MODELLING BATCH PROCESSING MACHINES PROBLEMS WITH SYMMETRY BREAKING AND ARC FLOW FORMULATION

Renan Spencer Trindade

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Marcia Helena Costa Fampa
Olinto César Bassi de Araújo

Rio de Janeiro
Maio de 2019

MODELLING BATCH PROCESSING MACHINES PROBLEMS WITH SYMMETRY BREAKING AND ARC FLOW FORMULATION

Renan Spencer Trindade

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Marcia Fampa

Prof. Marcia Helena Costa Fampa, D.Sc.

[illegible]

Prof. Olinto César Bassi de Araújo, D.Sc.

[illegible]

Prof. Luidi Gelabert Simonetti, D.Sc.

[illegible]

Prof. Yuri Abitbol de Menezes Frota, D.Sc.

[illegible]

Prof. Claudia D'Ambrosio, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 2019

Trindade, Renan Spencer

Modelling batch processing machines problems with symmetry breaking and arc flow formulation/Renan Spencer Trindade. – Rio de Janeiro: UFRJ/COPPE, 2019.

XIII, 79 p.: il.; 29, 7cm.

Orientadores: Marcia Helena Costa Fampa
Olinto César Bassi de Araújo

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 74 – 79.

1. Batch processing machine. 2. Scheduling. 3. Makespan. 4. Symmetry. 5. Arc flow. I. Fampa, Marcia Helena Costa *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicada aos meus pais, Elisabeth e Isac.
Muito obrigado por todo o apoio e por estarem sempre ao meu lado.
Amo muito vocês.

Agradecimentos

Estou extremamente feliz e finalmente aliviado após todos estes anos que se passaram. Nem preciso dizer o quão difícil foi tudo isto, mas quero agradecer a todos que fizeram com que esta jornada fosse possível. Minha vida mudou muito desde que comecei o doutorado, passei por várias situações difíceis, mas sou muito feliz e agradecido pela pessoa que me tornei. Alguns anos atrás eu não imaginava que eu pudesse concretizar tantos sonhos e tudo isso só foi possível por causa da minha família, meus professores e dos meus amigos.

Eu gostaria de agradecer aos meus pais, Elisabeth e Isac, por toda a paciência, apoio e amor que recebi nestes últimos anos. Nada disso seria possível sem vocês ao meu lado. Aos meus irmãos, Marcele, Fábian e Nathane e aos meus cunhados, Ricardo e Rochane, que sempre tiveram o cuidado de dedicar a atenção para uma conversa, um conselho ou para compartilharmos nossas ideias e planos. Obrigado também por trazerem ao mundo meus sobrinhos que tanto amo, Guilherme e Júlia. Muito obrigado aos primos Josiane Spencer, Paulo Pioner e Paula Spencer, pelos conselhos, conversas e por terem me recebido tão bem na cidade maravilhosa. Muito obrigado também ao meu avô, Arnaldo Trindade (*in memoriam*), que nos deixou durante esta jornada.

Aos meus amigos de longa data, Jhonatan, Mário, Bruno Vendrusculo, Simone, Bruno Nascimento, Fernando Guilherme, que mesmo distantes, sempre estiveram ao meu lado para me acalmar e me ajudar nas horas difíceis. À minha namorada, Gabriela, gostaria de agradecer imensamente por tudo. Passamos por momentos que nunca imaginaríamos em tão pouco tempo. Nosso chão se abriu tão de repente, mas vem se reconstruindo aos poucos, como deve ser. Ao seu pai, Ricardo Ribeiro (*in memoriam*), gostaria de dizer que você faz falta. Agradeço a Raquel Ribeiro e a Daniela, minha segunda família, por me aceitarem aqui com vocês.

Aos meus amigos que tive a imensa alegria de conhecer nesta jornada, Luiz Schirmer, Letícia Fausto, Guilherme Schardong, Joseany Almeida, Felipe Netto, Thaís Lópes e Fabrício, obrigado por dividirem comigo um pouco do dia-a-dia e de aceitarem um “desconhecido” na casa de vocês. É impossível não sentir saudade do convívio, das jogatinas, dos bolos, das experiências culinárias, do cafezinho, das bebidinhas, dos aniversários com charadas, do aquecedor estragado e da parede quebrada. Obrigado pela amizade de vocês!

Aos mamatas do PESC, um agradecimentos especial, Daniela, Evandro, Hugo Barbalho e Cinthya, Dani, Rebeca, Renan, Ana Flavia, Israel, Brunno, Natália, Pedro e Sávio. Muito obrigado por transformarem esta jornada difícil e extenuante em um momento descontraído. Eu não teria chegado até o final se não tivesse o apoio e a alegria de vocês.

Tenho muito orgulho de ter em minha bagagem a passagem pelas instituições UFSM e UFRJ, que me abriram as portas para a pesquisa científica e a pós-graduação. Conheci o professor Olinto ainda na minha graduação, completando mais de nove anos de trabalho e amizade. Logo após, conheci a professora Marcia Fampa, que aceitou as nossas ideias iniciais e trabalhou neste projeto nos últimos cinco anos. Foi uma jornada de dedicação, de novas ideias, de validação, de testes, de melhorias, de mais ideias, de resultados, de escrever artigos, de apresentar em congressos, da dissertação e da tese. Contudo, o meu maior agradecimento a eles é pela amizade e pelo amor à ciência, que eu aprendi a sentir quando os conheci.

Agradeço a Claudia D'Ambrosio e a École Polytechnique, pela oportunidade de fazer um intercâmbio e morar na França por cinco meses.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), 142205/2014-1, pelo suporte financeiro.

Obrigado a todos que estiveram comigo me dando aquela força, tanto presencialmente quanto em pensamento!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MODELAGEM DE PROBLEMAS DE MÁQUINAS DE PROCESSAMENTO EM LOTE UTILIZANDO FORMULAÇÕES COM QUEBRA DE SIMETRIAS E FLUXO EM ARCO

Renan Spencer Trindade

Maio/2019

Orientadores: Marcia Helena Costa Fampa
Olinto César Bassi de Araújo

Programa: Engenharia de Sistemas e Computação

Problemas de minimização do makespan no escalonamento de bateladas em máquinas de processamento são extensamente explorados pela literatura acadêmica, motivados principalmente por testes *burn-in* na indústria de semicondutores. Os problemas considerados neste trabalho consistem em agrupar tarefas em bateladas e escalonar o processamento em uma ou mais máquinas em paralelo. As tarefas possuem tempos de processamento e tamanhos não idênticos e o tamanho total da batelada não pode exceder a capacidade da máquina. Para cada batelada é definido um tempo de processamento que será igual ao maior tempo de processamento das tarefas que foram alocadas a ela. O problema pode considerar também tempos de liberação das tarefas não idênticos e, neste caso, as bateladas só poderão ser processadas depois que a tarefa com o maior tempo de liberação for disponibilizada. Este trabalho aborda quatro diferentes problemas de escalonamento de bateladas, que consideram diferentes características: máquina de processamento única $1|s_j, B|C_{max}$, máquinas de processamento paralelas idênticas $P_m|s_j, B|C_{max}$, máquina de processamento única e tarefas com tempos de liberação não idênticos $1|r_j, s_j, B|C_{max}$, máquinas de processamento paralelas idênticas e tarefas com tempos de liberação não idênticos $P_m|r_j, s_j, B|C_{max}$. São propostos novos modelos matemáticos com formulações que exploram o tratamento de simetria para estes problemas. Além disso, é apresentado um modelo baseado em fluxo em arco para os problemas $1|s_j, B|C_{max}$ e $P_m|s_j, B|C_{max}$. Os modelos matemáticos são resolvidos utilizando CPLEX e os resultados computacionais comprovam que os modelos propostos possuem um desempenho melhor do que outros modelos da literatura.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

MODELLING BATCH PROCESSING MACHINES PROBLEMS WITH SYMMETRY BREAKING AND ARC FLOW FORMULATION

Renan Spencer Trindade

May/2019

Advisors: Marcia Helena Costa Fampa
Olinto César Bassi de Araújo

Department: Systems Engineering and Computer Science

Problems of minimizing makespan in scheduling batch processing machines are widely exploited by academic literature, mainly motivated by burn-in tests in the semiconductor industry. The problems addressed in this work consist of grouping jobs in batches and scheduling this in parallel machines. The jobs have non-identical size and processing times. The total size of the batch cannot exceed the capacity of the machine. The processing time of each batch will be equal to the higher processing time of all the jobs assigned to it. Jobs can also consider non-identical release times; in this case, the batch can only be processed after the job with the longest release time is available. This thesis discusses four different batch scheduling problems, which consider different characteristics: single processing machine $1|s_j, B|C_{max}$, parallel processing machines $P_m|s_j, B|C_{max}$, single processing machine and non-identical release times $1|r_j, s_j, B|C_{max}$, parallel processing machines and non-identical release times $P_m|r_j, s_j, B|C_{max}$. New mathematical formulations are proposed exploiting the treatment of symmetry for these problems. In addition, an arc-flow-based model is presented for problems $1|s_j, B|C_{max}$ and $P_m|s_j, B|C_{max}$. The mathematical models are solved using CPLEX, and computational results show that the proposed models have a better performance than other models in the literature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the competitive environment of the manufacturing and service industries, companies need to be efficient in meeting deadlines with customers and reducing unnecessary operating costs. For this, sequencing and scheduling play a key role for companies to perform well in the marketplace. Scheduling is a widely used decision-making process in resource allocation and allows optimization in most production systems, information processing, transport, and distribution configurations, and other several real-world environments. To date, thousands of these problems have been modeled and studied, and machine scheduling configures an essential class of problems, where a collection of jobs require processing in a given environment.

This thesis focuses on scheduling problems in Batch Processing Machines (BPM), that have been extensively explored in the literature, motivated by a large number of applications in industries and also by the challenging solution of real world problems. The main goal in these problems is to group jobs in batches and process them simultaneously in a machine, to facilitate the tasks and to reduce the time spent in handling the material. Although there are many variations of the problem involving BPM, the versions addressed in this work are more suitable to model the scheduling problems that arise in reliability tests in the semiconductor industry, in operations called burn-in, presented in Uzsoy [2].

The burn-in operation is used to test electronic circuits and consists of designating them to industrial ovens, submitting them to thermal stress for a long period. The test of each circuit is considered here as a job and requires a minimum time inside the oven, which is referred to as the processing machine. The jobs cannot be processed directly on the machine, they need to be placed on a tray, respecting the capacity of the machine. Each group of jobs assigned to a tray is considered a batch. The minimum time of the circuit inside the oven is set a priori, based on the supplier requirements. It is possible to keep the circuit in the oven longer than necessary, with no prejudice, but it cannot be removed before its required processing time is fulfilled. Therefore, the processing time of a batch is determined by the longest processing time among all jobs assigned to it.

The burn-in tests are a bottleneck in final testing operations, and the efficient scheduling of these operations aims to maximize productivity and reduce flow time in the stock, which is a major concern for management. The processing time to test an electronic circuit can reach up to 120 hours in a constant temperature around 120°C, as presented in Lee *et al.* [3]. Tests reported in Tai [4] and Chung *et al.* [5], a liquid crystal display usually take 6 hours to complete the reliability test, which reinforces the importance of an efficient scheduling.

1.1 Problems description

For the BPM problems addressed in this paper, we are given a set J of jobs and a set K of batches. Each job $j \in J$ has a size s_j, a minimum processing time p_j, and must be assigned to a batch $k \in K$. The sum of the sizes of the jobs assigned to a batch cannot exceed the capacity limit B of the processing machine. The time P_k required to process a batch k is equal to the longest processing time of all jobs in the batch. In Figure 1.1 a graphical representation of a batch with three jobs, where the axes represent the time and the machine size, is presented.

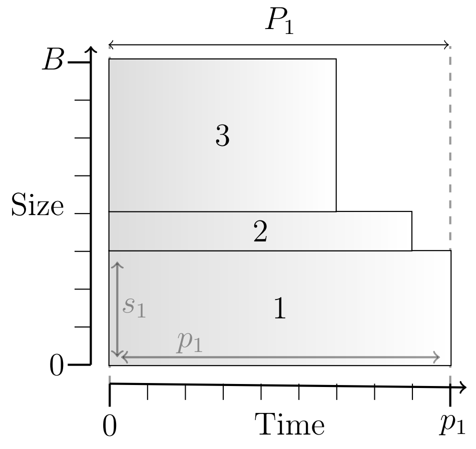


Figure 1.1: Graphical representation of a batch.

The batches must be scheduled on a machine, which can process only one batch at a time. For the case of parallel machines, the machines may be identical. Preemption is not allowed, which means that you can not interrupt the operation after it has started. The goal is to design and schedule the batches on the machines so that the completion time of the last job processed, called makespan and denoted by $C_{\max}$, is minimized. Finally, it is also possible that each job $j \in J$ has a different release time r_j, which is when the job becomes available to be processed. In this case, a batch can only be processed when all the jobs assigned to it are available.

We address four different versions of this BPM scheduling problem. On the two first, denoted by $1|s_j, B|C_{\max}$ and $P_m|s_j, B|C_{\max}$, we do not consider different release times for the jobs. On problem $1|s_j, B|C_{\max}$, only a single processing machine is used, and on $P_m|s_j, B|C_{\max}$, m parallel identical machines are used. The two other problems, denoted by $1|r_j, s_j, B|C_{\max}$ and $P_m|r_j, s_j, B|C_{\max}$, are defined equivalently to the two first ones, but with non-identical release times considered for the jobs.

1.2 Objective and Methodology

In this thesis, the approach address to the problems is twofold:

First, we present Mixed Integer Linear Programming (MILP) formulations for the four problems addressed, which were proposed in Melouk *et al.* [6], Chang *et al.* [7], Xu *et al.* [8] and Vélez Gallego [9]. We point out that all the four formulations present a large number of symmetric solutions in their feasible sets. Two types of symmetries are considered in our analysis. On the first one, two solutions are said to be symmetric if the designs of the batches are equal on both solutions and the batches assigned to a machine are processed in the same order. On the second, two solutions are said to be symmetric if the designs of the batches are still equal in both solutions, but the batches assigned to at least one machine are processed in a different order. Nevertheless, the modification in the order in which the batches are processed does not affect the makespan, and therefore also generates equivalent solutions. The existence of symmetric solutions in the feasible set of the problems leads to a very inefficient application of Branch-and-Bound (B&B) algorithms. Treatment of symmetry in integer programs is an intense area of research, where different strategies are suggested to mitigate the effect of symmetric solutions during the B&B execution (see, for example, Margot [10]). We present different strategies to deal with the symmetry observed on the MILP formulations from the literature.

On the second, we present two MILP formulations with an arc-flow approach for the problems that do not consider different release times for the jobs. This strategy has been applied to other problems, such as the cutting-stock problem in Valério de Carvalho [11], and the bin-packing problem in Valério de Carvalho [12] and Brandão and Pedroso [13]. The idea is to formulate $1|s_j, B|C_{\max}$ and $P_m|s_j, B|C_{\max}$ as problems of determining flows in graphs. In this approach, each physical space of the batch with capacity B is represented by a node, i.e., $V = \{0, \ldots, B\}$ and each arc (i, j) of the subset A^J represents the existence of at least one job k of size s_k, such that $s_k = j - i$. An important feature of the new model is that the number of variables does not change when the number of jobs increases. The respective formulations are compared with the solver `CPLEX` [14]. Table 1.1 shows the two different approaches to the four problems developed for this research.

	$1\|s_j, B\|C_{\max}$	$P_m\|s_j, B\|C_{\max}$	$1\|r_j, s_j, B\|C_{\max}$	$P_m\|r_j, s_j, B\|C_{\max}$
Symmetry breaking approach	MILP$_1^+$, pg. 16	MILP$_2^+$, pg. 44	MILP$_3^+$, pg. 65	MILP$_4^+$, pg. 69
Arc-flow approach	FLOW, pg. 24	FLOW$_2$, pg. 45		

Table 1.1: Methodology overview.

1.3 Thesis Statement and Contributions

To date, most of the work done for the problems addressed in this thesis uses heuristic and meta-heuristic approaches. The great contribution of this thesis is to present MILP formulations to find optimal solutions in a good computational time for the instances suggested in the literature. In this work, several sets of symmetric solutions for the problem addressed are pointed out, as well as symmetry breaking constraints to deal with them. These constraints motivated the study of symmetry breaking for the scheduling problems addressed in this paper. Furthermore, we take into account specific properties of the problems and their optimal solutions to propose new stronger formulations for them and avoid undesirable symmetric solutions in their feasible sets. We show the correctness of our models and explain how our different indexing choices for each problem allow a more efficient modeling. Our approach is currently the most recent work published in the literature in Trindade *et al.* [15]. We also present Arc-flow formulations that present even better computational results for two problems addressed in this thesis. We consider some properties to made possible a complete reformulation of the problems that do not involve times of releases. Thus, it is possible to increase the number of jobs of the instances without increasing the number of variables. It is only necessary to change the upper bound of the existing variables to representing new jobs in the Arc-flow graph. We can prove the optimality of instances with a large number of jobs, never considered before in the literature.

1.4 Outline

This thesis is organized as follows: In Chapter 2, we give an overview of the main results that we found in the literature, on the problems addressed and the similar approaches used.

In Chapter 3, we consider the $1|s_j, B|C_{\max}$ problem and present the MILP formulation from the literature, analyze symmetric solutions to these formulations and propose symmetry breaking constraints and a new formulation for the problem. We also present an Arc-flow formulation for this problem.

In Chapter 4, we consider the $P_m|s_j, B|C_{\max}$ problem, witch consider parallel machines. We also present the MILP formulation from the literature and propose symmetry breaking constraints and a new formulation for the problem. The Arc-flow approach is also applied.

In Chapter 5, we consider the $1|r_j, s_j, B|C_{\max}$ problem, witch consider jobs with different release times. We present the MILP formulations from the literature and we point out the differences for the symmetry breaking approach.

In Chapter 6, we consider the $P_m|r_j, s_j, B|C_{\max}$ problem, witch consider parallel machines and release times. We also applied symmetry breaking approach with the differences pointed out in Chapter 5 and present the new MILP formulation.

The computational results comparing the respective formulations are presented at the end of Chapters 3–6. In Chapter 7, we present some concluding remarks.

Chapter 2

Literature Review

In this chapter, we present a review of papers found in the scientific literature that address each of the problems discussed in this research, as well as the approaches adopted in our methodology. Since the 1950s, many works have been performed related to deterministic scheduling machine theory, opening up problems that are still under study today. To classify these numerous scheduling problems, the $\alpha|\beta|\gamma$ notation is commonly used, suggested by Graham *et al.* [16]. We use this notation, which describes the particular characteristic of each problem addressed, the α field describes the machine environment, the β field represents different processing characteristics and constraints, and the γ field describes the objective to be minimized. An extensive study of scheduling theories and applications appear in Pinedo [17].

In this work, four scheduling problems are treated in batch processing machines. All consider machine(s) with limited capacity B, and jobs with non-identical sizes s_j, in order to minimize the makespan C_{max}. We consider also the processing time (p_j) but this reference is omitted in $\alpha|\beta|\gamma$ notation. The distinction between problems is whether or not to consider non-identical release times r_j or to consider single machines (1) or parallel machines (P_m). The problems are listed below:

$1|s_j, B|C_{max}$: Minimizing makespan for single batch processing machine with non-identical job sizes.

$P_m|s_j, B|C_{max}$: Minimizing makespan for parallel batch processing machines with non-identical job sizes.

$1|r_j, s_j, B|C_{max}$: Minimizing makespan for single batch processing machine with release times and non-identical job sizes.

$P_m|r_j, s_j, B|C_{max}$: Minimizing makespan for parallel batch processing machines with release times and non-identical job sizes.

2.1 Batch Processing Machines problem

The research on BPM is recent, compared to the history of the semiconductor manufacturing, and consists of grouping the jobs into batches. These batches can be processed either in serial (named s-batching) or in parallel (named p-batching). In our problems, we focus only on p-batching problems. The paper Potts and Kovalyov [18] reviews research on scheduling models considering batch processing machines and point out that the research effort in designing algorithms for these problems is worthwhile. A survey related to BPM problems research founded in Mathirajan and Sivakumar [19], analyzing publications between 1986 and 2004 (part of 2004 only). Another survey that focus on BPM problems is published in Mönch *et al.* [20] and reveals that p-batching is much more important in semiconductor manufacturing comparing with s-batching.

2.1.1 A review of $1|s_j, B|C_{max}$ problem

Problem $1|s_j, B|C_{\max}$ was addressed for the first time in Uzsoy [2], where this NP-hard complexity is proved and a heuristic approach to solve it is proposed. Heuristics are also proposed for this problem in Ghazvini and Dupont [21], where instances with up to 100 jobs are considered. Two approximation algorithms are presented in [22] with approximation ratios of 3/2 and 7/4 of the optimal solution, in the worst case. In Melouk *et al.* [6], the simulated annealing meta-heuristic was applied to $1|s_j, B|C_{\max}$ and an MILP formulation was presented for the problem. This work also proposes configurations for test instances that were widely used in later works. Computational results are shown for instances with up to 100 jobs, comparing the heuristic solutions to the solutions obtained with the MILP formulation. Other meta-heuristics are also applied to problem $1|s_j, B|C_{\max}$ in the literature, namely, genetic algorithm (Damodaran *et al.* [23] and Kashan *et al.* [24]), tabu search (Meng and Tang [25]), and GRASP (Damodaran *et al.* [26]). These four papers consider instances with up to 100 jobs as well. In addition, the bee colony meta-heuristic is also applied to the problem in Al-Salamah [27], where results for instances with up to 200 jobs are shown. In Chen *et al.* [1], a heuristic based on a special case of the clustering problem is proposed, and test instances with up to 500 jobs are considered. In Li *et al.* [28], an enumeration scheme for heuristics is proposed. The work uses First Fit Longest Processing Time (FFLPT) and Best Fit Longest Processing Time (BFLPT) considering identical job sizes, and instances with up to 500 jobs are considered. In Lee and Lee [29], two heuristics are proposed based on a decomposition of the original problem, where relaxations of the problem are solved. Instances with up to 100 jobs are considered in this work. An exact approach is used in Rafiee Parsa *et al.* [30], where a formulation for problem $1|s_j, B|C_{\max}$ is presented, using a partition problem in the context of Dantzig-Wolfe decomposition. In this work, the branch-and-price method is applied to solve the problem to optimality. Instances with up to 500 jobs are considered in the computational experiments.

2.1.2 A review of $P_m|s_j, B|C_{max}$ problem

Concerning problem $P_m|s_j, B|C_{\max}$, the papers that address it, are mostly extensions of the works published for problem $1|s_j, B|C_{\max}$. In Chang *et al.* [7], the simulated annealing meta-heuristic is applied, and an MILP formulation is presented for the problem. This work also proves the NP-hard complexity of the problem, and show results for instances with up to 50 jobs. In Kashan *et al.* [31], a hybrid genetic algorithm is used to compute solutions for instances with up to 100 jobs, considering 2 and 4 parallel machines. In Damodaran *et al.* [32] a new application of the genetic algorithm is proposed, which solves instances with up to 100 tasks, also on 2 and 4 parallel machines. In Cheng *et al.* [33] an approximation algorithm is presented for the problem, with the approximation factor of 2. Finally, two other works that apply meta-heuristics (Cheng *et al.* [34] and Jia and Leung [35]), use the ant colony method and a meta-heuristic based on a max-min ant system for this problem. In Cheng *et al.* [34], results for instances with up to 500 jobs on 4 and 8 parallel machines are shown, whereas, in Jia and Leung [35], instances are solved with up to 100 jobs, on 2, 3, and 4 parallel machines.

2.1.3 A review of $1|r_j, s_j, B|C_{max}$ problem

There are only a few papers investigating problem $1|r_j, s_j, B|C_{\max}$ in the literature. To our knowledge, only three papers have been published. Solution approaches based on meta-heuristics are presented in Chou *et al.* [36] and Xu *et al.* [8], which apply hybrid genetic algorithms and the ant colony meta-heuristic, respectively. In both works, test instances with up to 100 jobs are considered. In Zhou *et al.* [37], three heuristics are proposed for this problem and computational results for instances with up to 300 jobs are shown. In Xu *et al.* [8], we can also find an MILP formulation for the problem.

2.1.4 A review of $P_m|r_j, s_j, B|C_{max}$ problem

Problem $P_m|r_j, s_j, B|C_{\max}$ was firstly addressed in Chung *et al.* [5]. In this paper, the authors propose an MILP formulation to solve the problem to optimality and three heuristics to handle instances with 7 and 15 jobs. In Vélez Gallego [9], the problem is proved to be NP-hard. The authors addressed the problem with the use of an MILP formulation and a column generation method. Also, five heuristics and two meta-heuristics are tested, including simulated annealing, and GRASP. Numerical experiments with instances with up to 50 jobs, and with 3 and 5 parallel machines, are presented. The results of this work are also published in Damodaran *et al.* [38].

2.2 Symmetry breaking

An integer linear program is symmetric if its variables can be permuted without changing the problem structure and all these solutions have the same value in the objective function. The B&B method becomes very inefficient when applied to formulations containing a large set of symmetric solutions because it solve several subproblems unnecessarily. In this case, the B&B is not efficient to prune branches of the enumeration tree with symmetric global optimum, since the the upper bound is not tight enough.

All formulations found in the literature, for the problems dealt with in this work, fit in this case. The difficulties related to symmetries in integer programming are depicted in Margot [10], where several strategies are cited for their treatment. The impact of symmetry breaking constraints on a particular software engineering application is also investigated in Köhler *et al.* [39]. In this work, several sets of symmetric solutions for the problem addressed are pointed out, as well as symmetry breaking constraints to deal with them. These constraints motivated the study of symmetry breaking for the scheduling problems addressed in this paper.

2.3 Arc Flow formulation

The arc flow approach has been used recently in classical optimization problems and allows modeling with a pseudo-polynomial number of variables and constraints. For a cutting-stock problem, Valério de Carvalho [11] proposes a branch-and-price approach for an arc-flow formulation. Next, it was extended for the bin-packing problem in Valério de Carvalho [12]. An alternative arc-flow formulation for the cutting-stock problem is proposed in Brandão and Pedroso [13] and Brandão [40], which uses a graph compression technique. These formulations were recently tested and compared in Delorme *et al.* [41] against several other models and problem-specific algorithms on one-dimensional bin packing and cutting stock problems. The results show that the arc-flow formulation outperforms all other models. In Martinovic *et al.* [42] the arc-flow model and the one-cut model are compared for the one-dimensional cutting-stock problem, and reduction techniques for both approaches are presented.

For the scheduling area, we are aware only two works that consider the arc-flow approach. In Kramer *et al.* [43] the problem of scheduling a set of jobs on a set of identical parallel machines, with the aim of minimizing the total weighted completion time, $P||\sum W_jC_j$ is considered. In Mrad and Souayah [44] the makespan minimization problem on identical parallel machines, $P||C_{max}$ is considered. It is important to note that these works do not consider more complex features in scheduling problems, such as batching machines, non-identical job sizes, and machine capacity.

2.4 Discussion

We can conclude from our literature review on the problems addressed in this work, that most of the effort made by researchers to solve them, concentrated in heuristic procedures. The MILP formulations presented for them were mostly used as a baseline to give a formal definition of the problems, and provide some evaluation for the heuristic solutions on small instances. As mentioned before, B&B algorithms become very inefficient when applied to these formulations due to the presence of symmetric solutions in their feasible sets. We take into account specific properties of the problems and their optimal solutions to propose new stronger formulations for them. Applying our models we are able to prove optimality of instances with sizes considered for the first time in the literature to be solved by exact methods.

Chapter 3

The $1|s_j, B|C_{max}$ problem

Problem $1|s_j, B|C_{\max}$ is the simplest job scheduling problem addressed in this work. It can be formally defined as follows. Given a set $J := \{1, \ldots, n_J\}$ of jobs, each job $j \in J$ has a processing time p_j and a size s_j. Each of them must be assigned to a batch $k \in K := \{1, \ldots, n_K\}$, not exceeding a given capacity limit B of the processing machine, i.e., the sum of the sizes of the jobs assigned to a single batch cannot exceed B. We assume that $s_j \leq B$, for all $j \in J$. The batches must be all processed in a single machine, one at a time, and all the jobs assigned to a single batch are processed simultaneously. The processing time P_k of each batch $k \in K$ is defined as longest processing time among all jobs assigned to it, i.e., $P_k := \max\{p_j : j \text{ is assigned to } k\}$. Jobs cannot be split between batches. It is also not possible to add or remove jobs from the machine while the batches are being processed. The goal is to design and schedule the batches so that the makespan ($C_{\max}$) is minimized, where the design of a batch is defined as the set of jobs assigned to it, to schedule the batches means to define the ordering in which they are processed in the machine, and the makespan is defined as the time required to finish processing the last batch.

The number of batches used on the solution is not fixed and should be optimized. It will depend on the number of jobs, their sizes, and the machine capacity. In the worst case, the number of batches n_K will be equal to the number of jobs n_J. Although it is possible to find feasible solutions with n_K smaller than n_J, we can not ensure that the model continues to represent the optimal solution. It is necessary to consider the case in which creating an additional batch in the solution can improve the makespan. We can illustrate in the Figure 3.1 an example where n_K in 3.1.(a) is smaller than in 3.1.(b), but the C'_{max} in 3.1.(b) is better than the solution C_{max} in 3.1.(a). Therefore, to assure the correctness of the formulations presented, it is assumed that $n_K = n_J$.

