**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Edson Antônio Gonçalves de Souza

**Bounded Dynamic Programming Approach to minimize
Makespan for the Blocking Flow Shop problem with
Sequence Dependent Setup Times Constraints**

São Carlos

2021

Edson Antônio Gonçalves de Souza

**Bounded Dynamic Programming Approach to minimize
Makespan for the Blocking Flow Shop problem with
Sequence Dependent Setup Times Constraints**

Master thesis presented to the School of Engineering of São Carlos of University of São Paulo in order to obtain the master's degree.
(This version is the one that has undergone corrections)

Research Field: Processes and Operations Management

Supervisor: Prof. Dr. Marcelo Seido Nagano

**São Carlos
2021**

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

- 635 Souza, Edson Antônio Gonçalves
Bounded Dynamic Programming Approach to minimize Makespan for the Blocking Flow Shop problem with Sequence Dependent Setup Times Constraints / Edson Antônio Gonçalves de Souza ; orientador Marcelo Seido Nagano. – São Carlos, 2021.
157 p.
- Master Thesis (Master's Degree) - Research Field in Processes and Operations Management – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2021.
1. blocking flow shop. 2. makespan. 3. Bounded Dynamic Programming. 4. setup times. 5. MILP. 6. B&B. I. Title

FOLHA DE JULGAMENTO

Candidato: Engenheiro **EDSON ANTONIO GONÇALVES DE SOUZA.**

Título da dissertação: "Análise de *Bounded Dynamic Programming* com minimização do *Makespan* para o problema de *Flow Shop* com bloqueio e restrições de tempos de *setup* dependentes da sequência".

Data da defesa: 14.05.2021

Comissão Julgadora

Resultado

Prof. Dr. **Marcelo Seido Nagano**

(Orientador)

(Escola de Engenharia de São Carlos/EESC-USP)

Aprovado

Prof. Dr. **Roberto Fernandes Tavares Neto**

(Universidade Federal de São Carlos/UFSCar)

Aprovado

Prof. Associado **José Francisco Ferreira Ribeiro**

(Faculdade de Economia, Administração e Contabilidade/FEA-USP)

Aprovado

Coordenadora do Programa de Pós-Graduação em Engenharia de Produção:

Profa. Dra. **Janaina Mascarenhas Hornos da Costa**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

*This thesis is dedicated to those who unwillingly have to face
the prejudice society imposes upon them constantly, however always find a way
to overcome those adversities and pave a better path to future generations.*

ACKNOWLEDGEMENTS

Over the past three years, I have been shown a world of possibilities that have made me realize that stepping out of my comfort zone essentially shaped me into a better version of myself. Despite being a rather slow and massive process, being part of the academic scenario had me reflect upon how much I have evolved and the depths I was able to reach. Occasionally, the characteristics that developed me into a researcher began to appear and, as subtle as these may seem, they are now imprinted in my core. The process of changing, however, did not depend exclusively on myself or few elements that are intrinsic to my research or the academic environment I am inserted in. It also depended on beings that I hold dear to my heart and have been pivotal in my evolution and the path I will follow next. Being able to design such work has demanded heavily from each of you, however I can gladly say that with your help, I am delivering a thesis I am proud of and therefore, I hope I can convey, through some words, my gratitude to you and your contribution.

First, I would like to thank the Universe for bestowing upon me time so that I could grow a more mature person regarding the ordeals that I have found throughout this period of my life, patience so I could understand that moving forward to the next steps may require the thorough comprehension of those that have been previously learned, even if they are just suitable for knowledge purposes at the moment, and ultimately energy so that I could learn how to deal with either ecstatic and discouraging phases that made part of this process.

To my family, my most wholehearted appreciation for playing an incredible role in supporting me when I decided to enter the masters program. To my parents, Edson and Ana Maria, and my grandma Luzia, who have always been patient and understanding towards stressful situations and, their motivational talks whenever it felt dismaying and uncomfortable but also for their proud eyes that kept pushing me forward. Additionally, I would also like to thank my aunt Maria de Lourdes and my cousin Mariana Santos for being such inspirations professionally and for their thoughtful insights regarding the academia and other aspects of life.

To Professor Dr. Marcelo Seido Nagano for being a mentor in the technical aspects of the topics that are comprised in the research as well as for being always kind and patient whenever challenging situations surfaced. Also, for the opportunity of helping me develop self-confidence and autonomy in a professional sense. Additionally, to my graduation supervisor, Professor Dr. Antônio Francisco Neto, my deepest gratitude for the mathematical formation in topics that have been crucial in this research and also for having encouraged me to broaden my horizons academically.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) - under grant number 88882.379101/2019-01.

To my friend Salvador Carvalho, who might, in fact, be the reason I am part of this program. Thank you for being the one who practically took me by the hand and had me apply for the selection process. You are truly an inspiration for me in terms of achievements, focus and determination.

To all my friends from Ouro Preto, especially Guilherme Tsuzuki, Laís Piol, Luan Pompeu, Mayara Fernanda, Mayra Cristina and Vitória Jeronimo who have, since the beginning of this journey, shown unconditional support and have also been there as a safe harbor by demonstrating kindness, sharing positive thoughts and boosting my morale whenever I needed.

I would also like to thank my Science without Borders, especially Bárbara Tobias, Markus Algayer, Olivia Aires and Rena Salzmänn who were essential in matters (related or unrelated to the research) that had been simultaneously set in motion. Dealing with all these feelings helped me become a much more focused and stronger person.

To all my lab (LAOR) partners who have become incredible friends over time: Caio Tomazella, Fernando Siqueira, Gustavo Rolim, Hugo Miyata, Marina Paiva, Rainne Florisbelo, Viviane Junqueira. Although the environment we found ourselves in demanded heavily from us, we managed to make these moments as smooth as possible by sharing work and personal experiences, laughter, knowledge and sometimes despair. Thank you for everything you have taught me and also for making this journey more pleasant. Furthermore, I would like to express my special gratitude for Caio, who has assisted me with programming tips that were crucial for the improvement of the algorithm in this research and also Hugo, who has made every possible effort to share his knowledge in scheduling, programming and statistics, hence leading this research to better standards.

Thank you Bruna Garrett, Maiara Rosa and Renato Naves for all the joyous moments and discussions we had, for being such incredible friends and hosts and for making my moments in Araraquara the best. Being able to create bonds like we did has certainly connected me with values such as thoughtfulness, patience and awareness.

I would like to thank Victor Hugo, who has welcomed me into his home with such an open heart and represented for me more than a roommate. You are a true friend who has always cared for me and also made me someone better by showing me how important chasing dreams one after another is.

To Angelica Camargo, Djaci Augusto, Felipe Rinaldi, Gustavo Freitas and Rachel Marascalchi. I wish I could express everything you represent to me because you are way more deserving than a paragraph, however I have neither space nor time to do so. For now I can only thank you for being the best people São Carlos could have gifted me with and

that you have been the ones who showed me how magical and special a place can be if you live intensely. Thank you for showing me that no matter how far from home you are, it is always possible to find a family in friends that are willing to advise, listen, trust and help you evolve in every sense of the word.

Last but not least, I would like to thank my dearest friend Paula Cavalcante, who has been since the beginning, the key to everything. You were there way before it all started and when I moved to São Carlos you were the missing piece of the puzzle that connected all the people surrounding me. Having you around definitely made it all much lighter and simpler. You have been there through the best and the worst, the chaos and the calm. Thank you for never abandoning me, for being optimistic when I felt desperate and for celebrating my small victories as if they were the greatest accomplishments because it all pushed me forward towards the finish line.

*“Nothing in life is to be feared, it is only to be understood.
Now is the time to understand more, so we may fear less.”*
Marie Curie

ABSTRACT

SOUZA, EAG. **Bounded Dynamic Programming Approach to minimize Makespan for the Blocking Flow Shop problem with Sequence Dependent Setup Times Constraints.** 2021. 157p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

Throughout the decades, the flow shop scheduling problem has appeared in a variety of manufacturing scenarios and, as a consequence of that fact, several branches of it have taken place and are frequently studied by researchers so as to develop efficient methods to solve these problems that stem from the classic version. A problem that is frequently found on the shop floor is the blocking flow shop with sequence dependent setup times (BFSP-SDST) with makespan optimization, which can be found on electronics, metallurgy, chemical and food industries and it comes to show the importance of improving systems that can be modelled via this problem. However, when gathering information on state-of-art references, one will notice that little has been made regarding BFSP-SDST and, consequently, only few methods have been applied with the intent of optimizing makespan. Furthermore, research has shown that dynamic programming is a method that has not been given much attention with regard to the flow shop environment and therefore, its contribution has become, thus far, limited. Hence, this research aims at applying Bounded Dynamic Programming (BDP), which is a DP-based method, to solve the BFSP-SDST so as to enlarge the number of methods that are related to such problem. The BDP is employed in two sets of problems and for the first set, it is compared to a MILP and B&B methods, while the comparison occurs only between the BDP and MILP for the second set. The results have shown that BDP outperforms the MILP in both scenarios and the B&B in terms of CPU times and success rate. In addition, a trade-off analysis is provided in order to determine which method is a better choice for the scheduler in terms of solution quality and CPU times and, once again, BDP proves to be better for the job and machine sets considered. Sequentially, some suggestions are furnished to improve the method and develop further research considering this approach.

Keywords: blocking flow shop, makespan, Bounded Dynamic Programming, setup times, MILP, B&B.

RESUMO

SOUZA, EAG. *Análise de Bounded Dynamic Programming com minimização do Makespan para o problema de Flow Shop com bloqueio e restrições de tempos de setup dependentes da sequência*. 2021. 157p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

Ao longo das décadas, o problema de sequenciamento em ambiente *flow shop* tem aparecido em uma variedade de cenários de manufatura e, como consequência, diversas ramificações se estabeleceram, sendo frequentemente estudadas por pesquisadores com o intuito de desenvolver métodos eficientes para resolver esses problemas originados da versão clássica do *flow shop*. Um problema comumente encontrado no chão de fábrica é o *flow shop* com bloqueio e tempos de *setup* dependentes da sequência (BFSP-SDST) com minimização do *makespan*, que pode ser encontrado em indústrias de eletrônicos, metalúrgicas, química e de alimentos e portanto, demonstra a importância de aperfeiçoar sistemas que possam ser modelados através do mesmo. Todavia, ao organizar informações de referências que são estado-da-arte, pode-se notar poucas referências acerca do BFSP-SDST e, por consequência, apenas poucos métodos foram aplicados com o intuito de minimizar o *makespan* para tal problema. Além do mais, as pesquisas mostraram que a programação dinâmica não tem recebido devida atenção quando se trata do ambiente *flow shop* e portanto, sua contribuição se tornou limitada, até o momento. Logo, essa pesquisa tem como objetivo aplicar *Bounded Dynamic Programming* (BDP), que é um método baseado em programação dinâmica, como solução para o BFSP-SDST para aumentar o número de métodos de solução para tal problema. O BDP é empregado em dois conjuntos de instâncias e, para o primeiro conjunto, o mesmo é comparado com os métodos MILP e B&B, enquanto a comparação ocorre apenas entre BDP e MILP para o segundo conjunto. Os resultados mostraram que BDP supera o MILP em ambos os cenários e o B&B em termos de eficiência computacional e qualidade de solução através da taxa de acerto do algoritmo. Adicionalmente, uma análise de *trade-off* é fornecida para se determinar qual método representa uma melhor escolha para o programador em termos de qualidade da solução e esforço computacional e, mais uma vez, BDP se mostra uma escolha melhor para os conjuntos de tarefas e máquinas considerados. Sequencialmente, algumas sugestões são fornecidas para refinar o método e desenvolver futuras pesquisas considerando essa abordagem.

Palavras-chave: *flow shop* com bloqueio, *makespan*, *Bounded Dynamic Programming*, tempos de *setup*, MILP, B&B.

