
	$u=5$	74.48	6	5	4	6	3	6	74.38	73.62	74.47	74.38	74.48	74.46
	$u=6$	74.47	6	5	4	6	3	6	74.39	73.63	74.47	74.37	74.30	74.47
	$u=7$	74.47	6	5	4	6	3	6	74.38	73.62	74.47	74.38	74.30	74.46
		74.41	6	5	4	6	3	6						
	$u=0$	75.66	6	5	5	4	6	4	75.66	74.79	74.74	74.95	75.07	73.60
	$u=1$	74.34	6	6	4	5	3	6	74.33	74.33	74.29	74.34	74.18	73.38
	$u=2$	74.34	6	6	4	5	3	6	74.20	74.13	74.28	74.34	74.18	73.69
3	$u=3$	74.32	6	6	4	5	3	6	74.20	74.32	74.29	73.85	74.18	73.68
	$u=4$	74.31	7	5	5	4	3	6	74.08	74.31	73.95	73.98	74.18	73.69
	$u=5$	74.18	7	5	5	4	3	6	74.08	74.13	73.95	73.98	74.18	73.68
		74.13	7	5	5	4	3	6						
	$u=0$	74.66	6	5	4	6	3	6	74.34	74.43	74.65	74.31	74.66	72.09
	$u=1$	74.52	5	5	7	4	3	6	73.35	74.12	73.66	74.51	74.52	74.47
	$u=2$	74.26	4	6	4	6	4	6	74.26	74.18	73.60	73.98	73.20	73.37
4	$u=3$	74.00	4	6	7	3	4	6	73.89	73.55	73.94	74.00	73.81	73.69
	$u=4$	74.00	4	6	4	6	4	6	74.00	73.69	73.99	73.83	73.80	73.68
	$u=5$	74.00	4	6	4	6	4	6	74.00	73.69	73.60	73.84	73.81	73.69
	$u=6$	73.81	4	6	7	3	4	6	73.77	73.69	73.62	73.59	73.81	73.68
		73.81	4	6	7	3	4	6						
	$u=0$	74.69	6	4	6	5	3	6	73.58	74.52	74.51	74.69	73.82	71.97
	$u=1$	74.47	8	4	4	4	4	6	74.15	74.44	74.30	74.23	72.89	74.47
	$u=2$	74.45	8	4	4	4	4	6	73.98	74.30	74.11	74.45	73.79	74.08
5	$u=3$	74.44	8	4	4	4	4	6	74.15	74.44	74.30	74.38	73.49	73.52
	$u=4$	74.30	8	4	4	4	4	6	74.15	74.16	74.30	74.23	73.79	73.69
	$u=5$	74.23	8	4	4	4	4	6	74.15	74.16	73.98	74.23	73.79	73.68
		74.22	8	4	4	4	4	6						

Finally, to show that the solution can be affected by the initialization of the values of α_j^0s, we made five replications by generating different α_j^0 values, assuming again $R = 0.950$ while keeping all other parameters the same. In the first replication of each problem instance, α_j^0s are set to $R^{(1/m)}$, which gives equal confidence levels to all workstations. For the other replications, a confidence level set consisting of random α_j^0 values is generated such that each α_j^0 can take a value between R and 1. Until the product of the confidence levels ($\prod_{j=1}^{m} \alpha_j^0$) is greater than or equal to the predefined confidence level R, a new set of confidence levels is generated. If this value is greater than R, the largest α_j^0 is gradually decreased until $\prod_{j=1}^{m} \alpha_j^0 = R$ holds true.

Table 4.4 displays the optimal solutions of the resulting five replications where the first replication corresponds to the solution depicted in Tables 4.2 and 4.3 and Figures 4.1 and 4.2. In this table, we provide the optimal cycle times, the number of tasks assigned to each workstation, and the cut-off values of the workloads of the workstations. The α_j^0 values used in these replications can be seen in Table 4.5.

Table 4.5: Initial predefined confidence levels in replications for 30-task problem, when $R = 0.950$ and $cv = 0.3$

	Given α_j^0 values					
Rep. #	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
1	0.9915	0.9915	0.9915	0.9915	0.9915	0.9915
2	0.9927	0.9932	0.9867	0.9994	0.9794	0.9977
3	0.9942	0.9752	0.9953	0.9935	0.9917	0.9993
4	0.9809	0.9939	0.9920	0.9935	0.9960	0.9928
5	0.9921	0.9886	0.9940	0.9870	0.9948	0.9925

The results show that even though the reliabilities of the assembly line are the same in these five replications, initial predefined confidence levels may have a significant impact on the optimal solutions, as well as the number of iterations, which changes the solution time. For example, while the final optimal cycle time is found as 74.41 seconds in the second replication, it is found as 73.81 seconds in the fourth one.

On the other hand, although the algorithm terminates at $u = 11$ with $ct^{11} = 74.06$ seconds in the first replication, it terminates at $u = 6$ with a better cycle time $ct^{6} = 73.81$ seconds in the fourth replication.

4.2.2 Computational Results

We compared the performance of the proposed matheuristic with that of the Liu et al.'s BHA [11]. The method of Liu et al. has two procedures. In the first procedure; namely, the bidirectional assignment procedure, from both sides of the assembly line, the tasks are assigned to the workstations in such a way that the cut-off value of the workload in each workstation does not exceed a pre-computed lower bound. All the remaining tasks are assigned to the middle workstation, i.e., workstation $j = \lceil \frac{m+1}{2} \rceil$. Due to the bidirectional assignment procedure, the cut-off value of the workload for the middle workstation can be larger than the predefined lower bound of the cycle time. In contrast, for the rest of the workstations, the cut-off values of the workloads are smaller. Therefore, the second procedure is applied to reduce the difference between cut-off values of the workloads of the workstations. The trade and transfer procedure initializes by using the assignments made in the first procedure and makes task trades between the workstations or transfers a task from one workstation to another if a decrease in difference between the cut-off values of the workloads is possible. Then, an upper bound is set to be equal to the highest cut-off value of the workloads. If the upper bound is greater than the lower bound, the lower bound is increased by a certain amount, and the algorithm iterates starting from the first procedure until the lower bound becomes greater than or equal to the upper bound. Finally, the cycle time is reduced gradually until the minimum cycle time satisfying the assembly line reliability condition is found.

In this computational study, the performance of the two algorithms is tested on 30-task and 45-task problems, using five different coefficient of variation values ($cv = 0.1, 0.2, 0.3, 0.4, 0.5$) and four different predefined assembly line reliabilities ($R = 0.900, 0.925, 0.950, 0.975$), which gives 40 ($2 \times 5 \times 4$) distinct problems. We further use five replications of these problems by setting different initial predefined confidence levels, in which Replication 1 stands for the ones initialized with equal pre-

defined confidence levels, i.e. $\alpha_j^0 = R^{\frac{1}{m}}$ for every workstation j. In total, solutions of 200 (40×5) instances are presented to evaluate the performance of the proposed matheuristic algorithm.

Table 4.6: Comparative results of the BHA and proposed matheuristic on the 30-task instances

				Proposed algorithm					
Ins. #	R	cv	BHA	Best rep.	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5
1	0.900	0.1	61.89	59.94	60.29	60.13	60.24	59.94	60.68
2	0.900	0.2	66.05	65.81	65.81	65.97	66.05	66.10	66.58
3	0.900	0.3	71.60	71.55	71.55	72.20	71.98	72.67	72.22
4	0.900	0.4	77.64	77.20	77.28	78.01	77.20	78.30	78.36
5	0.900	0.5	83.21	83.28	83.28	83.63	83.52	84.92	84.10
6	0.925	0.1	61.14	60.41	60.45	60.66	60.65	60.60	60.41
7	0.925	0.2	66.68	66.42	66.42	66.76	66.96	66.59	66.68
8	0.925	0.3	72.97	72.70	72.77	73.87	72.70	72.76	72.85
9	0.925	0.4	78.98	78.63	78.63	79.61	78.92	79.12	78.91
10	0.925	0.5	84.81	84.80	84.80	86.96	86.76	85.44	85.02
11	0.950	0.1	61.56	60.86	60.86	61.03	60.92	61.16	60.86
12	0.950	0.2	67.97	67.29	67.29	67.65	68.04	67.67	67.37
13	0.950	0.3	73.89	73.81	74.00	74.41	74.13	73.81	74.22
14	0.950	0.4	80.71	80.38	80.38	81.50	81.75	80.87	80.77
15	0.950	0.5	87.40	87.06	87.06	87.64	87.62	87.38	87.14
16	0.975	0.1	63.68	61.53	61.80	61.54	61.80	61.78	61.53
17	0.975	0.2	70.57	68.73	68.74	69.07	69.31	68.73	68.74
18	0.975	0.3	76.20	75.87	75.87	76.81	75.93	76.15	75.99
19	0.975	0.4	83.17	83.06	83.06	83.84	83.90	83.66	83.43
20	0.975	0.5	90.39	90.46	90.46	91.13	91.01	90.85	90.69

In Tables 4.6 and 4.7, we present the comparative results of the solutions of type-II SALBP-R, using the proposed matheuristic and the BHA. Each row in these ta-

bles corresponds to an instance with some specific parameters. The first column displays the instance number, and the parameter values are shown in the following two columns. Column 4 displays the cycle time values found by the BHA, while column 5 represents the best cycle time value among five replications solved using the proposed matheuristic. Cycle time values for five replications of an instance by the proposed matheuristic are given in the last five columns. To ease the comparison, the better cycle time value among the BHA and the proposed matheuristic is highlighted for each instance.