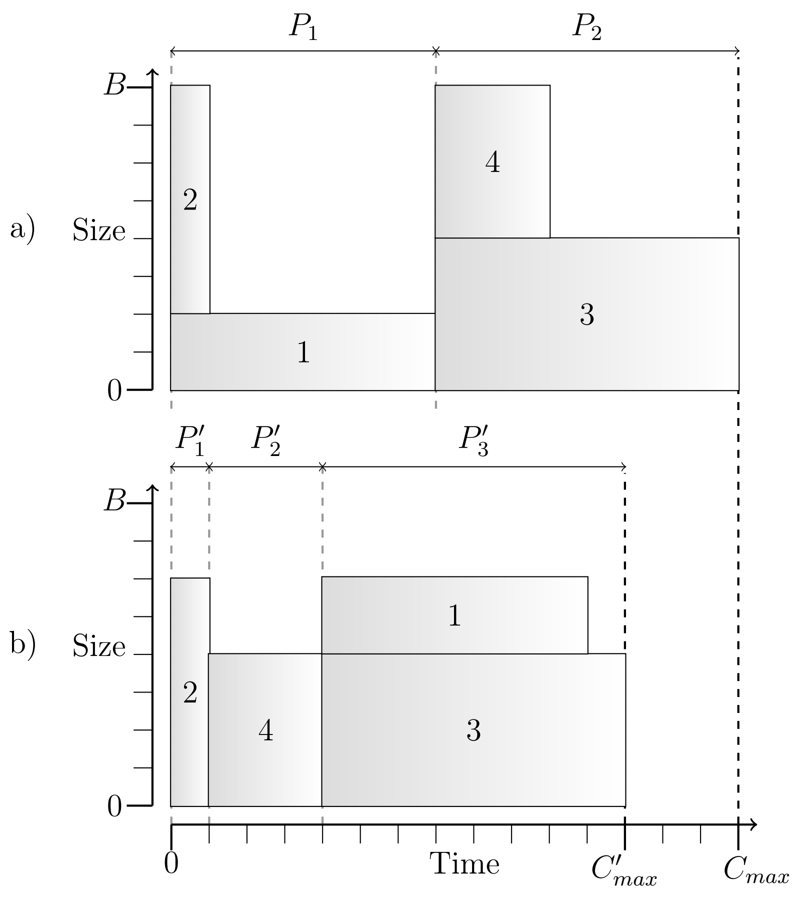


Figure 3.1: Example of solutions with $n_k = 2$ and $n_k = 3$.

3.1 Literature formulation

Let us consider then the following decision variables for all $j \in J$, $k \in K$:

$$x_{jk} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k; \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

$$y_k = \begin{cases} 1, & \text{if batch } k \text{ is used;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

$$P_k : \text{processing time of batch } k. \tag{3.3}$$

In Melouk *et al.* [6] the following MILP formulation is proposed for problem $1|s_j, B|C_{\max}$. Other very similar formulations and sometimes this exactly same one, are used in other papers as a comparative basis in computational experiments.

$$
\begin{aligned}
(\text{MILP}_1) \quad & \min \sum_{k \in K} P_k, && && (3.4)\\
& \sum_{k \in K} x_{jk} = 1, && \forall j \in J, && (3.5)\\
& \sum_{j \in J} s_j x_{jk} \leq B y_k, && \forall k \in K, && (3.6)\\
& P_k \geq p_j x_{jk}, && \forall j \in J, \forall k \in K, && (3.7)\\
& x_{jk} \leq y_k, && \forall j \in J, \forall k \in K, && (3.8)\\
& y_k \in \{0,1\}, && \forall k \in K, && (3.9)\\
& x_{jk} \in \{0,1\}, && \forall j \in J, \forall k \in K. && (3.10)
\end{aligned}
$$

The objective function (3.4) minimizes the makespan, given by the sum of the processing times of all batches. Constraints (3.5) determine that each job is assigned to a single batch. Constraints (3.6) determine that each batch if used does not exceed the capacity of the machine. Constraints (3.7) determine the processing times of the batches. Note that constraints (3.8) are redundant together with (3.6), but are added to strengthen the linear relaxation of the formulation.

3.2 Symmetry breaking approach

The formulation MILP$_1$ presented in the Section 3.1 allows a large number of symmetric solutions in the feasible sets of problems, which makes it difficult to solve using the traditional B&B algorithm. The trouble comes from the fact that many subproblems in the enumeration tree are isomorphic, forcing a wasteful duplication of effort. Our symmetry breaking approach is performed before running the solution algorithm, i.e., as a pre-solve procedure, we call such strategy as a Static Symmetry Breaking (SSB), and this approach is performed twofold: First, in the symmetry analysis, we distinguish two types of symmetry that may occur in this formulation. Second, we propose symmetry breaking constraints and a new reformulation to the problem.

3.2.1 Symmetry analysis

The first symmetry analyzed occurs when changing the processing order of the batches assigned to a machine does not affect the makespan. In this case, the processing time of a machine is given by the sum of the processing times of the batches assigned to it, the machine is never idle during their processing, and any permutation in the order in which those batches are processed lead to equivalent solutions, with the same makespan. This situation is exemplified in Figure 3.2, which depicts two equivalent solutions, 3.2(a) and 3.2(b), for an instance of the problem with 10 jobs. On both solutions all batches are equally designed, having the same jobs assigned to them. The makespan is consequently also the same for both solutions. The only difference between them is the sequence in which the batches are processed in the machine. Note that any other permutation would lead to another equivalent solution.

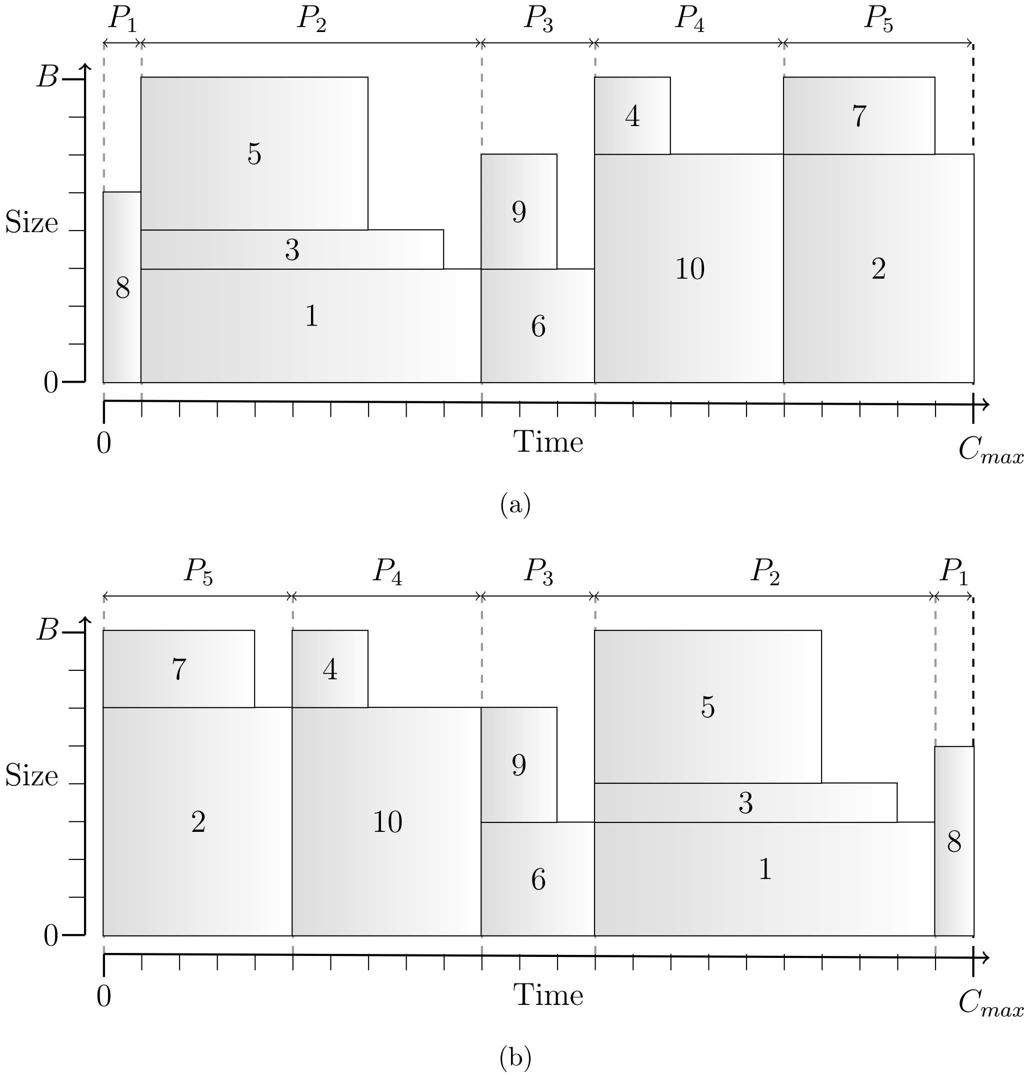


Figure 3.2: Symmetric solutions for problem $1|s_j, B|C_{\max}$.

The second symmetry analyzed occurs when the solutions where the batches are all equally designed and processed in the machine in the same order, may be represented as different solutions and coexist in the feasible set. This happens whenever the number of batches actually used or processed in a given machine is smaller than the number of batches available n_K. Clearly, this allows the solutions to be represented by different indexing of the batches, as illustrated in Figure 3.3, where three symmetric solutions for a problem with six jobs, equally grouped in

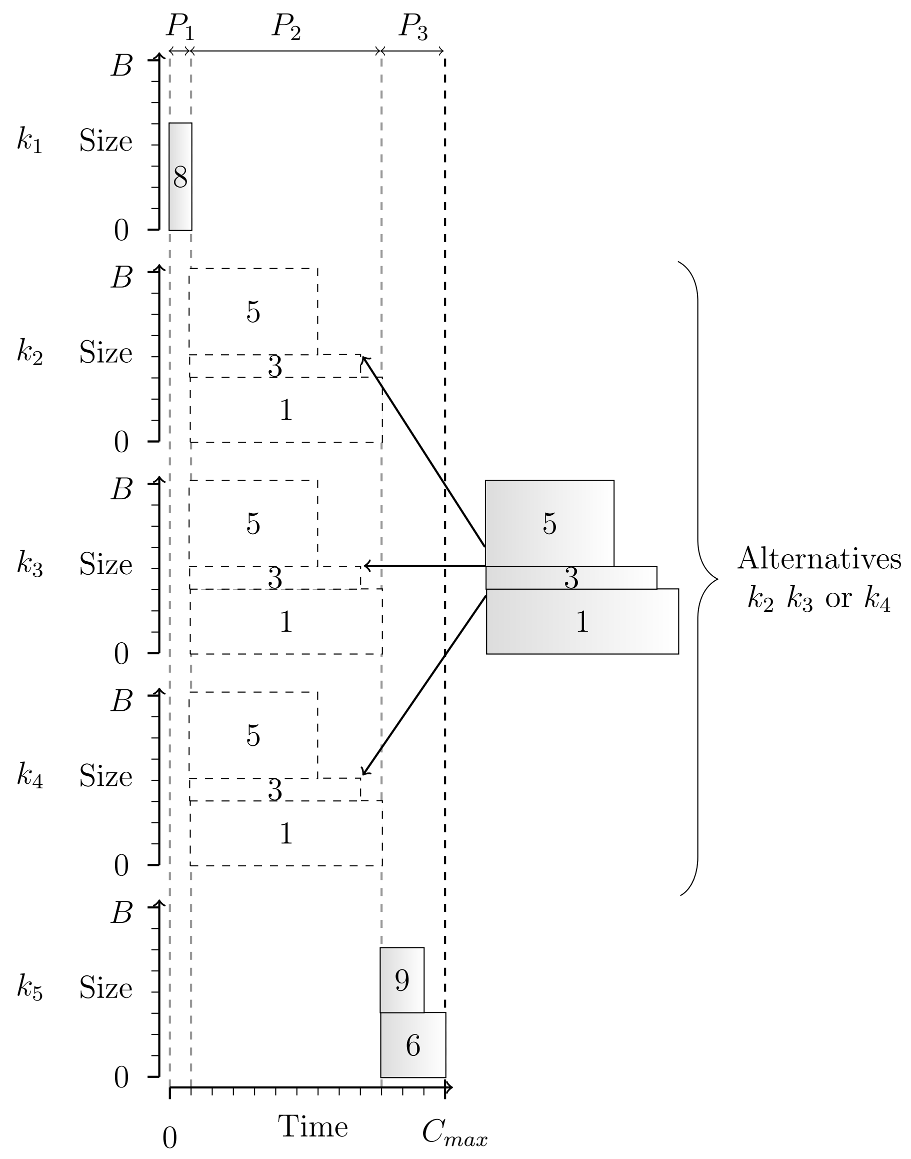


Figure 3.3: Symmetric solutions in which batches are represented by different indexes.

three batches, are depicted. Note that the only difference between the solutions is the indexing of the batch with jobs 1,3, and 5. Its index could be k_2, k_3, or k_4. The formulation presented allows this type of symmetry.

3.2.2 Symmetry breaking formulation

We deal with the symmetry of problem $1|s_j, B|C_{\max}$ with a threefold procedure. Firstly, we set the indexes of the jobs, ordering them by their processing time. More specifically, we consider:

$$p_1 \leq p_2 \leq \ldots \leq p_{n_J}. \tag{3.11}$$

Secondly, we set $n_K := n_J$ and determine that batch k can only be used if job k is assigned to it, for all $k \in K$. Thirdly, we determine that job j can only be assigned to batch k if $j \leq k$.

Note that with the assumptions made, we may only define variables x_{jk} in (3.1), for $j \leq k$, reducing the number of binary variables from n_J^2 to $n_J(n_J+1)/2$. More importantly, the assumptions lead to solutions where the processing time of batch k, if used, is equal to p_k, as job k is certainly assigned to it, and is also the job with longest processing time assigned to the batch. Consequently, there is no need to define the variables P_k (3.3), for all $k \in K$, in order to represent the processing times of the batches, or impose constraints (3.7) to determine them.

Besides reducing the number of variables, when compared to (MILP$_1$), the strategy described leaves only one possible processing ordering for the batches in a given solution, where the batches are ordered by non-decreasing processing time. Furthermore, there is only one possible way of idexing the batches on a given solution, where the index of the batch is equal to the largest index among the jobs assigned to it. Several equivalent solutions are therefore, eliminated from the feasible set of (MILP$_1$).

Considering the above, we propose next a new formulation for $1|s_j, B|C_{\max}$:

$$\text{(MILP}_1^+\text{)} \quad \min \sum_{k \in K} p_k x_{kk}, \tag{3.12}$$

$$\sum_{k \in K : k \geq j} x_{jk} = 1, \qquad \forall j \in J, \tag{3.13}$$

$$\sum_{j \in J : j \leq k} s_j x_{jk} \leq B x_{kk}, \qquad \forall k \in K, \tag{3.14}$$

$$x_{jk} \leq x_{kk}, \qquad \forall j \in J, \forall k \in K : j \leq k, \tag{3.15}$$

$$x_{jk} \in \{0,1\}, \qquad \forall j \in J, \forall k \in K : j \leq k. \tag{3.16}$$

The objective function (3.12) minimizes the makespan, given by the sum of the processing times of the batches used. Constraints (3.13) determine that each job j is assigned to a single batch k, such that $k \geq j$. Constraints (3.14) determine that the batches do not exceed the capacity of the machine. They also ensure that each batch k is used if and only if job k is assigned to it. Constraints (3.15) are redundant together with (3.14), but are included to strengthen the linear relaxation of the model.

It is straightforward to verify that the minimum makespan of (MILP$_1$) and (MILP$_1^+$) are the same. The following proposition formalizes this result.

Proposition 1. *The optimal makespan of problems (MILP$_1$) and (MILP$_1^+$) are the same.*

Proof. Let us consider w.l.o.g. that the indexes of jobs in J satisfy (3.11). Clearly, any feasible solution of (MILP$_1^+$) is also a feasible solution to (MILP$_1$) with the same objective function value. Therefore, it suffices to show that given any feasible solution to (MILP$_1$), with objective function value $\bar{C}_{\max}$, there is a feasible solution to (MILP$_1^+$) also with objective function value $\bar{C}_{\max}$. For that, let us first reset, if necessary, the indexes of the batches on the given solution of (MILP$_1$), determining them as the largest index among the ones of all jobs assigned to it. The solution is now a feasible to solution to (MILP$_1^+$), with the batches designed as in the given feasible solution to (MILP$_1$), but possibly processed in a different order. As this reordering of the batches does not affect the makespan in this problem, both solutions have the same objective function value. □

Remark. *We also consider a new set of constraints to treat some equivalent solutions remaining in the model (MILP$_1^+$). We have adapted the lexical ordering constraints proposed by Margot [10] to eliminate equivalent solutions in case where there are jobs with the same size and belonging to batches with the same processing time. At first glance, the set of constraints improved the computational times in instances with few jobs. However, in instances with more than 100 jobs, the number of constraints added was very large, and the computational results show that the computational times have increased in this situation.*

3.3 Arc Flow approach

The idea in this section is to formulate problem $1|s_j, B|C_{\max}$ as a problem of determining flows in graphs. With this goal, we initially define a directed graph $G = (V, A)$, in which each physical space of the batch with capacity B is represented by a node, i.e., $V = \{0, \dots, B\}$. The set of directed arcs A is divided into three subsets: the set of *job arcs* A^J, the set of *loss arcs* A^L, and the set with a *feedback arc* A^F. Therefore, $A = A^J \cup A^L \cup A^F$. Each arc (i, j) of the subset A^J represents the existence of at least one job k of size s_k, such that $s_k = j - i$. The subset A^J is more specifically defined as:

$$A^J := \{(i, j) : \exists k \in J, s_k = j - i \wedge i, j \in V \wedge i < j\}. \tag{3.17}$$

To compose valid paths and represent all possible solutions, it is necessary to include the *loss arcs* in G, which represent empty spaces at the end of a batch. The subset of arcs A^L is more specifically defined as:

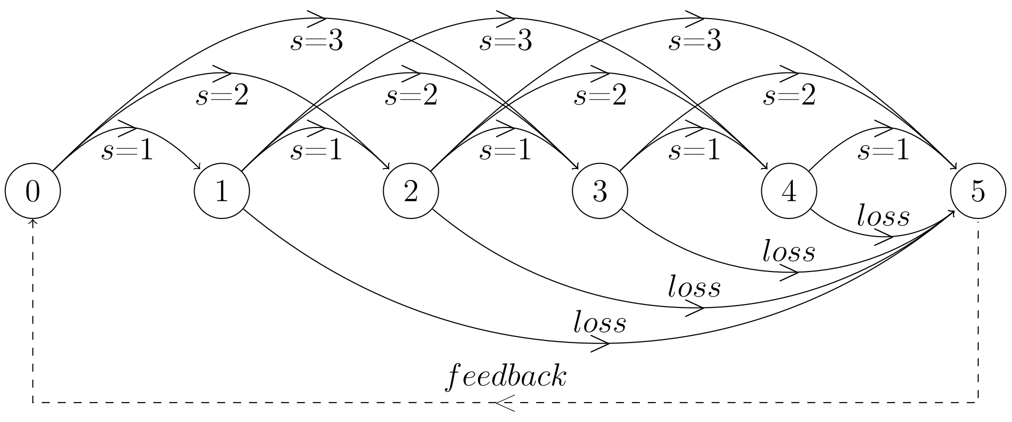
$$A^L := \{(i, B) : i \in V \wedge 0 < i < B\}. \tag{3.18}$$

Finally, the *feedback arc* is used to connect the last node to the first one, defined as:

$$A^F := \{(B, 0)\}. \tag{3.19}$$

Figure 3.4 shows the graph G for a machine with processing capacity $B = 5$, and a set of jobs $\{j_1, j_2, j_3, j_4, j_5\}$ with respective sizes $\{s_1 = 3, s_2 = 2, s_3 = 2, s_4 = 1, s_5 = 1\}$.

Figure 3.4: Example of the graph G representing the arc-flow structure used to model problem $1|s_j, B|C_{\max}$.

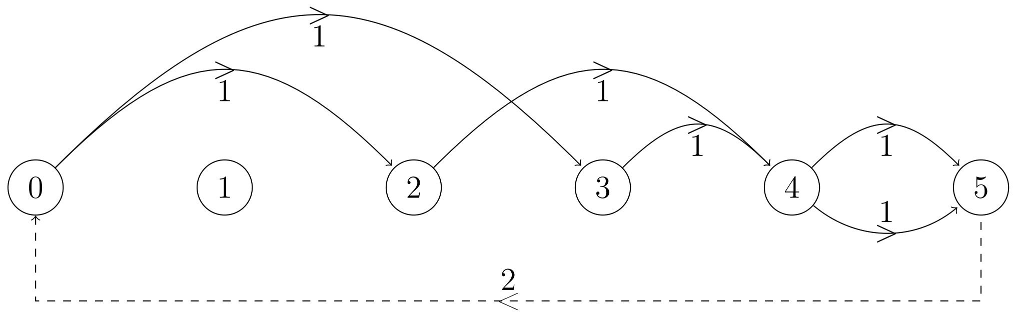


In our modeling approach, each unit flow in graph G, going from node 0 to node B, represents the configuration of a batch in the solution of the problem. Moreover, a unit flow from node B to node 0, going through the *feedback arc*, represents a solution using a single batch, and flows with more than one unit on the *feedback arc*, represent the use of several batches.

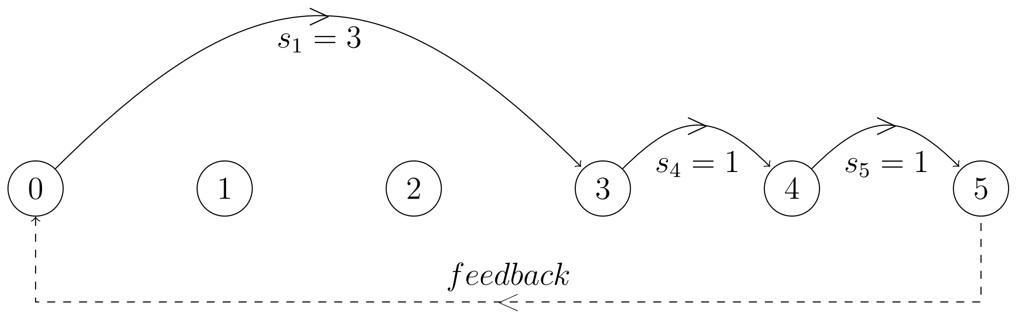
As all flows are non-negative and G is acyclic once the *feedback arc* is excluded, it is possible to decompose the multiple flow on the *feedback arc* into several unit flows, each corresponding to an oriented path from node 0 to node B. Each unit flow represents the configuration of a different batch and the flow on the *feedback arc* indicates the number of batches used in the solution.

Figure 3.5 depicts a solution for the problem represented by the graph G shown in Figure 3.4, and illustrates the decomposition of the flows on the multiple-flow graph of two units, that is shown in Figure 3.5(a). The multiple-flow graph is decomposed into two other graphs, each with a unit flow, in Figures 3.5(b) and 3.5(c). They represent two batches k_1 and k_2, such that jobs $\{j_1, j_4, j_5\}$ are assigned to batch k_1 and jobs $\{j_2, j_3\}$ are assigned to batch k_2. Arcs with null value are not shown in Figure 3.5.

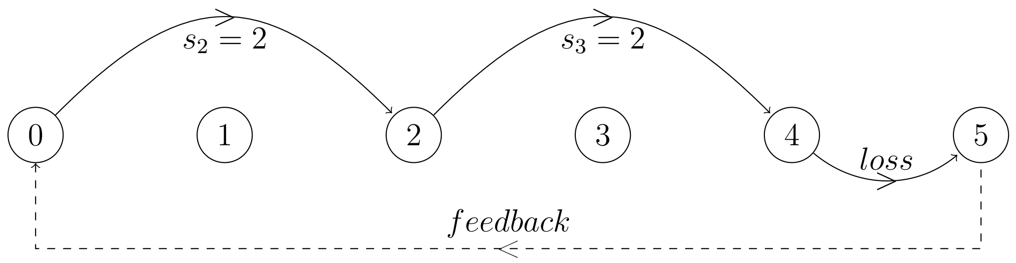
The graph G defined above, is then replicated for each different processing time of the problem in our modeling approach. Each replicated graph will be referred to



(a) Example of a solution for the arc-flow structure in Figure 3.4 with respective arcs' values, decomposed to 3.5(b) and 3.5(c).



(b)



(c)

Figure 3.5: Example of a solution obtained by the arc-flow approach for problem $1|s_j, B|C_{\max}$.

as an arc-flow structure for our problem. We consider $P := \{P_1, \ldots, P_\delta\}$ as the set with all the different processing times among all jobs, and $T := \{1, \ldots, \delta\}$ as the set of indexes corresponding to the arc-flow structures in the problem formulation. The arc-flow structure $t \in T$, will have the corresponding processing time fixed at P_t, which is the t^{th} shortest processing time in P, and only jobs j with processing time $p_j \leq P_t$ will be allowed to be assignment to this arc-flow structure. Moreover, if any job is assigned to this arc-flow structure, then at least one job with processing time P_t will also be assigned. The number of batches corresponding to each arc-flow structure is indicated by the value of the *feedback arc*. As each arc-flow structure can represent as many batches as needed, it is possible to represent all solutions of problem $1|s_j, B|C_{\max}$ with our arc-flow structures. Finally, we note that the maximum number of batches represented by the arc-flow structure t is given by the number of jobs with processing time P_t. This bound will only be attained when each job with processing time P_t is assigned to a different batch.

The Figure 3.7 shows an arc-flow representation of a solution presented in the Figure 3.6. The solution contains only jobs with processing times 3, 4, and 5 which are grouped in six batches, i.e., two with processing time 3, two with processing times 4 and two with processing time 5. In this case, the arc-flow approach needs only three structures to represent this solution, $P_1 = 3$, $P_2 = 4$, and $P_3 = 5$.

We note that the solution represented with our arc-flow structure does not specify exactly which are the jobs assigned to each batch, but only number of jobs with each size that are assigned to it. For example, the flow in Figure 3.7(a) represents two different batches in the solution. The first one has one job with size $s = 2$ and two jobs of size $s = 1$ and the second has two jobs with size $s = 2$. The mapping of arcs into jobs in the arc-flow solution is easily performed in polynomial time, since each *job arc* (i, j) in the arc-flow structure corresponding to processing time P_t, can be mapped to any job with size $j - i$ and processing time not greater that P_t.

A set of variables and constraints is required to ensure that all jobs are assigned to a batch, and that each arc-flow structure considers the correct number of jobs of each size that are available to be assigned to it. Our approach requires the jobs to be sorted by non-decreasing processing time. Let us then consider that the arc-flow structure t corresponding to a processing time P_t, has n jobs of size c still available to be assigned to it, i.e., these are the jobs with size c and processing time smaller than or equal to P_t, that have not already been assigned to an arc-flow structure with corresponding processing time smaller than P_t. A variable $z_{c,t}$ is defined in our model to represent how many of these n jobs of size c were not assigned to the arc-flow structure t. As n is the total number of jobs available and assuming that s is the number of jobs that were assigned to the structure t, the variable $z_{c,t}$ assumes value $n - s$. This variable will offer the remaining $n - s$ jobs to the next arc-flow structure, $t + 1$, corresponding to next processing time of the problem, greater than P_t. In the end, all jobs should be assigned to a structure, and, therefore, we should have $z_{c,\delta} = 0$, where $\delta := |T|$ is the index of the last arc-flow structure corresponding to the longest processing time.

3.3.1 Arc reduction and upper bounds

Some rules can be set to decrease the number of arcs in graph G and the number of variables in the problem. The set A^L of *loss arcs* defined in (3.18), represent empty

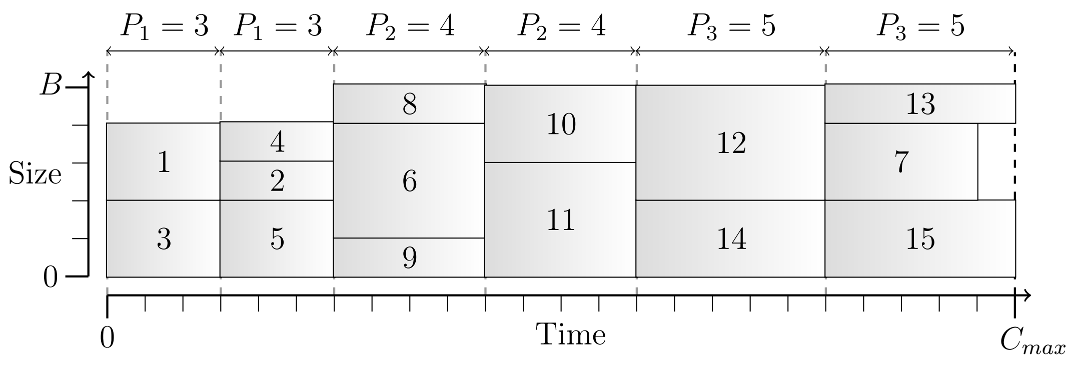
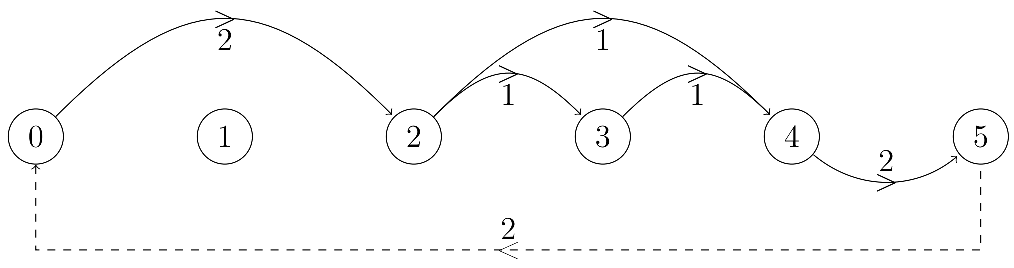
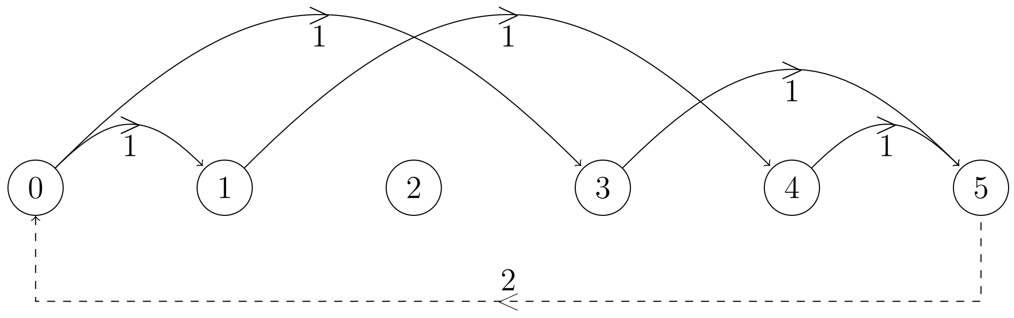


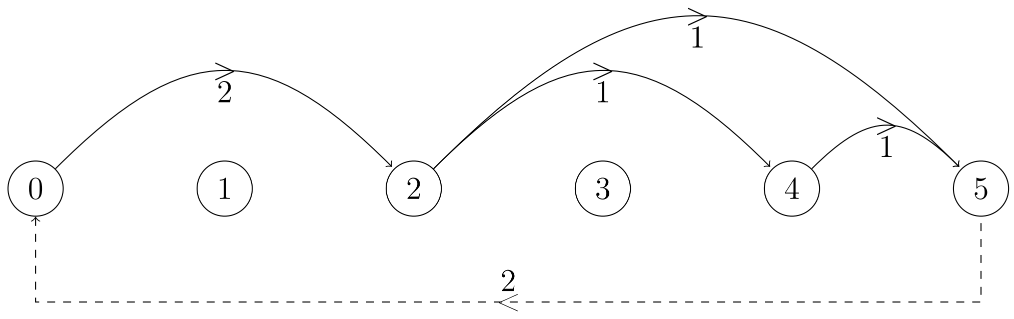
Figure 3.6: Example of solution with $C_{max} = 24$ and six batches.