LIST OF FIGURES

Figure 1 – Schematics of dominance rule and transitional state in the graph structure	103
Figure 2 – Representation of BDP and its perspective as a heuristic procedure considering the criteria for space state reduction.	105
Figure 3 – Boxplot relative to the RPD data	122
Figure 4 – Performance analysis with regard to ARPD and ARPT	124
Figure 5 – ARPD variation with respect to the number of jobs	125
Figure 6 – ARPD variation with respect to the first group of machines	132
Figure 7 – ARPD variation with respect to the second group of machines	132
Figure 8 – Boxplot relative to the RPD data for the second experimental set . . .	133
Figure 9 – Tradeoff with respect to ARPD and ARPT for the second experimental set	134
Figure 10 – ARPD variation with respect to the number of jobs for the second experimental set	135
Figure 11 – ARPD variation with respect to the number of machines for the second experimental set	136
Figure 12 – ARPD variation with respect to the setup rates for the second experi- mental set	137

LIST OF TABLES

Table 1 – Due-date related problems for single machine	43
Table 2 – Number of tardy jobs related problems for single machine	46
Table 3 – Completion Time related problems for single machine	48
Table 4 – Makespan problems for single machine	50
Table 5 – Multicriteria and Additional Functions problems for single machine . . .	58
Table 6 – Due date related problems for parallel machines	62
Table 7 – Number of tardy jobs related problems for parallel machines	65
Table 8 – Completion Time related problems for parallel machines	70
Table 9 – Makespan problems for parallel machines	74
Table 10 – Multicriteria and Additional Functions problems for parallel machines .	79
Table 11 – Due date related problems for flow shop	82
Table 12 – Makespan and Completion Time related problems for flow shop	86
Table 13 – Multicriteria and Additional Functions problems for flow shop	90
Table 14 – Problems in the job-shop environment	94
Table 15 – Problems in the open-shop environment	98
Table 16 – General data for the first experimental set	117
Table 17 – Verification for ANOVA	118
Table 18 – Pairwise comparison between methods according to Dunn’s Test	119
Table 19 – General data for the second experimental set	126
Table 20 – General data for the first experimental set	126
Table 21 – Pairwise comparison between methods according to Dunn’s Test for the second set	127

CONTENTS

1	INTRODUCTION	23
2	MACHINE SCHEDULING PRELIMINARIES	29
3	DYNAMIC PROGRAMMING APPLIED TO MACHINE SCHEDULING PROBLEMS: A REVIEW	35
3.1	Dynamic Programming	35
3.2	Literature Review Methodology	37
3.3	Dynamic Programming Approach for the Single Machine Environment	40
3.3.1	Classic due-dates related criteria	40
3.3.2	Number of Tardy Jobs	44
3.3.3	Completion Time and Flow Time	45
3.3.4	Makespan	48
3.3.5	Multi-criteria and Additional Objective Functions	49
3.4	Dynamic Programming approach for the Parallel Machine Environment	60
3.4.1	Classic due date related criteria	60
3.4.2	Number of Tardy Jobs	64
3.4.3	Completion time related criteria	66
3.4.4	Makespan	72
3.4.5	Multi-criteria and Additional Objective Functions	75
3.5	Dynamic Programming approach for the Flow Shop Environment .	81
3.5.1	Classic due dates-related criteria	81
3.5.2	Makespan and Completion Time-related criteria	83
3.5.3	Multi-criteria and Additional Objective Functions	87
3.6	Dynamic Programming Approach for the Job-Shop Environment . .	91
3.6.1	Classic due dates-related criteria	91
3.6.2	Makespan and Completion Time-related criteria	91
3.6.3	Multi-criteria and Additional Objective Functions	92
3.7	Dynamic Programming approach for the Open Shop Environment .	95
3.7.1	Classic due dates-related criteria	95
3.7.2	Makespan and Completion Times-related criteria	95
3.7.3	Multi-criteria and Additional Objective Functions	96

4	BOUNDED DYNAMIC PROGRAMMING APPLIED TO THE BLOCK- ING FLOW SHOP PROBLEM WITH SEQUENCE DEPENDENT SETUP TIMES	99
4.1	Problem Statement and Notations	100
4.2	Bounded Dynamic Programming for the $Fm block C_{\max}$ problem . .	101
4.3	Bounded Dynamic Programming extended for the $Fm s_{ijk}, block C_{\max}$ problem	105
5	METHODOLOGY	111
5.1	Computational Experiments	111
5.1.1	First set of problems	111
5.1.2	Second set of problems	113
5.2	Statistical Analysis	114
6	DISCUSSION AND RESULTS	117
6.1	First experimental set	117
6.2	Second experimental set	124
7	CONCLUSION	139
	BIBLIOGRAPHY	143

1 INTRODUCTION

As any other scenario in industry, the categorization of areas is necessary in order to guarantee that the product delivered to the costumer fits the requirements that have been established in the beginning of the supply chain. In classic industrial engineering those areas were often divided according to their qualitative (e.g. science management, ergonomics, engineering design process) or quantitative (e.g. operations research, quality engineering, logistics and supply chain) characteristics. However, modern industrial engineering refers to these areas as integrated portions of knowledge that combine management analysis, mathematical tools, modelling and computational methods and economical feasibility to optimize outputs throughout each step of the process (KÁDÁROVÁ et al., 2014).

Through these elements, one should be able to identify aspects that are usually studied in operations research. In fact, since its development, operations research has been quite productive despite the growing number of papers published within the area on a regular basis. This can be explained by the amount of problems in literature that have not yet been solved or might have their solutions improved regarding computational time, computational storage or even values (in case of heuristics). Furthermore, these problems are derived from daily problems that occur in several segments of industrial processes and are often mishandled.

In manufacturing systems, production planning and control is the area that plays a role of integration since it can be partitioned in three levels: operational, tactical and strategical. An optimal coordination among these three levels is responsible for placing companies ahead in competitiveness and, consequently, allowing these companies to own the largest shares of the market.(FUCHIGAMI; RANGEL, 2018) highlights that this is one of the reasons why decision making process is supplemented by operations research tools, modelling and simulation. At the operational level, one of the subjects that is frequently studied is **machine scheduling**.

According to (PINEDO, 2012), scheduling is a decision making process used regularly in a variety of manufacturing and services industries. It can be defined as the allocation of resources to tasks over given time periods aiming at optimizing a single objective function or a multi-objective function. The perspective of what is considered a resource varies according to the environment assessed. For instance, in the shop floor, machines are resources whereas runways are the analogous counterparts when considering airports systems. For this reason, as mentioned in (ABEDINNIA; GLOCK; SCHNEIDER, 2017), scheduling is widely applied in different fields such as project management, aerospace industry, computer science and personnel management.

Due to the behaviour of particular systems, it is paramount for the scheduler to be able to identify which aspects of the industrial scenario one might be dealing with. One of the settings that is essential in this investigation is the environment, which is associated with the number of machines available and the disposition of what is likely to be main element of scheduling, namely, jobs. For this work, the particular interest is taken in studying the **Flow shop** environment, which is one of the most popular classes in machine scheduling and that has largely contributed to the development of more complex shop environments.

Such environment has been progressively seen in either simple or sophisticated industry sections. (YEUNG; OĞUZ; CHENG, 2009) describe a series of applications in which the systems can be modelled as a flow shop problem, e.g. fabric, metallurgy, food, chemical and electronic components. Despite the differences in the final products, the modelling of these systems are treated equally and their only dissimilarities are regarding the objective function to be optimized and whether technological constraints are defined in the manufacturing process.

As it can be fathomed, the nature of the objective function poses a crucial role in the optimization model, since the singularities that each of them possesses can alter the pathway for solving a given problem. It is also interesting to consider which objective functions should be prioritized, once attempting to optimize a large number of those might be infeasible from several perspectives. In addition the inclusion of technological constraints can be accountable for defining or altering the complexity of a certain problem, because the more realistic the mathematical modelling becomes the higher the computational effort is, even for small instances of a job or machine set. Hence, one of the reasons that several problems are still open in literature or new ones are regularly proposed.

In order to picture the elements of the concepts previously mentioned, let us imagine the following situation: a pipe manufacturing facility receives a shipping with raw plastic material whose final products are pipes with different measurements in terms of radius, size and colours. Suppose that the contractor has demanded six different types of pipes that should undergo transformation in three machines. The first one performs a drilling activity to develop pipes with different radius, the second machine is accounted for the cutting activity and the last one paints the pipes with different colours regarding their technical specifications. Clearly, length, radius and colour are variables that define distinct processing times and, to add more realistic features to the problem, preparation is needed when a specific pipe arrives at a given machine to be processed. Furthermore, the order established for pipes is the same for all machines and there exists no buffers in between machines and an additional time might be incurred because if the next machine has not finished processing the last pipe or the preparation is ongoing, the current pipe is kept in the current machine until the next machine is available for processing.

These characteristics, despite being simple, describe the elements intrinsic to machine scheduling and are able to relate it to a real application. Each different specification of pipe represents a job, totalling six of those. There exists three machines assigned to these jobs and the serial configuration following the same pattern for every machine designates the flow shop environment. In addition, the preparation times and the zero buffer constraints are precisely referred to as setup times and blocking, respectively. Various other constraints could have been added but for sake of simplicity and relation to the problem that will be studied here we shall keep this visualization.

Since the interest of this research is to analyze a problem derived from flow shop scheduling, some information can be extracted from (FUCHIGAMI; RANGEL, 2018), who presented a survey in which the authors categorize the machine scheduling problems according to environment and objective functions, among others. The statistics showed that the flow shop environment is accountable for 17.65% of papers related to machine scheduling and it is the most deeply investigated among the shop environments. Another topic entailed by the authors is the use of diverse methods, which are classified in exact and non-exact (heuristics and metaheuristics), in order to solve a given problem subject to a single or multi-objective function and once again the odds are in favour of the non-exact methods, and even within these, some sub-methods have been more recurrent than others due to their probability of success for related problems.

In (NEUFELD; GUPTA; BUSCHER, 2016), a literature review on solution methods for various classes of flow shop problems is presented. Among these methods, there are some that have not been deeply explored due to some drawbacks that are intrinsic to their formulations. Dynamic programming is a type of method that falls into this category due to an effect called curse of dimensionality. This curse is similar to a trade-off that occurs between the exactness of solution and the complexity of the algorithm. In other words, the cost for finding an optimal solution is often linked to an algorithm that will demand both a large amount of computational time and storage as the number of instances become larger and this is one of the reasons why dynamic programming is often combined with some methods to create a more powerful tool in the search for optimal solutions. However, one of the advantages of this approach is that, once the problem can be partitioned into sub-problems and the optimal solution can be found for every sub-problem, then that the optimal solution can also be found for the original proposed problem via a collection of elements and values that are recursively stored.

A parallel can be established between flow shop scheduling problem and the dynamic programming premises. Partition can be conceived as the allocation of jobs at each stage. By making the best decision, there is a guarantee that the optimal solution is achieved and the cost is minimum for a given sub-problem. Furthermore the jobs that have already been selected are stored and their costs are added to the minimum cost attributed to a current

stage. Hence, flow shop scheduling fits the criteria necessary to incorporate dynamic programming as a solution method. The question that arises from this concatenation of ideas is: how could dynamic programming be applied as a tool in order to minimize classic performance measures in a flow shop scheduling problem and some variants of it?

Perhaps, applying pure dynamic programming formulations might not display the most efficient resolutions, however the use of its principles to construct hybrid algorithms have recently proven their advantage in improving solutions regarding problems involving flow shop. In fact, the association of methods that can guarantee high quality solutions, even though optimality is not reached for every case within a competitive time, is essential and often needed to sustain further approaches that might be developed as well as practical studies. Based on this information, the definition of the problem led to the investigation of a rather common problem in the shop floor, whereas with few occurrences in literature, which is the blocking flow shop with sequence dependent setup times problem aiming at minimizing the makespan. Logically, a solution method is needed and, in this case, a hybrid algorithm derived from a junction of Branch and Bound and Dynamic Programming and has been recently denominated Bounded Dynamic Programming.

Roughly explained, Bounded Dynamic Programming incorporates a graph-like structure that relies on two linked lists to develop states between two consecutive stages and a bounding scheme in order to reduce the state space. Additional features such as maximum size of states per stage and dominance rules are included so as to make the reduction more efficient. According to (BAUTISTA et al., 2012), the method might be either exact or heuristic if a given set of conditions is satisfied. Recently, this method has been an alternative solution when Dynamic Programming is involved due to its faster convergence to solutions and also the algorithm does not always require a recursive relation, which has been recognized as the main weighing factor for the curse of dimensionality.

Hence, this research can be divided in two objectives. The general objective is to derive a Bounded Dynamic Programming algorithm that can efficiently solve the blocking flow shop problem with sequence dependent setup times and compare the results obtained for other methods offered by sources found in literature considering the same range of jobs and machines. The specific objectives are outlined in the following manner:

- Gather relevant information, via state-of-art papers, on dynamic programming applications within the machine scheduling environment so as to compose a review paper, which has not been yet developed;
- Implement the Bounded Dynamic Programming algorithm proposed with an appropriate bounding scheme that includes the sequence dependent setup times to provide high quality solution efficiently for the BDP;

- Implement a Branch and Bound method and a Mixed Integer Linear Programming formulation for two sets of problems with prespecified characteristics on number of jobs, machines and setup times variation;
- Verify statistically the computational performance of the proposed method when tested against the other implemented methods;
- Propose suggestions for enhancing the proposed algorithm's capacity for jobs and machines that should be used in further studies.

For closure of this chapter, this master thesis will be outlined as follows: Section 2 explains briefly machine scheduling and notations. Section 3 is relative to the literature review involving dynamic programming applied to several problems in machine scheduling, which are divided externally by environment and internally by objective functions. Section 4 establishes a complete description of the problem investigated in this research as well as the method that is applied to solve it. Section 5 gives a detailed analysis of the methodology in terms of statistical and computational perspectives to familiarize the reader of the methods applied to derive the results necessary to verify the algorithm's efficiency. Section 6 shows the statistical evaluation of the methods that have been programmed and establishes a comparison in terms of CPU times and solution quality. Section 7 adds suggestions on future undertakings regarding this topic and other that might be derived from the work presented here.

2 MACHINE SCHEDULING PRELIMINARIES

Machine scheduling is defined as the allocation of jobs that will be processed on a set of machines responsible for performing a specific operation on each job. Because the order is unknown, the problem aims at establishing a sequence of jobs that will result in the minimum value (or an approximation) for a given performance measure or criterion (BŁAŻEWICZ et al., 1988), which are functions dependent on time. It is a classical set of problem in operations research and since its first occurrences, dating back to the fifties, extensive research in the area has been done and improved. Recently, theoretical and practical approaches have been converging and experts have shown more interest in developing algorithms in order to obtain better results for practical situations at manufacturing workstations. Here jobs will be denoted by $j \in J$, machines will be denoted by $k \in M$ and job sequences will be represented by $\pi \in \Pi$, with J being the set of jobs, M being the set of machines and Π being the set of all possible sequences.

The initial papers in machine scheduling were essentially a compilation of notes that featured strong mathematical background and properties about the behaviour of systems and also combinatorial nature of the problem (see (SMITH, 1956)). Over the years, with more advanced computational resources, it became possible to integrate mathematical modelling and programming. By joining these aspects, the application of an analytical model to numerous instances as well as consonance between mathematical and computational results seemed to be less distant. However, as stated in (BŁAŻEWICZ; DROR; WEGLARZ, 1991) the complexity of machine scheduling models has also increased and for this reason, most problems are considered to be NP-hard with some extensions (e.g. binary NP-hard) and a variety of methods have emerged over the years. It is relevant to say that when optimality is sought, most exact methods are only capable of solving small and medium-sized problems, yet at a very large computational time. Therefore, the worldwide urgent demand for faster methods caused the experts in the field to incorporate a trade-off between solution quality and time spent to solve the machine scheduling problems, which has expanded the capacity to solve larger problems with high quality solutions at a significant smaller amount of time.