Table 4.6 shows the comparative results for the 30-task problem instances. For the 30-task problem instances, the cycle times are the minimum possible values of the optimal cycle times found by the CP for type-II SALBP. Only for instance 5, where $R = 0.90$ and $cv = 0.5$, and instance 20, where $R = 0.90$ and $cv = 0.5$ the BHA found a better cycle time value than that of the proposed matheuristic. For other instances, the cycle time values of the proposed matheuristic were better than those of the BHA.

Table 4.7: Comparative results of the BHA and proposed matheuristic on the 45-task instances

				Proposed algorithm					
Ins. #	R	cv	BHA	Best rep.	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5
1	0.900	0.1	69.11	69.05	69.16	69.15	69.20	69.18	69.05
2	0.900	0.2	76.96	76.63	76.90	76.70	76.63	76.85	76.88
3	0.900	0.3	84.74	84.42	85.69	85.53	84.56	84.42	84.45
4	0.900	0.4	92.92	92.38	93.15	93.42	93.09	92.79	92.38
5	0.900	0.5	100.59	100.55	101.13	101.82	100.87	100.73	100.55
6	0.925	0.1	69.93	69.51	69.56	69.61	69.51	69.52	69.53
7	0.925	0.2	77.60	77.44	77.49	78.06	77.49	77.57	77.44
8	0.925	0.3	85.79	85.79	86.00	87.06	87.02	85.79	85.98
9	0.925	0.4	94.49	94.28	94.57	95.70	95.49	95.38	94.28
10	0.925	0.5	102.89	102.83	103.07	104.19	108.04	102.83	103.02
11	0.950	0.1	70.33	69.94	70.00	70.07	69.94	70.08	70.20
12	0.950	0.2	78.64	78.50	79.38	78.50	79.36	78.58	78.54
13	0.950	0.3	87.77	87.53	87.90	88.76	88.80	87.74	87.53
14	0.950	0.4	96.89	96.70	96.87	97.55	97.44	96.91	96.70
15	0.950	0.5	106.11	106.17	106.40	106.47	106.63	106.47	106.17
16	0.975	0.1	70.84	70.72	70.75	70.73	70.72	70.78	70.84
17	0.975	0.2	80.66	80.34	81.81	80.34	83.06	80.44	80.36
18	0.975	0.3	90.79	90.52	90.52	91.85	91.85	90.55	90.52
19	0.975	0.4	101.22	101.21	102.82	101.74	101.60	101.50	101.21
20	0.975	0.5	111.96	111.92	112.05	112.40	114.52	111.92	112.19

Table 4.7 shows the comparative results for the 45-task problem instances. For the 45-task problem instances, the cycle times are the minimum possible values of the best cycle times found by CP for type-II SALBP within the given one hour time limit. Only for instance 15, where $R = 0.950$ and $cv = 0.5$, the BHA found a better cycle time value than that of the proposed matheuristic. For instance 8, where $R = 0.925$ and $cv = 0.3$, both algorithms end up with the same cycle time value. For other

instances, the cycle time values of the proposed matheuristic were better than those of the BHA.

Table 4.8: CPU time results of the replications on 30-task problem instances

		CPU time (sc)				
R	cv	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5
0.900	0.1	370	759	284	841	220
0.900	0.2	652	274	248	450	400
0.900	0.3	212	403	347	256	332
0.900	0.4	293	247	200	134	142
0.900	0.5	357	114	182	246	107
0.925	0.1	624	268	361	423	275
0.925	0.2	469	365	528	461	385
0.925	0.3	364	131	281	221	204
0.925	0.4	501	361	481	116	183
0.925	0.5	415	82	220	133	306
0.950	0.1	1802	756	1693	170	375
0.950	0.2	373	454	247	319	378
0.950	0.3	383	286	175	271	228
0.950	0.4	236	61	38	209	400
0.950	0.5	192	147	335	162	174
0.975	0.1	166	2233	329	128	1800
0.975	0.2	2301	2145	114	265	283
0.975	0.3	122	46	349	144	94
0.975	0.4	92	819	33	53	98
0.975	0.5	84	50	90	129	123

Tables 4.8 and 4.9 show the CPU time results of replications using the proposed matheuristic algorithm on 30-task and 45-task problem instances, respectively. Depending on the problem size and the number of iterations to terminate, CPU times are changing. Due to the different configurations of computers used in the study of

Liu et al. and this thesis, direct CPU time comparison is not possible. For CPU time results of BHA, one can refer to the related research, i.e., see [11]. Since a mathematical formulation (CP for type-II SALBP) is solved in every iteration of the proposed matheuristic algorithm for type-II SALBP-R, the BHA proposed by Liu et al. finds solutions much faster than our proposed solution approach.

Table 4.9: CPU time results of the replications on 45-task problem instances

		CPU time (sc)				
R	cv	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5
0.900	0.1	144006	144005	144004	144004	144006
0.900	0.2	133240	110488	85043	112908	147604
0.900	0.3	40357	65232	51971	76599	88325
0.900	0.4	54976	69371	122872	79583	73129
0.900	0.5	84722	33140	54420	87543	90901
0.925	0.1	144005	180005	180007	144004	144005
0.925	0.2	84279	176416	124012	95343	96434
0.925	0.3	63041	33280	25734	90871	71012
0.925	0.4	7241	45055	59338	68421	26647
0.925	0.5	79893	22191	3714	84001	40068
0.950	0.1	144005	180007	180006	180006	144005
0.950	0.2	39639	106869	176416	100192	103404
0.950	0.3	57738	32637	2923	41270	25663
0.950	0.4	61430	25474	18435	69022	29354
0.950	0.5	43484	29260	11813	14845	29326
0.975	0.1	144004	177534	180006	183608	147605
0.975	0.2	36008	43326	460	18097	29458
0.975	0.3	3680	457	527	8185	7239
0.975	0.4	5	531	549	21687	3640
0.975	0.5	3646	36	236	7267	53

The results shown in Tables 4.6 and 4.7 show that the proposed matheuristic algorithm may be used as an alternative solution approach for type-II SALBP-R. Note that the results provided for the BHA are taken from the literature. One may replicate this algorithm as well to have a more fair comparison.

CHAPTER 5

CONCLUSION

As the first problem of this thesis, type-II robotic stochastic assembly line balancing problem (type-II RSALBP) is defined and examined for the first time. In this problem, given the numbers of robots and human workers, and tasks are to be assigned to a given number of workstations. The task times are assumed to be independent and normally distributed random variables whose parameters depend on the type of the operator performing the task. Robots have much less (possibly zero) variability in task processing times than human workers. It is assumed that human workers are capable of performing all the tasks while robots are able to perform a subset of the tasks. The objective is to minimize the cycle time in such a way that the workload of each workstation is less than or equal to the cycle time with a probability greater than or equal to the given confidence level α.

The type-II RSALBP is NP-hard and includes non-linearity. MISOCP and CP formulations are developed to solve the problem to optimality. Some problem instances from the literature are enriched by considering different values for different parameters in a comprehensive experimental design. The proposed formulations are tested on the generated instances and the effects of different problem parameters on the optimal cycle times are investigated. Several managerial insights are provided as a result of our computational experiments in terms of when the usage of robots would be beneficial or when it would be better to include a second robot on top of the first one. It was shown that the robots may be beneficial even when they are slower than human workers on average.

Both proposed formulations deliver excellent performance in solving small and moderate size problems in an hour. For the 45-task problems, the average relative and %

calculated gap values are less than 5% when the time limit is 1 hour. The computational experiments showed that for 25-task and 30-task problems, the performance of the MISOCP formulation is better; while the CP formulation outperforms the MISOCP formulation for larger size instances.

As the second problem of this thesis, type-II stochastic assembly line balancing problem with a reliability restriction (type-II SALBP-R) is studied. Given the number of workstations and a lower bound on the assembly line reliability, the aim is to find the minimum cycle time where the probability of not exceeding the cycle time by any workload in the whole assembly line is greater than or equal to the predefined lower bound. In this problem, all the operators are identical and the task times are assumed to be independent and normally distributed random variables.

As a solution approach for type-II SALBP-R, a matheuristic algorithm is proposed for the first time. In every iteration of this algorithm, a CP formulation for type-II SALBP is solved with fixed confidence levels for the workloads of each workstation satisfying the assembly line reliability condition. The algorithm iterates by manipulating these confidence levels while the updated confidence levels maintain the solution of the previous iteration. This way, the objective function values obtained from the CP formulation in each iteration are guaranteed to follow a descending trend.

The performance of the proposed matheuristic algorithm is evaluated by comparing the results on some instances for type-II SALBP-R with Liu et al.'s BHA. The computational experiments showed that the proposed matheuristic algorithm can be considered as an alternative method to solve type-II SALBP-R.

This thesis is the first to consider stochastic task times in the context of assembly line balancing with human-robot collaboration and proposes the first matheuristic for type-II SALBP-R. The thesis can be a good reference point when making decisions, with the need for changes brought about by the Industry 4.0 paradigm, during the transition from traditional manual assembly lines to those with human-robot collaboration, and developing effective solution approaches to solve type-II RSALBPs as well as type-II SALBP-Rs. In the future, extensions of type-II RSALBP and type-II SALBP-R can be studied in the context of other assembly line balancing problems such as mixed-model or U-shaped ALBPs. Different objective functions such as min-

imizing the number of workstations, carbon footprint, or cost can be considered in future works. Heuristic solution approaches can be developed to solve larger size problems. Moreover, a multi-objective version can be considered by integrating a cost component into RSALBP and/or SALBP-R.