(a) Arc-flow structure with $P_1 = 3$



(b) Arc-flow structure with $P_2 = 4$



(c) Arc-flow structure with $P_3 = 5$

Figure 3.7: Example of a solution for the arc-flow approach in Figure 3.6.

spaces that occur at the end of the batch. We first note that our modeling does not represent empty spaces between jobs or at the beginning of the batch, which avoids symmetric solutions on the feasible set of the problem. Because of that, two rules can be created to eliminate arcs that will never be used in the solution of the problem: Rule 1 eliminates *job arcs* that cannot be used in a valid flow starting at node 0, and Rule 2 eliminates *loss arcs* incident to nodes that are not incident to any *job arc*.

Rule 1. *Only* job arcs $(i, j) \in A^J$ *that belong to at least one continuous flow starting at node 0 can have a positive flow in the solution of the problem, and therefore, all the others may be eliminated from graph G. The* job arcs *that belong to at least one continuous flow can be selected by the following steps:*

1. *arc $(0, j)$ is selected, for all j, such that $(0, j) \in A^J$;*
2. *arc $(i, j) \in A^J$ is selected, if an arc (k, i) has been previously selected for some node k;*
3. *repeat step (2) until no arc is selected.*

Rule 2. *Only* loss arcs $(i, B) \in A^L$ *that are incident to a node i, which is incident to a remaining* job arc *after the application of Rule 1, can have a positive flow in the solution of the problem, and therefore, all the others may be eliminated from graph G.*

Figure 3.8 illustrates the application of Rules 1–2, where 2 *job arcs* and 1 *loss arc* are removed from graph G.

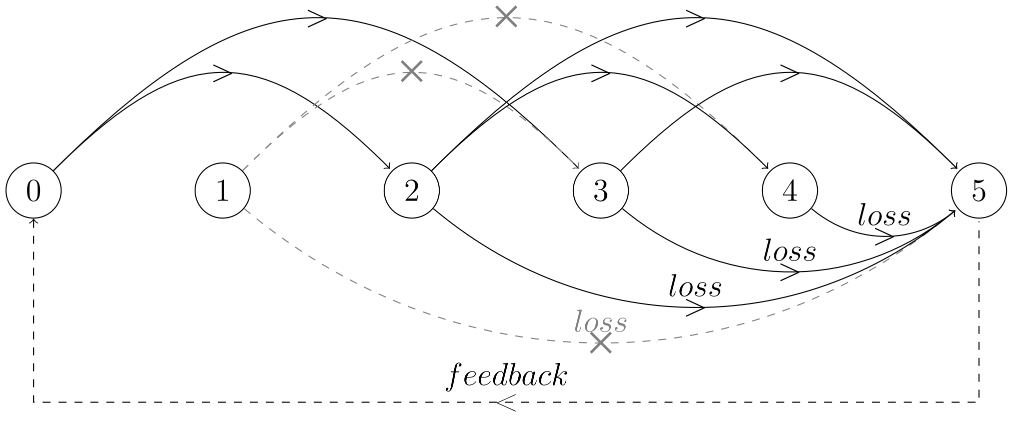


Figure 3.8: Example of arc reduction.

A second group of rules can be defined to bound the values of the flows in each arc of the arc-flow structures, and therefore bound the variables, strengthening our formulation.

Rule 3. *The flow on each arc $(i,j) \in A$ that belongs to the arc-flow structure corresponding to processing time p cannot be greater than the number of jobs with processing time p. This upper bound is more specifically given by*

$$\sum_{k \in J : p_k = p} 1.$$

Rule 4. *The flow on each* job arc $(i,j) \in A^J$ *that belongs to the arc-flow structure corresponding to processing time p cannot be greater than the number of jobs with size $j-i$ and processing time not greater than p. This upper bound is more specifically given by*

$$\sum_{\substack{k \in J : s_k = j-i, \\ p_k \le p}} 1.$$

3.3.2 Arc Flow formulation

Problem $1|s_j, B|C_{\max}$ is formulated as the problem of determining the minimum flow from node 0 to node B, for all the arc-flow structures. Our new formulation is presented below:

Sets and parameters

Max^p : maximum processing time among all jobs, $\max\{p_j, \forall j \in J\}$.

Min^p : minimum processing time among all jobs, $\min\{p_j, \forall j \in J\}$.

P : set with the range of processing times, $\{P_1, \ldots, P_\delta\}$, where $P_1 := Min^p$, $P_\delta := Max^p$, and δ is the number of different processing times among all jobs.

T : set of indexes of the arc-flow structures, $\{1, \ldots, \delta\}$

$NT_{c,t}$: number of jobs with size c and processing time P_t,

$$NT_{c,t} := \sum_{\substack{j \in J : s_j = c, \\ p_j = P_t}} 1;$$

$NT^+_{c,t}$: number of jobs with size c and processing time $\le P_t$.

$$NT^+_{c,t} := \sum_{\substack{j \in J : s_j = c, \\ p_j \le P_t}} 1;$$

NJ_t : number of jobs with processing time P_t,

$$NJ_t := \sum_{\substack{j \in J: \\ p_j = P_t}} 1.$$

Decision variables

$f_{i,j,t}$: flow on *job arc* $(i,j) \in A^J$ in arc-flow structure t. The variable indicates the quantity of batches created with position i occupied by jobs with size $j-i$.

$y_{i,j,t}$: flow on the *loss arc* $(i,B) \in A^L$ in arc-flow structure t.

v_t : flow on the *feedback arc* in arc-flow structure t. The variable indicates the number of batches required with processing time P_t.

$z_{c,t}$: number of jobs with size c, not allocated in the batches with processing time smaller than or equal to P_t. Theses jobs are allowed to be allocated in the batches with processing time P_{t+1}.

$$\text{(FLOW)} \quad \min \sum_{\forall t \in T} P_t.v_t \tag{3.20}$$

$$\left(\sum_{(i,j)\in A^J} f_{i,j,t} + \sum_{(i,j)\in A^L} y_{i,j,t}\right) - \left(\sum_{(j,i)\in A^J} f_{j,i,t} + \sum_{(j,i)\in A^L} y_{j,i,t}\right) = \begin{cases} -v_t & \text{if } j = 0; \\ v_t & \text{if } j = B; \\ 0 & \text{if } 0 < j < B. \end{cases} \qquad t \in T \tag{3.21}$$

$$NT_{c,t} - \sum_{\substack{(i,j)\in A^J: \\ j-i=c}} f_{i,j,t} = \begin{cases} z_{c,t} & \text{if } t = 1; \\ -z_{c,t-1} & \text{if } t = \delta; \\ z_{c,t} - z_{c,t-1} & \text{if } 1 < t < \delta. \end{cases} \qquad c \in \{1..B\} \tag{3.22}$$

$$f_{i,j,t} \le min(NJ_t, NT^+_{j-i,t}),\ f_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^J \tag{3.23}$$

$$v_t \le NJ_t,\ v_t \in \mathbb{Z} \qquad t \in T \tag{3.24}$$

$$y_{i,j,t} \le NJ_t,\ y_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^L \tag{3.25}$$

$$z_{c,t} \le NT^+_{c,t},\ z_{c,t} \in \mathbb{Z} \qquad t \in T : t < \delta, c \in \{1..B\} \tag{3.26}$$

The objective function (3.20) minimizes the makespan. The set of flow conservation constraints are defined by (3.21). Constraints (3.22) ensure that all jobs are assigned and also control the number of jobs to be assigned to each arc-flow structure. Constraints (3.23), (3.24), (3.25) and (3.26) define the domains of the variables and their respective upper bounds, defined by Rules 3–4.

3.4 Computational results

The models presented in this chapter were compared through computational tests performed with two sets of instances. The first set was created by the authors of the paper Chen *et al.* [1], who kindly sent it to us to use in our work. The second set is proposed in this thesis with parameters that make the instances more difficult. In all tests, we use the `CPLEX` version 12.7.1.0, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.70GHz `Intel Quad-Core Xeon E5-2697 v2` processor and 64GB of RAM. The computational time to solve each instance was limited in 1800 seconds.

3.4.1 Instances from the literature

The first set of test instances for problem $1|s_j, B|C_{\max}$ is the same one considered in Chen *et al.* [1]. For each job j, an integer processing time p_j and an integer job size s_j were generated from the respective uniform distribution depicted in Table 3.1. In total, there were generated 4200 instances, 100 for each of the 42 different combinations of number and size of the jobs.

Table 3.1: Parameter settings for set of test instances for $1|s_j, B|C_{\max}$, made available by the authors Chen *et al.* [1].

Number of jobs (n_J)	Processing time (p_j)	Jobs size (s_j)	Machine capacity (B)
10, 20, 50, 100, 300, 500	p_1: [1, 10] p_2: [1, 20]	s_1: [1, 10] s_2: [2, 4] s_3: [4, 8]	$B = 10$

The following statistics were considered in our analysis for the problem: The computational time of `CPLEX` in seconds ($T(s)$). We represent the time by the symbol "-" in our tables when `CPLEX` reaches the time limit of 1800 seconds on all instances of a given configuration. The makespan corresponding to the best solution obtained by `CPLEX` ($C_{\max}$). The duality gap of `CPLEX` at the end of its execution (Gap). The number of instances that the `CPLEX` ensure the optimal solution (#O).

We present in Table 3.2 comparison results among two models proposed in this work and the other model from the literature. All values presented in the Table 3.2 are the average results computed over the instances of the same configuration, as described in Table 3.1. Therefore, we note that it is possible to have the computational times of `CPLEX` for solving problems of a given configuration less than 1800 seconds while the gaps are non-zero. This happens when some of the instances in the group could be solved to optimality in 1800 seconds, and others could not.

Table 3.2: Computational results for the instances available by Chen *et al.* [1] for the $1|s_j.B|C_{\max}$ problem.

Instance		(MILP_1)				(MILP_1^+)				(FLOW)			
Jobs	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1.10]$													
10	p_1s_1	0.08	**36.86**	**0.00**	**100**	**0.00**	**36.86**	**0.00**	**100**	0.01	**36.86**	**0.00**	**100**
10	p_1s_2	0.04	**20.38**	**0.00**	**100**	**0.01**	**20.38**	**0.00**	**100**	0.02	**20.38**	**0.00**	**100**
10	p_1s_3	0.1	**43.79**	**0.00**	**100**	**0.00**	**43.79**	**0.00**	**100**	**0.00**	**43.79**	**0.00**	**100**
20	p_1s_1	42.62	**68.07**	0.06	99	**0.01**	**68.07**	**0.00**	**100**	0.03	**68.07**	**0.00**	**100**
20	p_1s_2	300.49	**37.13**	1.17	92	0.09	**37.13**	**0.00**	**100**	**0.06**	**37.13**	**0.00**	**100**
20	p_1s_3	24.37	**83.69**	0.03	99	**0.00**	**83.69**	**0.00**	**100**	0.01	**83.69**	**0.00**	**100**
50	p_1s_1	1618.35	164.17	7.83	15	0.84	**164.08**	**0.00**	**100**	**0.18**	**164.08**	**0.00**	**100**
50	p_1s_2	-	87.94	66.65	0	317.12	**87.39**	0.2	88	**0.36**	**87.39**	**0.00**	**100**
50	p_1s_3	1649.45	**202.03**	14.17	13	0.02	**202.03**	**0.00**	**100**	**0.01**	**202.03**	**0.00**	**100**
100	p_1s_1	-	325.38	73.99	0	70.24	**318.99**	0.02	97	**0.17**	**318.99**	**0.00**	**100**
100	p_1s_2	-	183.78	88.82	0	1689.67	**170.58**	1.41	7	**0.61**	**170.58**	**0.00**	**100**
100	p_1s_3	-	401.7	83.99	0	0.11	**396.96**	**0.00**	**100**	**0.01**	**396.96**	**0.00**	**100**
300	p_1s_1	-	1832.31	99.38	0	467.18	928.64	0.05	76	**0.23**	**928.63**	**0.00**	**100**
300	p_1s_2	-	2508.79	99.53	0	-	496.07	0.86	0	**1.04**	**495.66**	**0.00**	**100**
300	p_1s_3	-	1999.85	99.44	0	20.4	**1174.46**	0.0009	99	**0.09**	**1174.46**	**0.00**	**100**
500	p_1s_1	-	3112.85	99.77	0	825.1	1544.31	0.05	59	**0.17**	**1544.30**	**0.00**	**100**
500	p_1s_2	-	4970.07	99.8	0	-	832.43	0.72	0	**0.97**	**831.04**	**0.00**	**100**
500	p_1s_3	-	3347.23	99.88	0	34.39	**1949.76**	0.0005	99	**0.02**	**1949.76**	**0.00**	**100**
Instances with $p_2 = [1.20]$													
10	p_2s_1	0.07	**67.62**	**0.00**	**100**	**0.00**	**67.62**	**0.00**	**100**	0.02	**67.62**	**0.00**	**100**
10	p_2s_2	0.04	**40.22**	**0.00**	**100**	**0.02**	**40.22**	**0.00**	**100**	0.03	**40.22**	**0.00**	**100**
10	p_2s_3	0.09	**81.05**	**0.00**	**100**	**0.00**	**81.05**	**0.00**	**100**	0.01	**81.05**	**0.00**	**100**
20	p_2s_1	22.35	**133.09**	0.01	99	**0.01**	**133.09**	**0.00**	**100**	0.08	**133.09**	**0.00**	**100**
20	p_2s_2	215.7	**72.88**	0.96	94	**0.09**	**72.88**	**0.00**	**100**	0.13	**72.88**	**0.00**	**100**
20	p_2s_3	16.49	**159.11**	**0.00**	**100**	**0.00**	**159.11**	**0.00**	**100**	0.01	**159.11**	**0.00**	**100**
50	p_2s_1	1596.95	314.76	9.4	15	**0.38**	**314.57**	**0.00**	**100**	0.62	**314.57**	**0.00**	**100**
50	p_2s_2	-	169.93	65.57	0	178.7	**168.11**	0.06	94	**1.81**	**168.11**	**0.00**	**100**
50	p_2s_3	1701.53	384.15	15.56	8	**0.02**	**384.13**	**0.00**	**100**	**0.02**	**384.13**	**0.00**	**100**
100	p_2s_1	-	622.84	76.62	0	33.35	**610.64**	0.01	99	**0.86**	**610.64**	**0.00**	**100**
100	p_2s_2	-	357.69	89.35	0	1717.84	326.14	0.99	6	**4.71**	**326.11**	**0.00**	**100**
100	p_2s_3	-	775.5	83.96	0	0.1	**766.91**	**0.00**	**100**	**0.04**	**766.91**	**0.00**	**100**
300	p_2s_1	-	3979.09	99.42	0	329.03	1793.54	0.02	83	**0.96**	**1793.52**	**0.00**	**100**
300	p_2s_2	-	5925.99	99.66	0	-	965.42	1.08	0	**6.55**	**962.77**	**0.00**	**100**
300	p_2s_3	-	3983.88	99.43	0	21.78	**2247.39**	0.0005	99	**0.17**	**2247.39**	**0.00**	**100**
500	p_2s_1	-	6264.18	99.92	0	780.62	2964.62	0.03	63	**0.89**	**2964.57**	**0.00**	**100**
500	p_2s_2	-	9937.42	99.94	0	-	1592.72	0.86	0	**5.66**	**1587.50**	**0.00**	**100**
500	p_2s_3	-	6822.02	99.92	0	97.16	**3701.79**	0.0013	96	**0.13**	**3701.79**	**0.00**	**100**

The comparative tests clearly show that formulation (FLOW) is superior to (MILP_1) and (MILP_1^+) especially when the number of jobs increases. This is the first work that presents the proved optimal solution of all instances with up to 500 jobs in a fixed computational time. For instances with 50 jobs or less, formulation (MILP_1^+) can solve some instances in less computational time than (FLOW), but the difference between times is always a fraction of a second. Additionally, the duality gaps shown for (MILP_1) reveal the difficulty in obtaining good lower bounds. This difficulty is reduced with the use of (MILP_1^+), but only (FLOW) can show results with no-gap for any instances with more than 50 jobs. With formulation (FLOW), we were able to prove the optimality for 86% of the instances in less than 1 second, while with models (MILP_1^+) and (MILP_1), we prove optimality for 56.75% and 17.28% of the instances, respectively.

The total computational time spent on our 3600 test instances, when using model (FLOW), was 44 minutes and 29 seconds, while it was 15 days, 22 hours, 53 minutes and 56 seconds when using (MILP_1^+), and 49 days, 23 hours, 41 minutes and 13 seconds, when using (MILP_1).

Unlike what we have with models (MILP_1) and (MILP_1^+), the number of variables in (FLOW) does not grow when the number of jobs increases. Moreover, the flow graph does not change in this case. Only the bounds on the variables change. The flow graphs of two distinct instances will be the same if the settings in the parameters Processing Time, Job Size, and Machine Capacity are the same. In fact, this is a very important characteristic of the flow approach. We finally note that the computational time to construct the graphs for the flow formulation was not considered in these times. However, the maximum time to construct a graph for any instance in our experiments was 0.008 second.

Table 3.3 shows the solutions of the linear relaxations of models (MILP_1^+) and (FLOW). The following statistics were considered: the makespan corresponding to the optimal solution of the instance previous calculated ($C_{\max}*$), the makespan corresponding to the solution of the relaxations ($C_{\max}$), the gap given by: gap = ($C_{\max}*$ - $C_{\max}$)/$C_{\max}$, the number of simplex iterations (Iter.), and the computational time in seconds ($T(s)$). We can notice that the linear relaxation of (FLOW) is better than the linear relaxation of (MILP_1^+) for most instances, especially when the number of jobs increases. For instances of type s_2, the solutions of the linear relaxations are very close to each other, but (FLOW) is slightly better in this case. Another critical point is that (FLOW) presents lower computational time for solving the linear relaxations. With the number of jobs 500 in instances of type p_2s_2, the time for linear relaxation of (FLOW) is 0.66% of the time for the linear relaxation of (MILP_1^+). This time difference tends to increase with the increase in the number of jobs. These tests help us understand why (FLOW) overcomes (MILP_1^+).

Table 3.3: Comparison between the linear relaxations for (MILP$_1^+$) and (FLOW) for the $1|s_j, B|C_{\max}$ problem.

Instance			(MILP$_1^+$) - Relax				(FLOW) - Relax			
Jobs	Type	$C_{max}*$	$C_{\max}$	Gap	Iter.	$T(s)$	$C_{\max}$	Gap	Iter.	$T(s)$
Instances with $p_1 = [1, 10]$										
10	p_1s_1	36.86	32.34	12.26	**29.41**	**0.00**	**34.41**	**6.66**	79.52	**0.00**
10	p_1s_2	20.38	**18.61**	**8.70**	**36.66**	**0.00**	16.79	17.61	82.88	**0.00**
10	p_1s_3	43.79	34.82	20.49	34.09	**0.00**	**41.72**	**4.74**	**14.77**	**0.00**
20	p_1s_1	68.07	61.42	9.77	**80.21**	**0.00**	**65.34**	**4.02**	181.54	**0.00**
20	p_1s_2	37.13	**34.54**	**6.99**	**103.60**	**0.00**	33.38	10.11	133.90	**0.00**
20	p_1s_3	83.69	66.80	20.18	96.61	**0.00**	**81.49**	**2.63**	**26.04**	**0.00**
50	p_1s_1	164.08	151.94	7.40	330.65	0.01	**161.03**	**1.86**	**318.89**	**0.01**
50	p_1s_2	87.39	83.07	4.94	341.59	0.01	**83.15**	**4.85**	**168.99**	**0.00**
50	p_1s_3	202.03	165.65	18.01	385.43	0.01	**199.71**	**1.15**	**42.95**	**0.00**
100	p_1s_1	318.99	302.63	5.13	675.16	0.04	**315.98**	**0.94**	**386.88**	**0.01**
100	p_1s_2	170.58	165.68	2.87	720.13	0.05	**165.99**	**2.69**	**178.51**	**0.00**
100	p_1s_3	396.96	328.82	17.17	742.44	0.05	**394.65**	**0.58**	**56.13**	**0.00**
300	p_1s_1	928.63	902.86	2.77	2729.99	0.36	**925.87**	**0.30**	**467.62**	**0.01**
300	p_1s_2	495.66	490.81	0.98	3186.63	0.44	**490.85**	**0.97**	**180.92**	**0.01**
300	p_1s_3	1174.46	988.13	15.87	3043.30	0.40	**1172.12**	**0.20**	**62.20**	**0.00**
500	p_1s_1	1544.30	1513.43	2.00	4673.94	1.19	**1541.61**	**0.17**	**480.35**	**0.01**
500	p_1s_2	831.04	826.12	0.59	5387.19	1.64	**826.12**	**0.59**	**180.45**	**0.01**
500	p_1s_3	1949.76	1645.82	15.59	5354.30	1.47	**1947.47**	**0.12**	**61.54**	**0.00**
Instances with $p_2 = [1, 20]$										
10	p_2s_1	67.62	59.67	11.75	**30.16**	**0.00**	**62.33**	**7.82**	109.14	**0.00**
10	p_2s_2	40.22	**36.55**	**9.14**	**37.14**	**0.00**	32.69	18.72	101.74	**0.00**
10	p_2s_3	81.05	64.90	19.93	33.71	**0.00**	**77.00**	**5.00**	**18.71**	**0.00**
20	p_2s_1	133.09	119.97	9.86	**84.59**	**0.00**	**128.09**	**3.75**	253.47	0.01
20	p_2s_2	72.88	**68.24**	**6.37**	**105.44**	**0.00**	65.23	10.50	181.63	**0.00**
20	p_2s_3	159.11	129.26	18.76	97.66	**0.00**	**154.68**	**2.78**	**38.11**	**0.00**
50	p_2s_1	314.57	291.85	7.22	**331.51**	**0.01**	**309.22**	**1.70**	566.30	0.02
50	p_2s_2	168.11	**161.19**	**4.12**	352.33	0.01	159.89	4.89	**295.78**	**0.00**
50	p_2s_3	384.13	313.80	18.31	374.71	0.01	**379.53**	**1.20**	**73.38**	**0.00**
100	p_2s_1	610.64	579.74	5.06	**693.34**	0.05	**605.19**	**0.89**	782.51	**0.02**
100	p_2s_2	326.11	316.77	2.86	742.66	0.05	**317.20**	**2.73**	**335.60**	**0.01**
100	p_2s_3	766.91	633.63	17.38	783.24	0.05	**762.26**	**0.61**	**102.42**	**0.00**
300	p_2s_1	1793.52	1737.65	3.11	2786.36	0.36	**1788.23**	**0.29**	**963.18**	**0.03**
300	p_2s_2	962.77	952.38	1.08	3181.75	0.41	**952.79**	**1.04**	**357.50**	**0.01**
300	p_2s_3	2247.39	1892.29	15.80	3161.54	0.41	**2242.72**	**0.21**	**135.67**	**0.00**
500	p_2s_1	2964.57	2895.60	2.33	4966.34	1.25	**2959.25**	**0.18**	**1017.03**	**0.04**
500	p_2s_2	1587.50	1577.43	0.63	5496.04	1.51	**1577.56**	**0.63**	**360.63**	**0.01**
500	p_2s_3	3701.79	3138.60	15.21	5664.89	1.52	**3697.31**	**0.12**	**136.55**	**0.00**

All the results show that instances of configuration s_2 require more computational time and are more difficult when compared to the other instances for all formulations. The reason for this is the small sizes of the jobs when compared to the machine capacity, which allows more combinations of assignment to a batch. In this way, the number of feasible solutions for instances of type s_2 is higher than others instances.

With (FLOW), it is possible to find the optimal solution of all the instances proposed by Chen *et al.* [1] with good computational times. Therefore, we created new instances with the number of jobs up to 100 million of type p_2s_2, since they are the instances with higher computational times. Table 3.4 shows the computational results obtained. (FLOW) can find the optimum solution of instances with up to 100 million jobs with computational times between 3.25s and 6.05s, the same time frame the model needs to find the optimal solution of instances with 100 jobs. We emphasize that the column Construction Time considers the reading of the instance file, the sorting of the jobs and the creation of the parameters to the model. The construction of the model is a polynomial time method according to the number of jobs and can be longer than the resolution time when the number of jobs increases, which demonstrates the efficiency of (FLOW).

Table 3.4: Computational results for the instances available by Chen *et al.* [1] for the $1|s_j, B|C_{\max}$ problem.

Instance		(FLOW)				
Jobs	Type	$T(s)$	$C_{\max}$	Gap	Nodes	Construction Time
1000	p_2s_2	4.90	3241	0.00	8043	0.03
10000	p_2s_2	3.66	31719	0.00	3625	0.16
100000	p_2s_2	3.22	314945	0.00	4668	1.44
1000000	p_2s_2	2.80	3152697	0.00	3011	14.24
10000000	p_2s_2	6.05	31495193	0.00	17111	145.07
100000000	p_2s_2	3.25	314996812	0.00	4325	886.14

3.4.2 New instances proposed

The computational results from the last section shows that the instances proposed by Chen *et al.* [1] are not challenging enough, especially when using (FLOW). Therefore, a second set of test instances for problem $1|s_j, B|C_{\max}$ is proposed in this work, generated from the respective uniform distribution depicted in Table 3.5. The main idea is to generate more difficult instances that can be used in future work. For this, we explore the increase of the values of the parameters B, p_j and s_j, which directly affect the number of variables of (FLOW) model. The set considers three different values for the machine capacity, and the job sizes are generated by a uniform distribution in a range proportional to the size of the machine, i.e., the s_2 has the range $[0.2B, 0.4B]$, which means that for instances where the capacity of the machine is equal to 100, the distribution is between [20, 40]. This setting allows instances to maintain the characteristics of instances from Chen *et al.* [1] in terms of scale. Also, the range of p_2 varies according to the number of jobs in the instance and allows creating instances with a higher number of different processing times. In total there were generated 540 instances, 5 for each of the 108 different combinations of number of jobs, processing time, job size and machine capacity.

Table 3.5: Parameter settings for set of new instances proposed for $1|s_j, B|C_{\max}$.

Number of jobs (n_J)	Processing time (p_j)	Jobs size (s_j)	Machine capacity (B)
10, 50, 100, 500, 1000, 5000	p_1: [1, 20] p_2: [1, n_J]	s_1: [1, B] s_2: [$0.2B$, $0.4B$] s_3: [$0.4B$, $0.8B$]	$20, 50, 100$

We present in Tables 3.6–3.8 comparison results among the two models proposed in this work for the instances proposed in this section. When `CPLEX` cannot find an integer solution of a given configuration, we represent the $C_{\max}$ by "No solution" and the gap by "Infinite" in our tables.

The comparative tests show that new instances are more difficult for (FLOW), especially with processing time p_2 because the processing times of the jobs are very different, i.e., there are few jobs with the same processing time. In this way, (FLOW) needs to generate many flow structures, one for each processing time. The difference between the instances p_1 and p_2, however, are not influencing the number of variables in (MILP$_1^+$), and the computational results show that this model is more stable when the range of the processing time changes.

(MILP$_1^+$) is superior in instances of type p_2 especially when the capacity of the machine increases. Table 3.7 shows that (FLOW) was not able to find even a single integer solution in some cases. This behavior occurs in the worst case scenario for (FLOW), where a high value for machine capacity is combined with the type of instances p_2. When $B = 100$ the (FLOW) model is generated with many nodes in the arc-flow structures, which reflects in the computational performance. Another difference between this new set of instances and the instances tested in the previous section is that the range of jobs sizes varies with the number of jobs. The number of arcs in (FLOW) increases as the number of jobs increases.

Even with the difficulties of this new set, the (FLOW) model obtained superior results for instances of type p_1, especially when the number of jobs increases. In instances of type p_1 with B up to 20, it was possible to find the optimal solution for all instances. These tests show that both models are complementary, that is, there are clearly situations in which each model is superior to the other.

Table 3.6: Computational results for the new instances proposed for the $1|s_j, B|C_{\max}$ problem, with $B = 20$.