A problem in machine scheduling is defined by some elements. The notation introduced in (GRAHAM et al., 1979) is still widely used in papers and books that are reference in scheduling, even though recent ones have been developed. This notation is based on three parameters $\alpha|\beta|\gamma$. Parameter α is based on the manufacturing environment of the problem, which can be defined as the manner the jobs flow through the production system or, in other words, the environment that defines the problem. Parameter β is related to the technological constraints in the problem. In other words, there is a constraint

added (e.g. blocking) that will compose the configuration of the general problem. The last parameter, γ , represents the criterion or criteria to be optimized. It is necessary to understand and recognize the main elements that each parameter may assume. A detailed description provided by (PINEDO, 2012) of these elements is given as follows:

1. Flow pattern (α)

- Single machine (1): An environment in which every job has to be processed on a particular machine regardless its precedence. Note that the system may have more stages, however those will not be accounted for.
- Identical parallel machines (Pm): There are m identical machines and a job j can be processed on any one of the m machines or any one that belongs to a given subset.
- Uniform parallel machines (Qm): This environment differs from the last one by attributing a speed v_k to a given machine k . Therefore, the processing time is defined as $p_{jk} = p_j/v_k$.
- Unrelated parallel machine(Rm): This environment is a generalization of the last one where a speed v_{jk} is also associated to the processing of a given jobs, once that a given machine k might have its speed affected when processing a given job j . Hence, the processing time is defined as $p_{jk} = p_j/v_{jk}$.
- Flow shop (Fm): m machines are set up in series and each job has to be processed on each one of the machines and follows the same route on each machine.
- Job shop (Jm): each job has its own predetermined route to follow. There may be situations where a job visits a machine more than once.
- Open shop (Om): m machines are set up and each job has to be processed on each one of the machines, however there are no restrictions regarding the route followed by each job and different jobs may be processed on different machines.

2. Technological constraints (β)

- Release dates (r_j): Release date occurs to indicate that a job j cannot be processed before this specified time, i.e. the processing can only begin after r_j .
- Precedence ($prec$): Requires a specific job or set of jobs to be processed before others. It shows a relationship of dependence in the system. It is usual on a single machine or parallel machine environment.
- Preemption ($prmp$): Once a job has started its processing, the operation on this job is allowed to be interrupted so that another job that has preference take place on that machine.

- Sequence dependent setup times (s_{ij}): setup times that are incurred and depend on the order of job j that will be processed after job i . If it is also dependent on the machine, index k is added, representing a machine. Thus, the symbol becomes s_{ijk} . A setup time is related to the time prior to processing when operations are ongoing.
- Sequence independent setup times (s_j): Similar to s_{ij} in the definition, however the setup time does not depend on a precedence between jobs. Whenever the setup time is rendered different for distinct machines, the notation becomes s_{jk} .
- Controllable processing times ($p(y)$): This situation might emerge when processing times are functions instead of being a deterministic parameter of the system. The y notation represents one or more parameters the processing times might depend on (e.g. r_j , st_j).
- Learning Effect (Le): It occurs due to a decrease in the original value of processing times. It pictures the machine as an intelligent system that, over time, is able to learn how to work on a given job and by doing so, it can process it much faster than the usual deterministic processing time.
- Aging Effect (Ae): It has the contrary effect of Learning Effect. As any temporary resource, a machine tends to deterioration and, therefore its efficiency is led to decrease. Hence, the processing times are considered variables such that they are increased from the original value.
- Blocking (*block*): Blocking is a constraint likely to occur on flow shop environment. Suppose that the buffer between two machines cannot contain more jobs. In this case, a queue is formed and job i is blocked on machine $k - 1$ until job j on machine k has completed its processing.
- Permutation (*prmu*): Permutation is a phenomenon that also exists in the flow shop environment. It is characterized by ordering the queues according to the dispatching rule First In First Out (FIFO).
- No-wait (*nwt*): No-wait scenario happens when there can be no job waiting to be processed between two machines. By this it is implied that the starting time of jobs on the first machine has to be delayed to guarantee that no job has to wait on subsequent machines. Likely to be present in a flow shop environment.

3. Performance measures (γ)

- Makespan ($C_{\max}$): Completion time is defined as the time a given job j completes its processing on machine k . Whenever it is relative to the last machine, the notation is given by $C[j]$, where $[j]$ denotes the job placed in the j -th position on the sequence. Makespan is a criterion that can be mathematically described

by

$$C_{\max} = \max_{1 \leq j \leq n} C[j] \quad (2.1)$$

and it represents the maximum completion time of the last job to leave the system. In practical terms, the makespan is associated with usage of the machines and therefore, an optimal makespan implies an optimal utilization of resources.

- Maximum Tardiness ($T_{\max}$): Tardiness of a job j is given by

$$T_j = \begin{cases} C_j - d_j, & \text{if } C_j > d_j \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

where d_j is the due date associated with a given job. Similarly to Eq.2.1, maximum tardiness is defined as

$$T_{\max} = \max_{1 \leq j \leq n} T[j]. \quad (2.3)$$

Tardiness can be defined as a delay that occurs when the completion time for a job is longer than the due date established and some other functions might be derived from that concept, which is the case of Total Tardiness and Total Weighted Tardiness. The first one computes the sum of T_j for all jobs in the system. Clearly, this is crucial in a manufacturing system because the smaller the total tardiness, the closer the company will stand on meeting the agreements established with the contractor. Hence, minimizing the total tardiness implies a longer and profitable partnership. The Total Weighted Tardiness works analogously but the difference is that there are penalties w_j associated with designating a priority to a the jobs. Both criteria are mathematically represented by

- Total Tardiness ($\sum T_j$):

$$TT = \sum_{j=1}^n T[j]. \quad (2.4)$$

- Total Weighted Tardiness ($\sum w_j T_j$):

$$TWT = \sum_{j=1}^n w[j] T[j]. \quad (2.5)$$

Another due date related criterion that is often analyzed in the literature is the maximum lateness. Lateness of a job can be defined by

$$L_j = C_j - d_j. \quad (2.6)$$

Note that the expression can be positive is job is tardy and negative if job is early. The maximum lateness, which is denoted by $L_{\max}$, is given by

- Maximum Lateness ($L_{\max}$):

$$L_{\max} = \max_{1 \leq j \leq n} L_{[j]}. \quad (2.7)$$

Functions that often are of interest for researchers in the field are those related to the number of tardy jobs rather than the tardiness itself. The unit step function that represents a tardy job is given by

$$U_j = \begin{cases} 1, & \text{if } C_j > d_j \\ 0, & \text{otherwise} \end{cases} \quad (2.8)$$

and the two functions that are commonly related to it are

- Number of Tardy Jobs ($\sum U_j$):

$$NTJ = \sum_{j=1}^n U_{[j]} \quad (2.9)$$

and

- Weighted Number of Tardy Jobs ($\sum w_j U_j$):

$$WNTJ = \sum_{j=1}^n w_{[j]} U_{[j]}. \quad (2.10)$$

- Total Flow Time ($\sum F_j$): Flow time constitutes the time a given job j remains on the system. Mathematically it is defined by

$$F_j = C_j - r_j. \quad (2.11)$$

Several manufacturing problems consider $r_j = 0$. The total flow time (which is also denominated total completion time ($\sum C_j$) in this particular case) of a sequence is defined as the sum of flow times of each job. In mathematical notation

$$TFT = \sum_{j=1}^n F_{[j]} = \sum_{j=1}^n C_{[j]}. \quad (2.12)$$

Furthermore, there exists the analogous version of total flow time when there are penalties associated to the priorities established for jobs and it is called total weighted flow time (which is total weighted completion time for the case of $r_j = 0$). Mathematically it is expressed by

- Total Weighted Flow Time ($\sum w_j F_j$):

$$TWFT = \sum_{j=1}^n w_{[j]} F_{[j]} = \sum_{j=1}^n w_{[j]} C_{[j]}. \quad (2.13)$$

The main elements for each parameter were stated above. The ones that have not been provided here are either a variation of those that have been listed in this section (e.g. mean flow time) or do not have a representative role in this research. A detailed list is found on ([NEUFELD; GUPTA; BUSCHER, 2016](#)) and ([PINEDO, 2012](#)).

This chapter is relevant to understand the literature review that is presented in the next chapter, since the notations will be useful throughout the text. Furthermore, the reader should notice that the samples of articles linking flowshop scheduling and dynamic programming that will be presented are limited and, for that reason, it was acknowledged that all of them should be introduced here, even if they are not part of the initial scope of this project. Additionally, whenever it is necessary, the concept will be presented along the next chapter so as to contextualize the scenario to the reader for sake of completeness.

3 DYNAMIC PROGRAMMING APPLIED TO MACHINE SCHEDULING PROBLEMS: A REVIEW

This chapter's main goal is to enclose an extensive literature review about the applications of dynamic programming in machine scheduling organized in a chronological manner. The first section introduces dynamic programming from a general perspective in order to show which premises describe the use of the method. The second section is related to the selection of sources in the literature to compose the fundamentals for the research. The next sections incorporate a literature review by presenting multiple appearances of dynamic programming in machine scheduling. The outer segmentation is done according to the flow pattern parameter defined in the previous chapter, starting from the simplest configuration until it reaches the most complex one. The inner segmentation is focused on studying the main objective functions (which may include their variations, such as mean and weighted) also defined previously. In case additional objective functions are deemed relevant to the study, an extended subsection will be created.

3.1 Dynamic Programming

One of the first problems related to machine scheduling surfaced with (JOHNSON, 1954), which can be described as $F2||C_{\max}$ by Graham's notation. Johnson devised a polynomial-time algorithm so the system could function as Shortest Processing Time (*SPT*) - Longest Processing Time (*LPT*) pair when selecting jobs for allocation.

Parallel to Johnson's work in scheduling, (BELLMAN, 1952) had already begun investigating mathematical properties of what later on would be defined as dynamic programming. In this article, Bellman introduced a mathematical generalization of the discrete deterministic optimization problem associated with minimum cost required to accomplish a certain task. After publishing this first paper, Bellman consolidated the grounds on dynamic programming by presenting an extension of the discrete model to a continuous one. He also introduced a stochastic version of the model (BELLMAN et al., 1954).

The definition of dynamic programming (DP) can be associated with the division of a large problem (original problem) into subproblems that, in turn, present smaller dimensions if compared to the original one. Since these sub-problems are less complex to solve, their solutions are used as compounds to generate the final solution, which is that of the initial problem.

In dynamic programming, before fully understanding the model, some concepts and variables need to be defined. These are admitted as follows:

- Stage (l) : Associated to the level a given sub-problem is at and it is usually an indicator for the number of decision making variables that have been allocated.
- Control/State (u_l) : A control is the decision making variable associated with stage l . The set of decisions in stage l is denoted by U_l such that $U_l = \{u_{1l}, u_{2l}, \dots, u_{tl}\}$.
- State variable (x_l) : Function that yields a real value when a control is chosen in stage l . To obtain a state variable in the next stage, there is a dependence relation defined by $x_{l+1} = f(x_l, u_l)$.
- Policy (α) : Represents the set of controls that have been selected in each stage. Since optimal decisions are of interest, an optimal policy contains all the optimal controls in every stage, i.e. $\alpha^* = (\mu_0^*, \mu_1^*, \dots, \mu_N^*)$. Here, a finite horizon scenario is considered, and note that there are N stages to be accounted. Variable μ is relative to characteristics of each control.

By the definition of state variable, one can see a relation of recursion and this is the basis of dynamic programming. At each stage, a cost is bound to the control that has been chosen by the decision maker. Thus, the optimization problem, which is basis for dynamic programming formulation, is to obtain the minimum cost associated to these decisions. Therefore, the mathematical model for the discrete deterministic case is given by

$$\min_{u_l \in U_l} g_N(x_N) + \sum_{n=l}^{N-1} g_n(x_n, u_n) \quad (3.1)$$

s.t.

$$x_{l+1} = f(x_l, u_l) \quad (3.2)$$

$$x_1 = f(x_0, u_0) \quad (3.3)$$

$$u_l \in U_l. \quad (3.4)$$

Note that Eq.3.1 is the representation of an objective function to be minimized. Function g_n is the cost relative to a given state when x_n and u_n are provided. Since at the last stage (N), no decision needs to be made, the cost associated to this stage depends only on the state variable. The objective function describes the total cost function that is intrinsic to each problem, depending on the context. For example, if the problem referred to is a scheduling problem with performance measure defined as makespan, then this objective will be the makespan, while function g will be the completion times at each stage, which, in turn, is analogous to the position that will be occupied by a certain job j .

Furthermore, such job is a state, once its selection incurs an additional cost at a given stage.

In summary, (BAUTISTA; PEREIRA, 2009) define dynamic programming as a technique to solve multistage processes where any problem can be subdivided into smaller ones and their solutions are aggregated in order to obtain the solution of the initial problem. For each stage, a set of states is enumerated and these correspond to feasible partial solutions. The states of a given stage can be transformed to the next one through a process called transition T . Additionally, from a mathematical perspective, dynamic programming can be conceived as the exploration of a multistage graph $G(L, T)$, where L can be seen as the state set (or states belonging to the same stage also known as state space) that contains the vertices equivalent to a sub-problem and T is the set of transitions that are analogous to edges that indicate the allocation of a state at a given stage.

To solve the model, in (BELLMAN et al., 1954) the following principle has been developed:

Theorem 3.1.1 (Bellman’s Principle of Optimality). *An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with no regard to the state resulting from the first decisions.*

This principle is one of the foundations of dynamic programming and with modifications it can actually be used to find optimal solutions for cases that are neither deterministic nor discrete. In a simpler statement, this principle acknowledges that if the optimal solution has been found for sub-problems, and if an optimal policy is established for the remaining controls, then the solution for the original problem has been found.