REFERENCES

[1] P. Sivasankaran and P. Shahabudeen, "Literature review of assembly line balancing problems," *The International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9, pp. 1665–1694, 2014.

[2] K. Ağpak and H. Gökçen, "A chance-constrained approach to stochastic line balancing problem," *European Journal of Operational Research*, vol. 180, no. 3, pp. 1098–1115, 2007.

[3] H. Gökçen and Ö. F. Baykoç, "A new line remedial policy for the paced lines with stochastic task times," *International Journal of Production Economics*, vol. 58, no. 2, pp. 191–197, 1999.

[4] H.-S. Lau and A. Shtub, "An exploratory study on stopping a paced line when incompletions occur," *IIE transactions*, vol. 19, no. 4, pp. 463–467, 1987.

[5] N. Boysen, M. Fliedner, and A. Scholl, "A classification of assembly line balancing problems," *European Journal of Operational Research*, vol. 183, no. 2, pp. 674–693, 2007.

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

[7] M. Bortolini, E. Ferrari, M. Gamberi, F. Pilati, and M. Faccio, "Assembly system design in the industry 4.0 era: a general framework," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5700–5705, 2017.

[8] M. Salveson, "The assembly line balancing problem," *Journal of Industrial Engineering*, vol. 6, no. 3, pp. 18–25, 1955.

[9] C. L. Moodie and H. H. Young, "A heuristic method of assembly line balancing for assumptions of constant or variable work element times," *The Journal of Industrial Engineering*, vol. 16, no. 1, pp. 23–29, 1965.

[10] 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.

[11] S. Liu, H. Ong, and H. Huang, “A bidirectional heuristic for stochastic assembly line balancing Type II problem,” *The International Journal of Advanced Manufacturing Technology*, vol. 25, no. 1, pp. 71–77, 2005.

[12] R. L. Carraway, “A dynamic programming approach to stochastic assembly line balancing,” *Management Science*, vol. 35, no. 4, pp. 459–471, 1989.

[13] M. Held, R. M. Karp, and R. Shareshian, “Assembly-line balancing-dynamic programming with precedence constraints,” *Operations Research*, vol. 11, no. 3, pp. 442–459, 1963.

[14] M. Nkasu and K. Leung, “A stochastic approach to assembly line balancing,” *International Journal of Production Research*, vol. 33, no. 4, pp. 975–991, 1995.

[15] 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.

[16] A. Baykasoğlu and L. Özbakır, “Stochastic U-line balancing using genetic algorithms,” *The International Journal of Advanced Manufacturing Technology*, vol. 32, no. 1, pp. 139–147, 2007.

[17] 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.

[18] P. T. Zacharia and A. C. Nearchou, “Multi-objective fuzzy assembly line balancing using genetic algorithms,” *Journal of Intelligent Manufacturing*, vol. 23, no. 3, pp. 615–627, 2012.

[19] A. Bockmayr and N. Pisaruk, “Solving assembly line balancing problems by combining IP and CP,” in *Proceedings of the 6th Annual Workshop of the ERCIM Working Group on Constraints*, Prague, Czech Republic, 2001.

[20] M. Pınarbaşı, M. Yüzükırmızı, and B. Toklu, "Variability modelling and balancing of stochastic assembly lines," *International Journal of Production Research*, vol. 54, no. 19, pp. 5761–5782, 2016.

[21] M. Pınarbaşı and H. M. Alakaş, "Balancing stochastic type-II assembly lines: chance-constrained mixed integer and constraint programming models," *Engineering Optimization*, vol. 52, no. 12, pp. 2146–2163, 2020.

[22] J. Rubinovitz, J. Bukchin, and E. Lenz, "RALB–A heuristic algorithm for design and balancing of robotic assembly lines," *CIRP Annals*, vol. 42, no. 1, pp. 497–500, 1993.

[23] G. Nicosia, D. Pacciarelli, and A. Pacifici, "Optimally balancing assembly lines with different workstations," *Discrete Applied Mathematics*, vol. 118, no. 1-2, pp. 99–113, 2002.

[24] G. Levitin, J. Rubinovitz, and B. Shnits, "A genetic algorithm for robotic assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 811–825, 2006.

[25] J. M. Nilakantan, G. Q. Huang, and S. G. Ponnambalam, "An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems," *Journal of Cleaner Production*, vol. 90, pp. 311–325, 2015.

[26] B. Zhang, L. Xu, and J. Zhang, "Balancing and sequencing problem of mixed-model U-shaped robotic assembly line: Mathematical model and dragonfly algorithm based approach," *Applied Soft Computing*, vol. 98, p. 106739, 2021.

[27] J. M. Nilakantan, Z. Li, Q. Tang, and P. Nielsen, "Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems," *Journal of Cleaner Production*, vol. 156, pp. 124–136, 2017.

[28] Z. Li, M. N. Janardhanan, and S. Ponnambalam, "Cost-oriented robotic assembly line balancing problem with setup times: multi-objective algorithms," *Journal of Intelligent Manufacturing*, vol. 32, no. 4, pp. 989–1007, 2021.

[29] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Robotics and Autonomous Systems*, vol. 116, pp. 162–180, 2019.

[30] P. Chutima, "A comprehensive review of robotic assembly line balancing problem," *Journal of Intelligent Manufacturing*, vol. 33, no. 1, pp. 1–34, 2022.

[31] C. Weckenborg, K. Kieckhäfer, C. Müller, M. Grunewald, and T. S. Spengler, "Balancing of assembly lines with collaborative robots," *Business Research*, vol. 13, no. 1, pp. 93–132, 2020.

[32] Z. Li, M. N. Janardhanan, and Q. Tang, "Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots," *Neural Computing and Applications*, pp. 1–22, 2021.

[33] P. Samouei and J. Ashayeri, "Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations," *Applied Mathematical Modelling*, vol. 72, pp. 259–275, 2019.

[34] Z. A. Çil, Z. Li, S. Mete, and E. Özceylan, "Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human-robot collaboration," *Applied Soft Computing*, vol. 93, p. 106394, 2020.

[35] M. Tawarmalani and N. V. Sahinidis, "A polyhedral branch-and-cut approach to global optimization," *Mathematical Programming*, vol. 103, pp. 225–249, 2005.

[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[37] V. Blanco, "Ordered p-median problems with neighbourhoods," *Computational Optimization and Applications*, vol. 73, no. 2, pp. 603–645, 2019.

[38] D. Dinler and M. K. Tural, "Exact solution approaches for the workload smoothing in assembly lines," *Engineering Science and Technology, an International Journal*, vol. 24, no. 6, pp. 1318–1328, 2021.

[39] C. Timpe, "Solving planning and scheduling problems with combined integer and constraint programming," *OR spectrum*, vol. 24, no. 4, pp. 431–448, 2002.

[40] Y. Bukchin and T. Raviv, “Constraint programming for solving various assembly line balancing problems,” *Omega*, vol. 78, pp. 57–68, 2018.

[41] IBM, “ILOG CPLEX Optimization Studio.”

[42] A. Scholl, *Data of assembly line balancing problems*. Technische Hochschule Darmstadt, Institut für Betriebswirtschaftslehre, 1997.

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

[44] G. Suresh, V. Vinod, and S. Sahu, “A genetic algorithm for assembly line balancing,” *Production Planning & Control*, vol. 7, no. 1, pp. 38–46, 1996.

Appendix A

DETERMINISTIC TASK TIMES AND IMMEDIATE PREDECESSORS OF THE TASKS FOR THE 30-TASK, 35-TASK, AND 45-TASK PROBLEM INSTANCES

Table A.1: Deterministic task times and immediate predecessors of the tasks for the 30-task problem

Task	μ_{i2}	$P(i)$	Task	μ_{i2}	$P(i)$
1	8	-	16	10	3
2	7	-	17	2	3
3	19	-	18	10	17
4	10	1	19	18	18
5	2	1	20	16	14, 16
6	6	5	21	21	20
7	14	4, 6	22	14	15, 21
8	10	7	23	16	22
9	1	8	24	7	10, 20
10	4	-	25	17	24
11	14	2	26	9	9, 25
12	15	2	27	25	23, 26
13	5	12	28	7	27
14	12	13	29	14	27
15	9	14	30	2	29

Table A.2: Deterministic task times and immediate predecessors of the tasks for the 35-task problem

Task	μ_{i2}	$P(i)$	Task	μ_{i2}	$P(i)$
1	29	-	19	19	18
2	3	1	20	29	17, 19
3	5	2	21	6	16, 20
4	22	3	22	10	21
5	6	1	23	16	22
6	14	5	24	23	23
7	2	1, 6	25	5	21
8	5	6	26	5	25
9	22	8	27	5	24, 26
10	30	1	28	40	11, 13, 27
11	23	4	29	2	28
12	30	1	30	5	21
13	23	9	31	5	30
14	2	7, 10	32	1	21, 31
15	19	14	33	40	11, 13, 27, 32
16	29	15	34	2	27
17	2	-	35	2	33
18	2	7, 12			

Table A.3: Deterministic task times and immediate predecessors of the tasks for the 45-task problem