Instance			(MILP$_1^+$)				(FLOW)			
Jobs	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$ and $B = 20$										
10	20	p_1s_1	**0.01**	**62.20**	**0.00**	**5**	0.03	**62.20**	**0.00**	**5**
10	20	p_1s_2	**0.01**	**45.40**	**0.00**	**5**	0.12	**45.40**	**0.00**	**5**
10	20	p_1s_3	**0.00**	**71.80**	**0.00**	**5**	0.01	**71.80**	**0.00**	**5**
50	20	p_1s_1	**0.63**	**316.20**	**0.00**	**5**	1.23	**316.20**	**0.00**	**5**
50	20	p_1s_2	276.05	**181.00**	**0.00**	**5**	**13.67**	**181.00**	**0.00**	**5**
50	20	p_1s_3	**0.01**	**373.80**	**0.00**	**5**	0.02	**373.80**	**0.00**	**5**
100	20	p_1s_1	**0.87**	**629.60**	**0.00**	**5**	1.20	**629.60**	**0.00**	**5**
100	20	p_1s_2	1468.32	326.80	0.90	2	**20.91**	**326.40**	**0.00**	**5**
100	20	p_1s_3	0.05	**791.00**	**0.00**	**5**	**0.02**	**791.00**	**0.00**	**5**
500	20	p_1s_1	1145.40	2805.40	0.03	2	**2.52**	**2805.20**	**0.00**	**5**
500	20	p_1s_2	-	1613.60	1.80	0	**39.35**	**1595.20**	**0.00**	**5**
500	20	p_1s_3	2.48	**3869.80**	**0.00**	**5**	**0.07**	**3869.80**	**0.00**	**5**
1000	20	p_1s_1	1570.94	5675.20	0.04	1	**3.16**	**5674.80**	**0.00**	**5**
1000	20	p_1s_2	-	3193.60	1.75	0	**23.58**	**3148.60**	**0.00**	**5**
1000	20	p_1s_3	25.31	**7693.80**	**0.00**	**5**	**0.09**	**7693.80**	**0.00**	**5**
5000	20	p_1s_1	-	52548.60	522.43	0	**1.56**	**28037.80**	**0.00**	**5**
5000	20	p_1s_2	-	52520.40	100.00	0	**31.32**	**15735.40**	**0.00**	**5**
5000	20	p_1s_3	-	41035.60	6400.14	0	**0.07**	**38108.60**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$ and $B = 20$										
10	20	p_2s_1	**0.01**	**36.20**	**0.00**	**5**	0.05	**36.20**	**0.00**	**5**
10	20	p_2s_2	**0.02**	**24.20**	**0.00**	**5**	0.07	**24.20**	**0.00**	**5**
10	20	p_2s_3	**0.00**	**42.00**	**0.00**	**5**	0.00	**42.00**	**0.00**	**5**
50	20	p_2s_1	**0.14**	**690.20**	**0.00**	**5**	2.52	**690.20**	**0.00**	**5**
50	20	p_2s_2	**47.90**	**423.40**	**0.00**	**5**	110.56	**423.40**	**0.00**	**5**
50	20	p_2s_3	**0.02**	**1044.60**	**0.00**	**5**	0.06	**1044.60**	**0.00**	**5**
100	20	p_2s_1	**1.82**	**2849.40**	**0.00**	**5**	24.15	**2849.40**	**0.00**	**5**
100	20	p_2s_2	-	**1617.60**	**0.52**	**0**	-	**1617.60**	0.64	**0**
100	20	p_2s_3	**0.08**	**3889.40**	**0.00**	**5**	0.13	**3889.40**	**0.00**	**5**
500	20	p_2s_1	**504.40**	**69070.20**	**0.02**	**4**	994.67	69071.00	0.02	3
500	20	p_2s_2	-	38695.00	1.50	**0**	-	**38433.20**	**0.86**	**0**
500	20	p_2s_3	**6.18**	**91883.00**	**0.00**	**5**	30.57	**91883.00**	**0.00**	**5**
1000	20	p_2s_1	-	272294.40	0.06	0	**1423.96**	**272187.80**	**0.01**	**3**
1000	20	p_2s_2	-	155075.60	2.58	**0**	-	**153297.40**	**1.46**	**0**
1000	20	p_2s_3	**32.09**	**371483.00**	**0.00**	**5**	65.80	**371483.00**	**0.00**	**5**
5000	20	p_2s_1	-	12503592.80	421.05	**0**	-	**6659163.00**	**0.16**	**0**
5000	20	p_2s_2	-	12495060.40	100.00	**0**	-	**4175411.50**	**9.47**	**0**
5000	20	p_2s_3	-	**8993880.00**	**0.00**	**0**	-	8994409.60	0.01	**0**

Table 3.7: Computational results for the new instances proposed for the $1|s_j, B|C_{\max}$ problem, with $B = 50$.

Instance			(MILP$_1^+$)				(FLOW)			
Jobs	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$ and $B = 50$										
10	50	p_1s_1	**0.01**	**65.80**	**0.00**	**5**	0.13	**65.80**	**0.00**	**5**
10	50	p_1s_2	**0.02**	**40.60**	**0.00**	**5**	0.18	**40.60**	**0.00**	**5**
10	50	p_1s_3	**0.00**	**109.20**	**0.00**	**5**	0.00	**109.20**	**0.00**	**5**
50	50	p_1s_1	**0.22**	**299.80**	**0.00**	**5**	7.79	**299.80**	**0.00**	**5**
50	50	p_1s_2	72.71	**169.00**	**0.00**	**5**	**53.15**	**169.00**	**0.00**	**5**
50	50	p_1s_3	**0.01**	**429.00**	**0.00**	**5**	0.05	**429.00**	**0.00**	**5**
100	50	p_1s_1	**1.01**	**560.40**	**0.00**	**5**	11.33	**560.40**	**0.00**	**5**
100	50	p_1s_2	-	**317.80**	0.93	0	**654.39**	**317.80**	**0.08**	**4**
100	50	p_1s_3	**0.05**	**746.60**	**0.00**	**5**	0.08	**746.60**	**0.00**	**5**
500	50	p_1s_1	1465.28	2741.20	0.06	2	**84.02**	**2740.60**	**0.00**	**5**
500	50	p_1s_2	-	1602.40	2.77	0	**1625.40**	**1572.80**	**0.25**	**1**
500	50	p_1s_3	1.24	**4036.60**	**0.00**	**5**	**0.25**	**4036.60**	**0.00**	**5**
1000	50	p_1s_1	-	5519.80	26.27	0	**34.75**	**5517.00**	**0.00**	**5**
1000	50	p_1s_2	-	3219.60	3.80	0	**1654.71**	**3113.60**	**0.21**	**1**
1000	50	p_1s_3	9.58	**7901.00**	**0.00**	**5**	**0.14**	**7901.00**	**0.00**	**5**
5000	50	p_1s_1	-	52744.20	186.78	0	**57.91**	**27336.00**	**0.00**	**5**
5000	50	p_1s_2	-	52553.40	100.00	0	**1581.85**	**15769.60**	**0.03**	**1**
5000	50	p_1s_3	1766.61	38921.60	3400.02	1	**0.24**	**38914.80**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$ and $B = 50$										
10	50	p_2s_1	**0.00**	**46.80**	**0.00**	**5**	0.04	**46.80**	**0.00**	**5**
10	50	p_2s_2	**0.04**	**23.20**	**0.00**	**5**	0.13	**23.20**	**0.00**	**5**
10	50	p_2s_3	**0.00**	**48.80**	**0.00**	**5**	0.00	**48.80**	**0.00**	**5**
50	50	p_2s_1	**0.22**	**780.60**	**0.00**	**5**	20.29	**780.60**	**0.00**	**5**
50	50	p_2s_2	**89.05**	**411.40**	**0.00**	**5**	740.47	**411.40**	**0.00**	**5**
50	50	p_2s_3	**0.01**	**1010.40**	**0.00**	**5**	0.09	**1010.40**	**0.00**	**5**
100	50	p_2s_1	**4.58**	**2640.00**	**0.00**	**5**	1028.85	2640.20	0.10	3
100	50	p_2s_2	**1698.37**	1636.00	**0.68**	**1**	-	**1634.00**	1.06	0
100	50	p_2s_3	**0.05**	**4251.00**	**0.00**	**5**	0.21	**4251.00**	**0.00**	**5**
500	50	p_2s_1	**899.33**	**65347.80**	**0.03**	**3**	-	65388.80	0.13	0
500	50	p_2s_2	-	39222.40	2.71	**0**	-	**38819.00**	**1.37**	**0**
500	50	p_2s_3	**1.23**	**94798.00**	**0.00**	**5**	10.94	**94798.00**	**0.00**	**5**
1000	50	p_2s_1	**1555.41**	262289.20	0.26	**1**	1711.02	**261181.50**	**0.16**	**1**
1000	50	p_2s_2	-	**157692.20**	**4.07**	**0**	-	No solution	Infinite	0
1000	50	p_2s_3	**7.95**	**371115.80**	**0.00**	**5**	48.52	**371115.80**	**0.00**	**5**
5000	50	p_2s_1	-	**12573122.60**	**161.91**	**0**	-	No solution	Infinite	0
5000	50	p_2s_2	-	**12546134.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	50	p_2s_3	797.53	**9291848.20**	**0.00**	4	**524.10**	9291858.80	**0.00**	4

Table 3.8: Computational results for the new instances proposed for the $1|s_j, B|C_{\max}$ problem, with $B = 100$.

Instance			(MILP$_1^+$)				(FLOW)			
Jobs	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$ and $B = 100$										
10	100	p_1s_1	**0.01**	**77.20**	**0.00**	**5**	0.06	**77.20**	**0.00**	**5**
10	100	p_1s_2	**0.01**	**44.00**	**0.00**	**5**	0.24	**44.00**	**0.00**	**5**
10	100	p_1s_3	**0.00**	**94.80**	**0.00**	**5**	0.01	**94.80**	**0.00**	**5**
50	100	p_1s_1	**0.07**	**316.00**	**0.00**	**5**	3.39	**316.00**	**0.00**	**5**
50	100	p_1s_2	**9.49**	**180.60**	**0.00**	**5**	189.28	**180.60**	**0.00**	**5**
50	100	p_1s_3	**0.01**	**411.20**	**0.00**	**5**	0.07	**411.20**	**0.00**	**5**
100	100	p_1s_1	**2.05**	**586.80**	**0.00**	**5**	82.86	**586.80**	**0.00**	**5**
100	100	p_1s_2	**1372.59**	329.00	0.74	**2**	1588.41	**328.60**	**0.47**	**2**
100	100	p_1s_3	**0.05**	**811.20**	**0.00**	**5**	0.16	**811.20**	**0.00**	**5**
500	100	p_1s_1	877.55	2715.40	0.07	3	**664.78**	**2714.40**	**0.01**	**4**
500	100	p_1s_2	-	1601.60	3.09	**0**	-	**1595.20**	**2.26**	**0**
500	100	p_1s_3	1.71	**4096.20**	**0.00**	**5**	**1.06**	**4096.20**	**0.00**	**5**
1000	100	p_1s_1	-	5591.40	0.63	0	**902.17**	**5569.40**	**0.02**	**3**
1000	100	p_1s_2	-	3282.00	4.51	**0**	-	**3196.80**	**1.72**	**0**
1000	100	p_1s_3	13.43	**8094.80**	**0.00**	**5**	**1.46**	**8094.80**	**0.00**	**5**
5000	100	p_1s_1	-	52416.60	123.69	0	**883.92**	**26898.60**	**0.00**	**3**
5000	100	p_1s_2	-	52518.60	100.00	**0**	-	**15879.40**	**0.62**	**0**
5000	100	p_1s_3	1481.01	**39307.40**	**0.00**	**5**	**0.62**	**39307.40**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$ and $B = 100$										
10	100	p_2s_1	**0.00**	**32.60**	**0.00**	**5**	0.10	**32.60**	**0.00**	**5**
10	100	p_2s_2	**0.02**	**21.40**	**0.00**	**5**	0.18	**21.40**	**0.00**	**5**
10	100	p_2s_3	**0.00**	**49.40**	**0.00**	**5**	0.00	**49.40**	**0.00**	**5**
50	100	p_2s_1	**0.16**	**823.60**	**0.00**	**5**	281.80	**823.60**	**0.00**	**5**
50	100	p_2s_2	**40.72**	**441.60**	**0.00**	**5**	778.21	**441.60**	0.40	4
50	100	p_2s_3	**0.01**	**1041.40**	**0.00**	**5**	0.13	**1041.40**	**0.00**	**5**
100	100	p_2s_1	**0.63**	**3092.20**	**0.00**	**5**	654.44	3092.40	0.02	4
100	100	p_2s_2	-	**1631.60**	**0.59**	**0**	-	1636.40	1.59	**0**
100	100	p_2s_3	**0.06**	**4090.20**	**0.00**	**5**	0.68	**4090.20**	**0.00**	**5**
500	100	p_2s_1	**1207.01**	**65728.00**	**0.02**	**2**	-	69647.00	1.32	0
500	100	p_2s_2	-	**39355.60**	**3.43**	**0**	-	41833.00	9.15	**0**
500	100	p_2s_3	**0.76**	**98099.00**	**0.00**	**5**	6.06	**98099.00**	**0.00**	**5**
1000	100	p_2s_1	-	**262033.20**	**0.32**	**0**	-	No solution	Infinite	**0**
1000	100	p_2s_2	-	**168105.40**	**9.53**	**0**	-	178472.20	14.76	**0**
1000	100	p_2s_3	**6.72**	**381506.40**	**0.00**	**5**	46.58	**381506.40**	**0.00**	**5**
5000	100	p_2s_1	-	**12529354.80**	**107.49**	**0**	-	No solution	Infinite	**0**
5000	100	p_2s_2	-	**12488031.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	100	p_2s_3	**891.11**	**9404104.80**	**0.00**	4	1493.59	9404178.20	**0.00**	1

3.5 A Column Generation Approach

Another approaches have been tested for this problem, besides those presented, including a column generation for this problem proposed in Rafiee Parsa *et al.* [30]. Also, we propose a new set of sub-problems that allows adding more than one column to the master problem in a single interaction. Both sub-problems and the column generation approach are presented next. We have concluded that the column generation presented was not efficient because it presents computational times that are not competitive with the (FLOW) model. That is, the computational times of the generation of columns are worse than the times presented by the (FLOW) model. Also, the column generation provides only a bound for the problem and should be combined with some approach to finding the integer optimal solution, such as the branch-and-price.

We review in this section the Column Generation approach proposed by Rafiee Parsa *et al.* [30] for problem $1|s_j, B|C_{max}$, and then a new sub-problem formulation is presented. Both approaches are tested and compared with the (FLOW) model, presented in Section 3.3.

Column Generation (CG) is a technique usually used when a problem has a large number of decision variables. The best reduced cost of a nonbasic variable is found through a new optimization problem, in order to avoid the explicit calculation of the reduced cost for all nonbasic variables. From the Dantzig-Wolfe decomposition [45], the original problem is partitioned into Master Problem (MP) and Sub-Problem (SP).

For the MP below, each column corresponding to the binary variable x_k represents a batch $k \in 1..N$, where N is the number of batches generated by SP. In a given solution, if $x_k = 1$ the batch is used, and $x_k = 0$ otherwise. The constant a_{jk} determines whether a job j is assigned ($a_{jk} = 1$) or not ($a_{jk} = 0$) to the batch generated by SP. Coefficient P_k indicates the respective processing time of batch. The MP model can be formalized as follows:

$$\text{(MP)} \quad \min \sum_{k \in N} P_k x_k \tag{3.27}$$

$$\sum_{k \in N} a_{jk} x_k = 1 \qquad \forall j \in J \tag{3.28}$$

$$x_k \in \{0, 1\} \qquad \forall k \in K \tag{3.29}$$

The objective function (3.27) minimizes the total time required by the machine processing (makespan). Constraints (3.28) ensure that each job is assigned to a single batch. Constraint (3.29) guarantees the binary domain of the decision variable.

Through the linear relaxation of the binary variable x_k, a Relaxed Master Problem (RMP) is defined. The iterative process of Column Generation is described by the following steps:

1 - Determine an initial set of columns for the Master Problem;

2 - Solve the linear relaxation of the master problem (RMP);

3 - Solve the Sub-Problem using the dual values π_j of the solution obtained by the previous step;

4 - If the optimal solution of the previous step has negative reduced cost, insert a new column in RMP from this solution and repeat step 2. Otherwise, the optimal solution was found.

3.5.1 Sub-Problem from the literature

In Rafiee Parsa *et al.* [30] a Sub-Problem to generate a single column by iteration is presented. Each column can contain any batch design that can improve the solution to the master RMP problem. In this formulation, the variable y_j indicates whether the job j belongs or not to the batch. The variable P_k defines the processing time of the batch. The SP is formally defined as follows:

$$\text{(SP)} \quad \min\ P_k - \sum_{j \in J} \pi_j y_j \tag{3.30}$$

$$\sum_{j \in J} s_j y_j \leq B \qquad \forall j \in J \tag{3.31}$$

$$p_j y_j \leq P_k \qquad \forall j \in J \tag{3.32}$$

$$P_k \in \{0, 1\} \tag{3.33}$$

$$y_j \in \{0, 1\} \qquad \forall j \in J \tag{3.34}$$

The objective function (3.30) minimizes the reduced cost corresponding to the x_k variable of MP. Constraints (3.31) ensure that the capacity of the machine is not exceeded. Constraints (3.32) define the batch processing time. Constraints (3.33) and (3.34) define the domains of the decision variables.

The first step of Column Generation is the creation of a initial solution. In Rafiee Parsa *et al.* [30] an initial solution is suggested, using solutions of two heuristics widely used in the literature: Batch first-fit (BFF) and Batch best-fit (BBF) [2] [21]. Each heuristic is executed by sorting the instance into four different modes, as: increasing by processing times; decreasing by processing times; decreasing by jobs sizes; increasing by the ratio of p_j/s_j.

The procedure adds to model MP all the solutions found by the heuristics. Model MP is solved only once to find an initial solution of better quality, which will be used in the first iteration of CG.

3.5.2 Sub-Problem proposed

In this section we propose a new set of sub-problems for Column Generation, inspired by the symmetry breaking method presented in Section 3.2.1.

We decomposed the model SP into n_K different sub-problems, where $n_K = n_J$, and it is required to sort the jobs by non-decreasing processing time. Batch $k \in K$ can only have the jobs $1 \dots k$, with job k mandatory, with a fixed processing time p_k. Therefore, each sub-problem is designed to generate different batches, that is, a sub-problem k never creates a batch equal to that produced by sub-problem l, for $k \neq l$.

A new set of subproblems is formulated as follows:

$$\text{(SP}_2\text{)} \quad \min\ p_k - \pi_k - \sum_{j \in J, j<k} \pi_j y_j \tag{3.35}$$

$$\sum_{j \in J, j<k} s_j y_j \leq B - s_k \qquad \forall j \in J \tag{3.36}$$

$$y_j \in \{0,1\} \qquad \forall j \in J, j < k \tag{3.37}$$

for $\forall k \in K$.

Set SP$_2$ corresponds to n_K different knapsack sub-problems, one for each batch k. In this case, the solutions of n_K sub-problems correspond to a single iteration of CG. The objective function (3.35) minimizes the reduced cost that corresponds to the new solution, where p_k is a constant resulting from $y_k = 1$. Constraints (3.36) ensure that the new solution respects the remaining capacity of the machine. Constraints (3.37) guarantee the binary domain of the variable y_k.

Each sub-problem k generates a solution that may have a negative cost. In this approach, not only the most cost-effective solution is included in RMP, but all solutions that present a negative reduced cost in each iteration. In this case, SP$_2$ includes more columns in RMP than SP, to better exploit the computational effort and try to promote a decrease in the convergence time.

3.5.3 Dynamic Programming

To solve SP, Rafiee Parsa *et al.* [30] use an exact Dynamic Programming (DP) algorithm, which has a pseudo-polynomial time complexity of O(n^2B). Let $G(j, d, t)$ be the minimum objective value of SP for the first $j \in J$ jobs, with the capacity limitation $d \in \{0 \dots B\}$, and the batch processing time $t \in T$, where T is a set of distinct processing times among all job processing times. The recursive Dynamic Programming to solve SP is defined as follows:

$$G(j,d,t) = \begin{cases} G\big(j-1,d,t\big), & \text{if } d < s_j, \\ \min\begin{pmatrix} G\big(j-1,d,t\big), \\ G\big(j-1,d-s_j,\max(p_j,t)\big) + \max(p_j-t,0) - \pi_j \end{pmatrix} & \text{if } d \geq s_j, \end{cases}$$

for $\forall j \in J$, $\forall d \in \{0 \dots B\}$, and $\forall t \in T$.

Some initials conditions are defined as:

$$G(j,0,t) = 0, \qquad \forall d \in \{0\dots B\}, \forall t \in T, \tag{3.38}$$

$$G(0,d,t) = 0, \qquad \forall j \in J, \forall t \in T, \tag{3.39}$$

$$\text{if } d = B, \text{ then } t = 0, \qquad \forall j \in J, \forall t \in T. \tag{3.40}$$

There are two cases to be considered to define the structure of $G(j,d,t)$ for every job: If $d < s_j$, the job cannot be added in the solution because the machine does not have the necessary capacity in iteration d. Otherwise, it is required to choose the minimum solution between two situations: (1) the job is not included in the set; (2) the job is added to the optimal subset. Conditions (3.38) and (3.39) define the first row and column of the DP, where there are not jobs or capacity available, so the solution is 0.

3.5.4 Dynamic Programming proposed

We propose a different conception of Dynamic Programming for SP$_2$, using the classic implementation of 0–1 knapsack problem from Martello and Toth [46]. The formulation has a pseudo-polynomial time complexity of O(nB). Let $G_2(j,d)$ is the minimum objective value of SP$_2$ for the first $j \in J$ jobs, with the capacity limitation $d \in \{0 \dots B\}$. The recursive Dynamic Programming to solve SP$_2$ is defined as follows:

$$G_2(j,d) = \begin{cases} G_2\big(j-1,d\big), & \text{if } d < s_j, \\ \min\begin{pmatrix} G_2\big(j-1,d\big), \\ G_2\big(j-1,d-s_j\big) - \pi_j \end{pmatrix} & \text{if } d \geq s_j, \end{cases}$$

for $\forall j \in J$ and $\forall d \in \{0 \dots B\}$.

Some initials conditions are defined as:

$$G_2(j,0) = 0, \qquad \forall d \in \{0\dots B\}, \tag{3.41}$$

$$G_2(0,d) = 0, \qquad \forall j \in J. \tag{3.42}$$

Structure $G_2(j,d)$ works with the same logical of $G(j,d,t)$. The difference is that in this formulation, the solution obtained by $G_2(j,d)$ is only $\sum \pi_j$ of jobs that belong to the final solution. The reduced cost must be calculated by (3.43), after the Dynamic Programming runs. This allows us to use all the optimal subsets from $G_2(j, B)$, $\forall j \in J$. Through the solution reconstruction method from Dynamic Programming, all the solutions that present negative reduced cost, $RC < 0$, are added to the MP of the Column Generation procedure. It is important to make sure that $\forall j \in J$, a new column is only added if job j belongs to the solution.

$$RC = G_2(j, B) + p_j \qquad \forall j \in J. \tag{3.43}$$

3.5.5 Computational results

Both CG approaches presented in this chapter were compared with model (FLOW) through computational tests performed with the same set of instances presented in Section 3.4.1, proposed by Chen *et al.* [1]. In all tests, we use the CPLEX version 12.7.1.0, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.70GHz Intel Quad-Core Xeon E5-2697 v2 processor and 64GB of RAM.

The following statistics were considered in our analysis for the problem: the computational time in seconds ($T(s)$), the makespan corresponding to the solution obtained (C_{max}), the gap given by: gap = ($C_{\text{max}}*$ - C_{max})/C_{max}, and the number of instances for which CPLEX ensures the optimal solution (#O).

Table 3.9 shows the computational results. Set SP_2 of sub-problems shows an improvement in computational times compared to SP. This difference can be justified through the insertion of a larger number of columns in the master problem using SP_2. In all cases, the C_{max} values using SP are identical to the ones found by SP_2.

The results show that the column generation method spend more computational time than FLOW, in all instances tested. Thus, the FLOW model overcomes the Branch-and-Price method proposed in Rafiee Parsa *et al.* [30], because the solutions obtained through the Column Generation correspond to the initial node of the Branch-and-Price method, that need to run at least once node to find the optimal integer solution.

Table 3.9: Computational results for the instances available by Chen *et al.* [1] for $1|s_j.B|C_{\max}$ using Column Generation approach.

Instance		(CG with SP)			(CG with SP$_2$)			(FLOW)			
Jobs	Type	$T(s)$	$C_{\max}$	Gap	$T(s)$	$C_{\max}$	Gap	$T(s)$	$C_{\max}$	Gap	#O
					Instances with $p_1 = [1.10]$						
10	p_1s_1	1.65	36.49	1.02	0.75	36.49	1.02	**0.01**	**36.86**	**0.00**	**100**
10	p_1s_2	1.65	19.52	4.20	0.45	19.52	4.20	**0.02**	**20.38**	**0.00**	**100**
10	p_1s_3	1.05	43.52	0.63	0.60	43.52	0.63	**0.00**	**43.79**	**0.00**	**100**
20	p_1s_1	5.30	67.50	0.83	0.90	67.50	0.83	**0.03**	**68.07**	**0.00**	**100**
20	p_1s_2	12.23	35.83	3.49	2.70	35.83	3.49	**0.06**	**37.13**	**0.00**	**100**
20	p_1s_3	2.85	83.25	0.53	1.05	83.25	0.53	**0.01**	**83.69**	**0.00**	**100**
50	p_1s_1	39.59	162.92	0.70	3.90	162.92	0.70	**0.18**	**164.08**	**0.00**	**100**
50	p_1s_2	69.90	84.47	3.34	8.72	84.47	3.34	**0.36**	**87.39**	**0.00**	**100**
50	p_1s_3	19.52	201.14	0.44	4.69	201.14	0.44	**0.01**	**202.03**	**0.00**	**100**
100	p_1s_1	180.11	317.46	0.48	15.74	317.46	0.48	**0.17**	**318.99**	**0.00**	**100**
100	p_1s_2	236.52	166.37	2.47	40.90	166.37	2.47	**0.61**	**170.58**	**0.00**	**100**
100	p_1s_3	64.80	395.55	0.36	5.42	395.55	0.36	**0.01**	**396.96**	**0.00**	**100**
300	p_1s_1	1808.50	926.45	0.24	244.58	926.45	0.24	**0.23**	**928.63**	**0.00**	**100**
300	p_1s_2	1422.25	490.85	0.97	611.00	490.85	0.97	**1.04**	**495.66**	**0.00**	**100**
300	p_1s_3	543.01	1172.15	0.20	52.52	1172.15	0.20	**0.09**	**1174.46**	**0.00**	**100**
500	p_1s_1	5797.22	1541.76	0.16	1190.42	1541.76	0.16	**0.17**	**1544.30**	**0.00**	**100**
500	p_1s_2	3500.45	826.12	0.59	3161.47	826.12	0.59	**0.97**	**831.04**	**0.00**	**100**
500	p_1s_3	1523.24	1947.47	0.12	196.03	1947.47	0.12	**0.02**	**1949.76**	**0.00**	**100**
					Instances with $p_2 = [1.20]$						
10	p_2s_1	1.50	66.72	1.33	0.30	66.72	1.33	**0.02**	**67.62**	**0.00**	**100**
10	p_2s_2	1.05	38.24	4.92	1.50	38.24	4.92	**0.03**	**40.22**	**0.00**	**100**
10	p_2s_3	0.90	80.52	0.66	0.60	80.52	0.66	**0.01**	**81.05**	**0.00**	**100**
20	p_2s_1	6.60	132.02	0.80	2.00	132.02	0.80	**0.08**	**133.09**	**0.00**	**100**
20	p_2s_2	11.82	70.50	3.27	3.20	70.50	3.27	**0.13**	**72.88**	**0.00**	**100**
20	p_2s_3	4.20	158.30	0.51	0.90	158.30	0.51	**0.01**	**159.11**	**0.00**	**100**
50	p_2s_1	76.81	312.99	0.50	4.50	312.99	0.50	**0.62**	**314.57**	**0.00**	**100**
50	p_2s_2	132.47	164.07	2.41	10.27	164.07	2.41	**1.81**	**168.11**	**0.00**	**100**
50	p_2s_3	34.39	382.90	0.32	2.55	382.90	0.32	**0.02**	**384.13**	**0.00**	**100**
100	p_2s_1	362.50	608.82	0.30	15.75	608.82	0.30	**0.86**	**610.64**	**0.00**	**100**
100	p_2s_2	555.26	319.70	1.97	35.42	319.70	1.97	**4.71**	**326.11**	**0.00**	**100**
100	p_2s_3	142.42	765.08	0.24	5.55	765.08	0.24	**0.04**	**766.91**	**0.00**	**100**
300	p_2s_1	3497.57	1790.61	0.16	186.79	1790.61	0.16	**0.96**	**1793.52**	**0.00**	**100**
300	p_2s_2	3915.99	952.99	1.02	424.83	952.99	1.02	**6.55**	**962.77**	**0.00**	**100**
300	p_2s_3	1303.26	2243.85	0.16	48.61	2243.85	0.16	**0.17**	**2247.39**	**0.00**	**100**
500	p_2s_1	10829.96	2960.65	0.13	803.56	2960.65	0.13	**0.89**	**2964.57**	**0.00**	**100**
500	p_2s_2	9463.63	1577.57	0.63	1899.40	1577.57	0.63	**5.66**	**1587.50**	**0.00**	**100**
500	p_2s_3	3551.90	3697.54	0.11	158.00	3697.54	0.11	**0.13**	**3701.79**	**0.00**	**100**

Chapter 4

The $P_m|s_j, B|C_{max}$ problem

Problem $P_m|s_j, B|C_{\max}$ is very similar to problem $1|s_j, B|C_{\max}$. The only difference is the possibility of using more than one machine to process the batches. When defining problem $P_m|s_j, B|C_{\max}$, we consider all aspect presented for $1|s_j, B|C_{\max}$, where the $j \in J$ have non-identical processing times p_j and sizes s_j. This problem also assumes that the batches must be assigned to a specific machine $M := \{1, \ldots, n_M\}$. All machines are identical, and each one has its own processing time, defined by the time of the last batch processed on the machine. The objective is again to minimize the makespan ($C_{\max}$), now defined as the time required to finish processing the last machine.

The Figure 4.1 shows an example of a solution of problem $P_m|s_j, B|C_{max}$ with three parallel machines. This is the same instance used in Figure 3.2, but now considering the case of parallel machines. If a new machine M_4 is made available in this example, the solution will stay with the same makespan, defined by j_1, i.e., the availability of new parallel machines does not always lead to the improvement of the optimal solution. This occurs when the value of C_{max} is equal to the longest processing time $max\{p_j\}$.