Due to this principle, solutions via dynamic programming can be obtained either by using forward recursion or backward recursion. According to (BELLMAN et al., 1954), forward recursion defines an initial condition for the cost in the first stage and the optimal solution yielded from the optimal policy is found in the last stage. Analogously, the backward recursion sets an initial condition for the cost in the last stage and the optimal solution is achieved on the first stage. Some problems can be solved by both mechanisms, however depending on the problem, one of them can furnish a better and more efficient solution from the computational point of view.

3.2 Literature Review Methodology

As previously mentioned, scheduling problems have been abundantly explored in Operations Research, however some problems within this area have only recently been rewarded with solution methods that might be a useful groundwork for further analysis, as noted in the $Fm|s_{ijk}, block|C_{\max}$ problem.

Therefore the research baseline had been initially proposed as follows:

1. Define the environment of interest;
2. Define the objective function and the constraints that would be covered;
3. Define whether this method would be heuristic/exact/hybrid based on previous occurrences in literature;
4. Gather papers regarding the theme and propose a systematic or detailed review based on the data that has been found.

The first three steps are crucial in terms of decision, since the project and the experimental analysis rely on this type of selection. Flow shop environment was clearly the expected outcome of the first item because it is acknowledged as a fairly explored area with a vast number of papers indicating its common application to either classic or contemporary problems. Hence, these papers would allow us to identify some gaps that literature has failed to fill.

The second topic has also been justified upon the same criterion as the first one. Therefore, makespan has been chosen as the objective function of interest since a rather considerable amount reckoning it has been produced over the last 50 years. For the second part, however, rather than incorporating the most popular constraints, we chose to approach those that had received less attention throughout these years because it would be relevant to aggregate methods to a problem that had been provided so little in terms of solutions. Then, $Fm|s_{ijk}, block|C_{\max}$ has been selected, once it is commonly seen in industry, nevertheless rarely exploited in papers.

The third topic has been, perhaps, the most challenging among them, since not only did we have to decide a method, but also define its specificity. Heuristics and metaheuristics have been recently dominating the solution methods framework and exact methods rely usually on a B&B algorithm with several subroutines to generate improvements in the flow shop environment. Over the years, the use of dynamic programming has been weakly distributed in the flow shop environment when compared to less complex environments and when it came to the problem of interest in the research, no data would be presented considering DP as a solution method. Therefore, by applying DP we would start filling a gap in literature and could also generate potential collaboration for future considerations and improvements.

Although DP might seem a viable choice for the problem, its recursive formulations tend to demand excessively on computational requirements and the applications may be limited by the so-called curse of dimensionality. In addition, some authors, which have been overly cited in the previous sections, have been employing the Bounded Dynamic

Programming (BDP) method in machine scheduling because it seemingly demands much less in terms of computational costs. Hence, this final assertion was crucial to designate BDP for solving the problem that motivates the research.

Lastly, the fourth topic was to filter the given papers that had been scouted for the definition of problems and methods to support the research. Since BDP has not been widely applied in machine scheduling, the gathered content would not be significant enough to devise a consistent review. Similar conclusions can be drawn when cross-referencing DP and flow shop scheduling. In fact, DP has been more effectively applied in problems involving flexible flow shop than the classic version of the problem, however the environment does not belong to the scope of this research. Once that had been defined, the next step would be to filter articles that took into account lot sizing and batching because those consider additional assumptions that are also not in the scope of our review and would make the review article too dense for readers to relate in terms of notations and framework. Therefore, the final theme considered the junction of DP and machine scheduling, once a significant amount of papers has been extracted from the filtered search and a pattern established by many authors in integrating both themes.

The three search database that have been used were *Google Scholar*, *Web of Science* and *Scopus*. The first one was indeed the one that provided more papers and the results found were compared to those furnished by the other two sources, proving that the combination of papers of *Google Scholar* overlaid the remaining database. In addition the search employed the following keywords or statements:

- single machine and dynamic programming;
- single machine scheduling and dynamic programming;
- single machine and DP;
- parallel machine and dynamic programming;
- parallel machine and DP;
- flowshop scheduling and dynamic programming;
- flow-shop scheduling and dynamic programming;
- flow shop scheduling and dynamic programming;
- flowshop scheduling and DP;
- flow-shop scheduling and DP;
- flow shop scheduling and DP;

- job shop scheduling and dynamic programming;
- job shop scheduling and DP;
- open shop scheduling and dynamic programming;
- open shop scheduling and DP;
- dynamic programming applied to the single machine scheduling problem;
- dynamic programming applied to the parallel machine scheduling problem;
- dynamic programming applied to the flow shop scheduling problem;
- dynamic programming applied to the job shop scheduling problem;
- dynamic programming applied to the open shop scheduling problem.

After filtering the total amount of articles found in the three sources and each keyword shown above by excluding those that were not part of the scope of the research, 63 articles have been tracked for the single machine environment, 63 for parallel machine, 29 for flow shop, 10 for job shop and 12 for open shop, which totals 177 papers correlating DP and general machine scheduling problems. Also the time frame is relevant to our research, once we tried covering articles dating from 1952 up to August, 2020.

3.3 Dynamic Programming Approach for the Single Machine Environment

A single machine environment can be defined as follows: A set of jobs $J = 1, 2, \dots, n$ needs to be processed on one machine given that each job has a processing time p_j associated and those are treated independently. The objective is to find the best ordering that returns the minimum cost related to a specific objective function.

Due to its simplicity when compared to other environments in scheduling, the variations that come with this problem are abundant regarding the constraints as well as the objective functions that present potential interest in manufacturing. Therefore, experimenting on a single machine and finding solutions to tackle variants related to it may have positive impact on developing adapted methods for far more complex systems. Among those solutions, dynamic programming has shown substantial contribution when studying single machine problems, for both classic and contemporary approaches.

3.3.1 Classic due-dates related criteria

([LAWLER; MOORE, 1969](#)) model the $1||w_j E_j, 1|d, d'| \sum T_j$ and $1|d_j = d| \sum w_j T_j$ problems as knapsack-like formulations and, based on that premise, a DP approach is obtained from a previous DP solution for the knapsack problem. In addition, the

formulation is combined with a non-increasing w_j/p_j priority rule for $1||w_jE_j$, a SPT rule for $1|d, d'|\sum T_j$ and a non-decreasing w_j/p_j rule for $1|d_j = d|\sum w_jT_j$. Complexities for the first and third DP approaches are given by $O(nd)$ and $O(n^2d)$.

(SRINIVASAN, 1971) applies the theoretical background available on precedence relations to a DP formulation ("hybrid" algorithm) to solve the $1||\sum T_j$ via the principle of optimality. The tests are performed in data sets with up to 12 jobs and the CPU times results are better than those obtained by complete enumeration and full DP approach.

(LAWLER, 1977) proposes a pseudo-polynomial algorithm endorsed by properties of agreeable jobs and a DP formulation based on the principle of optimality to solve the $1||\sum w_jT_j$ problem. Although some options have been offered and employed to reduce the algorithm's complexity, the author proves it to be an $O(n^4 \sum p_j)$ or $O(n^5 \max\{p_j\})$ in a worst-case scenario.

(BAKER, 1977) focuses on applying the dominance rules proposed by Emmons to find efficient solutions for the $1||\sum T_j$. By coupling those rules with a DP formulation, the author manages to develop a forward recursion DP chain algorithm. For a 15-job data set the algorithm produces better results in terms of CPU times and storage than conventional DP and the one proposed by (SRINIVASAN, 1971). The comparison with a dual algorithm for a 20-job set generates better results, however the storage requirements start approximating those in conventional DP once cardinality of jobs is over 30.

(SCHRAGE; BAKER, 1978) consider an efficiency aspect of the DP formulation by devising a labelling scheme in order to reduce the computational effort when storing information from a previous stage for problems with precedence constraints. The method is applied to $1|prec|\sum T_j$ and $1|prec|\sum w_jT_j$ problems and compared to a chain algorithm, proving that DP outperforms the chain algorithm since the former explores a smaller number of feasible sets.

(POTTS; WASSENHOVE, 1982) combine features of the decomposition properties proposed by (LAWLER, 1977) and labelling scheme studied in (SCHRAGE; BAKER, 1978) in order to introduce a more powerful DP algorithm to solve $1||\sum T_j$, which resorts to decomposition if a given storage threshold is achieved. The experimental sets contain data ranging from 50 to 100 jobs, which are solved within reasonable time for instances up to 70 jobs and is proven to have $O(n^4)$ complexity.

(POTTS; WASSENHOVE, 1987) present a series of adapted DP algorithms to solve the $1||\sum T_j$ problem. The authors divide the algorithms according to their precedence or decomposition properties, which are included to reduce the state space of feasible sets. It is shown that decomposition algorithms outperformed precedence ones with the most efficient one being able to solve 100 jobs problems within reasonable time.

(ABDUL-RAZAQ; POTTS; WASSENHOVE, 1990) compare two DP formulations

based on Lawler and Schrage-Baker works and four B&B algorithms with lower bounds derived from Lagrangian, exponential, linear and state-space relaxations for the $1||\sum w_j T_j$ problem. The best results have relied on the B&B algorithms, especially the one with linear lower bound, which managed to solve up to 40 jobs. The DP algorithms were limited by core storage as problem sizes grew larger, despite yielding small time CPU times.

([SEN; BORAH, 1991](#)) develop a branching scheme based on ordering theorems to compare it to a DP approach for solving the $1||\sum T_j$ problem. Even though the comparison was not made to measure efficiency, parameters such as dispersion of CPU times and generated state space have been provided. The branching scheme presents a smaller solution set, whereas a more scattered group in CPU times.

([TANG; XUAN; LIU, 2007](#)) aim at a different approach for the $1|prec|\sum w_j T_j$ problem by proposing a Lagrangian relaxation of the model and applying a hybrid backward and forward DP algorithm for finding optimal solutions. This paper is considered a breakthrough in the field due to its capacity of assessing all precedence relationships and for also showing a faster convergence regarding the solutions for small and large sized problems.

([TUONG; SOUKHAL; BILLAUT, 2010](#)) discuss general aspects from the algorithm proposed by ([LAWLER; MOORE, 1969](#)) to solve the $1|d_j = d|\sum w_j T_j$ problem and some drawbacks such as the exclusion of straddling jobs are pointed out. Therefore, they propose a DP algorithm that takes into account that possibility, which is proven to have $O(n^2 d)$ complexity.

([ZHANG; LU; YUAN, 2010](#)) explore the nature of several problems that include the a penalty for rejected jobs as a bounded constraint (denoted here by θ_j^{rej}), which has been drawing considerable attention lately since rejection might imply reduction in costs for jobs that have larger processing times and do not alter significantly the composition of the process. The authors prove that $1|\sum \theta_j^{rej} \leq K|L_{\max}$ is NP-complete and they develop a DP formalism in which cases of acceptance and rejection are modelled recursively with $O(nK \sum p_j)$ complexity. In addition, a fully polynomial time approximation scheme (FPTAS) is derived for this problem

([TANAKA; FUJIKUMA, 2012](#)) develop a framework to solve several problems by improving the successive sublimation DP algorithm seen in ([TANAKA; FUJIKUMA; ARAKI, 2009](#)). These improvements take into account relaxations, reduction methods and shortened connections in networks. The algorithm is able to solve optimally 80-job instances for the $1|r_j|w_j T_j$ problem, thus outperforming previous methods.

([TANAKA; SATO, 2013](#)) take interest in studying the $1|prec|\sum w_j T_j$ by applying Lagrangian relaxation and reductions in paths generated by the precedence networks that compose the proposed successive sublimation DP algorithm. The algorithm is able to solve

Table 1: Due-date related problems for single machine

Authors	Problem Notation	Complexity	Method
(LAWLER; MOORE, 1969)	$1 \sum w_j E_j$	$O(nd)$	DP
	$1 d, d' \sum T_j$	—	DP
	$1 d_j = d \sum w_j T_j$	$O(n^2 d)$	DP
(SRINIVASAN, 1971)	$1 \sum T_j$	—	DP
(LAWLER, 1977)	$1 \sum w_j T_j$	$O(n^4 \sum p_j)$ or $O(n^5 \max p_j)$	DP
(BAKER, 1977)	$1 \sum T_j$	—	DP
(SCHRAGE; BAKER, 1978)	$1 prec \sum T_j$	—	DP
	$1 prec \sum w_j T_j$	—	DP
(POTTS; WASSENHOVE, 1982)	$1 \sum T_j$	$O(n^4)$	DP
(POTTS; WASSENHOVE, 1987)	$1 \sum T_j$	—	DP
(ABDUL-RAZAQ; POTTS; WASSENHOVE, 1990)	$1 \sum w_j T_j$	—	DP and B&B
(SEN; BORAH, 1991)	$1 \sum T_j$	—	DP
(CHENG; DING, 2000)	$1 p_j(s_j), d_j L_{\max}$	$O(n^6 \log n)$	DP
(TANG; XUAN; LIU, 2007)	$1 prec \sum w_j T_j$	—	Hybrid DP
(TUONG; SOUKHAL; BILLAUT, 2010)	$1 d_j = d \sum w_j T_j$	$O(n^2 d)$	DP
(ZHANG; LU; YUAN, 2010)	$1 \sum \theta_j^{rej} \leq K L_{\max}$	$O(nK \sum p_j)$	DP and FPTAS
(TANAKA; FUJIKUMA, 2012)	$1 r_j \sum T_j$	—	SSDP
(TANAKA; SATO, 2013)	$1 prec \sum w_j T_j$	—	SSDP
(ROSTAMI; CREEMERS; LEUS, 2019)	$1 prec \sum w_j T_j$	—	DP

all 50-job instances for the problem and most of the 100-job instances.