Task	μ_{i2}	$P(i)$	Task	μ_{i2}	$P(i)$
1	9	-	24	29	15
2	9	-	25	26	14
3	10	1	26	6	17, 25
4	10	2	27	5	17
5	17	3	28	24	22, 27
6	17	4	29	4	14
7	13	1	30	5	14
8	13	2	31	7	14
9	20	5, 7	32	4	14
10	20	6, 8	33	15	20, 23, 24, 27
11	10	-	34	7	26, 28, 36, 38
12	11	-	35	7	33
13	6	11, 12	36	9	33
14	22	7, 8, 13	37	4	12
15	11	13	38	3	33
16	19	15	39	5	-
17	12	14	40	4	34
18	4	20	41	21	10, 29, 30, 31, 32, 39, 40
19	3	15	42	12	41
20	7	16, 19	43	6	37
21	55	18	44	5	42
22	14	21	45	5	42
23	27	15			

Appendix B

RANDOMLY GENERATED CAPABILITIES OF THE ROBOT(S) FOR THE PROBLEM INSTANCES FOR TYPE-II RSALBP

Table B.1: Randomly generated capabilities of the robot(s) to perform each task for the 25-task problem instances for replications 1 to 5

		Randomly generated capabilities of the robot(s) to perform each task																								
Rep. #	Cap.	$i=1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	40%	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1
1	60%	1	1	1	0	1	1	1	1	0	1	0	0	0	1	0	1	0	1	0	0	0	1	1	1	1
	80%	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	1	0	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	1	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1
2	60%	0	1	0	0	1	1	1	1	1	0	1	1	0	1	0	1	1	0	0	1	1	0	1	0	1
	80%	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	1	1	0	0	1	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	1
3	60%	0	1	1	1	1	0	0	1	1	0	1	1	0	0	1	1	1	0	0	0	1	1	0	1	1
	80%	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	0	0	0	0	1	1
4	60%	0	1	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	1
	80%	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0
5	60%	1	0	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	0
	80%	1	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.2: Randomly generated capabilities of the robot(s) to perform each task for the 25-task problem instances for replications 6 to 10

		Randomly generated capabilities of the robot(s) to perform each task																								
Rep. #	Cap.	$i = 1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	40%	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0
6	60%	0	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	1	0
	80%	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	1	0	1	1	0	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0	0
7	60%	1	1	0	1	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1
	80%	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0
8	60%	0	1	1	0	0	1	0	0	0	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	0
	80%	0	1	1	1	0	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
9	60%	1	1	1	1	1	1	0	0	1	1	1	0	1	0	0	0	0	0	1	1	0	1	1	0	1
	80%	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	1	1	1	0	0
10	60%	0	0	1	1	1	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	1	1	1	1	0
	80%	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1	1	1	0	0	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.3: Randomly generated capabilities of the robot(s) to perform each task for the 30-task problem instances for replications 1 to 5

		Randomly generated capabilities of the robot(s) to perform each task																														
Rep. #	Cap.	$i = 1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	40%	0	1	1	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	1
1	60%	1	1	1	0	0	1	1	0	1	1	1	0	1	0	0	1	0	0	1	0	1	1	1	0	0	1	1	0	1	1
	80%	1	1	1	1	0	1	1	0	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	1	0	1	1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0
2	60%	1	0	1	0	0	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0
	80%	1	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	1	1	0	0	1	0	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	1
3	60%	0	1	0	1	1	1	0	0	1	0	1	0	1	1	0	1	1	1	1	1	0	0	0	0	1	1	1	0	1	1
	80%	0	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	0	0	0	1	1
4	60%	0	1	0	1	0	0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
	80%	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0
5	60%	1	0	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0
	80%	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.4: Randomly generated capabilities of the robot(s) to perform each task for the 30-task problem instances for replications 6 to 10

		Randomly generated capabilities of the robot(s) to perform each task																														
Rep. #	Cap.	$i=1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	40%	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	0	1	0	1	0	0
6	60%	0	1	0	0	1	0	1	1	1	0	1	1	0	1	0	0	1	1	1	1	1	1	1	1	0	1	0	1	0	0
	80%	0	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1	0	1	0	1	0
7	60%	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	0	1	1	1	0
	80%	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	1	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1	1	0
8	60%	0	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1
	80%	0	1	1	1	1	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	1	0	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0
9	60%	1	1	1	1	1	0	1	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	1	1	1	0
	80%	1	1	1	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	1	1	0	1	0	0
10	60%	0	0	1	1	1	1	1	0	1	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1	1	1	0	1	1	0
	80%	1	0	1	1	1	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.5: Randomly generated capabilities of the robot(s) to perform each task for the 35-task problem instances for replications 1 to 5

		Randomly generated capabilities of the robot(s) to perform each task																																		
Rep. #	Cap.	$i = 1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	40%	1	1	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	0	1
1	60%	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1	1	1	0	1
	80%	1	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0
2	60%	1	1	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	1	1	0	0	0	1	1	0
	80%	1	1	1	0	0	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	1	1	1	0	0	0	1	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1
3	60%	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	1
	80%	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	0	1	0	1	1	0	1	0	0	0	0	0	1	1
4	60%	0	1	0	0	1	0	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	0	1	1	1	1	1
	80%	1	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0
5	60%	1	0	0	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0
	80%	1	0	1	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.6: Randomly generated capabilities of the robot(s) to perform each task for the 35-task problem instances for replications 6 to 10

		Randomly generated capabilities of the robot(s) to perform each task																																		
Rep. #	Cap.	$i = 1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	40%	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	0	1	0	1	0	0
6	60%	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0
	80%	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0
7	60%	0	1	1	1	1	1	1	0	1	1	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	1	0	1	1	0	0	1	1	1	0
	80%	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	1	1	1	0
8	60%	0	0	1	1	1	0	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1
	80%	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	1	1	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	0
9	60%	1	1	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0	1	0	1	0	1	1	1	0	1	1	1	1	0
	80%	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	0	1	1	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1	0	0	0	0	0	1	1	1	0	1	0	0	0
10	60%	1	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	1	0	0	0	0	1	1	1	1	1	1	0	0
	80%	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.7: Randomly generated capabilities of the robot(s) to perform each task for the 45-task problem instances for replications 1 to 5

		Randomly generated capabilities of the robot(s) to perform each task																																												
Rep. #	Cap.	$i = 1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	40%	1	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	1	0	1	0	1
1	60%	1	1	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
	80%	1	1	1	0	1	0	0	1	0	1	1	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0
2	60%	1	1	0	1	0	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0
	80%	1	1	1	1	0	1	0	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1
3	60%	0	0	1	1	0	1	1	1	1	1	0	0	0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0	1	1	1
	80%	0	0	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1	1	1
4	60%	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1
	80%	1	1	1	0	1	0	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	1	0	0	1	0	0	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0
5	60%	0	1	0	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0	1	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1
	80%	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table B.8: Randomly generated capabilities of the robot(s) to perform each task for the 45-task problem instances for replications 6 to 10

		Randomly generated capabilities of the robot(s) to perform each task																																												
Rep. #	Cap.	$i=1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	40%	0	0	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	1	1	0	1	1	1	0	0	0	1	0	0	1	0	0	0
6	60%	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0	1	1
	80%	1	0	1	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	1	1	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0
7	60%	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0
	80%	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	1	1	0	1	1	1	1	0
8	60%	0	0	1	1	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1	0
	80%	1	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	1	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0	1	0
9	60%	1	1	1	1	1	0	1	0	0	1	0	1	1	1	0	1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1
	80%	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	40%	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0
10	60%	1	1	0	0	1	1	0	1	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1	1	1	1	0	1	0
	80%	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1	1	0	1	0
	100%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Appendix C

DETAILED SOLUTIONS OF SELECTED PROBLEM INSTANCES FOR TYPE-II RSALBP

Table C.1: Optimal cycle times, cut-off values of the workloads, and robot assignments to workstations of 25-task problem, when capability$= 80\%$, $r = 1$, $cv = 0.2$, $\alpha = 0.90$, $cvr = 0$, and $cmr = 0.6$

		Cut-off values of the workloads (sc)						Nb. of assigned tasks						Robots assigned					
#	ct (sc)	$j=1$	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	29.40	28.25	28.78	27.89	26.52	28.85	29.40	4	3	5	4	3	6	-	-	-	-	-	R
2	29.62	29.42	26.52	29.40	29.62	29.50	26.52	4	3	9	2	4	3	-	-	R	-	-	-
3	29.62	27.89	28.78	28.80	29.62	29.50	26.52	5	3	8	2	4	3	-	-	R	-	-	-
4	29.62	28.25	29.62	28.80	29.62	29.50	26.52	4	4	8	2	4	3	-	-	R	-	-	-
5	29.50	27.89	29.18	28.78	28.20	29.50	26.52	5	5	3	5	4	3	-	-	-	R	-	-
6	29.50	29.42	27.30	28.25	28.80	29.50	26.52	4	4	4	6	4	3	-	-	-	R	-	-
7	29.50	27.89	28.78	29.18	28.20	29.50	26.52	5	3	5	5	4	3	-	-	-	R	-	-
8	29.40	28.25	28.31	29.40	26.52	28.85	29.40	4	4	4	4	3	6	-	-	-	-	-	R
9	29.62	29.40	27.70	27.77	29.62	29.50	26.52	7	6	3	2	4	3	R	-	-	-	-	-
10	29.62	29.40	29.01	26.52	29.62	29.50	26.52	7	5	4	2	4	3	R	-	-	-	-	-