The decision-making for the problem $P_m|s_j, B|C_{max}$ consists of two parts: design of the batches and their scheduling in the processing machines. An approach to solve the problem could suggest the division of this problem into two distinct phases, in which the first one finds the optimal solution for the dimensioning of the batches without considering the parallel machines and the second allocates these batches in the parallel processing machines. However, this approach does not ensure the optimal solution to problem $P_m|s_j, B|C_{max}$. Figure 4.2 shows an example in which 4.2(a) uses the, the two-phases approach described above. The Figure 4.2(b) shows that the solution found in 4.2(a) is not optimal, since it is possible to modify the batches and find a better solution.

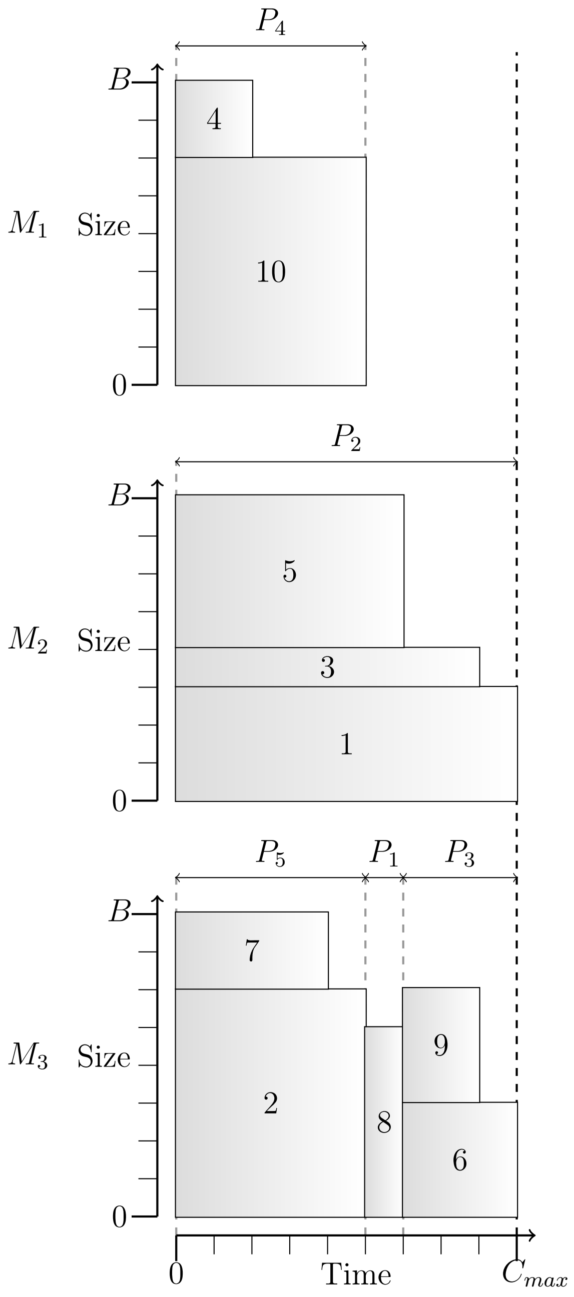
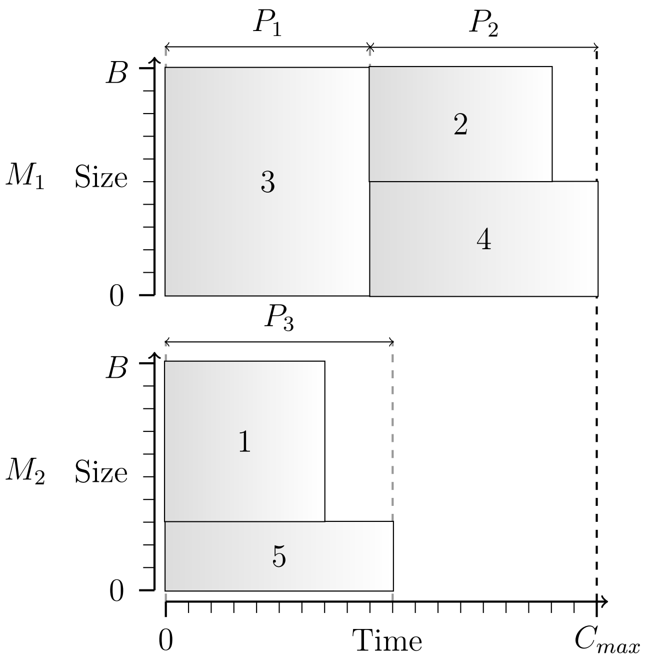
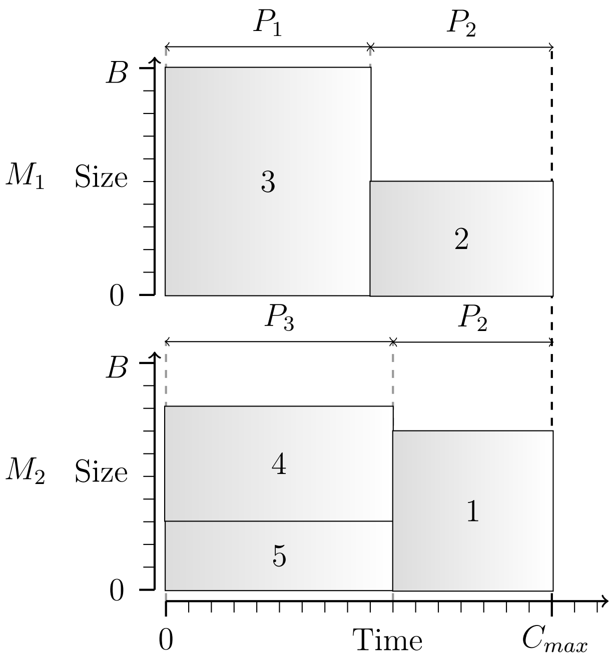


Figure 4.1: An example of solution for $P_m|s_j, B|C_{\max}$ with three parallel machines.



(a) A solution for problem $P_m|s_j, B|C_{max}$ obtained by the two-phase approach, where the first one finds the optimal solution $\sum P_i = 29$ for the design of the batches without considering the parallel machines. Then the optimal allocation of these batches in the processing machines is made. This solution has $C_{max} = 19$.



(b) The solution found in 4.2(a) is not optimal, since it is possible to modify the batches and get the optimal makepan $C_{max} = 17$.

Figure 4.2: Example of solutions for the $P_m|s_j, B|C_{max}$ problem with two-phase approach.

4.1 Literature formulation

Consider the following decision variables, for all $j \in J$, $k \in K$, and $m \in M$:

$$x_{jkm} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k \text{ processed in machine } m; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

$$P_{km} : \text{time to process batch } k \text{ in machine } m. \tag{4.2}$$

$$C_{max} : \text{the makespan.} \tag{4.3}$$

In Chang *et al.* [7] the following MILP formulation is proposed for $P_m|s_j, B|C_{\max}$:

$$\begin{aligned}
&(\text{MILP}_2) \quad \min\ C_{\max}, && && (4.4)\\
&\sum_{k \in K} \sum_{m \in M} x_{jkm} = 1, && \forall j \in J, && (4.5)\\
&\sum_{j \in J} \sum_{m \in M} s_j x_{jkm} \leq B, && \forall k \in K, && (4.6)\\
&P_{km} \geq p_j x_{jkm}, && \forall j \in J, \forall k \in K, \forall m \in M, && (4.7)\\
&C_{\max} \geq \sum_{k \in K} P_{km}, && \forall m \in M, && (4.8)\\
&x_{jkm} \in \{0,1\}, && \forall j \in J, \forall k \in K, \forall m \in M. && (4.9)
\end{aligned}$$

The objective function (4.4) minimizes the makespan. Constraints (4.5) and (4.6) ensure that each job is assigned to a single batch and a single machine, respecting the capacity of the machine. Constraints (4.7) determine the processing time of batch k in machine m. Constraints (4.8) determine the makespan, which is given by the longest sum of the processing times of all batches, among all machines. Note that formulation (MILP$_2$) takes into account that $n_K = n_J$, and therefore, all batches assigned to all machines on a given solution can be indexed by distinct indexes.

Note that constraints (4.6) take into account the fact that, although we have batches indexed by a given k, corresponding to all machines, a job can only be assigned to one of them, because of constraints (4.5). Therefore, a job j is only assigned to a unique pair (k, m).

Similar to model (MILP$_1$) for problem $1|s_j, B|C_{max}$, (MILP$_2$) can be considered highly symmetrical concerning the order in which the batches are scheduled in each one of the parallel machines. This is because the same solution can be represented in different ways, just by changing the sequence order of the batches, as well as the case explored in Section 3.2.1. Also, (MILP$_2$) presents the variable x_{jkm} with index m that represents the allocation of batches to the processing machines. This condition replicates all variables n_M times when compared to the variables x_{jk} of (MILP$_1$), which increases the number of symmetric solutions.

4.2 Symmetry breaking approach

Considering now problem $P_m|s_j, B|C_{\max}$, we note that the same symmetry mentioned above for problem $1|s_j, B|C_{\max}$ is also present in this problem, and we can use a similar symmetry breaking procedure to the one described above. For modeling problem $P_m|s_j, B|C_{\max}$, we initially note that variables x_{jkm} in (6.1) determine the design of the batches and also assign them to a specific machine. We propose the replacement of these variables with the binary variables x_{jk}, which determine only the design of the batches, as defined in (3.1), and the binary variables y_{km}, which determine whether or not batch k is processed in machine m, for all j and m. This replacement significantly reduces the number of binary variables. In (MILP$_2$), the number of variables x_{jkm} is equal to $(n_J^2)(n_M)$. In our proposed model, we will have $n_J(n_J+1)/2$ variables x_{jk}, as in (MILP$_1^+$), plus $n_J n_M$ variables y_{km}. Furthermore, using the same procedure described for $1|s_j, B|C_{\max}$ to eliminate equivalent solutions from the feasible set of the $P_m|s_j, B|C_{\max}$, we next propose a new formulation for this problem.

$$
\begin{aligned}
(\text{MILP}_2^+)\quad & \min\ C_{\max}, && && (4.10)\\
& \sum_{k\in K:k\geq j} x_{jk} = 1, && \forall j \in J, && (4.11)\\
& \sum_{j\in J:j\leq k} s_j x_{jk} \leq B x_{kk}, && \forall k \in K, && (4.12)\\
& x_{jk} \leq x_{kk}, && \forall j \in J, \forall k \in K, && (4.13)\\
& x_{kk} \leq \sum_{m\in M} y_{km}, && \forall k \in K, && (4.14)\\
& C_m \geq \sum_{k\in K} p_k y_{km}. && \forall m \in M, && (4.15)\\
& C_{\max} \geq C_m && \forall m \in M, && (4.16)\\
& x_{jk} \in \{0,1\} && \forall j \in J, \forall k \in K : j \leq k. && (4.17)
\end{aligned}
$$

The objective function (4.10) minimizes the makespan given by the latest time to finish processing all batches in all machines. Constraints (4.11)–(4.13) are exactly the same as (3.13)–(3.15) used in the model (MILP$_1^+$). Therefore, constraints (4.11) determine that each job j is assigned to a single batch k, such that $k \geq j$. Constraints (4.12) determine that the batches do not exceed the capacity of the machine. They also ensure that each batch k is used if and only if job k is assigned to it. Constraints (4.13) are redundant together with (4.12), but are included to strengthen the linear relaxation of the model. Constraints (4.14) ensure that each used batch is assigned to a machine. Constraints (4.15) and (4.16) determine the makespan.

This approach eliminates many of the symmetries found in the model (MILP$_2$), but we are aware that we are not dealing with the symmetries in the allocation of batches in parallel machines, i.e., the model considers all the permutations of batches between parallel machines as different solutions. These symmetries were investigated but until the present moment no approaches have been found that bring satisfactory computational results.

4.3 Arc Flow approach

We apply to problem $P_m|s_j, B|C_{\max}$ the same arc-flow model proposed for problem $1|s_j, B|C_{\max}$ in the Section 3.3. This problem is also formulated as the problem of determining the minimum flow from node 0 to node B, for all the arc-flow structures. However, two new sets of constraints were added to satisfy batch allocation on parallel machines. A new variable $w_{t,m}$ is created to determine the number of batches with processing time P_t that will be allocated on the machine m. Also, the objective function has been reformulated.

Our new formulation is presented below:

$f_{i,j,t}$: flow on *job arc* $(i,j) \in A^J$ in arc-flow structure t. The variable indicates the quantity of batches created with position i occupied by jobs with size $j-i$.

$y_{i,j,t}$: flow on the *loss arc* $(i,B) \in A^L$ in arc-flow structure t.

v_t : flow on the *feedback arc* in arc-flow structure t. The variable indicates the number of batches required with processing time P_t.

$z_{c,t}$: number of jobs with size c, not allocated in the batches with processing time smaller than or equal to P_t. Theses jobs are allowed to be allocated in the batches with processing time P_{t+1}.

$w_{t,m}$: number of batches with processing time P_t, allocated to machine m.

$$(\text{FLOW}_2) \quad \min\ C_{max} \tag{4.18}$$

$$\left(\sum_{(i,j)\in A^J} f_{i,j,t} + \sum_{(i,j)\in A^L} y_{i,j,t}\right) - \left(\sum_{(j,i)\in A^J} f_{j,i,t} + \sum_{(j,i)\in A^L} y_{j,i,t}\right) = \begin{cases} -v_t & \text{if } j = 0; \\ v_t & \text{if } j = B; \\ 0 & \text{if } 0 < j < B. \end{cases} \qquad t \in T \tag{4.19}$$

$$NT_{c,t} - \sum_{\substack{(i,j)\in A^J: \\ j-i=c}} f_{i,j,t} = \begin{cases} z_{c,t} & \text{if } t = 1; \\ -z_{c,t-1} & \text{if } t = \delta; \\ z_{c,t} - z_{c,t-1} & \text{if } 1 < t < \delta. \end{cases} \qquad c \in \{1..B\} \tag{4.20}$$

$$\sum_{m \in M} w_{t,m} \geq v_t \qquad t \in T \qquad (4.21)$$

$$\sum_{t \in T} P_t w_{t,m} \leq C_{max} \qquad m \in M \qquad (4.22)$$

$$f_{i,j,t} \leq min(NJ_t, NT^{+}_{j-i,t}),\ f_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^J \qquad (4.23)$$

$$v_t \leq NJ_t,\ v_t \in \mathbb{Z} \qquad t \in T \qquad (4.24)$$

$$y_{i,j,t} \leq NJ_t,\ y_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^L \qquad (4.25)$$

$$z_{c,t} \leq NT^{+}_{c,t},\ z_{c,t} \in \mathbb{Z} \qquad t \in T : t < \delta, c \in \{1..B\} \qquad (4.26)$$

$$w_{t,m} \in \mathbb{Z} \qquad t \in T, m \in M \qquad (4.27)$$

The objective function (4.18) minimizes the makespan. The set of flow conservation constraints are defined by constraints (4.19). Constraints (4.20) ensure that all jobs are assigned and also control the number of jobs to be assigned to each arc-flow structure. Constraints (4.21) ensure that all batches used are assigned to a machine. Constraints (4.22) determine the makespan as the time required to finish processing the last batch on all machines. Constraints (4.23–4.27) define the domains of the variables and their respective upper bounds. We emphasize that (4.21) and (4.22) are the constraints that make it possible for the arc-flow model to handle batch allocation on parallel machines.

4.4 Computational results

The models presented in this chapter were compared through computational tests performed with two sets of instances. The first set was created by the authors of the paper Chen *et al.* [1], which kindly sent us to use in our work. The second set is proposed in this thesis with parameters that make the instances more difficult. In all tests, we use the `CPLEX` version 12.7.1.0, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.70GHz `Intel Quad-Core Xeon E5-2697 v2` processor and 64GB of RAM. The computational time to solve each instance was limited in 1800 seconds.

4.4.1 Instances from the literature

The set of test instances for problem $P_m|s_j, B|C_{\max}$ is the same considered in Chen *et al.* [1] for the $1|s_j, B|C_{\max}$ problem. For each job j, an integer processing time p_j and an integer job size s_j were generated from the respective uniform distribution depicted in Table 4.1. In total, 4200 instances were generated, 100 for each of the 42 different combinations of number and size of the jobs. We test each instance with three different numbers of parallel machines.

Table 4.1: Parameter settings.

Number of jobs (n_J)	Processing time (p_J)	Jobs size	Machine capacity (B)	Parallel machines (n_M)
10, 20, 50, 100	p_1: [1, 10]	s_1: [1, 10]	$B = 10$	2, 4, 8
200, 300, 500	p_2: [1, 20]	s_2: [2, 4]		
		s_3: [4, 8]		

We present in Table 4.2–4.4 comparison results among the two models proposed in this work and another one from the literature included in this thesis. All values presented are the average results computed over the instances of the same configuration, as described in Table 4.1.

The comparative tests clearly show that formulation (FLOW) is superior to (MILP) and (MILP$^+$), especially when the number of jobs increases. Model (FLOW) did not prove the optimality of only one instance from the set of test problems. For instances with 20 jobs or less, (MILP$^+$) can solve some instances in less computational time than (FLOW), but the difference between times is always a fraction of a second. Additionally, the duality gaps shown for (MILP) reveal the difficulty in obtaining good lower bounds.

Unlike what we have with models (MILP) and (MILP$^+$), the number of variables in (FLOW) does not grow when the number of jobs increases. Moreover, the flow graph does not change in this case. Only the bounds on the variables change. The flow graphs of two distinct instances will be the same if the settings in the parameters Processing Time, Job Size and Machine Capacity are the same. In fact, this is a very important characteristic of the flow approach. We finally note that the computational time to construct the graphs for the flow formulation was not considered in these times. However, the maximum time to construct a graph for any instance in our experiments was 0.008 second.

The results show that instances of configuration s_2 require more computational time and are more difficult compared to the other instances for all formulations. The reason for this is the small sizes of the jobs when compared to the machine capacity, which allows more combinations of assignment to a batch.

4.4.2 New instances proposed

For the computational experiments on problem $P_m|s_j, B|C_{\max}$, we used the same set of instances tested in the Section 3.4.2 for problem $1|s_j, B|C_{\max}$. In addition, three new categories were included corresponding to the numbers of parallel machines: 2, 4 and 8 machines, and each instance was tested for the three different numbers of parallel machines. The sizes, the processing times and the release times of the jobs

Table 4.2: Computational results for $P_m|s_j.B|C_{\max}$ - 2 parallel machines.

Instance		(MILP)			(MILP$^+$)			(FLOW)		
Jobs	Type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
					2 parallel machines					
10	p_1s_1	**18.76**	0.13	**0.00**	**18.76**	**0.01**	**0.00**	**18.76**	0.02	**0.00**
10	p_1s_2	**11.03**	0.05	**0.00**	**11.03**	**0.02**	**0.00**	**11.03**	**0.02**	**0.00**
10	p_1s_3	**22.13**	0.19	**0.00**	**22.13**	0.01	**0.00**	**22.13**	**0.00**	**0.00**
10	p_2s_1	**34.50**	0.12	**0.00**	**34.50**	**0.01**	**0.00**	**34.50**	0.03	**0.00**
10	p_2s_2	**21.71**	0.05	**0.00**	**21.71**	**0.02**	**0.00**	**21.71**	0.03	**0.00**
10	p_2s_3	**40.87**	0.17	**0.00**	**40.87**	**0.01**	**0.00**	**40.87**	**0.01**	**0.00**
20	p_1s_1	**34.27**	1308.41	5.54	**34.27**	**0.03**	**0.00**	**34.27**	0.04	**0.00**
20	p_1s_2	**18.83**	884.08	8.16	**18.83**	0.11	**0.00**	**18.83**	**0.04**	**0.00**
20	p_1s_3	**42.13**	1412.74	6.27	**42.13**	0.02	**0.00**	**42.13**	**0.01**	**0.00**
20	p_2s_1	**66.79**	1287.70	4.35	**66.79**	**0.03**	**0.00**	**66.79**	0.09	**0.00**
20	p_2s_2	**36.87**	651.70	7.05	**36.87**	0.15	**0.00**	**36.87**	**0.09**	**0.00**
20	p_2s_3	**79.82**	1395.83	5.60	**79.82**	0.02	**0.00**	**79.82**	**0.01**	**0.00**
50	p_1s_1	83.07	-	58.36	**82.30**	2.48	**0.00**	**82.30**	**0.08**	**0.00**
50	p_1s_2	46.56	-	59.68	**43.94**	529.33	0.52	**43.94**	**0.07**	**0.00**
50	p_1s_3	101.74	-	60.69	**101.30**	0.02	**0.00**	**101.30**	**0.01**	**0.00**
50	p_2s_1	159.08	-	61.30	**157.52**	5.12	**0.00**	**157.52**	**0.33**	**0.00**
50	p_2s_2	88.96	-	62.44	**84.32**	478.37	0.19	**84.32**	**0.55**	**0.00**
50	p_2s_3	192.95	-	64.02	**192.34**	0.03	**0.00**	**192.34**	**0.02**	**0.00**
100	p_1s_1	171.60	-	87.71	**159.78**	192.10	0.07	**159.78**	**0.11**	**0.00**
100	p_1s_2	98.19	-	86.49	**85.56**	1743.59	1.73	**85.56**	**0.10**	**0.00**
100	p_1s_3	206.66	-	86.52	**198.75**	0.15	**0.00**	**198.75**	**0.01**	**0.00**
100	p_2s_1	328.38	-	89.47	**305.58**	84.36	0.02	**305.58**	**0.42**	**0.00**
100	p_2s_2	188.69	-	88.60	163.39	1770.79	1.21	**163.31**	**1.58**	**0.00**
100	p_2s_3	398.94	-	89.28	**383.73**	0.20	**0.00**	**383.73**	**0.03**	**0.00**
200	p_1s_1				314.93	332.53	0.05	**314.92**	**0.07**	**0.00**
200	p_1s_2				167.44	-	1.58	**166.97**	**0.15**	**0.00**
200	p_1s_3				**393.36**	79.14	0.01	**393.36**	**0.02**	**0.00**
200	p_2s_1				599.00	495.03	0.05	**598.96**	**0.67**	**0.00**
200	p_2s_2				320.16	-	1.52	**318.85**	**3.43**	**0.00**
200	p_2s_3				**752.78**	42.39	0.00	**752.78**	**0.05**	**0.00**
300	p_1s_1				464.59	639.71	0.08	**464.54**	**0.10**	**0.00**
300	p_1s_2				250.62	-	1.89	**248.06**	**0.14**	**0.00**
300	p_1s_3		Unperformed		**587.49**	241.24	0.02	**587.49**	**0.02**	**0.00**
300	p_2s_1				897.09	764.46	0.05	**897.00**	**0.57**	**0.00**
300	p_2s_2				487.55	-	2.09	**481.61**	**3.25**	**0.00**
300	p_2s_3				**1123.96**	274.67	0.02	**1123.96**	**0.09**	**0.00**
500	p_1s_1				772.54	1084.33	0.11	**772.38**	**0.11**	**0.00**
500	p_1s_2				421.92	-	1.98	**415.76**	**0.17**	**0.00**
500	p_1s_3				**975.15**	382.95	0.02	**975.15**	**0.01**	**0.00**
500	p_2s_1				1483.02	1365.87	0.09	**1482.58**	**0.59**	**0.00**
500	p_2s_2				806.24	-	2.10	**794.00**	**2.78**	**0.00**
500	p_2s_3				**1851.16**	488.38	0.01	**1851.16**	**0.06**	**0.00**

Table 4.3: Computational results for $P_m|s_j.B|C_{\max}$ - 4 parallel machines.

Instance		(MILP)			(MILP$^+$)			(FLOW)		
jobs	type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
					4 parallel machines					
10	p_1s_1	**10.87**	0.16	**0.00**	**10.87**	**0.02**	**0.00**	**10.87**	**0.02**	**0.00**
10	p_1s_2	**9.49**	0.10	**0.00**	**9.49**	**0.01**	**0.00**	**9.49**	**0.01**	**0.00**
10	p_1s_3	**12.18**	0.25	**0.00**	**12.18**	0.02	**0.00**	**12.18**	**0.01**	**0.00**
10	p_2s_1	**20.26**	0.16	**0.00**	**20.26**	**0.02**	**0.00**	**20.26**	0.03	**0.00**
10	p_2s_2	**18.68**	0.11	**0.00**	**18.68**	**0.01**	**0.00**	**18.68**	0.02	**0.00**
10	p_2s_3	**22.67**	0.23	**0.00**	**22.67**	0.02	**0.00**	**22.67**	**0.01**	**0.00**
20	p_1s_1	**17.47**	1316.19	8.11	**17.47**	**0.05**	**0.00**	**17.47**	0.06	**0.00**
20	p_1s_2	**10.43**	56.14	0.49	**10.43**	0.32	**0.00**	**10.43**	**0.05**	**0.00**
20	p_1s_3	**21.29**	1629.93	11.78	**21.29**	0.03	**0.00**	**21.29**	**0.01**	**0.00**
20	p_2s_1	**33.95**	1122.49	5.29	**33.95**	**0.07**	**0.00**	**33.95**	0.14	**0.00**
20	p_2s_2	**20.51**	92.62	0.64	**20.51**	0.35	**0.00**	**20.51**	**0.14**	**0.00**
20	p_2s_3	**40.21**	1731.24	12.27	**40.21**	0.05	**0.00**	**40.21**	**0.02**	**0.00**
50	p_1s_1	42.69	-	70.03	**41.43**	2.54	**0.00**	**41.43**	**0.11**	**0.00**
50	p_1s_2	23.76	-	58.83	**22.18**	269.31	0.56	**22.18**	**0.26**	**0.00**
50	p_1s_3	51.85	-	71.91	**50.90**	0.05	**0.00**	**50.90**	**0.01**	**0.00**
50	p_2s_1	81.23	-	71.19	**78.97**	0.90	**0.00**	**78.97**	**0.80**	**0.00**
50	p_2s_2	45.55	-	57.37	**42.38**	283.70	0.19	**42.38**	**1.41**	**0.00**
50	p_2s_3	97.89	-	73.39	**96.40**	0.07	**0.00**	**96.40**	**0.04**	**0.00**
100	p_1s_1	93.06	-	93.44	**80.09**	82.33	0.05	**80.09**	**0.18**	**0.00**
100	p_1s_2	50.26	-	81.80	43.06	1409.33	1.67	**43.04**	**0.26**	**0.00**
100	p_1s_3	110.60	-	92.94	**99.64**	0.55	**0.00**	**99.64**	**0.02**	**0.00**
100	p_2s_1	177.17	-	93.54	**153.03**	51.98	0.02	**153.03**	**1.83**	**0.00**
100	p_2s_2	96.18	-	86.63	82.03	1679.11	1.32	**81.88**	**3.12**	**0.00**
100	p_2s_3	213.47	-	93.38	**192.11**	0.52	**0.00**	**192.11**	**0.05**	**0.00**
200	p_1s_1				157.71	209.19	0.06	**157.70**	**0.16**	**0.00**
200	p_1s_2				84.05	1788.54	1.69	**83.67**	**0.53**	**0.00**
200	p_1s_3				**196.93**	38.03	0.01	**196.93**	**0.02**	**0.00**
200	p_2s_1				299.78	396.85	0.06	**299.75**	**21.23**	**0.00**
200	p_2s_2				160.95	-	1.94	**159.68**	**31.02**	**0.01**
200	p_2s_3				**376.64**	59.36	0.01	**376.64**	**0.07**	**0.00**
300	p_1s_1				232.56	422.79	0.09	**232.52**	**0.17**	**0.00**
300	p_1s_2				126.22	-	2.40	**124.28**	**0.33**	**0.00**
300	p_1s_3		Unperformed		**293.99**	146.32	0.03	**293.99**	**0.02**	**0.00**
300	p_2s_1				448.84	568.62	0.06	**448.79**	**1.19**	**0.00**
300	p_2s_2				244.96	-	2.49	**241.07**	**21.59**	**0.00**
300	p_2s_3				**562.24**	230.89	0.02	**562.24**	**0.11**	**0.00**
500	p_1s_1				386.62	1009.64	0.15	**386.47**	**0.17**	**0.00**
500	p_1s_2				211.91	-	2.31	**208.12**	**0.40**	**0.00**
500	p_1s_3				**487.84**	266.44	0.03	**487.84**	**0.02**	**0.00**
500	p_2s_1				741.93	1306.98	0.12	**741.56**	**1.91**	**0.00**
500	p_2s_2				405.30	-	2.58	**397.30**	**134.68**	**0.01**
500	p_2s_3				**925.84**	345.92	0.02	**925.84**	**0.07**	**0.00**

Table 4.4: Computational results for $P_m|s_j.B|C_{\max}$ - 8 parallel machines.