(ROSTAMI; CREEMERS; LEUS, 2019) gather information on precedence theorems in order to develop a more efficient exact algorithm to solve the $1|prec|\sum w_j T_j$ problem. They construct a DP algorithm with inbuilt formulations related to those theorems in order to diminish the computational effort in generation of states and compare it to that proposed by (TANAKA; SATO, 2013). The results were superior for the most recent method, since it could solve some set of instances that had not been solved by its predecessor and also larger ones.

A summary of the papers in this subsection can be found in Table 1.

3.3.2 Number of Tardy Jobs

(MOORE, 1968) presents a DP algorithm to solve the $1||\sum U_j$ problem, by including SPT, LPT and earliest due date rules in its composition. It is classified as a polynomial time method with $O(n \log n)$ time complexity.

A DP approach has also been found in (LAWLER; MOORE, 1969). The authors use similar mechanisms seen in [Classic due-dates related criteria](#) and resort to an earliest due date (EDD) priority rule to solve the $1||\sum U_j$ problem. Tardy jobs are identified whenever zero processing times are yielded from the DP formulation and its complexity is given by $O(n \min\{\sum p_j, \max\{d_j\}\})$.

(SAHNI, 1976) uses its previous work concerning $1||\sum w_j U_j$ to formulate a DP algorithm that also employs EDD priority rule. In addition, an elimination criterion is applied so as to reduce the state space by comparing tuples that include p_j and w_j . This generates a $O(\min\{2^n, nM\})$ time complexity algorithm, where $M = \min\{\sum w_j, \sum p_j, d_n\} + 1$ tuples. The author also studies properties and presents an approximation algorithm to solve the same problem.

(KISE; IBARAKI; MINE, 1978) aim at minimizing the $1|r_j|\sum U_j$ problem by imposing a non-decreasing restriction such that $r_i \leq r_j$ implies $d_i \leq d_j$. The authors design an $O(n^2)$ polynomial time DP algorithm and validate its capacity to solve the problem.

(GENS; LEVNER, 1981) design an approximation scheme to minimize the $1|d_j = d|\sum w_j U_j$ problem by inserting a DP scheme with a proposed modification to Sahni's approximation algorithm so as to eliminate redundant solutions, thus delivering a smaller set of feasible solutions and $O(n^2 \log n + n^2/\varepsilon)$ time complexity.

(POTTS; WASSENHOVE, 1988) develop eight algorithms and analyze their performance in solving the $1||\sum w_j U_j$ problem. The algorithms are obtained by deriving lower bounds from (LAWLER; MOORE, 1969) (DPLM), a LP relaxation (BBLP), a branch and bound method (BBDP) and number of tardy jobs lower bound (BBNTJ). In addition, all of them are also implemented by applying the Reduction Theorem (R). The most promising results are obtained from the R+BBDP and R+DPLM algorithms, since they can be used for solving large problems up to 1000 jobs.

(LAWLER, 1990) incorporates aspects of a preemptive EDD priority rule to a DP formulation to solve the $1|r_j, prmp|\sum w_j U_j$ and $1|r_j, prmp|\sum U_j$. The DP algorithm is classified as pseudopolynomial with $O(nk^2W^2)$ time complexity and $O(k^2W)$ space complexity for the first problem and with time complexity $O(n^3k^2)$ for the second, where k is the number of distinct release dates and $W = \sum w_j$. The algorithm is also extended to the special case $1|r_j|\sum U_j$, producing the knapsack DP formulation seen in (LAWLER; MOORE, 1969).

In (HARIRI; POTTS, 1994) a bounding scheme is presented by using state-space

relaxation and job penalties dependent on DP formulations to develop a B&B to solve $1|\bar{d}_j|\sum w_j U_j$. The experiments range from 50 to 300 jobs and good results are obtained, provided that processing times are not too large since the algorithm is bounded by $O(n\sum p_j)$ time complexity.

(LEE, 1996) investigates a series of optimization problems involving resumable and non-resumable availability constraints. Among them, the author points that the algorithm in (MOORE, 1968) with the inclusion of the unavailability period in the completion times solves the $1|r - a|\sum U_j$ problem optimally.

(DONDETI; MOHANTY, 1998) model the $1|Fe, Le|\sum w_j U_j$ as a 0-1 knapsack problem and combine a DP formulation with a maximum-weighted network path algorithm in order to solve it. The validation of such equivalence is established by lemmas based on EDD rule and modifications on the recursive relation used for building the DP algorithm.

(BAPTISTE, 1999a) several properties for the $1|r_j, prmp|\sum U_j$ problem, which are the foundation for a DP algorithm devised for solving such problem. The author is able to reduce time and space complexities (compared do Lawler's), respectively, from $O(n^5)$ to $O(n^4)$ and $O(n^3)$ to $O(n^2)$. Experiments to test the DP efficiency are designed for 50 and 100 jobs and solved in reasonable CPU times.

(BAPTISTE, 1999b) proposes DP-based polynomial time algorithms to solve $1|p_j = p, r_j|\sum w_j U_j$ and $1|prmp, p_j = p, r_j|\sum w_j U_j$ problems, respectively. By examining their properties, the author is able to prove that the first algorithm has $O(n^7)$ time complexity while the second is $O(n^{10})$.

(WOEGINGER, 2000) outlines a series of cases reasoning that, given some properties, some scheduling problems could provide a FPTAS by altering their DP formulations. Among them, the author suggests that, employing properties seen in (SAHNI, 1976) and (GENS; LEVNER, 1981), the DP approaches generate a FPTAS for $1||\sum w_j U_j$.

(ZHANG; LU; YUAN, 2010) analyze a DP formulation that depends on the makespan of early jobs that have been accepted and a total rejection penalty for the $1|\sum \theta_j^{rej} \leq K|\sum w_j U_j$ problem and it is bounded by $O(nK\sum p_j)$. The problem is categorized as NP-hard.

Table 2 summarizes all the articles in this subsection.

3.3.3 Completion Time and Flow Time

(LEE, 1996) proves that the $1|r - a|\sum w_j C_j$ problem is NP-hard and presents a pseudo-polynomial DP algorithm to solve it based on weighted shortest processing time algorithm for $1||\sum w_j C_j$ as well as a recursive equation for the objective function. The DP is bounded by $O(n(st_j)p_{\max})$, where st_j is the start time of maintenance for all jobs j .

(GÉLINAS; SOUMIS, 1997) present a thorough analysis regarding the $1|r_j|\sum C_j$

Table 2: Number of tardy jobs related problems for single machine

Authors	Problem Notation	Complexity	Method
(MOORE, 1968)	$1 \sum U_j$	$O(n \log n)$	DP
(LAWLER; MOORE, 1969)	$1 \sum U_j$	$O(n \min\{\sum p_j, \max\{d_j\}\})$	DP
(SAHNI, 1976)	$1 \sum w_j U_j$	$O(\min\{2^n, nM\})$	DP
(KISE; IBARAKI; MINE, 1978)	$1 r_j \sum U_j$	$O(n^2)$	DP
(GENS; LEVNER, 1981)	$1 d_j = d \sum w_j U_j$	$O(n^2 \log n + n^2/\varepsilon)$	Approximation with DP
(POTTS; WASSENHOVE, 1988)	$1 \sum w_j U_j$	—	DP, B&B and LP relaxation
(LAWLER, 1990)	$1 r_j, prmp \sum w_j U_j$	$O(nk^2W^2) - O(k^2W)$	DP
	$1 r_j, prmp \sum U_j$	$O(n^3k^2)$	DP
(HARIRI; POTTS, 1994)	$1 \bar{d}_j, d_j \sum w_j U_j$	$O(n \sum p_j)$	B&B with DP
(LEE, 1996)	$1 r - a \sum U_j$	$O(n \log n)$	DP
(DONDETI; MOHANTY, 1998)	$1 Fe, Le \sum w_j U_j$	—	DP with weighted network path
(BAPTISTE, 1999a)	$1 r_j, prmp \sum U_j$	$O(n^4) - O(n^2)$	DP
(BAPTISTE, 1999b)	$1 p_j = p, r_j \sum w_j U_j$	$O(n^7)$	DP
	$1 prmp, p_j = p, r_j \sum w_j U_j$	$O(n^{10})$	DP
(WOEGINGER, 2000)	$1 \sum w_j U_j$	—	FPTAS with DP
(ZHANG; LU; YUAN, 2010)	$1 \sum \theta_j^{rej} \leq K \sum w_j U_j$	$O(nK \sum p_j)$	DP

problem by studying a DP formulation coupled with a labelling scheme and an elimination test in order to select reduced and feasible sets of solution, a complexity investigation for each step or modification in the formulation and also a pseudo-polynomial DP algorithm that can solve problems up to 100 jobs out of 200-job instances used in the experiments.

(BIANCO; DELL'OLMO; GIORDANI, 1999) address the $1|r_j, p_j(\pi_{ij})|\sum C_j$ problem with sequence dependent processing times. In order to solve this problem, the authors establish its equivalence to the Cumulative TSP and propose a DP formulation, from which two lower bounds are derived. Furthermore, two heuristics are defined and a comparative performance analysis with the bounds is done for a realistic data set, showing that heuristics are more efficient in such case.

(CHENG; DING, 2000) examine the complexity analysis for $1|p_j(st_j), d_j|\sum C_j$, considering equal deteriorating rates $b_j = b$. Such problem is proved to be solved by a polynomial time DP algorithm of $O(n^5)$ complexity.

(JI; HE; CHENG, 2006) formulate a DP-based pseudo-polynomial time algorithm to deal with solutions for the $1|nr - a, p_j(st_j)|\sum C_j$ problem, with unavailability in the (t_1, t_2) interval. In addition, proof on the NP-hardness of the problem is shown and a

pseudo-polynomial DP algorithm with $O(n(t_1 - t_0)t_2 \prod (1 + b_j))$ is formulated. Ultimately, a heuristic procedure is also provided for the same problem.

(KACEM; CHU; SOUISSI, 2008) propose a DP algorithm, an integer programming formulation and a B&B for finding an optimal schedule regarding the $1|r - a| \sum w_j C_j$ problem. After tests being run and comparisons being made in terms of CPU time and storage requirements among the methods, the authors conclude the DP algorithm outperforms the other two, solving instances with up to 3000 jobs.

(FAN et al., 2011) contribute with theoretical analysis on the complexity of $1|h(\mu), r - a, p_j(t)| \sum C_j$, where μ corresponds to the number of unavailable periods. Two DP algorithms are furnished to solve a special case when $\mu = 1$ with resumability with $O(n^2(t_2 - t_1)^2)$ and $O(n^2(t_2 - t_1)^2 \prod (1 + b_j))$ complexities, respectively. Furthermore, a complexity assessment is given for both scenarios and for the general case, no polynomial time approximation is likely to occur for constant deteriorating rate.

(TANAKA; FUJIKUMA, 2012) also apply the modified successive sublimation DP method for $1|r_j| \sum w_j C_j$ and obtain optimal schedules for problems containing up to 200 jobs.

(LI; FAN, 2012) intensify the characteristics of the problems previously examined by (FAN et al., 2011), however considering only non-resumable scenarios and weighted completion time. For single unavailable intervals, an $O(nt_1)$ pseudo-polynomial DP algorithm is developed and a FPTAS is derived from it by using a state-space trimming technique. In addition, the authors prove that the double unavailable interval case in NP-hard and no polynomial time algorithm can be devised unless $P = NP$.

(GU et al., 2016) study the time-dependent aging effect with an associated processing speed ($v(t)$) and optional maintenance (ma) constraints applied to the total completion time optimization problem. A complexity analysis is made for makespan and, subsequently, total completion time criterion, with both being classified as NP-complete. Moreover, two DP algorithms are presented to solve the $1|v(t), ma| \sum C_j$ problem with $O(2^{n+1})$ and $O(n^2 \sum p_j)$ time complexities, respectively.

(CHENG; KRAVCHENKO; LIN, 2020) analyze the $1|p_j(st_j)| \sum C_j$ problem, given a b_j . The study of the complexity is evaluated and the problem is classified as binary NP-hard. The authors also work on a pseudo-polynomial DP algorithm and show its complexity can be evaluated at $O(n^3 \sum a_j \sum (a_j + b_j))$, where a_j is the ordinary processing time.

Table 3 summarizes the content of this subsection.

Table 3: Completion Time related problems for single machine

Authors	Problem Notation	Complexity	Method
(LEE, 1996)	$1 r - a \sum w_j C_j$	$O(ns_j p_{\max})$	DP
(GÉLINAS; SOUMIS, 1997)	$1 r_j \sum C_j$	—	DP
(BIANCO; DELL'OLMO; GIORDANI, 1999)	$1 r_j, p_j(\pi_{ij}) \sum C_j$	—	DP and Heuristics
(CHENG; DING, 2000)	$1 p_j(s_j), d_j \sum C_j$	$O(n^5)$	DP
(JI; HE; CHENG, 2006)	$1 nr - a, p_j(s_j) \sum C_j$	$O(n(t_1 - t_0)t_2 \prod (1 + b_j))$	DP and Heuristic
(KACEM; CHU; SOUISSI, 2008)	$1 r - a \sum w_j C_j$	—	DP, integer programming and B&B
(FAN et al., 2011)	$1 r - a, p_j(t) \sum C_j$	$O(n^2(t_2 - t_1)^2)$	DP
	$1 r - a, p_j(t) \sum C_j$	$O(n^2(t_2 - t_1)^2 \prod (1 + b_j))$	DP
(TANAKA; FUJIKUMA, 2012)	$1 r_j \sum w_j C_j$	—	SSDP
(LI; FAN, 2012)	$1 nr - a, p_j(t) \sum w_j C_j$	$O(nt_1)$	FPTAS with DP
(GU et al., 2016)	$1 v(t), ma \sum C_j$	$O(2^{n+1})$	DP
		$O(n^2 \sum p_j)$	DP
(CHENG; KRAVCHENKO; LIN, 2020)	$1 p_j(s_j) \sum C_j$	$O(n^3 \sum a_j \sum (a_j + b_j))$	DP

3.3.4 Makespan

(KUNNATHUR; GUPTA, 1990) propose several approaches to deal with $1|p_j(d_j)|C_{\max}$ problem, which are composed of two DP methods, a B&B and five heuristics. Although, from a theoretical point of view, DP is sustained by strong properties, its computational requirements were considered infeasible. Therefore, the experimental analysis was conducted from a comparison between the B&B and the best heuristic out of the five that have been provided.