Table C.2: Optimal cycle times, cut-off values of the workloads, and robot assignments to workstations of 25-task problem, when capability$= 80\%$, $r = 1$, $cv = 0.4$, $\alpha = 0.95$, $cvr = 0$, and $cmr = 1.2$

		Cut-off values of the workloads (sc)						Nb. of assigned tasks						Robots assigned					
#	ct (sc)	$j = 1$	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	38.87	36.48	38.37	37.80	38.40	36.86	38.87	5	6	4	4	2	4	-	-	-	R	-	-
2	38.87	38.69	36.24	37.67	38.40	35.91	38.87	4	4	7	3	3	4	-	-	-	R	-	-
3	38.87	38.69	38.11	35.85	38.40	35.91	38.87	4	6	5	3	3	4	-	-	-	R	-	-
4	41.70	40.38	41.70	40.80	36.86	34.97	32.05	5	7	4	2	4	3	-	-	R	-	-	-
5	38.87	37.75	37.09	37.80	38.40	35.91	38.87	5	6	4	3	3	4	-	-	-	R	-	-
6	38.87	37.75	37.09	37.80	38.40	35.91	38.87	5	6	4	3	3	4	-	-	-	R	-	-
7	38.87	37.75	37.09	37.80	37.79	36.00	38.87	5	6	4	3	3	4	-	-	-	-	R	-
8	38.87	38.69	38.09	35.84	38.40	36.86	38.87	4	5	6	4	2	4	-	-	-	R	-	-
9	41.98	41.49	40.80	41.98	40.40	24.87	38.87	5	5	6	4	1	4	-	R	-	-	-	-
10	38.87	38.69	38.09	35.84	38.40	35.91	38.87	4	5	6	3	3	4	-	-	-	R	-	-

Table C.3: Optimal cycle times, cut-off values of the workloads, and robot assignments to workstations of 30-task problem, when capability$= 60\%$, $r = 1$, $cv = 0.4$, $\alpha = 0.95$, $cvr = 0.5$, and $cmr = 0.8$

		Cut-off values of the workloads (sc)						Nb. of assigned tasks						Robots assigned					
#	ct (sc)	$j = 1$	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	68.09	68.09	68.04	67.53	67.83	67.99	67.45	7	5	4	5	5	4	-	-	-	R	-	-
2	68.45	68.45	68.30	67.62	68.36	67.99	67.45	5	7	4	5	5	4	-	R	-	-	-	-
3	68.54	68.45	67.90	68.54	68.37	67.99	67.45	5	8	4	4	5	4	-	R	-	-	-	-
4	68.35	68.30	68.25	66.78	68.35	67.76	68.07	6	6	5	5	3	5	-	-	R	-	-	-
5	68.07	67.76	67.23	68.03	67.63	67.76	68.07	7	4	5	6	3	5	-	-	R	-	-	-
6	68.20	67.55	68.20	68.20	67.48	67.53	67.45	6	5	5	5	5	4	-	-	-	R	-	-
7	68.45	68.45	68.25	68.34	68.07	66.57	67.45	5	6	6	4	5	4	-	-	R	-	-	-
8	68.21	67.76	67.23	68.21	67.99	66.94	68.07	7	4	5	5	4	5	-	-	-	-	R	-
9	68.52	68.52	67.63	67.52	68.36	67.99	67.45	6	4	6	5	5	4	-	-	R	-	-	-
10	68.45	68.45	68.25	68.34	68.07	66.57	67.45	5	6	6	4	5	4	-	-	R	-	-	-

Table C.4: Optimal cycle times, cut-off values of the workloads, and robot assignments to workstations of 35-task problem, when capability= 100%, $r = 1$, $cv = 0.4$, $\alpha = 0.95$, $cvr = 0.0$, and $cmr = 0.8$

	Cut-off values of the workloads (sc)									Nb. of assigned tasks									Robots Assigned								
ct	$j = 1$	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
71.69	71.20	71.69	71.20	70.81	70.83	70.06	71.20	71.52	71.39	3	9	3	2	4	5	3	2	4	R	-	-	-	-	-	-	-	-

Table C.5: Best found cycle time, cut-off values of the workloads, and robot assignments to workstations of 45-task problem, when capability= 100%, $r = 2$, $cv = 0.4$, $\alpha = 0.95$, $cvr = 0.0$, and $cmr = 0.6$

	Cut-off values of the workloads (sc)									Nb. of assigned tasks									Robots Assigned								
ct	$j = 1$	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
64.50	64.50	64.48	63.60	64.42	64.45	64.20	63.74	64.26	64.17	7	6	3	5	4	5	7	3	5	-	-	R	-	-	R	-	-	-

Appendix D

DETAILED CPU TIME RESULTS OF PROPOSED FORMULATIONS ON ALL PROBLEM INSTANCES FOR TYPE-II RSALBP

Table D.1: Average CPU time results of 10 replications for MISOCP formulation on 25-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	6.0	5.3	5.4	5.0	3.6	4.8	3.0	3.5
2	0.0	0.6	0.2	0.95	5.8	5.9	5.3	5.2	4.3	4.9	3.4	4.0
3	0.0	0.6	0.4	0.90	5.8	5.3	5.1	5.1	4.3	4.9	4.0	4.8
4	0.0	0.6	0.4	0.95	6.3	5.8	5.1	5.1	5.1	4.5	4.5	7.0
5	0.0	0.8	0.2	0.90	5.7	5.6	5.1	4.9	4.1	5.4	3.1	4.5
6	0.0	0.8	0.2	0.95	7.2	6.1	5.4	5.3	5.0	5.4	4.0	3.7
7	0.0	0.8	0.4	0.90	6.0	5.9	5.4	5.2	5.2	5.6	3.3	4.7
8	0.0	0.8	0.4	0.95	5.7	5.5	5.4	5.2	5.5	5.7	3.1	5.1
9	0.0	1.0	0.2	0.90	6.7	5.5	5.5	5.0	7.6	5.7	3.9	4.0
10	0.0	1.0	0.2	0.95	5.4	8.5	5.5	5.2	7.2	6.2	5.3	6.9
11	0.0	1.0	0.4	0.90	5.9	5.9	6.0	5.3	7.8	5.8	5.5	7.6
12	0.0	1.0	0.4	0.95	5.9	6.3	6.1	5.7	8.5	6.2	4.6	7.2
13	0.0	1.2	0.2	0.90	6.7	6.8	7.2	6.8	7.6	7.3	9.2	9.5
14	0.0	1.2	0.2	0.95	6.9	6.3	7.3	7.1	7.7	7.8	10.7	10.0
15	0.0	1.2	0.4	0.90	6.9	7.5	7.6	7.8	8.5	9.1	8.7	15.9
16	0.0	1.2	0.4	0.95	7.0	7.9	7.6	7.7	8.9	9.3	11.4	18.7
17	0.5	0.6	0.2	0.90	5.8	7.1	5.9	6.8	5.5	9.6	4.7	13.0
18	0.5	0.6	0.2	0.95	5.9	6.6	5.7	7.1	6.0	8.8	3.8	16.7
19	0.5	0.6	0.4	0.90	6.0	6.7	6.2	6.6	6.2	9.3	4.7	13.0
20	0.5	0.6	0.4	0.95	6.3	7.1	5.9	6.6	6.2	7.9	4.3	13.1
21	0.5	0.8	0.2	0.90	6.1	7.2	5.9	7.1	6.4	9.4	2.8	12.2
22	0.5	0.8	0.2	0.95	6.2	7.2	5.8	6.9	6.5	9.1	3.7	14.9
23	0.5	0.8	0.4	0.90	6.6	7.0	6.5	7.3	7.0	10.4	3.9	15.7
24	0.5	0.8	0.4	0.95	6.8	9.4	6.7	7.6	7.5	10.2	6.5	17.2
25	0.5	1.0	0.2	0.90	8.1	8.6	8.3	8.6	8.7	10.9	13.2	15.5
26	0.5	1.0	0.2	0.95	8.3	8.2	8.5	9.2	8.7	11.0	12.2	15.9
27	0.5	1.0	0.4	0.90	7.4	8.4	8.4	9.4	9.6	11.3	11.0	17.6
28	0.5	1.0	0.4	0.95	8.8	9.2	8.7	9.4	10.3	12.3	14.8	23.7
29	0.5	1.2	0.2	0.90	6.4	6.7	6.7	6.7	7.7	7.9	12.5	12.3
30	0.5	1.2	0.2	0.95	7.0	6.7	7.2	6.3	8.3	8.0	9.5	13.8
31	0.5	1.2	0.4	0.90	6.9	7.1	7.4	7.4	8.9	9.6	10.9	21.1
32	0.5	1.2	0.4	0.95	7.4	7.3	7.1	7.6	8.9	12.6	14.4	13.5