Instance		(MILP)			(MILP^{+})			(FLOW)		
jobs	type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
					8 parallel machines					
10	p_1s_1	**9.54**	0.23	**0.00**	**9.54**	**0.01**	**0.00**	**9.54**	0.02	**0.00**
10	p_1s_2	**9.49**	0.25	**0.00**	**9.49**	**0.01**	**0.00**	**9.49**	0.02	**0.00**
10	p_1s_3	**9.42**	0.33	**0.00**	**9.42**	**0.01**	**0.00**	**9.42**	**0.01**	**0.00**
10	p_2s_1	**18.55**	0.21	**0.00**	**18.55**	**0.01**	**0.00**	**18.55**	0.02	**0.00**
10	p_2s_2	**18.68**	0.24	**0.00**	**18.68**	**0.01**	**0.00**	**18.68**	0.02	**0.00**
10	p_2s_3	**18.27**	0.34	**0.00**	**18.27**	**0.01**	**0.00**	**18.27**	**0.01**	**0.00**
20	p_1s_1	**10.51**	276.62	2.44	**10.51**	0.09	**0.00**	**10.51**	**0.06**	**0.00**
20	p_1s_2	**9.81**	2.76	**0.00**	**9.81**	0.07	**0.00**	**9.81**	**0.03**	**0.00**
20	p_1s_3	11.61	760.27	7.24	**11.60**	0.15	**0.00**	**11.60**	**0.01**	**0.00**
20	p_2s_1	**20.76**	328.01	3.34	**20.76**	**0.13**	**0.00**	**20.76**	0.18	**0.00**
20	p_2s_2	**19.52**	2.80	**0.00**	**19.52**	0.08	**0.00**	**19.52**	**0.04**	**0.00**
20	p_2s_3	22.31	958.29	8.28	**22.30**	0.26	**0.00**	**22.30**	**0.04**	**0.00**
50	p_1s_1	22.30	-	55.90	**20.96**	2.99	**0.00**	**20.96**	**0.25**	**0.00**
50	p_1s_2	12.83	1783.20	27.02	**11.77**	850.12	4.12	**11.77**	**0.39**	**0.00**
50	p_1s_3	26.78	-	62.67	**25.71**	0.10	**0.00**	**25.71**	**0.02**	**0.00**
50	p_2s_1	42.41	-	55.68	**39.72**	**1.25**	**0.00**	**39.72**	2.47	**0.00**
50	p_2s_2	24.41	1775.39	28.90	22.46	1198.77	3.91	**22.45**	**11.60**	**0.00**
50	p_2s_3	50.33	-	61.21	**48.45**	0.17	**0.00**	**48.45**	**0.08**	**0.00**
100	p_1s_1	59.57	-	96.84	**40.34**	51.78	0.05	**40.34**	**0.19**	**0.00**
100	p_1s_2	28.80	-	84.72	21.82	872.63	2.04	**21.75**	**1.49**	**0.00**
100	p_1s_3	69.72	-	98.03	**50.07**	0.22	**0.00**	**50.07**	**0.03**	**0.00**
100	p_2s_1	123.38	-	97.70	**76.81**	59.45	0.01	**76.81**	**10.88**	**0.00**
100	p_2s_2	57.82	-	94.95	41.34	1251.21	1.45	**41.23**	**37.99**	**0.03**
100	p_2s_3	139.99	-	98.19	**96.34**	0.81	**0.00**	**96.34**	**0.11**	**0.00**
200	p_1s_1				79.13	213.02	0.12	**79.10**	**0.15**	**0.00**
200	p_1s_2				42.41	1599.90	1.97	**42.10**	**0.95**	**0.00**
200	p_1s_3				**98.74**	2.72	**0.00**	**98.74**	**0.02**	**0.00**
200	p_2s_1				150.18	341.94	0.08	**150.14**	**54.67**	**0.01**
200	p_2s_2				81.27	1796.74	2.64	**80.08**	**141.71**	**0.04**
200	p_2s_3				**188.53**	21.83	0.01	**188.53**	**0.14**	**0.00**
300	p_1s_1				116.58	480.57	0.16	**116.51**	**18.20**	**0.01**
300	p_1s_2				63.40	1779.89	2.44	**62.39**	**0.57**	**0.00**
300	p_1s_3		Unperformed		**147.25**	94.40	0.03	**147.25**	**0.02**	**0.00**
300	p_2s_1				224.76	641.81	0.12	**224.61**	**2.84**	**0.00**
300	p_2s_2				123.61	-	3.20	**120.78**	**358.94**	**0.13**
300	p_2s_3				**281.31**	115.95	0.02	**281.31**	**0.31**	**0.00**
500	p_1s_1				193.74	1067.41	0.29	**193.53**	**0.24**	**0.00**
500	p_1s_2				106.63	1787.69	2.72	**104.38**	**1.67**	**0.00**
500	p_1s_3				**244.11**	134.96	0.03	**244.11**	**0.04**	**0.00**
500	p_2s_1				371.50	1136.26	0.20	**371.03**	**2.08**	**0.00**
500	p_2s_2				203.42	-	2.83	**198.96**	**376.70**	**0.09**
500	p_2s_3				**463.16**	189.74	0.02	**463.16**	**0.24**	**0.00**

were randomly selected in the ranges shown in Table 4.5.

Table 4.5: Parameter settings for set of new instances proposed for $P_m|s_j, B|C_{\max}$.

Number of jobs (n_J)	Processing time (p_j)	Jobs size (s_j)	Machine capacity (B)	Parallel machines (n_M)
10, 50, 100, 500, 1000, 5000	p_1: [1, 20] p_2: [1, n_J]	s_1: [1, B] s_2: [$0.2B$, $0.4B$] s_3: [$0.4B$, $0.8B$]	$20, 50, 100$	2, 4, 8

The comparative tests presented in Tables 4.6 – 4.14 maintain the patterns observed in Section 3.4.2 for the problem that does not consider parallel machines. Instances of type p_2 are more difficult for (FLOW), especially when B and the number of parallel machines increases. In the most difficult instances, presented in Table 4.14, (FLOW) was not able to find a single integer solution for most of the p_2 instances with more than 500 jobs. Model (MILP$_2^+$) presents better results for this type of instance, even if they have a gap close to 100% in some cases.

The FLOW model is superior for most instances of type p_1, even though no integer solution is found in some extreme cases. In easier instances presented in Table 4.14, with 2 parallel machines and $B = 20$, (FLOW) is able to find the optimal solution of all instances of type p_1. The tests show that increasing the number of parallel machines makes the instances more difficult for both models, as expected.

4.5 Considerations

Even if some symmetries have not been treated, the tests with the model (MILP$_2^+$) present good computational results. The symmetries relating to exchange batches between machines are not treated, i.e., if all batches processed in machine m_j are processed in machine m_k and vice versa, we will have a symmetric solution that is not treated in this model. We have tested some approaches to eliminate these symmetries, especially with lexical ordering proposed by Margot [10] and adapted to this problem, but the computational effort was not satisfactory. This is because the instances for this problem generally consider a very small number of parallel machines in comparison to the number of jobs.

Table 4.6: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 20$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 2$, and $B = 20$											
10	2	20	p_1s_1	**0.01**	**31.60**	**0.00**	**5**	0.04	**31.60**	**0.00**	**5**
10	2	20	p_1s_2	**0.02**	**25.00**	**0.00**	**5**	0.07	**25.00**	**0.00**	**5**
10	2	20	p_1s_3	**0.00**	**36.00**	**0.00**	**5**	0.01	**36.00**	**0.00**	**5**
50	2	20	p_1s_1	5.87	**158.40**	**0.00**	**5**	**1.54**	**158.40**	**0.00**	**5**
50	2	20	p_1s_2	196.31	**90.60**	**0.00**	**5**	**4.46**	**90.60**	**0.00**	**5**
50	2	20	p_1s_3	**0.02**	**187.00**	**0.00**	**5**	0.03	**187.00**	**0.00**	**5**
100	2	20	p_1s_1	**0.90**	**315.20**	**0.00**	**5**	1.45	**315.20**	**0.00**	**5**
100	2	20	p_1s_2	-	163.80	1.46	0	**13.71**	**163.40**	**0.00**	**5**
100	2	20	p_1s_3	0.09	**395.60**	**0.00**	**5**	**0.03**	**395.60**	**0.00**	**5**
500	2	20	p_1s_1	1455.63	1404.00	0.14	1	**2.20**	**1402.80**	**0.00**	**5**
500	2	20	p_1s_2	-	823.40	3.75	0	**53.49**	**797.60**	**0.00**	**5**
500	2	20	p_1s_3	4.61	**1935.20**	**0.00**	**5**	**0.21**	**1935.20**	**0.00**	**5**
1000	2	20	p_1s_1	-	2841.60	0.24	0	**2.28**	**2837.80**	**0.00**	**5**
1000	2	20	p_1s_2	-	10411.80	84.93	0	**23.05**	**1574.40**	**0.00**	**5**
1000	2	20	p_1s_3	372.40	**3847.00**	0.01	4	**0.09**	**3847.00**	**0.00**	**5**
5000	2	20	p_1s_1	-	52548.60	100.00	0	**1.61**	**14019.00**	**0.00**	**5**
5000	2	20	p_1s_2	-	52520.40	100.00	0	**41.34**	**7867.80**	**0.00**	**5**
5000	2	20	p_1s_3	-	45738.60	50.97	0	**0.14**	**19054.40**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 2$, and $B = 20$											
10	2	20	p_2s_1	**0.00**	**18.60**	**0.00**	**5**	0.06	**18.60**	**0.00**	**5**
10	2	20	p_2s_2	**0.03**	**13.00**	**0.00**	**5**	0.08	**13.00**	**0.00**	**5**
10	2	20	p_2s_3	**0.01**	**21.40**	**0.00**	**5**	**0.01**	**21.40**	**0.00**	**5**
50	2	20	p_2s_1	**0.29**	**345.20**	**0.00**	**5**	3.41	**345.20**	**0.00**	**5**
50	2	20	p_2s_2	356.27	**212.00**	**0.00**	**5**	**85.10**	**212.00**	**0.00**	**5**
50	2	20	p_2s_3	**0.04**	**522.40**	**0.00**	**5**	0.07	**522.40**	**0.00**	**5**
100	2	20	p_2s_1	**4.20**	**1425.00**	**0.00**	**5**	100.63	**1425.00**	**0.00**	**5**
100	2	20	p_2s_2	-	810.40	0.87	**0**	-	**809.00**	**0.62**	**0**
100	2	20	p_2s_3	**0.17**	**1945.00**	**0.00**	**5**	0.23	**1945.00**	**0.00**	**5**
500	2	20	p_2s_1	1558.70	**34550.00**	**0.08**	1	**1299.25**	34582.00	0.18	**2**
500	2	20	p_2s_2	-	20191.40	5.65	**0**	-	**19334.80**	**1.48**	**0**
500	2	20	p_2s_3	**53.90**	**45941.80**	**0.00**	**5**	258.84	**45941.80**	**0.00**	**5**
1000	2	20	p_2s_1	-	136245.00	0.19	0	**1600.48**	**136243.40**	**0.12**	**1**
1000	2	20	p_2s_2	-	502970.20	85.00	**0**	-	**83794.33**	**9.20**	**0**
1000	2	20	p_2s_3	**110.80**	**185741.80**	**0.00**	**5**	257.97	**185741.80**	**0.00**	**5**
5000	2	20	p_2s_1	-	**12503592.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	20	p_2s_2	-	**12495060.40**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	20	p_2s_3	-	**4497072.80**	**0.00**	**0**	-	4497712.20	0.02	**0**

Table 4.7: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 20$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 4$, and $B = 20$											
10	4	20	p_1s_1	**0.01**	**19.00**	**0.00**	**5**	0.03	**19.00**	**0.00**	**5**
10	4	20	p_1s_2	**0.01**	**18.60**	**0.00**	**5**	0.04	**18.60**	**0.00**	**5**
10	4	20	p_1s_3	**0.01**	**19.60**	**0.00**	**5**	0.02	**19.60**	**0.00**	**5**
50	4	20	p_1s_1	**6.04**	**79.60**	**0.00**	**5**	27.46	**79.60**	**0.00**	**5**
50	4	20	p_1s_2	410.08	**45.60**	0.47	4	**8.37**	**45.60**	**0.00**	**5**
50	4	20	p_1s_3	0.08	**93.80**	**0.00**	**5**	**0.06**	**93.80**	**0.00**	**5**
100	4	20	p_1s_1	**2.41**	**157.80**	**0.00**	**5**	119.07	**157.80**	**0.00**	**5**
100	4	20	p_1s_2	1455.20	82.20	1.22	2	**24.13**	**81.80**	**0.00**	**5**
100	4	20	p_1s_3	0.20	**198.00**	**0.00**	**5**	**0.10**	**198.00**	**0.00**	**5**
500	4	20	p_1s_1	1464.10	702.60	0.20	1	**6.63**	**701.60**	**0.00**	**5**
500	4	20	p_1s_2	-	411.20	3.60	0	**97.22**	**399.20**	**0.00**	**5**
500	4	20	p_1s_3	6.43	**967.80**	**0.00**	**5**	**0.26**	**967.80**	**0.00**	**5**
1000	4	20	p_1s_1	-	1421.20	0.28	0	**6.78**	**1419.40**	**0.00**	**5**
1000	4	20	p_1s_2	-	10411.80	92.47	0	**765.81**	**787.60**	**0.06**	**3**
1000	4	20	p_1s_3	43.12	**1923.80**	**0.00**	**5**	**0.15**	**1923.80**	**0.00**	**5**
5000	4	20	p_1s_1	-	52548.60	100.00	0	**3.25**	**7009.80**	**0.00**	**5**
5000	4	20	p_1s_2	-	52520.40	100.00	0	**775.28**	**3934.20**	**0.01**	**3**
5000	4	20	p_1s_3	-	52524.60	81.86	0	**0.17**	**9527.60**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 4$, and $B = 20$											
10	4	20	p_2s_1	**0.01**	**10.80**	**0.00**	**5**	0.04	**10.80**	**0.00**	**5**
10	4	20	p_2s_2	**0.00**	**10.00**	**0.00**	**5**	0.05	**10.00**	**0.00**	**5**
10	4	20	p_2s_3	**0.01**	**11.40**	**0.00**	**5**	0.01	**11.40**	**0.00**	**5**
50	4	20	p_2s_1	**0.43**	**172.80**	**0.00**	**5**	8.81	**172.80**	**0.00**	**5**
50	4	20	p_2s_2	510.57	**106.20**	**0.00**	**5**	**261.78**	**106.20**	**0.00**	**5**
50	4	20	p_2s_3	**0.09**	**261.40**	**0.00**	**5**	0.11	**261.40**	**0.00**	**5**
100	4	20	p_2s_1	**20.20**	**713.00**	**0.00**	**5**	510.37	**713.00**	0.03	4
100	4	20	p_2s_2	-	**405.60**	**0.99**	**0**	-	**405.60**	1.12	**0**
100	4	20	p_2s_3	0.57	**972.80**	**0.00**	**5**	**0.41**	**972.80**	**0.00**	**5**
500	4	20	p_2s_1	1709.31	**17285.80**	**0.14**	**1**	**1658.60**	17357.20	0.56	**1**
500	4	20	p_2s_2	-	10079.40	5.50	**0**	-	**9906.60**	**3.82**	**0**
500	4	20	p_2s_3	493.26	**22971.00**	**0.00**	4	**402.28**	**22971.00**	**0.00**	**5**
1000	4	20	p_2s_1	-	**68180.60**	0.27	**0**	-	69855.00	**0.13**	**0**
1000	4	20	p_2s_2	-	502970.20	92.50	**0**	-	**38502.50**	**2.28**	**0**
1000	4	20	p_2s_3	**465.01**	**92871.20**	**0.00**	**5**	885.58	92872.60	0.00	3
5000	4	20	p_2s_1	-	**12503592.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	20	p_2s_2	-	**12495060.40**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	20	p_2s_3	-	**2248781.80**	**0.02**	**0**	-	2249805.80	0.06	**0**

Table 4.8: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 20$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 8$, and $B = 20$											
10	8	20	p_1s_1	**0.00**	**17.60**	**0.00**	**5**	0.05	**17.60**	**0.00**	**5**
10	8	20	p_1s_2	**0.01**	**18.60**	**0.00**	**5**	0.05	**18.60**	**0.00**	**5**
10	8	20	p_1s_3	**0.00**	**18.60**	**0.00**	**5**	0.02	**18.60**	**0.00**	**5**
50	8	20	p_1s_1	**0.75**	**40.00**	**0.00**	**5**	12.95	**40.00**	**0.00**	**5**
50	8	20	p_1s_2	371.22	**23.80**	0.87	4	**30.08**	**23.80**	**0.00**	**5**
50	8	20	p_1s_3	0.15	**47.20**	**0.00**	**5**	**0.15**	**47.20**	**0.00**	**5**
100	8	20	p_1s_1	**2.23**	**79.20**	**0.00**	**5**	170.59	**79.20**	**0.00**	**5**
100	8	20	p_1s_2	1756.47	41.60	2.38	1	**499.59**	**41.20**	**0.45**	**4**
100	8	20	p_1s_3	0.34	**99.20**	**0.00**	**5**	**0.17**	**99.20**	**0.00**	**5**
500	8	20	p_1s_1	1517.25	351.80	0.34	1	**704.92**	**351.20**	**0.06**	**4**
500	8	20	p_1s_2	-	208.00	4.70	0	**1542.16**	**200.20**	**0.40**	**1**
500	8	20	p_1s_3	8.40	**484.40**	**0.00**	**5**	**0.33**	**484.40**	**0.00**	**5**
1000	8	20	p_1s_1	-	711.40	0.34	0	**6.66**	**709.80**	**0.00**	**5**
1000	8	20	p_1s_2	-	10411.80	96.23	0	**1094.20**	**394.20**	**0.15**	**2**
1000	8	20	p_1s_3	125.75	**962.00**	**0.00**	**5**	**0.26**	**962.00**	**0.00**	**5**
5000	8	20	p_1s_1	-	52548.60	100.00	0	**368.80**	**3505.40**	**0.01**	**4**
5000	8	20	p_1s_2	-	52520.40	100.00	0	**1201.02**	**1967.60**	**0.03**	**2**
5000	8	20	p_1s_3	-	33457.40	54.56	0	**0.32**	**4764.00**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 8$, and $B = 20$											
10	8	20	p_2s_1	**0.00**	**9.40**	**0.00**	**5**	0.02	**9.40**	**0.00**	**5**
10	8	20	p_2s_2	**0.00**	**10.00**	**0.00**	**5**	0.04	**10.00**	**0.00**	**5**
10	8	20	p_2s_3	**0.00**	**9.20**	**0.00**	**5**	0.02	**9.20**	**0.00**	**5**
50	8	20	p_2s_1	**78.78**	**86.80**	**0.00**	**5**	496.99	**86.80**	0.25	4
50	8	20	p_2s_2	**768.31**	**56.20**	0.74	**4**	1008.26	**56.20**	**0.73**	**4**
50	8	20	p_2s_3	**0.32**	**131.00**	**0.00**	**5**	1.06	**131.00**	**0.00**	**5**
100	8	20	p_2s_1	**76.61**	**356.80**	**0.00**	**5**	797.35	358.60	0.58	3
100	8	20	p_2s_2	-	208.40	3.54	**0**	-	**206.60**	**2.95**	**0**
100	8	20	p_2s_3	**10.89**	**486.60**	**0.00**	**5**	14.64	**486.60**	**0.00**	**5**
500	8	20	p_2s_1	-	**8657.60**	**0.32**	**0**	-	8675.80	0.53	**0**
500	8	20	p_2s_2	-	5063.40	5.93	**0**	-	**4972.25**	**4.66**	**0**
500	8	20	p_2s_3	**1023.62**	**11487.20**	**0.02**	**3**	1261.86	11494.20	0.08	2
1000	8	20	p_2s_1	-	**34193.20**	**0.58**	**0**	-	No solution	Infinite	**0**
1000	8	20	p_2s_2	-	502970.20	96.25	**0**	-	**20346.50**	**6.47**	**0**
1000	8	20	p_2s_3	**1520.26**	**46446.80**	**0.03**	**1**	1566.09	46452.40	0.04	1
5000	8	20	p_2s_1	-	**12503592.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	20	p_2s_2	-	**12495060.40**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	20	p_2s_3	-	1124676.40	**0.04**	**0**	-	**1124219.50**	0.07	**0**

Table 4.9: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 50$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 2$, and $B = 50$											
10	2	50	p_1s_1	**0.01**	**34.40**	**0.00**	**5**	0.13	**34.40**	**0.00**	**5**
10	2	50	p_1s_2	**0.02**	**22.20**	**0.00**	**5**	0.11	**22.20**	**0.00**	**5**
10	2	50	p_1s_3	**0.00**	**55.20**	**0.00**	**5**	0.01	**55.20**	**0.00**	**5**
50	2	50	p_1s_1	**0.27**	**150.20**	**0.00**	**5**	9.07	**150.20**	**0.00**	**5**
50	2	50	p_1s_2	370.42	**84.60**	0.22	4	**48.46**	**84.60**	**0.00**	**5**
50	2	50	p_1s_3	**0.02**	**214.80**	**0.00**	**5**	0.07	**214.80**	**0.00**	**5**
100	2	50	p_1s_1	**2.49**	**280.40**	**0.00**	**5**	10.81	**280.40**	**0.00**	**5**
100	2	50	p_1s_2	-	159.80	1.62	0	**776.96**	**159.20**	**0.27**	**3**
100	2	50	p_1s_3	**0.09**	**373.60**	**0.00**	**5**	0.12	**373.60**	**0.00**	**5**
500	2	50	p_1s_1	-	1373.60	0.32	0	**94.69**	**1370.80**	**0.00**	**5**
500	2	50	p_1s_2	-	830.40	6.07	0	**1579.70**	**786.20**	**0.22**	**2**
500	2	50	p_1s_3	3.28	**2018.40**	**0.00**	**5**	**0.23**	**2018.40**	**0.00**	**5**
1000	2	50	p_1s_1	-	2767.80	0.53	0	**28.09**	**2758.80**	**0.00**	**5**
1000	2	50	p_1s_2	-	10332.20	85.01	0	**1623.97**	**1560.80**	**0.48**	**1**
1000	2	50	p_1s_3	47.25	**3950.80**	**0.00**	**5**	**0.18**	**3950.80**	**0.00**	**5**
5000	2	50	p_1s_1	-	52744.20	100.00	0	**182.83**	**13668.20**	**0.00**	**5**
5000	2	50	p_1s_2	-	52553.40	100.00	0	**1567.58**	**7899.80**	**0.22**	**1**
5000	2	50	p_1s_3	1561.67	19458.20	0.01	1	**1.01**	**19457.80**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 2$, and $B = 50$											
10	2	50	p_2s_1	**0.00**	**23.80**	**0.00**	**5**	0.04	**23.80**	**0.00**	**5**
10	2	50	p_2s_2	**0.02**	**13.00**	**0.00**	**5**	0.09	**13.00**	**0.00**	**5**
10	2	50	p_2s_3	**0.00**	**24.80**	**0.00**	**5**	0.00	**24.80**	**0.00**	**5**
50	2	50	p_2s_1	**0.46**	**390.60**	**0.00**	**5**	77.14	**390.60**	**0.00**	**5**
50	2	50	p_2s_2	**393.20**	**206.20**	**0.10**	**4**	1150.07	206.40	0.76	2
50	2	50	p_2s_3	**0.03**	**505.60**	**0.00**	**5**	0.09	**505.60**	**0.00**	**5**
100	2	50	p_2s_1	**21.79**	**1320.20**	**0.00**	**5**	1054.65	**1320.20**	0.13	3
100	2	50	p_2s_2	-	**820.20**	**1.22**	**0**	-	820.80	1.58	**0**
100	2	50	p_2s_3	**0.09**	**2125.60**	**0.00**	**5**	0.28	**2125.60**	**0.00**	**5**
500	2	50	p_2s_1	**1295.58**	**32706.80**	**0.14**	**2**	-	32843.50	0.29	0
500	2	50	p_2s_2	-	20240.00	5.78	**0**	-	**19788.25**	**3.85**	**0**
500	2	50	p_2s_3	**3.96**	**47399.40**	**0.00**	**5**	29.17	**47399.40**	**0.00**	**5**
1000	2	50	p_2s_1	**1700.55**	**131697.20**	0.69	**1**	1716.32	132847.50	**0.07**	**1**
1000	2	50	p_2s_2	-	504224.40	85.04	**0**	-	**98230.00**	**21.94**	**0**
1000	2	50	p_2s_3	**38.93**	**185558.20**	**0.00**	**5**	448.30	**185558.20**	**0.00**	4
5000	2	50	p_2s_1	-	**12573122.60**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	50	p_2s_2	-	**12546134.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	50	p_2s_3	1302.20	**4645926.20**	**0.00**	**3**	**1011.89**	4646106.80	0.00	**3**

Table 4.10: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 50$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 4$, and $B = 50$											
10	4	50	p_1s_1	**0.00**	**21.20**	**0.00**	**5**	0.15	**21.20**	**0.00**	**5**
10	4	50	p_1s_2	**0.00**	**19.80**	**0.00**	**5**	0.12	**19.80**	**0.00**	**5**
10	4	50	p_1s_3	**0.02**	**29.80**	**0.00**	**5**	**0.02**	**29.80**	**0.00**	**5**
50	4	50	p_1s_1	**0.46**	**75.40**	**0.00**	**5**	376.28	**75.40**	0.29	4
50	4	50	p_1s_2	**429.56**	**42.60**	**0.44**	**4**	1167.93	42.80	1.54	2
50	4	50	p_1s_3	**0.06**	**107.60**	**0.00**	**5**	0.10	**107.60**	**0.00**	**5**
100	4	50	p_1s_1	**1.29**	**140.40**	**0.00**	**5**	49.86	**140.40**	**0.00**	**5**
100	4	50	p_1s_2	-	80.20	1.76	0	**1433.91**	**80.00**	**1.24**	**2**
100	4	50	p_1s_3	0.19	**187.00**	**0.00**	**5**	**0.14**	**187.00**	**0.00**	**5**
500	4	50	p_1s_1	-	687.20	0.32	0	**1725.22**	**686.20**	**0.12**	**1**
500	4	50	p_1s_2	-	408.40	4.50	**0**	-	**394.60**	**0.83**	**0**
500	4	50	p_1s_3	8.76	**1009.40**	**0.00**	**5**	**0.35**	**1009.40**	**0.00**	**5**
1000	4	50	p_1s_1	-	1383.80	0.52	0	**234.11**	**1379.60**	**0.00**	**5**
1000	4	50	p_1s_2	-	10332.20	92.51	**0**	-	**786.80**	**1.35**	**0**
1000	4	50	p_1s_3	79.96	**1975.80**	**0.00**	**5**	**0.37**	**1975.80**	**0.00**	**5**
5000	4	50	p_1s_1	-	52744.20	100.00	0	**1103.19**	**6834.80**	**0.01**	**2**
5000	4	50	p_1s_2	-	52553.40	100.00	**0**	-	**3978.40**	**0.94**	**0**
5000	4	50	p_1s_3	1790.96	18218.40	16.28	1	**0.88**	**9729.20**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 4$, and $B = 50$											
10	4	50	p_2s_1	**0.01**	**13.00**	**0.00**	**5**	0.06	**13.00**	**0.00**	**5**
10	4	50	p_2s_2	**0.00**	**10.00**	**0.00**	**5**	0.10	**10.00**	**0.00**	**5**
10	4	50	p_2s_3	**0.01**	**13.00**	**0.00**	**5**	**0.01**	**13.00**	**0.00**	**5**
50	4	50	p_2s_1	**1.35**	**195.40**	**0.00**	**5**	461.05	**195.40**	0.10	4
50	4	50	p_2s_2	**431.81**	**103.40**	**0.19**	**4**	1505.28	103.80	1.97	1
50	4	50	p_2s_3	**0.08**	**253.00**	**0.00**	**5**	0.17	**253.00**	**0.00**	**5**
100	4	50	p_2s_1	**63.97**	**660.20**	**0.00**	**5**	1506.97	663.20	0.71	1
100	4	50	p_2s_2	-	**414.60**	**2.31**	**0**	-	419.00	3.68	**0**
100	4	50	p_2s_3	**0.24**	**1062.80**	**0.00**	**5**	0.59	**1062.80**	**0.00**	**5**
500	4	50	p_2s_1	**1524.31**	**16358.00**	**0.17**	**1**	-	16472.00	0.59	0
500	4	50	p_2s_2	-	**10126.60**	**5.84**	**0**	-	10601.33	10.70	**0**
500	4	50	p_2s_3	**11.93**	**23699.80**	**0.00**	**5**	87.02	**23699.80**	**0.00**	**5**
1000	4	50	p_2s_1	-	**66165.00**	**1.16**	**0**	-	No solution	Infinite	**0**
1000	4	50	p_2s_2	-	**504224.40**	**92.52**	**0**	-	No solution	Infinite	**0**
1000	4	50	p_2s_3	**315.94**	**92779.40**	**0.00**	**5**	1368.75	92823.00	0.05	2
5000	4	50	p_2s_1	-	**12573122.60**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	50	p_2s_2	-	**12546134.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	50	p_2s_3	-	2323120.80	0.01	0	**1662.90**	**2314720.00**	**0.01**	**1**

Table 4.11: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 50$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 8$, and $B = 50$											
10	8	50	p_1s_1	**0.00**	**19.00**	**0.00**	**5**	0.11	**19.00**	**0.00**	**5**
10	8	50	p_1s_2	**0.00**	**19.80**	**0.00**	**5**	0.10	**19.80**	**0.00**	**5**
10	8	50	p_1s_3	**0.00**	**19.20**	**0.00**	**5**	0.01	**19.20**	**0.00**	**5**
50	8	50	p_1s_1	**1.39**	**38.00**	**0.00**	**5**	419.18	38.20	0.61	4
50	8	50	p_1s_2	504.58	**22.80**	**0.00**	**5**	**400.76**	**22.80**	0.95	4
50	8	50	p_1s_3	**0.12**	**54.20**	**0.00**	**5**	0.17	**54.20**	**0.00**	**5**
100	8	50	p_1s_1	**3.01**	**70.60**	**0.00**	**5**	323.91	**70.60**	**0.00**	**5**
100	8	50	p_1s_2	-	**40.60**	2.47	0	**1561.40**	**40.60**	**1.98**	**1**
100	8	50	p_1s_3	**0.24**	**93.80**	**0.00**	**5**	0.28	**93.80**	**0.00**	**5**
500	8	50	p_1s_1	-	344.40	0.53	0	**1194.22**	**343.60**	**0.18**	**2**
500	8	50	p_1s_2	-	206.40	5.41	**0**	-	**198.20**	**1.22**	**0**
500	8	50	p_1s_3	9.48	**505.00**	**0.00**	**5**	**0.58**	**505.00**	**0.00**	**5**
1000	8	50	p_1s_1	-	696.20	1.17	0	**1027.20**	**690.40**	**0.06**	**3**
1000	8	50	p_1s_2	-	10332.20	96.25	**0**	-	**392.40**	**1.06**	**0**
1000	8	50	p_1s_3	140.24	**988.00**	**0.00**	**5**	**0.66**	**988.00**	**0.00**	**5**
5000	8	50	p_1s_1	-	52744.20	100.00	0	**1661.61**	**3418.40**	**0.02**	**1**
5000	8	50	p_1s_2	-	52553.40	100.00	**0**	-	**1975.40**	**0.24**	**0**
5000	8	50	p_1s_3	1736.05	23832.40	36.30	1	**1.04**	**4865.00**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 8$, and $B = 50$											
10	8	50	p_2s_1	**0.00**	**10.00**	**0.00**	**5**	0.06	**10.00**	**0.00**	**5**
10	8	50	p_2s_2	**0.00**	**10.00**	**0.00**	**5**	0.09	**10.00**	**0.00**	**5**
10	8	50	p_2s_3	**0.00**	**9.80**	**0.00**	**5**	0.01	**9.80**	**0.00**	**5**
50	8	50	p_2s_1	**11.43**	**98.00**	**0.00**	**5**	1110.82	**98.00**	0.43	3
50	8	50	p_2s_2	**641.64**	**54.40**	**0.39**	**4**	1423.61	55.20	3.00	2
50	8	50	p_2s_3	**0.41**	**126.60**	**0.00**	**5**	0.45	**126.60**	**0.00**	**5**
100	8	50	p_2s_1	**131.77**	**330.20**	**0.00**	**5**	-	337.20	2.37	0
100	8	50	p_2s_2	-	**210.60**	**3.72**	**0**	-	216.00	6.38	**0**
100	8	50	p_2s_3	**1.29**	**531.60**	**0.00**	**5**	5.31	**531.60**	**0.00**	**5**
500	8	50	p_2s_1	-	**8196.40**	**0.39**	**0**	-	8258.50	0.88	**0**
500	8	50	p_2s_2	-	**5085.00**	**6.26**	**0**	-	5727.00	17.92	**0**
500	8	50	p_2s_3	**429.71**	**11850.60**	**0.00**	**4**	841.71	11854.00	0.03	3
1000	8	50	p_2s_1	-	**33137.00**	**1.34**	**0**	-	No solution	Infinite	**0**
1000	8	50	p_2s_2	-	**504224.40**	**96.26**	**0**	-	No solution	Infinite	**0**
1000	8	50	p_2s_3	**1063.52**	**46393.60**	**0.01**	**3**	-	46448.60	0.13	0
5000	8	50	p_2s_1	-	**12573122.60**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	50	p_2s_2	-	**12546134.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	50	p_2s_3	-	1161601.00	0.01	**0**	-	**1157414.50**	**0.01**	**0**