(KUBIAK; VELDE, 1998) apply theoretical and empirical approaches to solve the $1|p_j(D_b, d_c)|C_{\max}$, where d_c is a common due date at which jobs start deteriorating and D_b is a bounded due date at which jobs no further deteriorate. The theoretical content encloses three pseudo-polynomial DP algorithms to solve the unbounded and bounded job deteriorating cases while the empirical methodology encompasses a B&B algorithm and heuristics. The complexities of the DP algorithms are described in Table 4.

(CHENG; DING, 2000) proves that solving the $1|p_j(st_j), d_j|C_{\max}$ problem is equivalent to finding the optimal solution to the flow time case. Therefore, the DP approach can be applied with no modifications required. Ultimately, the DP algorithm presents itself as an alternative solution for $1|p_j(st_j), d_j|L_{\max}$ with $O(n^6 \log n)$ (see Table 1) time complexity.

(WOEGINGER, 2000) studies the effect of the general transformation scheme of DP algorithms in FPTAS for the $1|p_j(d)|C_{\max}$ problem as well. Theoretical background is

extracted from (KUBIAK; VELDE, 1998) so as to perform the shifting procedure.

(JENG; LIN, 2004) explore two optimal procedures to solve the $1|p_j(d_j, s_j)|C_{\max}$ problem, where the processing times are defined as nonlinear step functions. The first approach is an $O(n(\max a_j + \max d_j))$ pseudo-polynomial DP algorithm and the second one relies on a B&B. Each procedure is modified by the inclusion of dominance rules, however, as experiments show, the procedures reach memory capacity, suggesting further research should focus on improvements.

(BOSIO; RIGHINI, 2009) evaluate the $1|p_j(r_j, st_j), r_j|C_{\max}$ problem. A DP algorithm associated with construction of upper and lower bounds is provided and an overall analysis regarding the range of r_j and p_j and their effect upon the upper and lower bounds are described through computational experiments.

(ZHANG; LU; YUAN, 2010) propose a DP formulation to solve the $1|r_j, \sum \theta_j^{rej} \leq K|C_{\max}$ problem taking into account a recursive relation that is dependent on the makespan of the accepted jobs for each stage with $O(n(r_n + \sum p_j))$ complexity. The problem is categorized as NP-hard, due to the complexity analysis performed on the problem without the release dates.

(DAVARI et al., 2020) address the $1|r_j, inv|C_{\max}$, which implies that the inventory is now bounded by a capacity constraint. The authors study the complexity and the problem is proved to be strongly NP-hard even for an infinite capacity inventory. In addition, two MIP formulations, a B&B method and a guess-and-check DP algorithm are provided. Experiments are carried out on, at most, 50 jobs and the guess-and-check DP outperforms the other method in terms of CPU times and number of solved problems.

Problems in which makespan is considered the main criterion are summarized in Table 4.

3.3.5 Multi-criteria and Additional Objective Functions

(WASSENHOVE; GELDERS, 1978) propose four approaches, two of which are DP, to solve the bicriteria $1||\sum w_j T_j + \lambda_j C_j$ problem. The authors resort to dominance rules based on precedence constraints and experimental tests are performed in a comparative manner for 25 jobs maximum. The DP algorithms outperformed the other methods for cases up to 20 jobs, however dual algorithms might be more efficient as the number of jobs increases.

(BARNES; VANSTON, 1981) present a DP method by coupling B&B features (DPBB) to solve the problem of minimizing the sum of delay penalties and setup costs. This method reduces the nodes explored in the DP states, thus reducing the effects of the curse of dimensionality inherent to a "pure" DP formulation. The computational experiments are performed for 10, 15 and 20-job data sets and compared with B&B and heuristics. The

Table 4: Makespan problems for single machine

Authors	Problem Notation	Complexity	Method
(KUNNATHUR; GUPTA, 1990)	$1 p_j(s_j) C_{\max}$	—	DP, B&B and Heuristics
(KUBIAK; VELDE, 1998)	$1 p_j(d_c) C_{\max}$	$O(nd_c \sum p_j) - O(nd_c)$	DP and B&B
	$1 p_j(d_c, D_b) C_{\max}$	$O(n^2 d_c (D_b - d_c) \sum p_j) - O(nd_c (D_b - d_c))$	DP and Heuristics
	$1 p_j(d_c, D_b) C_{\max}$	$O(nd_c \sum b_j (\sum p_j)^2) - O(nd_c \sum b_j \sum p_j)$	DP and Heuristics
(CHENG; DING, 2000)	$1 p_j(s_j), d_j C_{\max}$	—	DP
(WÖEGINGER, 2000)	$1 p_j(d) C_{\max}$	—	FPTAS with DP
(JENG; LIN, 2004)	$1 p_j(d_j, s_j) C_{\max}$	$O(n(\max a_j + \max d_j))$	DP and B&B
(BOSIO; RIGHINI, 2009)	$1 p_j(r_j, s_j), r_j C_{\max}$	—	DP
(ZHANG; LU; YUAN, 2010)	$1 r_j, \sum \theta_j^{rej} \leq K C_{\max}$	$O(n(r_n + \sum p_j))$	DP
(DAVARI et al., 2020)	$1 r_j, inv C_{\max}$	—	Guess-and-Check DP

DPBB method has shown best performance regarding solution quality and CPU times for the 15 and 20-job sets whereas the B&B presented better outcomes for the 10-job set.

(CHAND; SCHNEEBERGER, 1988) build a modified Smith heuristic and a DP algorithm to minimize weighted earliness criterion subject to no tardy jobs. The DP algorithm operates in backward recursion and makes use of an efficient frontier, which stores optimal values of weighted earliness and tardy jobs, to schedule jobs as further stages are created. The authors acknowledged that DP was generated to serve as benchmark for studying heuristics. In fact, the proposed heuristic provides good quality solutions and better CPU times.

(ABDUL-RAZAQ; POTTS, 1988) derive six lower bounds from a DP state-space relaxation to reduce the number of states generated by job assignments. Each bound is built upon a combination of modifiers and job penalties (similar to Lagrangian multipliers) so as to make them tighter. In addition, three B&B algorithms are implemented with less combinations than those designed for the DP formulation. Ultimately, the chosen problem to apply the methods was $1||\sum (\alpha_j E_j + \beta_j T_j)$ with DP being suitable for small-sized instances whereas B&B performs better for medium-sized instances with small processing times.

(HALL; POSNER, 1991) show that the $1|d_j = d|\sum w_j|d - C_j|$ problem (d being unrestrictively late) is NP-complete in the ordinary sense and presents a DP algorithm along with four special cases of the problem, which can each be solved with polynomial time algorithms. The second portion of this article is seen in (HALL; KUBIAK; SETHI, 1991) and investigates $1|d_j = d|\sum |d - C_j|$ (d placed early enough for the allocation process) by

proving it to be NP-complete in the ordinary sense. A DP pseudo-polynomial algorithm is introduced and shows promising results for being able to solve up to 1000 jobs.

(DE; GHOSH; WELLS, 1991) focus on presenting properties related to V-shaped structures for $1||d_j = d|\sum(\alpha_j E_j + \beta_j T_j)$ with asymmetrical weights and state that this problem is NP-complete in the strong sense. Furthermore, a DP algorithm is developed to ensure an exact approach to the problem as well as an approximation algorithm. Ultimately some particular cases are displayed.

(YANO; KIM, 1991) evaluate the use of a DP procedure for solving optimally the $1||\sum(\alpha_j E_j + \beta_j T_j)$ problem, assuming the sequence of jobs is already given and focusing on timing as a decision making process. They also present theoretical and experimental B&B and heuristics formulations to solve a specific case when penalties are proportional to the jobs' processing times.

(HOOGEVEEN; VELDE, 1991) explore the complexity of $1|d_j = d|\sum w_j|d - C_j|$ when $d < \sum p_j$ is imposed. The authors prove the problem is NP-hard and present a DP algorithm with time and space complexities of $O(n^2d)$ and $O(nd)$, respectively. Moreover, some specific cases within this scenario are presented as polynomially solvable.

(POTTS; WASSENHOVE, 1992a) incorporate additional methods to the $1||\sum Y_j$ problem by devising approximation algorithms derived from DP formulations and B&B. The approximation methods have been developed by aggregating rounding techniques into the DP so as to produce higher quality and less complex (in the computational sense) algorithms.

(POTTS; WASSENHOVE, 1992b) analyze properties related to the $1||\sum Y_j$ problem as well as complexity analysis by showing it is NP-hard. Moreover, the authors develop a heuristic for setting an initial sequencing and a pseudo-polynomial DP algorithm of $O(nUB)$ complexity, which is able to yield solution up to 10000-job instances due to space-state reductions performed via optimality properties.

(CARRAWAY et al., 1992) apply generalized DP to solve the nonlinear multi-criteria $1||\sum(e^{k_1 w_j T_j} + \lambda e^{k_2 w_j U_j})$ problem and compare the results with a DP-based heuristic and a DP formulation to generate feasible sets. The heuristic and generalized DP show similar results and both outperform the DP formulation, since the last one may also admit sub-optimal schedules and fails rapidly due to the curse of dimensionality. The authors suggest using the technique so as to develop further studies regarding different nonlinear multi-criteria problems.

(DE; GHOSH; WELLS, 1992) devise a pseudo-polynomial DP algorithm by adding V-shaped properties to identify optimal solutions for $1||\sum(C_j - \overline{C}_j)$ with overall complexity of $O(n^2 \sum p_j)$. Experiments are carried out for at most 100 jobs and DP procedure outperforms a B&B algorithm and an enumeration method for the same time limit.

Nevertheless the algorithm shows greater efficiency when processing times are small, therefore an approximation algorithm is also developed. Additionally, the DP approach is extended to the $1||(\alpha\bar{C}_j + (1 - \alpha)(C_j - \bar{C}_j))$ bi-criteria problem, using V-shaped structure to construct feasible sets that contain optimal solutions.

(BARD; VENKATRAMAN; FEO, 1993) combine the generation of state spaces inherent to DP approaches with a B&B algorithm in order to reach optimality for the $1||\sum(\alpha_j E_j + \lambda_j C_j)$ problem. A GRASP heuristic is also programmed so as to furnish a tight upper bound and fortify the fathoming in the B&B-DP. Optimality has been provided up to 30 jobs regardless the composition of the data in less than 12 minutes and with steady computational effort.

(IBARAKI; NAKAMURA, 1994) apply successive sublimation DP to solve the $1||\sum(\alpha_j E_j + \beta_j T_j)$ problem. The procedure includes heuristics in order establish an initial upper bound and recursive lower bounds are derived from modifiers, job penalties and state-space relaxation. The method is executed for data sets up to 35 jobs and compared with an original DP formulation, obtaining superior outcomes both in CPU times and number of generated states.

(CHAND; CHHAJED; TRAUB, 1994) propose a DP algorithm to minimize the lead time (LT) and earliness costs considering fixed-delivery intervals (d_i, d_f) by setting a first-come first-deliver priority rule. Reductions in the state space are provided through several dominance rules and compared to a full DP approach in an experiment containing up to 150 jobs. Results are considered better in terms of CPU times for the former approach.

(LIMAN; RAMASWAMY, 1994) describe properties related to restrictive and non-restrictive common due windows for the problem $1|d_w|\sum(\alpha_j E_j + w_j U_j)$. In addition they show the NP-completeness for both situations and develop a DP pseudo-polynomial algorithms for each case with overall complexity $O(n^2(d_w) \sum p_j)$.

(HARIRI; POTTS; WASSENHOVE, 1995) bring a theoretical work on preemptive and non-preemptive $1||\sum w_j Y_j$ by stating complexity analysis for both problems. In addition, the authors offer a DP-based B&B algorithm, in which DP, in conjunction with reduction properties, is used for designing upper and lower bounds efficiently. Instances ranging from 100 to 700 jobs show of DP bounds efficiency by obtaining a small number of nodes before reaching optimal solution.

(VENTURA; WENG, 1995) improve the method proposed in (HALL; KUBIAK; SETHI, 1991) by providing a pseudo-polynomial DP algorithm that stems from an elimination process of two subroutines in the original version. This enhances the method significantly in computational requirements and yields an $O(n(d + \sum p_j))$ overall complexity.

(WENG; VENTURA, 1996a) propose a pseudo-polynomial DP algorithm and

a heuristic to minimize $1|d_j = d, t_j| \sum (E_j + T_j)$. Some properties involving LPT-SPT ordering are outlined and optimality is ensured by DP if $p_i \leq p_j$ implies $t_i \leq t_j$. Comparative tests are performed for $n = 8, 9, 10$ with both methods being quite efficient and DP reaching near optimal solutions, indicating it could be used to ease the search for optimal solutions in larger job sizes.

(WENG; VENTURA, 1996b) describe properties related to the $1|d_j = d| \sum [(C_j - d)^2]/n$ problem by associating optimal scheduling with V-shaped structure. Furthermore, the authors design an $O(n \sum p_j)$ pseudo-polynomial DP algorithm for the tightly restrictive case, for which no strategy had yet been developed, and provide optimal solutions up to 100-job data sets in less than two seconds in CPU time.

(LANN; MOSHEIOV, 1996) explore the minimization of $1|| \sum (D_j + U_j)$ by providing algorithms for solving the job-independent and job-dependent with symmetrical and asymmetrical costs scenarios (in Table 5, *asy* and *sy* stand for asymmetrical and symmetrical costs, respectively), where DP formulations are given for the last two cases. They develop a polynomial DP algorithm of $O(n^2)$ time complexity for the symmetrical cost situation and a pseudo-polynomial DP method with $O(n \max\{\sum p_j, d_{\max}\})$ complexity, which is the basis for two heuristics used to solve the asymmetrical case.