Table D.2: Average CPU time results of 10 replications for CP formulation on 25-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	15.5	14.2	13.8	16.2	14.3	15.4	13.8	33.2
2	0.0	0.6	0.2	0.95	15.1	14.9	13.8	17.8	14.2	13.1	16.5	20.8
3	0.0	0.6	0.4	0.90	15.8	16.9	14.9	15.4	15.3	13.8	19.3	19.2
4	0.0	0.6	0.4	0.95	16.1	15.6	14.4	16.9	15.3	13.5	22.2	22.6
5	0.0	0.8	0.2	0.90	14.1	14.6	13.0	13.3	17.1	15.1	20.6	23.6
6	0.0	0.8	0.2	0.95	14.6	14.4	12.8	14.2	14.4	15.7	18.4	40.5
7	0.0	0.8	0.4	0.90	15.5	16.0	15.1	16.0	15.3	16.5	27.3	41.5
8	0.0	0.8	0.4	0.95	14.9	15.2	14.6	16.2	15.6	13.9	17.3	46.7
9	0.0	1.0	0.2	0.90	13.7	14.9	13.2	12.5	14.3	17.0	16.2	57.8
10	0.0	1.0	0.2	0.95	15.6	15.2	13.2	13.9	14.4	17.2	16.6	45.5
11	0.0	1.0	0.4	0.90	16.6	16.3	14.2	14.5	15.6	15.9	19.1	38.4
12	0.0	1.0	0.4	0.95	15.0	15.4	15.0	15.2	17.0	15.6	20.8	42.1
13	0.0	1.2	0.2	0.90	13.9	17.4	13.6	13.2	15.1	15.9	17.6	38.6
14	0.0	1.2	0.2	0.95	15.0	14.9	13.4	13.3	15.8	16.2	19.2	52.3
15	0.0	1.2	0.4	0.90	15.5	15.0	14.0	14.2	16.1	16.7	14.2	38.6
16	0.0	1.2	0.4	0.95	14.6	15.4	13.5	13.4	13.3	15.7	15.5	37.8
17	0.5	0.6	0.2	0.90	15.4	15.7	14.9	16.0	17.6	17.5	22.8	44.6
18	0.5	0.6	0.2	0.95	16.3	15.8	15.8	19.2	16.9	17.5	16.1	39.1
19	0.5	0.6	0.4	0.90	16.0	17.4	16.2	18.9	18.0	15.7	17.6	33.4
20	0.5	0.6	0.4	0.95	16.1	16.7	15.8	18.9	17.8	14.6	17.3	38.9
21	0.5	0.8	0.2	0.90	14.9	15.8	15.4	14.8	18.4	19.7	19.7	54.1
22	0.5	0.8	0.2	0.95	16.3	15.7	14.9	16.3	18.1	18.6	28.9	50.6
23	0.5	0.8	0.4	0.90	17.5	17.4	16.0	16.1	18.3	18.1	17.6	31.2
24	0.5	0.8	0.4	0.95	15.9	16.9	14.9	14.7	17.6	17.6	18.6	45.4
25	0.5	1.0	0.2	0.90	15.4	16.1	13.6	14.4	16.2	17.6	16.0	45.2
26	0.5	1.0	0.2	0.95	15.5	17.6	15.1	16.7	18.4	18.8	19.5	37.1
27	0.5	1.0	0.4	0.90	16.5	16.8	16.1	15.3	17.1	19.8	16.7	36.0
28	0.5	1.0	0.4	0.95	14.7	16.8	14.8	14.9	15.0	19.3	18.6	39.5
29	0.5	1.2	0.2	0.90	14.5	15.4	14.1	15.0	15.9	19.0	17.7	46.9
30	0.5	1.2	0.2	0.95	15.6	17.3	14.5	15.2	18.0	18.6	23.8	32.6
31	0.5	1.2	0.4	0.90	16.5	15.9	16.5	16.4	19.4	21.3	25.9	55.0
32	0.5	1.2	0.4	0.95	15.2	17.8	16.2	16.1	20.1	21.1	22.6	68.7

Table D.3: Average CPU time results of 10 replications for MISOCP formulation on 30-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	10.7	8.6	12.4	8.5	28.2	28.7	58.7	135.2
2	0.0	0.6	0.2	0.95	12.2	10.1	12.5	8.2	33.3	21.5	83.6	143.6
3	0.0	0.6	0.4	0.90	12.0	10.0	11.0	8.0	29.2	18.1	76.7	198.2
4	0.0	0.6	0.4	0.95	11.8	11.0	11.4	9.1	35.1	17.4	83.2	230.8
5	0.0	0.8	0.2	0.90	14.9	10.1	19.8	8.7	46.3	29.7	124.2	124.3
6	0.0	0.8	0.2	0.95	13.7	10.6	19.4	9.3	57.2	27.6	185.4	161.2
7	0.0	0.8	0.4	0.90	12.4	10.7	19.8	10.1	49.9	28.6	114.8	207.1
8	0.0	0.8	0.4	0.95	12.1	10.8	15.4	9.6	40.3	39.6	79.0	248.7
9	0.0	1.0	0.2	0.90	20.8	13.4	33.8	13.7	74.2	65.1	97.2	316.6
10	0.0	1.0	0.2	0.95	22.0	12.9	30.8	12.4	68.8	66.3	74.3	288.0
11	0.0	1.0	0.4	0.90	23.5	17.5	39.0	14.0	81.4	44.9	109.5	141.4
12	0.0	1.0	0.4	0.95	25.3	13.9	44.5	16.0	101.1	47.6	190.7	227.2
13	0.0	1.2	0.2	0.90	22.5	14.1	29.3	24.2	59.3	48.8	109.5	161.6
14	0.0	1.2	0.2	0.95	23.1	16.2	35.4	21.9	71.6	61.1	105.3	145.4
15	0.0	1.2	0.4	0.90	27.0	16.5	42.4	24.0	81.4	97.2	136.5	208.2
16	0.0	1.2	0.4	0.95	32.5	20.6	46.5	23.9	116.2	99.2	347.0	326.3
17	0.5	0.6	0.2	0.90	10.5	12.1	12.0	12.3	73.3	37.7	85.8	342.7
18	0.5	0.6	0.2	0.95	12.1	13.2	14.8	13.7	49.4	44.4	94.1	404.5
19	0.5	0.6	0.4	0.90	11.1	12.6	16.4	13.9	43.1	49.0	103.1	560.2
20	0.5	0.6	0.4	0.95	12.7	15.3	16.2	12.9	42.0	52.1	183.5	476.8
21	0.5	0.8	0.2	0.90	14.5	14.2	29.4	15.2	55.7	62.4	159.1	361.3
22	0.5	0.8	0.2	0.95	17.4	13.9	30.1	15.2	62.7	73.7	266.5	226.1
23	0.5	0.8	0.4	0.90	18.6	14.5	36.2	14.2	86.8	57.7	184.1	332.7
24	0.5	0.8	0.4	0.95	22.2	15.4	34.2	16.2	103.0	74.0	285.7	461.7
25	0.5	1.0	0.2	0.90	25.6	18.4	47.7	23.5	83.5	94.1	383.9	307.9
26	0.5	1.0	0.2	0.95	25.7	18.9	47.9	24.9	104.3	83.1	258.2	447.6
27	0.5	1.0	0.4	0.90	31.2	21.2	45.8	25.9	123.9	149.7	322.5	1216.1
28	0.5	1.0	0.4	0.95	32.7	19.5	53.6	24.6	144.7	155.2	230.2	526.8
29	0.5	1.2	0.2	0.90	18.6	16.8	27.5	27.4	82.0	78.9	98.6	564.8
30	0.5	1.2	0.2	0.95	20.3	17.8	35.9	25.5	89.7	125.5	183.6	255.3
31	0.5	1.2	0.4	0.90	27.0	20.2	45.8	29.4	93.8	98.2	225.5	336.5
32	0.5	1.2	0.4	0.95	33.2	21.6	51.2	30.6	121.0	117.9	210.2	608.5

Table D.4: Average CPU time results of 10 replications for CP formulation on 30-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	20.2	14.5	26.9	17.2	73.9	112.7	284.9	2227.4
2	0.0	0.6	0.2	0.95	22.5	16.3	26.2	16.5	74.6	91.1	355.7	2095.8
3	0.0	0.6	0.4	0.90	20.2	14.9	22.6	15.1	59.6	74.6	277.7	1381.4
4	0.0	0.6	0.4	0.95	19.1	14.8	21.8	14.9	59.6	46.4	272.5	1525.8
5	0.0	0.8	0.2	0.90	31.2	17.5	59.5	21.9	185.4	149.1	807.3	1556.3
6	0.0	0.8	0.2	0.95	31.7	20.9	62.6	19.5	203.2	142.6	948.8	1776.3
7	0.0	0.8	0.4	0.90	23.0	15.6	27.7	16.5	94.5	96.6	281.6	1174.9
8	0.0	0.8	0.4	0.95	20.2	17.0	24.0	16.3	80.5	107.0	259.0	1682.2
9	0.0	1.0	0.2	0.90	38.6	20.0	84.3	34.3	256.1	311.0	967.2	3039.6
10	0.0	1.0	0.2	0.95	39.5	20.9	80.3	32.0	230.1	277.6	918.3	2601.9
11	0.0	1.0	0.4	0.90	35.2	19.5	66.0	33.2	200.8	124.3	799.3	957.5
12	0.0	1.0	0.4	0.95	30.7	18.8	63.1	17.1	209.7	146.4	752.6	1080.5
13	0.0	1.2	0.2	0.90	37.7	22.0	89.5	49.1	229.9	306.3	728.2	1785.8
14	0.0	1.2	0.2	0.95	41.2	23.6	96.6	43.3	253.4	320.0	867.4	2448.7
15	0.0	1.2	0.4	0.90	43.0	22.2	86.4	40.9	222.3	255.2	765.1	1856.1
16	0.0	1.2	0.4	0.95	39.2	18.9	82.6	35.6	250.5	240.9	816.8	2120.5
17	0.5	0.6	0.2	0.90	23.5	17.7	29.9	18.9	86.1	129.7	336.9	2635.5
18	0.5	0.6	0.2	0.95	25.1	18.7	29.0	18.4	89.5	132.5	394.0	2968.0
19	0.5	0.6	0.4	0.90	22.6	16.3	29.9	18.2	79.8	124.0	325.1	3019.9
20	0.5	0.6	0.4	0.95	20.4	16.4	26.6	17.7	80.8	108.7	320.2	2528.3
21	0.5	0.8	0.2	0.90	40.3	18.8	77.7	28.6	230.0	278.9	954.4	1839.8
22	0.5	0.8	0.2	0.95	33.9	19.2	70.6	22.9	234.0	194.6	1039.5	2000.8
23	0.5	0.8	0.4	0.90	34.5	19.8	70.4	19.1	215.8	173.1	1037.8	1759.9
24	0.5	0.8	0.4	0.95	35.8	20.4	73.5	19.2	212.5	147.4	1126.0	1574.4
25	0.5	1.0	0.2	0.90	42.2	21.3	93.7	45.2	250.7	303.7	899.9	2841.9
26	0.5	1.0	0.2	0.95	39.9	22.7	88.7	44.6	254.9	286.4	986.8	2853.6
27	0.5	1.0	0.4	0.90	42.5	20.5	87.5	45.0	257.5	316.5	983.2	2767.7
28	0.5	1.0	0.4	0.95	40.2	22.6	91.6	31.5	264.4	304.4	743.4	2925.2
29	0.5	1.2	0.2	0.90	38.6	23.4	97.8	57.2	240.4	439.2	779.6	3344.3
30	0.5	1.2	0.2	0.95	40.5	23.5	104.0	49.2	257.1	369.7	857.8	2360.9
31	0.5	1.2	0.4	0.90	46.1	24.5	99.6	61.3	273.8	351.4	994.0	2344.0
32	0.5	1.2	0.4	0.95	47.4	22.2	104.2	46.7	283.0	350.4	977.5	2572.3