Table 4.12: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 100$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 2$, and $B = 100$											
10	2	100	p_1s_1	**0.01**	**39.00**	**0.00**	**5**	0.06	**39.00**	**0.00**	**5**
10	2	100	p_1s_2	**0.01**	**23.20**	**0.00**	**5**	0.24	**23.20**	**0.00**	**5**
10	2	100	p_1s_3	**0.00**	**47.60**	**0.00**	**5**	0.01	**47.60**	**0.00**	**5**
50	2	100	p_1s_1	**0.12**	**158.20**	**0.00**	**5**	3.54	**158.20**	**0.00**	**5**
50	2	100	p_1s_2	193.32	**90.60**	**0.00**	**5**	**52.28**	**90.60**	**0.00**	**5**
50	2	100	p_1s_3	**0.02**	**205.80**	**0.00**	**5**	0.09	**205.80**	**0.00**	**5**
100	2	100	p_1s_1	**16.74**	**293.80**	**0.00**	**5**	29.84	**293.80**	**0.00**	**5**
100	2	100	p_1s_2	1587.80	165.20	1.31	1	**976.67**	**164.60**	**0.63**	**3**
100	2	100	p_1s_3	**0.10**	**406.00**	**0.00**	**5**	0.21	**406.00**	**0.00**	**5**
500	2	100	p_1s_1	1646.86	1360.60	0.31	1	**413.61**	**1357.40**	**0.00**	**5**
500	2	100	p_1s_2	-	823.60	5.72	**0**	-	**805.60**	**3.25**	**0**
500	2	100	p_1s_3	4.92	**2048.40**	**0.00**	**5**	**0.93**	**2048.40**	**0.00**	**5**
1000	2	100	p_1s_1	-	2806.60	1.11	0	**348.53**	**2784.60**	**0.00**	**5**
1000	2	100	p_1s_2	-	10531.40	85.12	**0**	-	**1613.40**	**2.65**	**0**
1000	2	100	p_1s_3	742.17	**4047.60**	0.01	3	**0.73**	**4047.60**	**0.00**	**5**
5000	2	100	p_1s_1	-	52416.60	100.00	0	**1214.55**	**13449.40**	**0.00**	**4**
5000	2	100	p_1s_2	-	52518.60	100.00	**0**	-	**8078.67**	**2.30**	**0**
5000	2	100	p_1s_3	1239.39	19654.40	0.00	4	**0.67**	**19653.80**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 2$, and $B = 100$											
10	2	100	p_2s_1	**0.01**	**17.00**	**0.00**	**5**	0.10	**17.00**	**0.00**	**5**
10	2	100	p_2s_2	**0.03**	**11.20**	**0.00**	**5**	0.22	**11.20**	**0.00**	**5**
10	2	100	p_2s_3	**0.00**	**25.00**	**0.00**	**5**	**0.00**	**25.00**	**0.00**	**5**
50	2	100	p_2s_1	**0.34**	**412.00**	**0.00**	**5**	436.22	**412.00**	0.10	4
50	2	100	p_2s_2	**224.77**	**221.00**	**0.00**	**5**	1579.39	221.20	0.85	2
50	2	100	p_2s_3	**0.04**	**521.00**	**0.00**	**5**	0.16	**521.00**	**0.00**	**5**
100	2	100	p_2s_1	**1.18**	**1546.40**	**0.00**	**5**	669.74	1546.60	0.04	4
100	2	100	p_2s_2	-	**817.80**	**1.05**	**0**	-	843.40	4.58	**0**
100	2	100	p_2s_3	**0.15**	**2045.40**	**0.00**	**5**	0.95	**2045.40**	**0.00**	**5**
500	2	100	p_2s_1	**1607.79**	**32894.40**	**0.13**	**1**	-	No solution	Infinite	0
500	2	100	p_2s_2	-	**20471.20**	**7.23**	**0**	-	No solution	Infinite	**0**
500	2	100	p_2s_3	**1.33**	**49049.60**	**0.00**	**5**	19.92	**49049.60**	**0.00**	**5**
1000	2	100	p_2s_1	-	**280458.40**	**30.87**	**0**	-	No solution	Infinite	**0**
1000	2	100	p_2s_2	-	501531.00	84.89	**0**	-	**127916.00**	**39.51**	**0**
1000	2	100	p_2s_3	**11.29**	**190753.40**	**0.00**	**5**	546.25	**190753.40**	**0.00**	**5**
5000	2	100	p_2s_1	-	**12529354.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	100	p_2s_2	-	**12488031.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	2	100	p_2s_3	**1191.10**	**4702054.40**	**0.00**	**3**	-	4702357.60	0.00	0

Table 4.13: Computational results for the new instances proposed for the $P_m|s_j,B|C_{\max}$ problem, with $B = 100$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 4$, and $B = 100$											
10	4	100	p_1s_1	**0.01**	**22.80**	**0.00**	**5**	0.07	**22.80**	**0.00**	**5**
10	4	100	p_1s_2	**0.00**	**18.60**	**0.00**	**5**	0.17	**18.60**	**0.00**	**5**
10	4	100	p_1s_3	**0.01**	**26.40**	**0.00**	**5**	0.02	**26.40**	**0.00**	**5**
50	4	100	p_1s_1	**0.19**	**79.20**	**0.00**	**5**	14.14	**79.20**	**0.00**	**5**
50	4	100	p_1s_2	**366.68**	**45.60**	**0.38**	**4**	1453.81	45.80	2.16	1
50	4	100	p_1s_3	**0.05**	**103.20**	**0.00**	**5**	0.11	**103.20**	**0.00**	**5**
100	4	100	p_1s_1	**9.98**	**147.00**	**0.00**	**5**	645.74	147.20	0.13	4
100	4	100	p_1s_2	**1688.92**	83.00	**1.40**	**1**	-	**82.80**	1.93	0
100	4	100	p_1s_3	**0.19**	**203.40**	**0.00**	**5**	0.30	**203.40**	**0.00**	**5**
500	4	100	p_1s_1	1459.27	**680.80**	**0.33**	1	**1349.94**	682.60	0.59	**2**
500	4	100	p_1s_2	-	409.80	5.17	**0**	-	**396.00**	**1.62**	**0**
500	4	100	p_1s_3	2.18	**1024.60**	**0.00**	**5**	**1.70**	**1024.60**	**0.00**	**5**
1000	4	100	p_1s_1	-	**1406.20**	1.30	0	**1691.34**	1419.33	**0.05**	**1**
1000	4	100	p_1s_2	-	10531.40	92.56	**0**	-	**804.75**	**1.65**	**0**
1000	4	100	p_1s_3	466.31	**2024.00**	0.01	4	**1.70**	**2024.00**	**0.00**	**5**
5000	4	100	p_1s_1	-	**52416.60**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	100	p_1s_2	-	52518.60	100.00	**0**	-	**3965.20**	**0.52**	**0**
5000	4	100	p_1s_3	1696.55	26826.40	32.51	1	**2.01**	**9827.00**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 4$, and $B = 100$											
10	4	100	p_2s_1	**0.02**	**9.60**	**0.00**	**5**	0.11	**9.60**	**0.00**	**5**
10	4	100	p_2s_2	**0.00**	**9.80**	**0.00**	**5**	0.17	**9.80**	**0.00**	**5**
10	4	100	p_2s_3	**0.01**	**13.20**	**0.00**	**5**	0.01	**13.20**	**0.00**	**5**
50	4	100	p_2s_1	**0.82**	**206.20**	**0.00**	**5**	930.97	206.40	0.32	4
50	4	100	p_2s_2	**282.99**	**110.60**	**0.00**	**5**	-	112.20	3.10	0
50	4	100	p_2s_3	**0.07**	**260.80**	**0.00**	**5**	0.21	**260.80**	**0.00**	**5**
100	4	100	p_2s_1	**2.90**	**773.40**	**0.00**	**5**	1197.72	775.00	0.26	4
100	4	100	p_2s_2	-	**412.80**	**2.04**	**0**	-	425.40	5.41	**0**
100	4	100	p_2s_3	**0.35**	**1022.80**	**0.00**	**5**	1.93	**1022.80**	**0.00**	**5**
500	4	100	p_2s_1	**1659.26**	**16471.00**	**0.28**	**1**	-	No solution	Infinite	0
500	4	100	p_2s_2	-	**10143.80**	**6.41**	**0**	-	No solution	Infinite	**0**
500	4	100	p_2s_3	**4.19**	**24525.00**	**0.00**	**5**	153.57	**24525.00**	**0.00**	**5**
1000	4	100	p_2s_1	-	**239423.80**	**36.16**	**0**	-	No solution	Infinite	**0**
1000	4	100	p_2s_2	-	**501531.00**	**92.45**	**0**	-	No solution	Infinite	**0**
1000	4	100	p_2s_3	**46.37**	95377.00	**0.00**	**5**	1326.33	**94294.00**	0.01	2
5000	4	100	p_2s_1	-	**12529354.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	100	p_2s_2	-	**12488031.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	4	100	p_2s_3	-	**2351119.80**	**0.00**	**0**	-	No solution	Infinite	**0**

Table 4.14: Computational results for the new instances proposed for the $P_m|s_j, B|C_{\max}$ problem, with $B = 100$.

Instance				(MILP$_2^+$)				(FLOW)			
Jobs	Machines	Capacity	Type	$T(s)$	$C_{\max}$	Gap	#O	$T(s)$	$C_{\max}$	Gap	#O
Instances with $p_1 = [1, 20]$, $n_M = 8$, and $B = 100$											
10	8	100	p_1s_1	**0.00**	**18.40**	**0.00**	**5**	0.06	**18.40**	**0.00**	**5**
10	8	100	p_1s_2	**0.00**	**18.60**	**0.00**	**5**	0.20	**18.60**	**0.00**	**5**
10	8	100	p_1s_3	**0.00**	**19.80**	**0.00**	**5**	0.01	**19.80**	**0.00**	**5**
50	8	100	p_1s_1	**0.41**	**39.80**	**0.00**	**5**	52.65	**39.80**	**0.00**	**5**
50	8	100	p_1s_2	**486.87**	**23.80**	**0.71**	**4**	1474.00	24.20	3.34	1
50	8	100	p_1s_3	**0.10**	**51.80**	**0.00**	**5**	0.18	**51.80**	**0.00**	**5**
100	8	100	p_1s_1	**18.64**	**73.60**	**0.00**	**5**	689.92	73.80	0.26	4
100	8	100	p_1s_2	**1494.83**	**42.00**	**1.87**	**1**	1590.99	42.20	2.80	**1**
100	8	100	p_1s_3	**0.20**	**102.20**	**0.00**	**5**	0.49	**102.20**	**0.00**	**5**
500	8	100	p_1s_1	1485.79	**341.20**	**0.41**	1	**1478.52**	344.00	0.69	**2**
500	8	100	p_1s_2	-	208.40	6.68	**0**	-	**199.40**	**2.11**	**0**
500	8	100	p_1s_3	5.42	**512.40**	**0.00**	**5**	**1.96**	**512.40**	**0.00**	**5**
1000	8	100	p_1s_1	-	**702.20**	1.20	0	**1127.93**	708.25	**0.04**	**3**
1000	8	100	p_1s_2	-	10531.40	96.28	**0**	-	**400.60**	**1.96**	**0**
1000	8	100	p_1s_3	85.57	**1012.00**	**0.00**	**5**	**26.01**	**1012.00**	**0.00**	**5**
5000	8	100	p_1s_1	-	**52416.60**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	100	p_1s_2	-	52518.60	100.00	**0**	-	**1983.60**	**0.54**	**0**
5000	8	100	p_1s_3	1376.08	4914.40	0.01	2	**1.28**	**4914.00**	**0.00**	**5**
Instances with $p_2 = [1, n_J]$, $n_M = 8$, and $B = 100$											
10	8	100	p_2s_1	**0.01**	**9.00**	**0.00**	**5**	0.14	**9.00**	**0.00**	**5**
10	8	100	p_2s_2	**0.01**	**9.80**	**0.00**	**5**	0.18	**9.80**	**0.00**	**5**
10	8	100	p_2s_3	**0.00**	**9.40**	**0.00**	**5**	0.01	**9.40**	**0.00**	**5**
50	8	100	p_2s_1	**1.84**	**103.20**	**0.00**	**5**	1091.00	104.20	1.44	2
50	8	100	p_2s_2	**610.75**	**59.20**	**0.36**	**4**	-	59.80	5.14	0
50	8	100	p_2s_3	**0.18**	**130.60**	**0.00**	**5**	1.25	**130.60**	**0.00**	**5**
100	8	100	p_2s_1	**26.48**	**387.00**	**0.00**	**5**	-	400.80	3.79	0
100	8	100	p_2s_2	-	**212.80**	**4.88**	**0**	-	219.60	8.23	**0**
100	8	100	p_2s_3	**2.76**	**511.60**	**0.00**	**5**	16.50	**511.60**	**0.00**	**5**
500	8	100	p_2s_1	-	**8242.00**	**0.35**	**0**	-	No solution	Infinite	**0**
500	8	100	p_2s_2	-	**5104.20**	**6.98**	**0**	-	No solution	Infinite	**0**
500	8	100	p_2s_3	**39.32**	**12262.80**	**0.00**	**5**	1131.33	32756.60	18.03	2
1000	8	100	p_2s_1	-	**407140.20**	**75.02**	**0**	-	No solution	Infinite	**0**
1000	8	100	p_2s_2	-	**501531.00**	**96.22**	**0**	-	No solution	Infinite	**0**
1000	8	100	p_2s_3	**934.07**	47689.20	**0.00**	**4**	-	**47167.33**	0.05	0
5000	8	100	p_2s_1	-	**12529354.80**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	100	p_2s_2	-	**12488031.00**	**100.00**	**0**	-	No solution	Infinite	**0**
5000	8	100	p_2s_3	-	**1175647.80**	**0.01**	**0**	-	No solution	Infinite	**0**

Chapter 5

The $1|r_j, s_j, B|C_{max}$ problem

Problem $1|r_j, s_j, B|C_{\max}$ is also very similar to problem $1|s_j, B|C_{\max}$, the only difference is that in this problem jobs have non-identical release times. All other definitions are similar to the ones presented for problem $1|s_j, B|C_{\max}$. The $1|r_j, s_j, B|C_{\max}$ problem admits that each job $j \in J$ has a release time r_j in addition to the processing time p_j and the size s_j. The jobs can only be processed after released, and consequently, each batch can only be processed when all jobs assigned to it have been released. The release time of batch k is then defined as $R_k := \max\{r_j : j \text{ is assigned to } k\}$.

An analysis of how release times variation affects the difficulty of the problem is reported in Xu *et al.* [8]. If the distribution of release times is short, jobs will be available at a similar time, and release times will have little effect on the solution, making this problem similar to $1|s_j, B|C_{\max}$. On the other hand, if this variation of release times is large, the jobs will be distributed at long intervals, reducing the solution space of the problem. In extreme situations, the optimal solution may determine that each job is assigned to a different batch.

5.1 Literature formulation

Let us consider now the following decision variables:

$$x_{jk} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k; \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

$$P_k : \text{processing time of batch } k. \tag{5.2}$$

$$S_k : \text{time when batch } k \text{ starts to be processed,} \tag{5.3}$$

for all $j \in J$, $k \in K$.

In Xu *et al.* [8] the following MILP formulation is proposed for $1|r_j, s_j, B|C_{\max}$.

$$
\begin{aligned}
(\text{MILP}_3) \quad & \min\ S_{n_K} + P_{n_K}, && && (5.4)\\
& \sum_{k\in K} x_{jk} = 1 && \forall j \in J && (5.5)\\
& \sum_{j\in J} s_j x_{jk} \le B, && \forall k \in K, && (5.6)\\
& P_k \ge p_j x_{jk} && \forall j \in J, \forall k \in K && (5.7)\\
& S_k \ge r_j x_{jk}, && \forall j \in J, \forall k \in K, && (5.8)\\
& S_k \ge S_{k-1} + P_{k-1}, && \forall k \in K : k > 1, && (5.9)\\
& x_{jk} \in \{0,1\} && \forall j \in J, \forall k \in K && (5.10)
\end{aligned}
$$

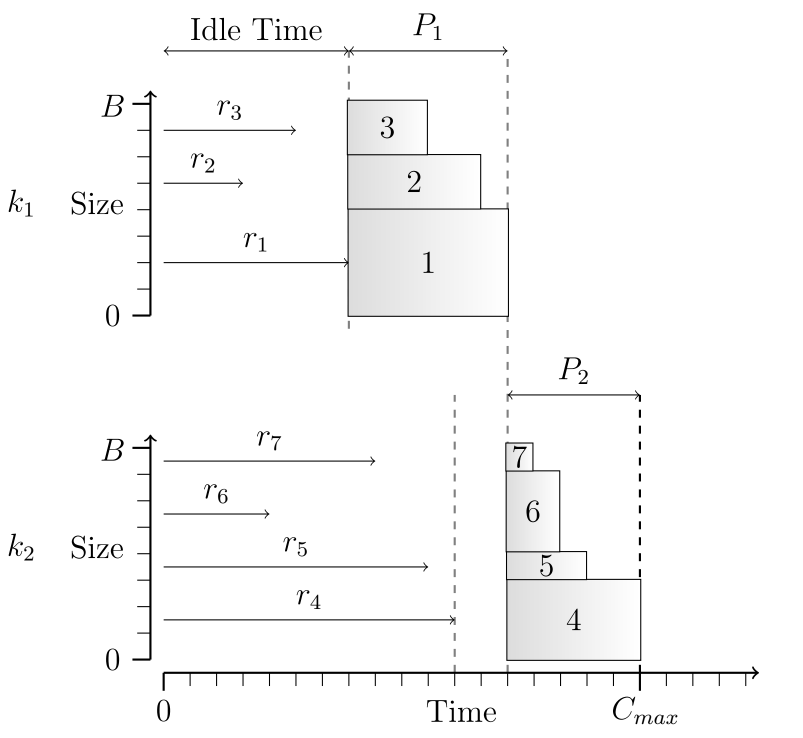
The objective function (5.4) minimizes the makespan, given by the starting time of the last batch processed added to its processing time. Constraints (5.5) determine that each job is assigned to a single batch. Constraints (5.6) ensure that each batch respects the capacity of the machine. Constraints (5.7) determine the processing times of the batches. Constraints (5.8)-(5.9) determine the time when each batch starts to be processed. The constraints (5.5), (5.7), and (5.10) are the same as the constraints used in the model (MILP$_1$) (3.5), (3.7), and (3.10) respectively. Note that the binary variables y_k, defined in (3.2), and previously used to strengthen the linear relaxation of problem $1|s_j, B|C_{\max}$, are not used in this formulation.

5.2 Symmetry breaking approach

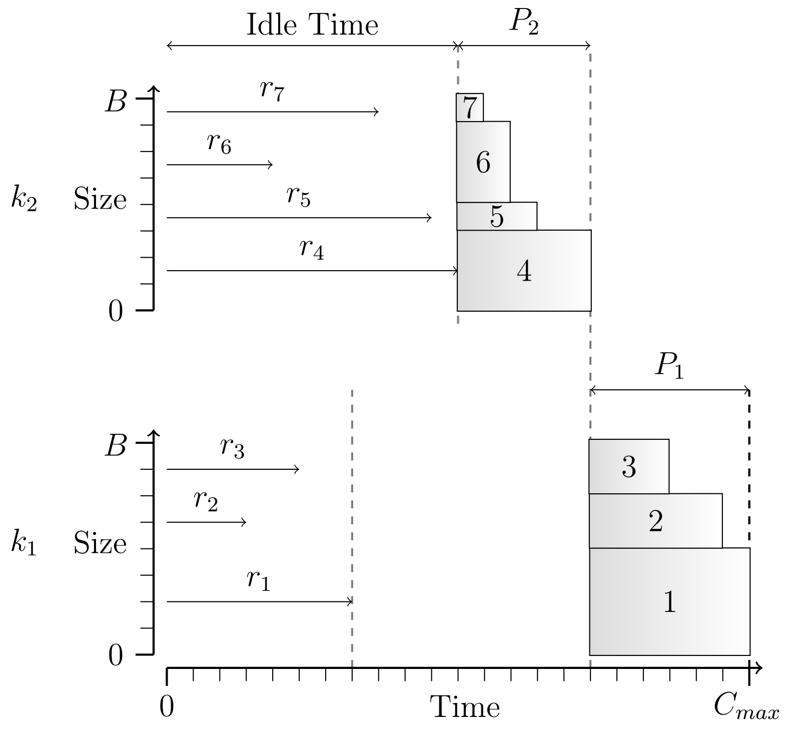
Unlike the previous case where jobs have identical release times, when the jobs have different release times, the order in which the batches are processed in a machine may modify the resulting makespan. The reason for this is the possible modification on the amount of time in which the machine stays idle, when permutations on the processing order are made.

Figure 5.1 depicts two different solutions for a problem, where a permutation on the processing order of the batches leads to a different makespan. In the example, batches k_1 and k_2 shown in Figure 5.1(a) switch places in the processing order. The modification, shown in Figure 5.1(b), increases the idle time of the processing machine leading to an increase on the makespan.

Nevertheless, although a modification in the processing order of the batches leads in general, to nonequivalent solutions, symmetric solutions may still occur. Two cases are illustrated in Figure 5.2. In Case 1, two consecutive batches k and k' have the same release time. Thus, switching their indexes clearly does not affect the value of the makespan. In Case 2, batch k is released before k', but the machine is not idle at this time, due to the processing of the previously scheduled batch k''. In this configuration, switching k and k' does not affect the value of the makespan



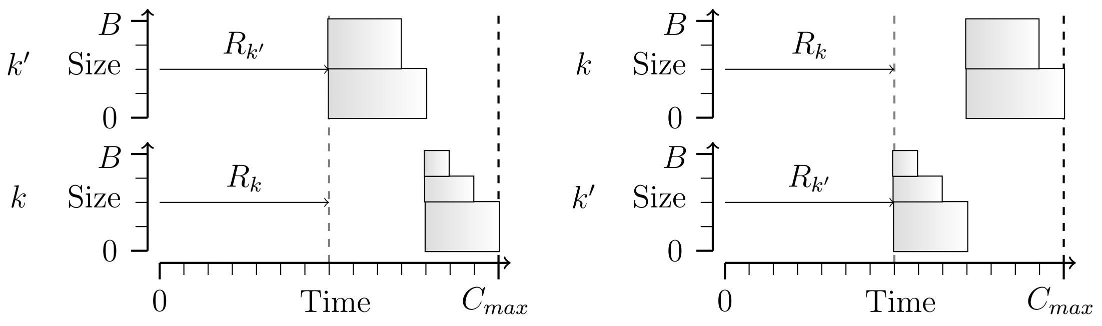
(a) Solution for $1|r_j, s_j, B|C_{\max}$ with batch k_1 processed before k_2.



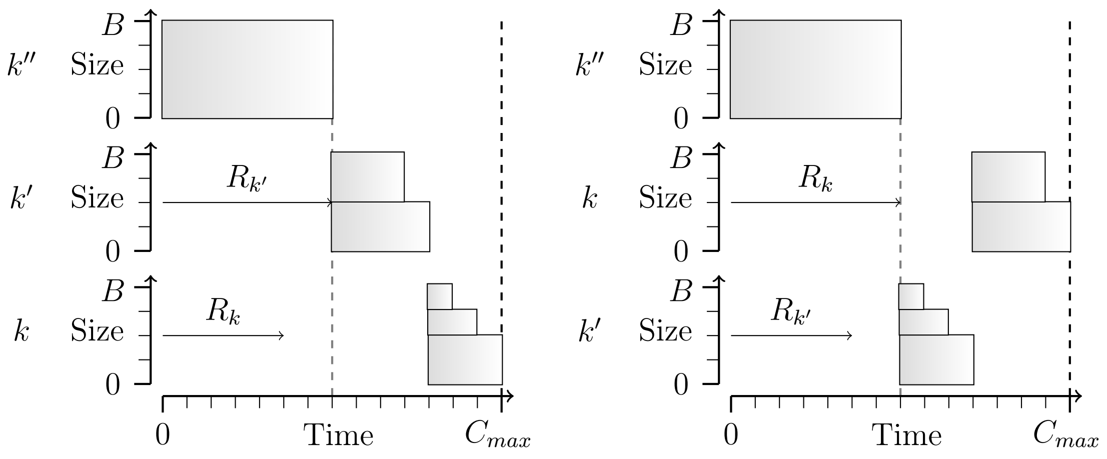
(b) Solution for $1|r_j, s_j, B|C_{\max}$ with batch k_1 processed after k_2.

Figure 5.1: An example of solution for $1|r_j, s_j, B|C_{\max}$.

as well.



(a) Case 1



(b) Case 2

Figure 5.2: Cases where switching batches in the processing order does not modify the makespan.

5.2.1 Symmetry breaking formulation

To deal with the symmetry of problem $1|r_j, s_j, B|C_{\max}$, we use a similar approach to the one proposed for problems where jobs have identical release times. However, now the jobs must be ordered by non-decreasing release times, and not by non-decreasing processing times as before. More specifically, we consider that the jobs are indexed satisfying:

$$r_1 \leq r_2 \leq \ldots \leq r_{n_J}. \tag{5.11}$$

As done for problems $1|s_j, B|C_{\max}$ and $P_m|s_j, B|C_{\max}$, the symmetry of problem $1|r_j, s_j, B|C_{\max}$ can be addressed by considering the variables x_{jk} in (5.1), only for $j \leq k$, together with the constraints (3.13)-(3.15), that assure that each job j is assigned to a single batch k, such that $j \leq k$. Once more, if batch k is used, we enforce job k to be the job with the highest index assigned to it. In this case, however, job k determines the release time of batch k, given by r_k, and not the processing time. Thus, the solutions presented by the formulation proposed, also present a specific ordering for the batches scheduled in the processing machine. The batches that are used, are processed in a non-decreasing order of their release times. Clearly, for a given set of designed batches, no other processing order for them would lead to a smaller makespan. Finally, for a given solution, the procedure allows only one possible way of indexing the batches, where the index of each batch is again given by the largest index among the jobs assigned to it.

We propose the following formulation for $1|r_j, s_j, B|C_{\max}$:

$$
\begin{aligned}
(\text{MILP}_3^+)\quad & \min\ S_{n_K} + P_{n_K}, && && (5.12)\\
& \sum_{k\in K: j\le k} x_{j,k} = 1 && \forall j \in J && (5.13)\\
& \sum_{j\in J: j\le k} s_j x_{j,k} \le B x_{k,k} && \forall k \in K && (5.14)\\
& x_{j,k} \le x_{k,k} && \forall j \in J, \forall k \in K : j < k && (5.15)\\
& P_k \ge p_j x_{jk}, && \forall j \in J, \forall k \in K : j \le k, && (5.16)\\
& S_k \ge r_k x_{kk}, && \forall k \in K, && (5.17)\\
& S_k \ge S_{k-1} + P_{k-1}, && \forall k \in K : k > 1, && (5.18)\\
& x_{j,k} \in \{0, 1\} && \forall j \in J, \forall k \in K : j \le k && (5.19)
\end{aligned}
$$

The objective function (5.12) minimizes the makespan, given by the time required to finish processing the last batch. Constraints (5.13) determine that each job j is assigned to a single batch k, such that $k \ge j$. Constraints (5.14) determine that the batches do not exceed the capacity of the machine. They also ensure that each batch k is used if and only if job k is assigned to it. Constraints (5.15) are redundant together with (5.14), but are included to strengthen the linear relaxation of the model. Constraints (5.16) determine the processing time of the batches. Constraints (5.17)-(5.18) determine when each batch starts to be processed. The constraints (5.13), (5.14), (5.15), (5.19) are the same as the constraints (3.13), (3.14), (3.15), and (3.16) respectively, used in the model (MILP$_1$).

The following proposition certifies that our formulation is equivalent to formulation (MILP$_3$), concerning the optimal solution value.