(CHENG; CHEN; LI, 1996) analyze the $1|p_j(x_j)|(U_j, \sum x_j)$ bi-criteria problem, where x_j represents resources allocated to a given job and $p_j(x_j)$ indicates controllable processing times dependent on this resources. Complexity theorems stating the NP-hardness are shown and sequencing properties based on EDD rule are given. In addition pseudo-polynomial DP algorithms are provided for a linear non-decreasing function and a general non-decreasing function of x_j .

(CHENG; JANIAK; KOVALYOV, 1998) examine several bi-criteria minimization problems with resource dependent processing times, considering a non-negative time compressing rate q_j and initial processing time a_j . The DP algorithms are applied to the $1|(\sum v_j x_j \leq K, \sum w_j U_j)$ and $1|(\sum v_j x_j, \sum w_j U_j \leq K)$ problems with $O(KnUB \sum \tau_j)$, where $\tau_j \leq a_j/q_j$. In addition, the DP algorithms can be altered into approximation algorithms.

(WOEGINGER, 2000) also applies the properties delineated by (POTTS; WASSENHOVE, 1992a) and (HARIRI; POTTS; WASSENHOVE, 1995) to show the development of a FPTAS is likely to occur when the DP formulations for $1|| \sum Y_j$ and $1|| \sum w_j Y_j$ are modified accordingly.

(KLAMROTH; WIECEK, 2001) develop a DP algorithm in order to find the optimal schedule for a general multiple criteria optimization problem, whose objective functions are dependent on time and also bounded by a resource availability constraint. The authors model the problem as a multiple criteria knapsack problem and develop the

DP based on previous references of DP algorithms for the related knapsack formulation.

(YEUNG; OĞUZ; CHENG, 2001) introduce optimal properties for the $1|d_w|\sum(\alpha_j D_j + \beta_j U_j)$ problem and prove the NP-completeness associated to it. They also develop an $O(n^2 p_{\max}(1 + \sum p_j/2))$ pseudo-polynomial DP algorithm, which is unfolded in two types for solving problems with either presence or absence of straddling jobs. Computational tests are performed and the algorithm is limited to solving problems with 100 jobs within reasonable time.

(YEUNG; OGUZ; CHENG, 2001) discuss optimal schedule properties related to the multi-criteria $1|d_w|\sum(\alpha_j E_j + \beta_j T_j + \rho_j D_j + \theta_j U_j) + L(d_w)$ and $1|d_w|\sum(\alpha_j E_j + \beta_j T_j + w_j C_j) + L(d_w)$ problems bounded by a given due window. Two pseudo-polynomial DP algorithms, one for each problem, are designed with time complexity given by $O(n^3(K+1)p_{\max}^2(\sum p_j + 1)(\sum p_j - K + 1)^2)$. Additional polynomially solvable cases are presented.

(VENTURA; KIM; GARRIGA, 2002) aim at finding the optimal schedule as well as the release dates that minimize the $1|r_j|\sum(f(r_j) + \alpha_j E_j + \beta_j T_j)$ problem, in which f is a non-increasing function of release dates. A DP algorithm imbued with V-shaped ordering properties is conceived and heuristics are also provided. Despite small CPU times, the DP approach does not seem to be more advantageous than the heuristics since memory overreaching occurs for instances larger than 40 jobs while heuristics achieve high quality solutions by demanding less computational effort.

(HENDEL; SOURD, 2005) apply the DP concepts to generate a formulation to solve the $1||\sum(E_j + T_j)$. Despite being the main element for a pairwise interchange approach, the DP formulation, which is of $O(n^2)$ complexity, is seen as an auxiliary method that can be coupled within a neighbourhood search algorithm to finally solve the problem.

(KEDAD-SIDHOUM; SOLIS; SOURD, 2008) evaluate the performance of several lower bounds and a DP-based heuristic to solve the $1||\sum(E_j + T_j)$. The $O(n^2)$ polynomial DP formulation is included in the process in order to reduce the memory usage when storing previous information regarding the heuristic solutions. Experimental results are provided for at most 90 jobs and 6 machines and the heuristic produces the best results when compared to the proposed bounds.

(TANAKA; FUJIKUMA; ARAKI, 2009) investigate the $1|no-idle|\sum f_j(C_j)$ problem. A series of theorems are outlined and applied in order to reduce the computational space requirements to store the DP states in the successive sublimation DP algorithm proposed to solve the scheduling problem. These modifications lead to an efficient algorithm with capacity to optimally solve up to 300 jobs within reasonable time.

(CHENG; SUN, 2009) develop several DP algorithms and FPTAS for the $1|p_j(st_j), r_j, rej|\sum w_j C_j + \sum \theta_j^{rej}$, $1|p_j(st_j), r_j, rej|C_{\max} + \sum \theta_j^{rej}$ and $1|p_j(st_j), r_j, rej|(T_{\max}/L_{\max}) + \sum \theta_j^{rej}$ problems. A detailed assessment of each algorithm's complexity is given (see Table 5) and an $O(n^2)$

DP algorithm is provided for $1|p_j(st_j), r_j, rej|C_{\max} + \sum \theta_j^{rej}$ with deteriorating rates $b_j = b$.

(ZHANG; LU; YUAN, 2009) analyze the complexity formulations the $1|rej, r_j|C_{\max} + \sum \theta_j^{rej}$ and categorize the problem as NP-hard. In addition, the authors provide polynomial and pseudo-polynomial DP algorithms as well as approximation schemes (see Table 5) for the general problem and specific cases.

(ZHAO; TANG, 2011) supplement the work done in (FAN et al., 2011) by extending the completion time model to two other optimization problems: $1|h(\mu), r - a, p_j(t)|\alpha \sum C_j + \beta \sum T_j$ and $1|h(\mu), r - a, p_j(t)|\alpha \sum C_j + \beta \sum E_j$. Each problem is solved from a DP perspective and the authors examine the complexity of the algorithms and classify both problems as NP-hard.

(TANAKA; FUJIKUMA, 2012) extend the successive sublimation algorithm DP for the following two cases: $1||\sum (\alpha_j E_j + \beta_j T_j)$ and $1|r_j|\sum (\alpha_j E_j + \beta_j T_j)$. The SSDP has been able to find optimal solution for instances with up to 200 jobs.

(TANAKA; SATO, 2013) also apply the successive sublimation algorithm to the $1|no - idle, prec|\sum (\alpha_j E_j + \beta_j T_j)$ problem and the results are quite promising, given that solution has been found for all instances in the 50-job data set.

(YIN et al., 2013) bring a contemporary approach to a class of problems that include common due date as an endogenous element in the decision making process, considering positional learning effect and past sequence-dependent delivery times (psd). Four different objective functions are considered and also some specific cases are outlined. An $O(n^3)$ time complexity DP algorithm is developed to minimize the $1|Le, q_{psd}|\alpha \sum E_j + \sum w_j U_j + \gamma d$. Moreover, another $O(n^3)$ time complexity DP approach is offered as solution to the $1|Le, q_{psd}|\gamma \sum d + \sum w_j$ problem.

(ZHAO et al., 2014) evaluate solutions to minimize the $1|p_j(st_j)|\sum w_j U_j + \gamma d$, where d is also to be determined by the optimization process. The authors prove the problem is NP-hard and design two pseudo-polynomial DP algorithms, which possess $O(n \sum w_j)$ and $O(a_j \prod (1 + b_j))$ complexities, and an approximation scheme to deal with the optimization.

(XINGONG; YONG, 2015) analyze two problems with positional learning effect, which main objective is to minimize $F(d, \pi) = \phi(d) + \alpha \sum E_j + \sum \beta_j$, where $\phi(d)$ is a variable that represents the assigned due dates and β_j denotes a penalty associated with a discarded job. Common (CON) and slack due date (SLK) rules are applied for the due date assignment. $O(n^4)$ and $O(n^3)$ polynomial time DP algorithms are employed as solution methods for each problem, respectively.

(BEN-YEHOSHUA; MOSHEIOV, 2016) investigate the complexity behaviour related to the $1||\sum V_j$ problem and present proofs for its NP-hardness in the ordinary sense. Furthermore, a pseudo-polynomial DP algorithm with $O(n \sum p_j)$ complexity is

proposed and its results show that it can solve instances with up to 200 jobs efficiently.

(YIN et al., 2016) study the $1|ma|\sum Y_j$ problem, given that a fixed interval (t_1, t_2) is defined for the maintenance to occur. The authors propose two pseudo-polynomial DP algorithms, whose complexities are given by $O(nt_1(t_1 + U))$ and $O(nU(t_1 + U))$ (with U being a threshold), respectively. Additionally, a FPTAS is also provided and experiments are performed on each algorithm. The second DP algorithm outperforms the first in terms of CPU time and optimal solutions have been found for at most 45 jobs for either. The FPTAS reduces the state space generated by the DP algorithms and is able to find solution with up to 85 jobs depending on the approximation parameter.

(WANG et al., 2017) address the two-agent problem denoted by $1||\sum Y_j : L_{\max} \leq K$. The article focuses on exploring exact methods, which are comprised of two pseudo-polynomial DP algorithms and a B&B, and a Tabu Search (TS) in order to analyze their performances. Although the DP approaches for this problem are deemed fairly good methods, their practical usage is limited to small-sized instances and therefore, the comparison is estimated for the B&B and TS. For the complexity description seen in Table 5 we consider two parameters, which are $SP = (\sum p_j)_A + (\sum p_j)_B$ and $sp = \max\{(p_j)_A + (d_j)_A - 1\}$.

(BÜLBÜL; KEDAD-SIDHOUM; ŞEN, 2019) analyze solvability for the following problems $1|h(\mu), r-a|\sum (E_j + T_j)$, $1|h(\mu), sr-a|\sum (E_j + T_j)$ and $1|h(\mu), nr-a|\sum (E_j + T_j)$. The authors prove that the non-resumable and semi-resumable cases about a common non-restrictive common due date is NP-hard in the strong sense and for special cases involving the three scenarios, in which the problem is NP-hard in the ordinary sense, pseudo-polynomial DP algorithms are developed for optimal scheduling.

(LI; YUAN, 2020) focus on a variety of problems related to a multi-agent scheduling with the intent of minimizing the total weighted late work criterion. They introduce some notations in order to separate the various forms of minimization, classify some problems regarding their complexity and develop an exact pseudo-polynomial DP algorithm with $O(n^2 \prod UB_x)$ and an approximation method for a specific case in which Pareto-optimal solutions are sought with a fixed number x of agents.

(GERSTL; MOSHEIOV, 2020) develop seven pseudo-polynomial DP algorithms, in which three of them are designed for solving the $1|nr - a, d|\sum (E_j + T_j)$ and their usage depends on where d is placed regarding the interval of unavailability. The next three algorithms are similar, with respect to their usage, however are made for solving the $1|nr - a, d|\sum (F(E_j) + G(T_j))$, in which the functions are job-independent and non-decreasing. Lastly, the remaining algorithm is set for the job-dependent case and the three first methods are employed in case idle-time is allowed. Experiments are also performed in order to test the efficiency of the methods in each scenario and their complexities are dependent on n and $p_{\max}$.

([ZOU; YUAN, 2020](#)) provide a compilation of 16 problems involving several functions in combination with penalties associated with job rejection. Time complexities are furnished for each problem and two pseudo-polynomial DP algorithms are devised in order to provide solutions for the entire set of problems described in the paper.