Table D.5: Average CPU time results of 10 replications for MISOCP formulation on 35-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	72.2	67.7	101.6	68.0	314.6	167.2	340.4	795.9
2	0.0	0.6	0.2	0.95	118.5	89.3	140.9	74.5	306.6	181.4	1581.7	227.2
3	0.0	0.6	0.4	0.90	129.0	112.3	132.8	69.9	537.0	123.0	291.2	486.6
4	0.0	0.6	0.4	0.95	139.1	139.0	100.4	66.5	260.9	134.3	418.7	347.7
5	0.0	0.8	0.2	0.90	209.7	97.0	238.0	116.4	435.1	343.0	316.6	1801.3
6	0.0	0.8	0.2	0.95	139.9	113.6	202.6	147.7	305.5	363.6	305.3	893.7
7	0.0	0.8	0.4	0.90	138.0	125.7	195.5	144.1	372.6	305.7	1376.7	1060.9
8	0.0	0.8	0.4	0.95	186.0	123.1	233.2	126.9	397.9	244.1	715.1	225.4
9	0.0	1.0	0.2	0.90	224.5	181.3	328.3	424.9	620.3	1216.9	1133.3	3460.7
10	0.0	1.0	0.2	0.95	356.9	140.5	569.2	317.7	1162.8	930.2	1153.1	3600.1
11	0.0	1.0	0.4	0.90	322.8	166.5	602.3	566.7	637.8	934.9	1288.4	2220.2
12	0.0	1.0	0.4	0.95	317.1	227.6	380.2	322.0	655.2	1036.7	689.5	2357.6
13	0.0	1.2	0.2	0.90	168.8	194.7	275.6	268.6	356.1	642.4	915.0	2018.9
14	0.0	1.2	0.2	0.95	267.2	163.8	399.3	465.4	581.2	1453.5	3259.6	2019.0
15	0.0	1.2	0.4	0.90	396.1	283.9	520.3	505.0	844.1	1254.2	2893.8	3600.1
16	0.0	1.2	0.4	0.95	500.6	337.8	512.6	570.5	1028.1	1543.0	979.6	3600.1
17	0.5	0.6	0.2	0.90	123.2	139.4	134.1	158.3	511.8	447.8	434.3	858.3
18	0.5	0.6	0.2	0.95	146.6	118.6	241.3	209.2	296.7	408.9	1543.3	3600.1
19	0.5	0.6	0.4	0.90	201.9	142.2	225.9	164.7	773.5	358.1	1267.3	912.2
20	0.5	0.6	0.4	0.95	144.8	234.0	238.3	216.1	492.9	393.0	568.5	1843.0
21	0.5	0.8	0.2	0.90	132.0	117.1	286.6	229.8	671.7	663.9	911.7	3311.1
22	0.5	0.8	0.2	0.95	151.9	153.5	293.1	223.8	873.2	722.0	1536.1	3600.1
23	0.5	0.8	0.4	0.90	523.3	230.2	302.8	331.9	736.4	1345.8	927.2	3600.1
24	0.5	0.8	0.4	0.95	213.7	165.5	545.6	366.8	711.3	1050.0	861.3	2274.1
25	0.5	1.0	0.2	0.90	209.3	322.5	692.8	414.4	586.8	917.7	884.0	1343.1
26	0.5	1.0	0.2	0.95	280.0	183.2	560.1	362.9	659.4	995.3	1014.9	3600.1
27	0.5	1.0	0.4	0.90	333.1	262.9	526.0	581.5	841.4	1790.4	3600.2	3600.1
28	0.5	1.0	0.4	0.95	213.8	288.7	531.7	920.1	1125.8	1474.0	884.9	3600.1
29	0.5	1.2	0.2	0.90	331.9	383.6	397.2	365.0	583.9	1009.1	419.1	1453.6
30	0.5	1.2	0.2	0.95	269.6	341.6	463.8	722.6	668.2	1025.6	1072.3	1700.2
31	0.5	1.2	0.4	0.90	397.0	358.8	492.4	703.1	1001.9	1373.6	3600.2	3600.1
32	0.5	1.2	0.4	0.95	551.7	298.4	824.6	580.1	1191.1	2364.1	1435.6	3600.1

Table D.6: Average CPU time results of 10 replications for CP formulation on 35-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	49.6	53.8	74.8	45.5	235.4	249.5	1555.2	1652.5
2	0.0	0.6	0.2	0.95	48.8	46.4	75.3	45.2	212.5	167.2	925.2	562.8
3	0.0	0.6	0.4	0.90	45.6	46.7	77.3	38.4	199.6	117.7	474.4	322.5
4	0.0	0.6	0.4	0.95	46.6	48.3	50.7	35.7	129.0	75.5	505.6	358.6
5	0.0	0.8	0.2	0.90	98.4	72.1	233.3	127.5	648.6	599.5	2096.7	3600.1
6	0.0	0.8	0.2	0.95	94.8	63.6	190.4	106.5	363.0	516.7	552.1	3600.1
7	0.0	0.8	0.4	0.90	69.2	59.8	119.6	71.7	307.1	283.5	1007.2	3549.8
8	0.0	0.8	0.4	0.95	63.9	59.8	109.6	56.1	272.7	152.2	1166.8	518.6
9	0.0	1.0	0.2	0.90	87.1	79.1	208.5	298.1	546.0	1318.1	1366.4	3600.1
10	0.0	1.0	0.2	0.95	92.5	78.9	215.1	236.4	528.1	951.1	1969.4	3600.1
11	0.0	1.0	0.4	0.90	78.2	66.2	180.6	168.2	431.2	779.2	1051.2	3600.2
12	0.0	1.0	0.4	0.95	80.6	66.7	172.9	151.0	421.6	703.0	1512.8	3600.2
13	0.0	1.2	0.2	0.90	106.3	95.3	223.2	311.3	556.4	1172.5	2019.0	3600.2
14	0.0	1.2	0.2	0.95	96.4	93.2	227.2	300.0	521.1	1033.3	1411.6	3600.2
15	0.0	1.2	0.4	0.90	114.7	86.6	266.7	285.2	723.4	687.6	1722.8	3111.8
16	0.0	1.2	0.4	0.95	88.8	76.7	172.9	245.3	422.5	813.0	999.9	3600.2
17	0.5	0.6	0.2	0.90	82.0	55.9	94.8	70.8	252.8	255.4	1188.9	2971.2
18	0.5	0.6	0.2	0.95	72.8	63.6	103.9	70.8	249.9	278.3	1418.5	2887.7
19	0.5	0.6	0.4	0.90	59.2	66.7	118.7	68.7	355.6	270.2	1646.9	746.2
20	0.5	0.6	0.4	0.95	64.5	58.4	108.0	62.1	304.8	178.8	1194.0	769.1
21	0.5	0.8	0.2	0.90	93.9	78.4	233.0	240.9	701.9	737.8	2609.1	3600.1
22	0.5	0.8	0.2	0.95	96.6	79.2	239.4	180.8	739.4	671.8	2458.6	3600.1
23	0.5	0.8	0.4	0.90	83.4	70.9	200.2	175.8	482.8	882.5	1056.3	3600.2
24	0.5	0.8	0.4	0.95	90.4	68.0	209.4	162.8	512.8	899.7	1181.7	3600.1
25	0.5	1.0	0.2	0.90	109.7	93.4	216.3	267.9	537.8	857.6	1830.1	3600.1
26	0.5	1.0	0.2	0.95	108.8	97.6	228.0	327.8	543.6	1003.3	1391.9	3600.1
27	0.5	1.0	0.4	0.90	120.4	115.3	305.9	312.0	817.0	984.4	2585.1	3600.1
28	0.5	1.0	0.4	0.95	108.9	98.8	211.6	302.4	509.6	983.4	1439.0	3600.2
29	0.5	1.2	0.2	0.90	90.0	105.3	210.0	253.9	503.2	866.9	1005.4	3586.5
30	0.5	1.2	0.2	0.95	93.5	92.4	219.0	270.6	506.0	959.4	1089.3	2624.6
31	0.5	1.2	0.4	0.90	101.1	114.3	231.5	331.4	598.1	1157.1	1888.2	3600.2
32	0.5	1.2	0.4	0.95	126.3	113.7	284.9	358.4	637.7	1349.1	2179.6	3600.2