Proposition 2. *The optimal makespan of problems (MILP$_3$) and (MILP$_3^+$) are the same.*

Proof. Let us consider w.l.o.g. that the indexes of jobs in J satisfy (5.11). Clearly, any feasible solution of (MILP$_3^+$) is also a feasible solution to (MILP$_3$) with the same objective function value. Therefore, it suffices to show that given an optimal solution to (MILP$_3$), with objective function value $\bar{C}_{\max}$, there is a feasible solution to (MILP$_3^+$) also with objective function value $\bar{C}_{\max}$. For that, let us first reset, if necessary, the indexes of the batches on the given solution of (MILP$_3$), determining them as the largest index among the ones of all jobs assigned to it. The solution is now a feasible to solution to (MILP$_3^+$), with the batches designed as in the given feasible solution to (MILP$_3$), but possibly processed in a different order. Note that as the batches are now ordered by non-decreasing release times, the reordering cannot increase the idle time of the machine and, thus, cannot increase the makespan. Therefore the feasible solution to (MILP$_3^+$) also has the minimum makespan $C_{\max}$, as the objective function value. □

5.3 Computational results

The set of instances considered for problem $1|r_j, s_j, B|C_{\max}$ was generated according to the methodology proposed in Xu *et al.* [8] and Chou *et al.* [36]. For each job j, a processing time p_j, a release time r_j, and a size s_j are randomly selected as described in Table 5.1. In total, 280 instances were generated, 20 for each of the 14 combinations of number and size of jobs.

Table 5.1: Parameter settings for set of test instances for $1|r_j, s_j, B|C_{\max}$.

Number of jobs (n_J)	Processing time (p_j)	Jobs size (s_j)	Release time (r_j)	Machine capacity (B)
10, 20, 50, 100, 300, 500	p_1: [8, 48]	s_1: [1, 15] s_2: [15, 35]	r_1: [0,C]	$B = 40$

The following steps were considered to generate the release times of the jobs:

1. For each instance, the size and processing time of each job, are randomly selected in the intervals presented in Table 5.1.

2. A lower bound C on the optimal makespan for each instance is computed. This bound does not consider the release times of the jobs, which have not been defined yet. For this computation, a simple batch-first-fit heuristic, proposed in [2], is applied generating a feasible solution for the problem with no release times.

3. Finally, the job release times for each instance, are randomly generated in the interval [0, C], where C is the value of the lower bound found in the previous step.

In all tests, we use the `CPLEX` version 12.5, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.83 GHz `Intel Quad-Core Xeon X3360` processor, and 8GB of RAM. The computational time to solve each instance was limited in 1800 seconds.

Table 5.2 presents the average computational results for the 20 instances tested with each configuration. The two first columns show the instances configuration, now specified by the pair of parameters n_J, s_i, for $i = 1, 2$, according to Table 5.1. The other columns show the statistics analyzed for both formulations (MILP$_3$) and (MILP$_3^+$). The time required by `CPLEX` to obtain the best solution, in seconds $T_b(s)$.

Table 5.2: Computational results for $1|r_j, s_j, B|C_{\max}$.

Instance		(MILP$_3$)				(MILP$_3^+$)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
10	$s1$	**117.80**	0.16	0.09	**0.00**	**117.80**	**0.02**	**0.02**	**0.00**
10	$s2$	**316.95**	0.57	0.20	**0.00**	**316.95**	**0.01**	**0.00**	**0.00**
20	$s1$	**193.80**	192.34	27.78	0.03	**193.80**	**0.33**	**0.32**	**0.00**
20	$s2$	**560.70**	270.13	7.97	0.44	**560.70**	**0.01**	**0.01**	**0.00**
50	$s1$	396.50	-	1674.20	40.64	**389.45**	**456.99**	**129.53**	**0.63**
50	$s2$	1351.80	-	1725.32	75.01	**1298.55**	**0.06**	**0.06**	**0.00**
100	$s1$	920.05	-	1744.30	94.50	**760.45**	**1744.24**	**327.94**	**5.51**
100	$s2$	3368.70	-	1640.57	94.90	**2578.35**	**0.39**	**0.36**	**0.00**
200	$s1$	2884.10	**-**	**177.67**	98.33	**1576.75**	**-**	1761.36	**15.38**
200	$s2$	9257.15	-	82.61	99.40	**5049.35**	**1.19**	**1.17**	**0.00**
300	$s1$	4355.70	**-**	1785.86	99.94	**2526.05**	**-**	**1388.04**	**21.49**
300	$s2$	13707.75	-	1754.98	99.86	**7483.25**	**1.99**	**1.93**	**0.00**
500	$s1$	7152.60	**-**	-	100.00	**5805.50**	**-**	**1179.36**	**43.76**
500	$s2$	23320.85	-	-	100.00	**12589.80**	**3.50**	**3.43**	**0.00**

Both formulations (MILP$_3$) and (MILP$_3^+$) are efficient for solving instances of reduced size and have proved optimality for all instances with 10 jobs. However, the remaining results show that (MILP$_3^+$) overcomes (MILP$_3$), both in computational time and in the quality of the solutions obtained. Formulation (MILP$_3$) cannot prove optimality of any instance with more than 20 jobs, while (MILP$_3^+$) proves optimality for all instances with 10 and 20 jobs in no more than 0.32 seconds. Considering instances with 50 and 100 jobs, (MILP$_3^+$) continues to outperform (MILP$_3$) in every respect.

Table 5.2 also shows that instances of type $s1$ are more difficult than than instances of type $s2$ for both formulations, which was already verified in previous computational experiments. As before, the reduced size of the jobs generates a larger number of combinations in the batch configuration. Formulation (MILP$_3^+$) finds all optimal solutions for instances of type $s2$, with average computational time less than 3.5 seconds. For instances of type $s1$, with more than 50 jobs, (MILP$_3^+$) cannot prove optimality for all instances, but obtains much better solutions and lower bounds than (MILP$_3$).

Chapter 6

The $P_m|r_j, s_j, B|C_{max}$ problem

$P_m|r_j, s_j, B|C_{\max}$ is the last problem considered in this work and gathers characteristics of all the problems presented previously. When defining problem $P_m|r_j, s_j, B|C_{\max}$, we consider all aspects presented to $1|s_j, B|C_{\max}$, where each job $j \in J$ have non-identical processing times p_j and sizes s_j. This problem considers that the jobs can only be processed after released as in $1|r_j, s_j, B|C_{\max}$, and also assumes that the batches must be assigned to a specific identical-machine $M := \{1, \ldots, n_M\}$, as in $P_m|s_j, B|C_{\max}$. The objective is again to minimize the makespan ($C_{\max}$), defined as the time required to finish processing the last batch in all machines.

6.1 Literature formulation

Let us consider now the following decision variables:

$$x_{jkm} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k \text{ processed in machine } m; \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

$$S_{km} : \text{time when batch } k \text{ starts to be processed in machine } m,, \tag{6.2}$$

$$C_{max} : \text{the makespan} \tag{6.3}$$

for all $j \in J$, $k \in K$, and $m \in M$.

In Vélez Gallego [9] the following MILP formulation is proposed for $P_m|r_j, s_j, B|C_{\max}$.

$$
\begin{aligned}
&(\text{MILP}_4) \quad \min\ C_{\max}, && && (6.4)\\
&\sum_{k\in K}\sum_{m\in M} x_{j,k,m} = 1 && \forall j \in J && (6.5)\\
&\sum_{j\in J} s_j x_{jkm} \leq B, && \forall k \in K, \forall m \in M && (6.6)\\
&S_{km} \geq r_j x_{jkm}, && \forall j \in J, \forall k \in K, \forall m \in M, && (6.7)\\
&S_{km} \geq S_{(k-1)m} + p_j x_{j(k-1)m}, && \forall j \in J, \forall k \in K : k > 1, \forall m \in M, && (6.8)\\
&C_{\max} \geq S_{n_K m} + p_j x_{jn_K m}, && \forall m \in M, && (6.9)\\
&x_{j,k,m} \in \{0,1\} && \forall j \in J, \forall k \in K, \forall m \in M && (6.10)
\end{aligned}
$$

The objective function (6.4) minimizes the makespan. Constraints (6.5) and (6.6) ensure that each job is assigned to a single batch and a single machine, respecting the capacity of the machine. Constraints (6.7)-(6.8) determine the time when each batch k starts to be processed in each machine m. Constraints (6.9) determine the makespan. The constraints (6.5), and (6.10) are the same as the constraints (4.5), and (4.9) respectively, used in the (MILP$_2$) model.

6.2 Symmetry breaking approach

As we did for problem $P_m|s_j, B|C_{\max}$, we will now replace the variable x_{jkm} in (MILP$_4$) with the binary variables x_{jk} and y_{km}. By doing so, we can apply to $P_m|r_j, s_j, B|C_{\max}$, the same symmetry breaking strategy used for $1|r_j, s_j, B|C_{\max}$. Considering that the indexes of the jobs satisfy the relation (5.11), we propose the following formulation for $P_m|r_j, s_j, B|C_{\max}$.

$$
\begin{aligned}
&(\text{MILP}_4^+) \quad \min\ C_{\max}, && && (6.11)\\
&\sum_{k\in K, j\leq k} x_{jk} = 1, && \forall j \in J, && (6.12)\\
&\sum_{j\in J: j\leq k} s_j x_{jk} \leq B x_{k,k}, && \forall k \in K, && (6.13)\\
&x_{jk} \leq x_{kk}, && \forall j \in J, \forall k \in K : j < k, && (6.14)\\
&x_{kk} \leq \sum_{m\in M} y_{km}, && \forall k \in K, \forall m \in M, && (6.15)\\
&P_{km} \geq p_j(x_{jk} + y_{km} - 1), && \forall j \in J, \forall k \in K : j \leq k, \forall m \in M, && (6.16)\\
&S_{km} \geq r_k y_{km}, && \forall k \in K, \forall m \in M, && (6.17)\\
&S_{km} \geq S_{(k-1)m} + P_{(k-1)m}, && \forall k \in K : k > 1, \forall m \in M, && (6.18)
\end{aligned}
$$

$$C_{\max} \geq S_{n_K m} + P_{n_K m}, \qquad \forall m \in M, \qquad (6.19)$$

$$x_{jk} \in \{0,1\}, \qquad \forall j \in J, \forall k \in K : j \leq k. \qquad (6.20)$$

The objective function (6.11) minimizes the makespan. Constraints (6.12) ensure that all batches used are assigned to a machine. Constraints (6.16) determine the processing time of each batch in each machine. Constraints (6.17)-(6.18) determine when each batch starts to be processed on the machine to which it is assigned. Constraints (6.19) determine the makespan as the time required to finish processing the last batch on all machines. The constraints (6.12), (6.13), (6.14) and (6.20) are the same as the constraints (3.13), (3.14), (3.15), and (3.16) respectively, used in the model (MILP$_1^+$).

6.3 Computational results

For the computational experiments on problem $P_m|r_j, s_j, B|C_{\max}$, we used the same set of instances used for problem $1|r_j, s_j, B|C_{\max}$. In addition, three new categories were included corresponding to the numbers of parallel machines: 2, 4 and 8 machines, and each instance was tested for the three different numbers of parallel machines. The sizes, the processing times and the release times of the jobs were randomly selected in the ranges shown in Table 6.1.

Table 6.1: Parameter settings for set of test instances for $1|r_j, s_j, B|C_{\max}$.

Number of jobs (n_J)	Processing time (p_j)	Jobs size (s_j)	Release time (r_j)	Machine cap.(B)	Parallel machines (n_M)
10, 20, 50, 100	p_1: [8, 48]	s_1: [1, 15] s_2: [15, 35] s_3: [4, 8]	r_1: [0,C]	$B = 40$	2, 4, 8

According to Table 6.1, 8 instance configurations were generated, defined by the combination of four different number of jobs, and two ranges for the job sizes. For each configuration, 20 instances were generated, summing up to 160 instances. Each instance was tested for the three numbers of parallel machine, leading to 480 tests performed. The release times generation following the same procedure described in the Section 5.3, for problem $1|r_j, s_j, B|C_{\max}$.

In all tests, we use the `CPLEX` version 12.5, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.83 GHz `Intel Quad-Core Xeon X3360` processor, and 8GB of RAM. The computational time to solve each instance was limited in 1800 seconds.

Table 6.2 show average computational results for each group of 100 instances with each configuration. The two first columns show the configuration of the instance, according to Table 6.1. The other columns show the statistics analyzed for both formulations (MILP$_4$) and (MILP$_4^+$).

Table 6.2: Computational results for $P_m|r_j, s_j, B|C_{\max}$ - 2-8 parallel machines.

Instance		(MILP$_4$)				(MILP$_4^+$)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
				2 parallel machines					
10	$s1$	**106.80**	**0.03**	**0.03**	**0.00**	**106.80**	**0.03**	**0.03**	**0.00**
10	$s2$	**281.15**	0.06	0.04	**0.00**	**281.15**	**0.01**	**0.01**	**0.00**
20	$s1$	**177.05**	1.28	0.48	**0.00**	**177.05**	**0.13**	**0.11**	**0.00**
20	$s2$	**521.25**	10.28	1.09	**0.00**	**521.25**	**0.02**	**0.02**	**0.00**
50	$s1$	**362.15**	706.72	47.89	0.74	**362.15**	**2.29**	**2.01**	**0.00**
50	$s2$	1254.05	1653.55	182.58	40.69	**1253.40**	**0.06**	**0.06**	**0.00**
100	$s1$	707.40	-	1659.15	94.79	**691.00**	**53.53**	**28.30**	**0.00**
100	$s2$	2666.95	-	1558.00	90.79	**2501.45**	**0.18**	**0.18**	**0.00**
				4 parallel machines					
10	$s1$	**106.60**	**0.02**	**0.02**	**0.00**	**106.60**	0.03	0.03	**0.00**
10	$s2$	**278.10**	0.03	0.03	**0.00**	**278.10**	**0.01**	**0.01**	**0.00**
20	$s1$	**176.70**	0.67	0.15	**0.00**	**176.70**	**0.28**	**0.13**	**0.00**
20	$s2$	**518.35**	0.89	0.23	**0.00**	**518.35**	**0.03**	**0.03**	**0.00**
50	$s1$	**361.55**	367.34	10.23	**0.00**	**361.55**	**2.54**	**1.11**	**0.00**
50	$s2$	**1251.40**	1624.65	12.46	8.23	**1251.40**	**0.19**	**0.13**	**0.00**
100	$s1$	**690.05**	-	451.21	73.25	**690.05**	**28.76**	**9.31**	**0.00**
100	$s2$	**2498.55**	1636.68	599.11	74.42	**2498.55**	**1.32**	**0.66**	**0.00**
				8 parallel machines					
10	$s1$	**106.60**	0.09	0.08	**0.00**	**106.60**	**0.07**	**0.06**	**0.00**
10	$s2$	**278.10**	0.04	0.03	**0.00**	**278.10**	**0.02**	**0.02**	**0.00**
20	$s1$	**176.70**	**0.47**	**0.13**	**0.00**	**176.70**	1.39	0.23	**0.00**
20	$s2$	**518.35**	0.99	0.13	**0.00**	**518.35**	**0.04**	**0.04**	**0.00**
50	$s1$	**361.55**	336.81	7.95	0.01	**361.55**	**6.46**	**1.92**	**0.00**
50	$s2$	**1251.40**	950.79	7.75	1.45	**1251.40**	**0.47**	**0.25**	**0.00**
100	$s1$	**690.05**	1653.79	108.07	30.62	**690.05**	**45.43**	**12.39**	**0.00**
100	$s2$	**2498.55**	1657.19	322.27	83.48	**2498.55**	**4.13**	**1.46**	**0.00**

The results shown in this subsection indicate that, in general, (MILP$_4^+$) obtain better solutions than (MILP$_4$) in less computational time. (MILP$_4^+$) proves optimality for all instances with up to 50 jobs, while (MILP$_4$) can only prove optimality for all instances with up to 20 jobs. Furthermore, even for these instances, it takes more time to run than (MILP$_4^+$).

For instances with 50 jobs, the superiority of (MILP$_4^+$) over (MILP$_4$) is even clearer. It can prove optimality for all instances in reduced times, while (MILP$_4$) has difficulty and does not prove optimality for the majority of the instances. The computational times and the duality gaps for (MILP$_4$) are big, especially when compared to (MILP$_4^+$), which solves the instances in less than 6.46 seconds on average. Following the pattern of the previous tests, instances of type s_2 consume more computational effort and are more difficult than those of type s_1.

Chapter 7

Conclusions and Future Work

In this thesis, we address four different versions of batch scheduling problems, which have been identified in the literature as suited models for problems that appear in reliability tests in the semiconductor industry. The economic importance of the problems have motivated the investigation of good solution approaches for them, and their NP-hardness have led the majority of this research to focus on heuristic approaches. We show that applying good MILP formulations for these scheduling problems we can go a step further in the exact resolution of applied problems, having presented optimal solutions for test instances with sizes never considered in the literature by exact methods. Even for instances which we could not solve to optimality in our time limit of 1800 seconds, we were able to present much better average results with the formulations proposed than the ones obtained with formulations previously presented in the literature.

The enhancement in the models was mainly based on the idea of eliminating symmetric solutions from the feasible sets of the problems. The development of symmetry breaking cuts is widely pursued in the MILP literature and some general approaches can certainly be applied to scheduling problems. Nevertheless, using some well-known properties of the optimal solutions of the problems addressed, we propose a specific indexing of the jobs to be processed, for each version of the problem. The indexing not only allows the reduction of the feasible sets by eliminating symmetric solutions but also significantly reduces the number of variables and constraints in the models, when compared to the ones in the literature, leading to simplified and stronger formulations. Finally, the referred properties of optimal solutions are the backbone to the proof of correctness of the models presented in this paper.

We also propose arc-flow formulations to problems that do not consider release times. In the new models, the number of variables does not change when the number of jobs increases, which enabled us to find the optimum solution of instances with 100 million jobs in 3.25 seconds. These results define a new threshold for the size of the instances, because the maximum number of jobs previously treated by the works in the literature was 500, using heuristic approaches in instances with the same parameters.

As future research, we would like to investigate if the good performance of the models presented can be replicated when symmetry breaking constraints and the arc-flow approach are applied to other problems in the vast area of scheduling applications as, for example, the problem of scheduling a batch processing machine with incompatible job families.

Bibliography

[1] CHEN, H., DU, B., HUANG, G. Q. “Scheduling a batch processing machine with non-identical job sizes: a clustering perspective”, *International Journal of Production Research*, v. 49, n. 19, pp. 5755–5778, oct 2011. ISSN: 0020-7543, doi: 10.1080/00207543.2010.512620.

[2] UZSOY, R. “Scheduling a single batch processing machine with non-identical job sizes”, *International Journal of Production Research*, v. 32, n. 7, pp. 1615–1635, jul 1994. ISSN: 0020-7543, doi: 10.1080/00207549408957026.

[3] LEE, C.-Y., UZSOY, R., MARTIN-VEGA, L. A. “Efficient Algorithms for Scheduling Semiconductor Burn-In Operations”, *Operations Research*, v. 40, n. 4, pp. pp. 764–775, aug 1992. ISSN: 0030364X, doi: 10.1287/opre.40.4.764.

[4] TAI, Y. *The Study on the Production Scheduling Problems for Liquid Crystal Display Module Assembly factories.* Ph.D. Thesis, National Chiao Tung University, jul 2008.

[5] CHUNG, S., TAI, Y., PEARN, W. “Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes”, *International Journal of Production Research*, v. 47, n. 18, pp. 5109–5128, sep 2009. ISSN: 0020-7543, doi: 10.1080/00207540802010807.

[6] MELOUK, S., DAMODARAN, P., CHANG, P.-Y. “Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing”, *International Journal of Production Economics*, v. 87, n. 2, pp. 141–147, jan 2004. ISSN: 09255273, doi: 10.1016/S0925-5273(03)00092-6.

[7] CHANG, P.-Y., DAMODARAN *, P., MELOUK, S. “Minimizing makespan on parallel batch processing machines”, *International Journal of Production Research*, v. 42, n. 19, pp. 4211–4220, oct 2004. ISSN: 0020-7543, doi: 10.1080/00207540410001711863.

[8] XU, R., CHEN, H., LI, X. “Makespan minimization on single batch-processing machine via ant colony optimization”, *Computers & Operations Research*, v. 39, n. 3, pp. 582–593, mar 2012. ISSN: 03050548, doi: 10.1016/j.cor.2011.05.011.

[9] VÉLEZ GALLEGO, M. C. *Algorithms for Scheduling Parallel Batch Processing Machines with Non-Identical Job Ready Times*. Ph.D. Thesis, Florida International University, mar 2009, doi: 10.25148/etd.FI10022555.

[10] MARGOT, F. “Symmetry in Integer Linear Programming”. In: *50 Years of Integer Programming 1958-2008*, Springer Berlin Heidelberg, pp. 647–686, Berlin, Heidelberg, nov 2010. ISBN: 978-3-540-68279-0, doi: 10.1007/978-3-540-68279-0_17.

[11] VALÉRIO DE CARVALHO, J. “Exact solution of cutting stock problems using column generation and branch-and-bound”, *International Transactions in Operational Research*, v. 5, n. 1, pp. 35–44, 1998. ISSN: 1475-3995, doi: 10.1111/j.1475-3995.1998.tb00100.x.

[12] VALÉRIO DE CARVALHO, J. “Exact solution of bin-packing problems using column generation and branch-and-bound”, *Annals of Operations Research*, v. 86, pp. 629–659, jan 1999. ISSN: 02545330, doi: 10.1023/A:1018952112615.

[13] BRANDÃO, F., PEDROSO, J. P. “Bin packing and related problems: General arc-flow formulation with graph compression”, *Computers and Operations Research*, v. 69, pp. 56–67, may 2016. ISSN: 03050548, doi: 10.1016/j.cor.2015.11.009.

[14] IBM CORP. *IBM ILOG CPLEX 12.7 Optimization User’s Manual*. 2016.

[15] TRINDADE, R. S., DE ARAÚJO, O. C. B., FAMPA, M. H. C., et al. “Modelling and symmetry breaking in scheduling problems on batch processing machines”, *International Journal of Production Research*, v. 56, n. 22, pp. 7031–7048, nov 2018. ISSN: 0020-7543, doi: 10.1080/00207543.2018.1424371.

[16] GRAHAM, R., LAWLER, E., LENSTRA, J., et al. “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey”. In: *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada*

and SIAM Banff, Aha. and Vancouver, v. 5, pp. 287–326, 1979, doi: 10.1016/S0167-5060(08)70356-X.

[17] PINEDO, M. L. *Scheduling: Theory, algorithms, and systems, fifth edition*. Cham, Springer International Publishing, 2016. ISBN: 9783319265803, doi: 10.1007/978-3-319-26580-3.

[18] POTTS, C. N., KOVALYOV, M. Y. "Scheduling with batching: A review", *European Journal of Operational Research*, v. 120, n. 2, pp. 228–249, jan 2000. ISSN: 03772217, doi: 10.1016/S0377-2217(99)00153-8.

[19] MATHIRAJAN, M., SIVAKUMAR, A. "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor", *The International Journal of Advanced Manufacturing Technology*, v. 29, n. 9-10, pp. 990–1001, jan 2006. ISSN: 0268-3768, doi: 10.1007/s00170-005-2585-1.

[20] MÖNCH, L., FOWLER, J. W., DAUZÈRE-PÉRÈS, S., et al. "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations", *Journal of Scheduling*, v. 14, n. 6, pp. 583–599, jan 2011. ISSN: 1094-6136, doi: 10.1007/s10951-010-0222-9.

[21] GHAZVINI, F. J., DUPONT, L. "Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes", *International Journal of Production Economics*, v. 55, pp. 273–280, aug 1998, doi: 10.1016/S0925-5273(98)00067-X.

[22] ZHANG, G., CAI, X., LEE, C.-Y., et al. "Minimizing makespan on a single batch processing machine with nonidentical job sizes", *Naval Research Logistics*, v. 48, n. 3, pp. 226–240, apr 2001. ISSN: 0894-069X, doi: 10.1002/nav.4.

[23] DAMODARAN, P., KUMAR MANJESHWAR, P., SRIHARI, K. "Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms", *International Journal of Production Economics*, v. 103, n. 2, pp. 882–891, oct 2006. ISSN: 09255273, doi: 10.1016/j.ijpe.2006.02.010.

[24] KASHAN, A. H., KARIMI, B., JOLAI, F. "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes", *International Journal of Production Research*, v. 44, n. 12, pp. 2337–2360, jun 2006. ISSN: 0020-7543, doi: 10.1080/00207540500525254.

[25] MENG, Y., TANG, L. “A tabu search heuristic to solve the scheduling problem for a batch-processing machine with non-identical job sizes”. In: *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*, v. 3, pp. 1703–1707. IEEE, jan 2010. ISBN: 978-1-4244-7331-1. doi: 10.1109/ICLSIM.2010.5461274.

[26] DAMODARAN, P., GHRAYEB, O., GUTTIKONDA, M. C. “GRASP to minimize makespan for a capacitated batch-processing machine”, *The International Journal of Advanced Manufacturing Technology*, v. 68, n. 1-4, pp. 407–414, jan 2013. ISSN: 0268-3768, doi: 10.1007/s00170-013-4737-z.

[27] AL-SALAMAH, M. “Constrained binary artificial bee colony to minimize the makespan for single machine batch processing with non-identical job sizes”, *Applied Soft Computing*, v. 29, pp. 379–385, apr 2015. ISSN: 15684946, doi: 10.1016/j.asoc.2015.01.013.

[28] LI, X. L., LI, Y. P., WANG, Y. “Minimising makespan on a batch processing machine using heuristics improved by an enumeration scheme”, *International Journal of Production Research*, v. 55, n. 1, pp. 176–186, jun 2017. ISSN: 1366588X, doi: 10.1080/00207543.2016.1200762.

[29] LEE, Y. H., LEE, Y. H. “Minimising makespan heuristics for scheduling a single batch machine processing machine with non-identical job sizes”, *International Journal of Production Research*, v. 51, n. 12, pp. 3488–3500, jun 2013. ISSN: 0020-7543, doi: 10.1080/00207543.2012.748226.

[30] RAFIEE PARSA, N., KARIMI, B., HUSSEINZADEH KASHAN, A. “A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes”, *Computers & Operations Research*, v. 37, n. 10, pp. 1720–1730, oct 2010. ISSN: 03050548, doi: 10.1016/j.cor.2009.12.007.

[31] KASHAN, A. H., KARIMI, B., JENABI, M. “A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes”, *Computers & Operations Research*, v. 35, n. 4, pp. 1084–1098, apr 2008. ISSN: 03050548, doi: 10.1016/j.cor.2006.07.005.

[32] DAMODARAN, P., HIRANI, N. S., GALLEGO, M. C. V. “Scheduling identical parallel batch processing machines to minimise makespan using genetic algorithms”, *European Journal of Industrial Engineering*, v. 3, n. 2, pp. 187, sept 2009. ISSN: 1751-5254, doi: 10.1504/EJIE.2009.023605.

[33] CHENG, B., YANG, S., HU, X., et al. “Minimizing makespan and total completion time for parallel batch processing machines with non-identical job sizes”, *Applied Mathematical Modelling*, v. 36, n. 7, pp. 3161–3167, jul 2012. ISSN: 0307904X, doi: 10.1016/j.apm.2011.09.061.

[34] CHENG, B., WANG, Q., YANG, S., et al. “An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes”, *Applied Soft Computing*, v. 13, n. 2, pp. 765–772, feb 2013. ISSN: 15684946, doi: 10.1016/j.asoc.2012.10.021.

[35] JIA, Z.-H., LEUNG, J. Y.-T. “A meta-heuristic to minimize makespan for parallel batch machines with arbitrary job sizes”, *European Journal of Operational Research*, v. 240, n. 3, pp. 649–665, feb 2015. ISSN: 03772217, doi: 10.1016/j.ejor.2014.07.039.

[36] CHOU, F.-D., CHANG, P.-C., WANG, H.-M. “A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem”, *The International Journal of Advanced Manufacturing Technology*, v. 31, n. 3-4, pp. 350–359, nov 2005. ISSN: 0268-3768, doi: 10.1007/s00170-005-0194-7.

[37] ZHOU, S., CHEN, H., XU, R., et al. “Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes”, *International Journal of Production Research*, v. 52, n. 8, pp. 2258–2274, apr 2014. ISSN: 0020-7543, doi: 10.1080/00207543.2013.854937.

[38] DAMODARAN, P., VÉLEZ-GALLEGO, M. C., MAYA, J. “A GRASP approach for makespan minimization on parallel batch processing machines”, *Journal of Intelligent Manufacturing*, v. 22, n. 5, pp. 767–777, jun 2009. ISSN: 0956-5515, doi: 10.1007/s10845-009-0272-z.

[39] KÖHLER, V., FAMPA, M., ARAÚJO, O. “Mixed-Integer Linear Programming Formulations for the Software Clustering Problem”, *Computational Optimization and Applications*, v. 55, n. 1, pp. 113–135, oct 2012. ISSN: 0926-6003, doi: 10.1007/s10589-012-9512-9.

[40] BRANDÃO, F. D. A. *Cutting & Packing Problems: General Arc-flow Formulation with Graph Compression*. Ph.D. Thesis, Universidade do Porto, 2017.

[41] DELORME, M., IORI, M., MARTELLO, S. “Bin packing and cutting stock problems: Mathematical models and exact algorithms”, *European Journal*

of Operational Research, v. 255, n. 1, pp. 1–20, nov 2016. ISSN: 03772217, doi: 10.1016/j.ejor.2016.04.030.

[42] MARTINOVIC, J., SCHEITHAUER, G., VALÉRIO DE CARVALHO, J. M. "A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems", *European Journal of Operational Research*, v. 266, n. 2, pp. 458–471, apr 2018. ISSN: 03772217, doi: 10.1016/j.ejor.2017.10.008.

[43] KRAMER, A., DELL'AMICO, M., IORI, M. "Enhanced arc-flow formulations to minimize weighted completion time on identical parallel machines", *European Journal of Operational Research*, v. 275, n. 1, pp. 67–79, may 2019. ISSN: 03772217, doi: 10.1016/j.ejor.2018.11.039.

[44] MRAD, M., SOUAYAH, N. "An Arc-Flow Model for the Makespan Minimization Problem on Identical Parallel Machines", *IEEE Access*, v. 6, pp. 5300–5307, jan 2018. ISSN: 21693536, doi: 10.1109/ACCESS.2018.2789678.

[45] DANTZIG, G. B., WOLFE, P. "Decomposition Principle for Linear Programs", *Operations Research*, v. 8, n. 1, pp. 101–111, feb 1960. ISSN: 0030-364X, doi: 10.1287/opre.8.1.101.

[46] MARTELLO, S., TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA, John Wiley & Sons, Inc., 1990. ISBN: 0-471-92420-2.