Table D.7: Average CPU time results of 10 replications for MISOCP formulation on 45-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	397.4	410.5	430.2	750.8	2546.4	2533.7	3600.2	3607.7
2	0.0	0.6	0.2	0.95	75.1	104.7	396.2	427.9	2534.9	2528.5	3605.2	3600.2
3	0.0	0.6	0.4	0.90	404.5	69.1	379.5	367.7	2526.9	2527.6	3600.6	3600.2
4	0.0	0.6	0.4	0.95	49.1	64.9	394.9	370.2	2528.0	2544.5	3600.3	3600.1
5	0.0	0.8	0.2	0.90	124.7	268.1	549.9	436.2	2556.3	2544.4	3601.6	3604.3
6	0.0	0.8	0.2	0.95	123.1	246.5	474.0	437.6	2524.5	2532.2	3600.3	3605.4
7	0.0	0.8	0.4	0.90	104.1	148.8	422.6	393.4	2549.3	2533.6	3600.2	3604.1
8	0.0	0.8	0.4	0.95	70.5	117.7	413.2	398.5	2539.3	2533.5	3600.2	3602.3
9	0.0	1.0	0.2	0.90	2684.9	1252.4	2658.6	1008.4	3061.2	2626.4	3612.2	3600.3
10	0.0	1.0	0.2	0.95	392.8	810.7	904.3	701.8	2661.7	2569.6	3600.5	3600.3
11	0.0	1.0	0.4	0.90	331.7	338.8	534.7	524.4	2618.7	2546.5	3604.3	3600.2
12	0.0	1.0	0.4	0.95	283.9	259.1	512.1	583.4	2602.6	2539.3	3603.6	3600.2
13	0.0	1.2	0.2	0.90	3602.1	3600.8	3604.0	3603.7	3603.6	3603.8	3605.1	3605.3
14	0.0	1.2	0.2	0.95	527.5	2191.2	690.0	1493.7	2745.4	2754.6	3605.2	3602.0
15	0.0	1.2	0.4	0.90	218.6	832.2	685.2	777.3	2626.1	2676.5	3605.5	3604.7
16	0.0	1.2	0.4	0.95	439.2	293.4	604.3	736.7	2619.1	2624.5	3600.2	3604.4
17	0.5	0.6	0.2	0.90	42.7	194.5	389.1	413.5	2553.3	2559.4	3600.7	3604.6
18	0.5	0.6	0.2	0.95	56.8	99.9	427.3	375.8	2532.5	2535.5	3600.3	3603.6
19	0.5	0.6	0.4	0.90	72.4	196.8	372.7	736.5	2534.9	2525.9	3600.3	3603.8
20	0.5	0.6	0.4	0.95	62.4	86.2	379.3	383.7	2527.3	2523.1	3600.3	3604.0
21	0.5	0.8	0.2	0.90	232.6	458.8	559.7	649.8	2525.9	2525.4	3600.3	3603.1
22	0.5	0.8	0.2	0.95	279.6	440.6	487.8	564.2	2599.1	2526.0	3600.3	3604.2
23	0.5	0.8	0.4	0.90	208.0	367.2	529.4	648.9	2568.1	2567.9	3601.0	3600.6
24	0.5	0.8	0.4	0.95	192.0	190.6	464.3	536.0	2556.4	2560.6	3600.4	3606.5
25	0.5	1.0	0.2	0.90	3602.4	3600.5	3613.1	3355.8	3603.1	3604.6	3603.3	3605.4
26	0.5	1.0	0.2	0.95	734.9	941.9	1144.5	1006.7	2828.2	2967.6	3609.2	3604.4
27	0.5	1.0	0.4	0.90	744.1	822.8	624.4	978.8	2720.4	2713.7	3603.8	3603.4
28	0.5	1.0	0.4	0.95	367.2	602.2	655.5	831.2	2625.3	2577.3	3601.3	3600.2
29	0.5	1.2	0.2	0.90	3603.0	3602.0	3602.6	3602.9	3603.2	3602.2	3600.2	3600.2
30	0.5	1.2	0.2	0.95	3625.8	3601.3	3607.0	3604.3	3451.9	3602.7	3600.4	3600.2
31	0.5	1.2	0.4	0.90	311.6	345.5	582.6	541.3	2101.8	1865.8	3215.4	2273.9
32	0.5	1.2	0.4	0.95	225.4	404.9	529.9	593.0	2115.8	2368.0	3600.2	3600.2

Table D.8: Average CPU time results of 10 replications for CP formulation on 45-task problem instances

					40% Capability		60% Capability		80% Capability		100% Capability	
Ins.#	cvr	cmr	cv	α	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	0.0	0.6	0.2	0.90	12.9	13.5	369.7	369.2	2523.4	2522.7	3600.1	3600.1
2	0.0	0.6	0.2	0.95	10.7	12.9	370.3	369.6	2523.8	2523.0	3600.2	3600.1
3	0.0	0.6	0.4	0.90	9.8	10.0	369.4	368.5	2523.8	2522.2	3600.1	3600.2
4	0.0	0.6	0.4	0.95	8.4	9.2	367.0	367.6	2522.9	2522.1	3600.1	3600.1
5	0.0	0.8	0.2	0.90	12.2	13.8	370.2	369.8	2523.7	2523.0	3600.1	3600.1
6	0.0	0.8	0.2	0.95	12.2	12.1	370.3	369.4	2523.8	2523.3	3600.2	3600.1
7	0.0	0.8	0.4	0.90	10.4	11.0	369.8	368.6	2523.7	2522.4	3600.2	3600.2
8	0.0	0.8	0.4	0.95	8.6	9.6	368.8	368.0	2522.8	2522.6	3600.1	3600.1
9	0.0	1.0	0.2	0.90	252.5	14.7	418.6	369.8	2531.2	2524.1	3600.2	3600.1
10	0.0	1.0	0.2	0.95	11.5	14.7	371.2	369.0	2524.3	2524.1	3600.1	3600.2
11	0.0	1.0	0.4	0.90	11.4	11.0	370.9	368.5	2524.2	2523.1	3600.2	3600.2
12	0.0	1.0	0.4	0.95	8.7	9.9	370.2	368.4	2523.5	2522.9	3600.2	3600.1
13	0.0	1.2	0.2	0.90	3600.2	3600.1	3600.1	3600.1	3600.1	3600.1	3600.1	3600.1
14	0.0	1.2	0.2	0.95	16.8	692.2	373.6	437.7	2524.4	2525.8	3600.1	3600.1
15	0.0	1.2	0.4	0.90	11.2	11.5	371.4	368.9	2524.4	2523.7	3600.2	3600.1
16	0.0	1.2	0.4	0.95	8.7	10.4	369.7	368.5	2524.3	2523.3	3600.1	3600.1
17	0.5	0.6	0.2	0.90	13.1	15.1	371.4	370.7	2524.5	2523.5	3600.1	3600.1
18	0.5	0.6	0.2	0.95	12.2	12.9	372.1	370.0	2524.3	2523.2	3600.1	3600.1
19	0.5	0.6	0.4	0.90	12.1	11.8	371.7	370.0	2523.9	2523.1	3600.2	3600.2
20	0.5	0.6	0.4	0.95	9.3	10.9	370.2	369.9	2523.2	2522.5	3600.1	3600.2
21	0.5	0.8	0.2	0.90	12.9	15.7	372.2	371.4	2524.3	2524.3	3600.1	3600.1
22	0.5	0.8	0.2	0.95	12.8	13.4	371.8	369.7	2524.5	2523.5	3600.1	3600.1
23	0.5	0.8	0.4	0.90	11.7	11.7	371.8	369.6	2524.5	2523.4	3600.1	3600.1
24	0.5	0.8	0.4	0.95	9.5	11.5	370.1	369.6	2524.4	2523.3	3600.1	3600.1
25	0.5	1.0	0.2	0.90	3600.1	2062.3	3600.2	935.3	3600.1	2570.2	3600.1	3600.1
26	0.5	1.0	0.2	0.95	13.1	12.3	372.2	371.4	2525.0	2524.7	3600.1	3600.1
27	0.5	1.0	0.4	0.90	12.2	12.1	372.1	370.5	2525.3	2523.7	3600.1	3600.1
28	0.5	1.0	0.4	0.95	10.7	11.3	371.2	369.5	2523.8	2524.0	3600.1	3600.1
29	0.5	1.2	0.2	0.90	3600.1	3600.2	3600.1	3600.2	3600.1	3600.1	3600.1	3600.1
30	0.5	1.2	0.2	0.95	2505.1	3600.2	1921.2	3600.1	980.2	3600.2	2493.3	3600.2
31	0.5	1.2	0.4	0.90	11.6	11.4	14.9	12.5	16.5	15.1	12.6	18.4
32	0.5	1.2	0.4	0.95	9.9	11.4	12.9	10.7	17.4	14.3	31.9	18.4