Table 5: Multicriteria and Additional Functions problems for single machine

Authors	Problem Notation	Complexity	Method
(WASSENHOVE; GELDERS, 1978)	$1 \sum w_j T_j + \lambda_j C_j$	—	DP and Dual algorithm
(BARNES; VANSTON, 1981)	$1 \sum F(s_{ij}) + G(w_j U_j)$	—	DPBB
(CHAND; SCHNEEBERGER, 1988)	$1 (\sum w_j E_j, \sum U_j = 0)$	—	DP and Heuristic
(ABDUL-RAZAQ; POTTS, 1988)	$1 \sum (\alpha_j E_j + \beta_j T_j)$	—	DP and B&B
(HALL; POSNER, 1991)	$1 d_j = d \sum w_j d - C_j $	$O(n \sum p_j)$	DP and FPTAS
(HALL; KUBIAK; SETHI, 1991)	$1 d_j = d \sum d - C_j $	$O(n \sum p_j)$	DP
(DE; GHOSH; WELLS, 1991)	$1 d_j = d \sum (\alpha_j E_j + \beta_j T_j)$	—	DP and Approximation algorithm
(YANO; KIM, 1991)	$1 \sum (\alpha_j E_j + \beta_j T_j)$	—	DP
(HOOGEVEEN; VELDE, 1991)	$1 d_j = d \sum w_j d - C_j $	$O(n^2 d) - O(nd)$	DP
(POTTS; WASSENHOVE, 1992b)	$1 \sum Y_j$	$O(nUB)$	DP
(POTTS; WASSENHOVE, 1992a)	$1 \sum Y_j$	—	Approximation with DP and B&B
(CARRAWAY et al., 1992)	$1 (e^{k_1} w_j T_j + \lambda e^{k_2} w_j U_j)$	—	Heuristic with DP and B&B
(DE; GHOSH; WELLS, 1992)	$1 \sum (C_j - \bar{C}_j)$	$O(n^2 \sum p_j)$	DP
(BARD; VENKATRAMAN; FEO, 1993)	$1 (\alpha \bar{C}_j + (1 - \alpha)(C_j - \bar{C}_j))$	—	DP
(IBARAKI; NAKAMURA, 1994)	$1 \sum (\alpha_j E_j + \lambda_j C_j)$	—	B&B with DP and Heuristic
(CHAND; CHHAJED; TRAUB, 1994)	$1 \sum (\alpha_j E_j + \beta_j T_j)$	—	SSDP
(LIMAN; RAMASWAMY, 1994)	$1 (d_i, d_f) \sum E_j + \sigma_j LT$	—	DP
(HARIRI; POTTS; WASSENHOVE, 1995)	$1 d_w \sum (\alpha_j E_j + w_j U_j)$	$O(n^2(d_w) \sum p_j)$	DP
(VENTURA; WENG, 1995)	$1 \sum w_j Y_j$	—	B&B with DP
(WENG; VENTURA, 1996a)	$1 d_j = d \sum d - C_j $	$O(n(d + \sum p_j))$	DP
(WENG; VENTURA, 1996b)	$1 d_j = d, t_j \sum (E_j + T_j)$	—	DP and Heuristic
(LANN; MOSHEIOV, 1996)	$1 d_j = d \sum [(C_j - d)^2]/n$	$O(n \sum p_j)$	DP
(CHENG; CHEN; LI, 1996)	$1 \sum (D_j + U_j)$	—	DP and Heuristics
(CHENG; JANIAK; KOVALYOV, 1998)	$1 \sum (D_j^{sy} + U_j^{sy})$	$O(n^2)$	DP
(WEOGINGER, 2000)	$1 \sum (D_j^{asy} + U_j^{asy})$	$O(n \max\{\sum p_j, d_{\max}\})$	DP
(KLAMROTH; WIECEK, 2001)	$1 p_j(x_j) (U_j, \sum x_j)$	—	DP and Approximation algorithm
(YEUNG; OĞUZ; CHENG, 2001)	$1 (\sum v_j x_j \leq K, \sum w_j U_j)$	$O(KnUB \sum \tau_j)$	DP and Approximation algorithm
(YEUNG; OĞUZ; CHENG, 2001)	$1 (\sum v_j x_j, \sum w_j U_j \leq K)$	$O(KnUB \sum \tau_j)$	FPTAS with DP
	$1 \sum Y_j$	—	FPTAS with DP
	$1 \sum w_j Y_j$	—	DP
	$1 (f_i(t)), \sum x_j$	—	DP
	$1 d_w \sum (\alpha_j D_j + \beta_j U_j)$	$O(n^2 p_{\max}(1 + \sum p_j/2))$	DP
	$1 d_w \sum (\alpha_j E_j + \beta_j T_j + \rho_j D_j + \theta_j U_j) + L(d_w)$	$O(n^3(K + 1)p_{\max}^2(\sum p_j + 1)(\sum p_j - K + 1)^2)$	DP

Table 5 continued from previous page

(VENTURA; KIM; GARRIGA, 2002)	$1 d_w \sum (\alpha_j E_j + \beta_j T_j + w_j C_j) + L(d_w)$	$O(n^3(K+1)p_{\max}(\sum p_j + 1)(\sum p_j - K + 1)^2)$	DP
(HENDEL; SOURD, 2005)	$1 r_j \sum (f(r_j) + \alpha_j E_j + \beta_j T_j)$	—	DP
(KEDAD-SIDHOUM; SOLIS; SOURD, 2008)	$1 \sum (E_j + T_j)$	$O(n^2)$	Heuristic with DP
(TANAKA; FUJIKUMA; ARAKI, 2009)	$1 \sum (E_j + T_j)$	$O(n^2)$	Heuristic with DP and LBs
(CHENG; SUN, 2009)	$1 no - idle \sum f_j(C_j)$	—	SSDP
	$1 p_j(s_j), r_j, rej \sum w_j C_j + \sum \theta_j^{rej}$	$O(n \sum \theta_j^{rej})$	DP and FPTAS
	$1 p_j(s_j), r_j, rej C_{\max} + \sum \theta_j^{rej}$	$O(n \sum \theta_j^{rej})$	DP and FPTAS
	$1 p_j(s_j), r_j, rej (T_{\max}/L_{\max}) + \sum \theta_j^{rej}$	$O(n \prod (1 + b_j) \sum \theta_j^{rej})$	DP and FPTAS
	$1 p_j(s_j), b_j = b, r_j, rej C_{\max} + \sum \theta_j^{rej}$	$O(n^2)$	DP
(ZHANG; LU; YUAN, 2009)	$1 rej, r_j C_{\max} + \sum \theta_j^{rej}$	$O(n \sum \theta_j^{rej})$	DP and Approximation algorithm
	$1 rej, r_j C_{\max} + \sum \theta_j^{rej}$	$O(n(r_{\max} + \sum p_j))$	DP and Approximation algorithm
	$1 rej, r_j, \theta_j^{rej} = \theta^{rej} C_{\max} + \sum \theta_j^{rej}$	$O(n^2)$	DP
	$1 rej, r_j, p_j = p C_{\max} + \sum \theta_j^{rej}$	$O(n^3)$	DP
(ZHAO; TANG, 2011)	$1 h(\mu), r - a, p_i(t) \alpha \sum C_j + \beta \sum T_j$	$O(nt_1(\frac{\alpha}{\beta} \prod (1 + bp_j) - 1) + t_2(\prod (1 + bp_j) - 1))$	DP
(TANAKA; FUJIKUMA, 2012)	$1 h(\mu), r - a, p_i(t) \alpha \sum C_j + \beta \sum E_j$	$O(nt_1(\frac{\alpha}{\beta} \prod (1 + bp_j) - 1) + t_2(\prod (1 + bp_j) - 1))$	DP
	$1 \sum (\alpha_j E_j + \beta_j T_j)$	—	SSDP
	$1 r_j \sum (\alpha_j E_j + \beta_j T_j)$	—	SSDP
(TANAKA; SATO, 2013)	$1 no - idle, prec \sum (\alpha_j E_j + \beta_j T_j)$	—	SSDP
(YIN et al., 2013)	$1 Le, q_{psd} \alpha \sum E_j + \sum w_j U_j + \gamma d$	$O(n^3)$	DP
	$1 Le, q_{psd} \gamma \sum d + \sum w_j$	$O(n^3)$	DP
(ZHAO et al., 2014)	$1 p_j(s_j) \sum w_j U_j + \gamma d$	$O(n \sum w_j)$	DP
	$1 p_j(s_j) \sum w_j U_j + \gamma d$	$O(a_j \prod (1 + b_j))$	DP
(XINGONG; YONG, 2015)	$1 Le, SLK \phi(d) + \alpha \sum E_j + \sum \beta_j$	$O(n^4)$	DP
	$1 Le, CON \phi(d) + \alpha \sum E_j + \sum \beta_j$	$O(n^3)$	DP
(BEN-YEHOSHUA; MOSHEIOV, 2016)	$1 \sum V_j$	$O(n \sum p_j)$	DP
(YIN et al., 2016)	$1 ma \sum Y_j$	$O(nt_1(t_1 + U))$	DP and FPTAS
	$1 ma \sum Y_j$	$O(nU(t_1 + U))$	DP and FPTAS
(WANG et al., 2017)	$1 \sum Y_j : L_{\max} \leq K$	$O(nAn_B \min\{(SP, sp, (\sum d_j)_B\})$	DP, B&B and TS
	$1 \sum Y_j : L_{\max} \leq K$	$O(nAn_B (\sum p_j)_A)$	DP, B&B and TS
(BÜLBÜL; KEDAD-SIDHOUM; ŞEN, 2019)	$1 h(\mu), r - a \sum (E_j + T_j)$	—	DP
	$1 h(\mu), sr - a \sum (E_j + T_j)$	—	DP
	$1 h(\mu), nr - a \sum (E_j + T_j)$	—	DP
(LI; YUAN, 2020)	$1 P(\sum w_j Y_j^{(1)}, \dots, \sum w_j Y_j^{(w)})$	$O(n^2 \prod UB_x)$	DP and Approximation algorithm

3.4 Dynamic Programming approach for the Parallel Machine Environment

Dynamic Programming has also contributed for an extensive list of solutions regarding the parallel machine environment. Several mathematical properties have been developed for this multi-machine scenario and this fact has enabled sole or combined methods relying on dynamic programming resources to be designed.

3.4.1 Classic due date related criteria

(LAWLER; MOORE, 1969) extend the functional DP formulation for the same problems studied in the single machine environment to parallel machines. Since loss functions are part of the formulation, they might be equal or not. This feature, combined to the behaviour of processing times, generates solutions for identical and uniform machines.

(BAPTISTE, 2000) proves that, since pairwise monotonicity and non-decreasing behaviour can be observed for some classes of functions in scheduling, a decomposition algorithm can be applied and a DP formulation is derived from it. Due to that, it is possible to develop a polynomial solution of $O(n^{2m+2})$ complexity for $Pm|p_j = p, r_j| \sum T_j$.

(HALL; LESAOANA; POTTS, 2001) explore a series of problems concerning fixed delivery dates, which will be denoted as d_s with s indicating the number of fixed deliveries. The authors formulate two DP algorithms with $O(n(\sum p_j)^{m-1})$ and $O(n(\sum p_j)^{m\bar{s}-1})$ complexities that provide optimal solutions for the $Pm|d_s|L_{\max}$ and one of them is able to yield optimal solution for the $Pm|d_s, s = \bar{s}| \sum T_j$ and $Pm|d_s, s = \bar{s}| \sum w_j T_j$.

(LOPES; CARVALHO, 2007) analyze a branch-and-price (B&P) algorithm to minimize $Rm|s_{ijk}, r_j, a_k| \sum w_j T_j$. Due to the problem's robustness, the algorithm relies on a series of improvements and one of those is directed to the DP formulations that compose the pricing algorithm embedded in the B&P, which is named *primal box*. The method is applied in instances with up to 150 jobs and 50 machines and promising results are attained.

(TUONG; SOUKHAL; BILLAUT, 2010) extend their single-machine DP formulation to the identical and uniform machines under modifications. The assumptions of their DP are based on the existence of straddling jobs and WSPT priority rule for the fully tardy jobs. These properties generate two algorithms, each designated to its associated problem, with $O(n^{m+1}d^{2m}(\sum p_j)^{m-1})$ and $O(mn^{m+1}d^{2m}(\sum p_j))$.

(PESSOA et al., 2010) present a Branch-Cut-and-Price (BCP) algorithm in order to solve the $Pm|| \sum w_j T_j$. This algorithm relies on a forward and backward DP recursion to fix variables related to Lagrangian bounds that optimize the pricing algorithm, which is one of the essential elements in the internal structure of the BCP. Experiments are placed for at most 100 jobs and 4 machines and results are promising for the scenario with the authors emphasizing the need of sophisticated methods to large-sized arc-time indexed

problems such as the one presented.

([YOO; LEE, 2016](#)) investigate a series of problems concerning the dependent and independent cases of maintenance on parallel machines. Their analysis include classification of $Pm|ind|L_{\max}$, $Pm|ind|\sum L_j$, $Pm|dep|L_{\max}$ and $Pm|dep|\sum L_j$ as NP-hard in the strong sense and DP formulations are proposed in order to solve them. Their complexities are functions of some elements relative to the maintenance activity and those are defined as $R_1 = \min\{T - (t_2 - t_1), (\sum p_j + (m+1)(t_2 - t_1) - p_{\max})/m + p_{\max}\}$, $R_2 = (\sum p_j - p_{\max})/m + (t_2 - t_1) + p_{\max}$, $R_3 = \min\{T - (m - k + 1), (\sum p_j + (m-1)(t_2 - t_1) + (m-2)p_{\max})/m + p_{\max}\}$ and $R_4 = (\sum p_j + (m-1)p_{\max})/m + (t_2 - t_1) + p_{\max}$. Furthermore, the authors extend the DP approaches for problems with identical processing times for all the jobs. Detailed information about their complexities is shown in [Table 6](#).

Table 6: Due date related problems for parallel machines

Authors	Problem Notation	Complexity	Method
(LAWLER; MOORE, 1969)	$Pm \sum w_j E_j$	–	DP
	$Qm \sum w_j E_j$	–	DP
	$Pm d, d' \sum T_j$	–	DP
	$Qm d, d' \sum T_j$	–	DP
	$Pm d_j = d \sum w_j T_j$	–	DP
	$Qm d_j = d \sum w_j T_j$	–	DP
(BAPTISTE, 2000)	$Pm p_j = p, r_j \sum T_j$	$O(n^{2m+2})$	DP and Decomposition algorithm
(HALL; LESAOANA; POTTS, 2001)	$Pm d_s L_{\max}$	$O(n(\sum p_j)^{m-1})$	DP
	$Pm d_s L_{\max}$	$O(n(\sum p_j)^{\bar{m}\bar{s}-1})$	DP
	$Pm d_s, s = \bar{s} \sum T_j$	$O(n(\sum p_j)^{\bar{m}\bar{s}-1})$	DP
	$Pm d_s, s = \bar{s} \sum w_j T_j$	$O(n(\sum p_j)^{\bar{m}\bar{s}-1})$	DP
(LOPES; CARVALHO, 2007)	$Rm s_{ijk}, r_j, a_k \sum w_j T_j$	–	B&P with DP
(TUONG; SOUKHAL; BILLAUT, 2010)	$Pm d_j = d \sum w_j T_j$	$O(n^{m+1} d^{2m} (\sum p_j)^{m-1})$	DP
	$Qm d_j = d \sum w_j T_j$	$O(mn^{m+1} d^{2m} (\sum p_j)^{m-1})$	DP
(PESSOA et al., 2010)	$Pm \sum w_j T_j$	–	BCP with DP
(YOO; LEE, 2016)	$Pm ind L_{\max}$	$O(nm(R_1 + R_2)^m)$	DP
	$Pm ind \sum L_j$	$O(nm(R_1 + R_2)^m)$	DP
	$Pm dep L_{\max}$	$O(nmR_3^m R_4^m T^m)$	DP
	$Pm dep \sum L_j$	$O(nmR_3^m R_4^m T^m)$	DP
	$Pm dep, p_j = p L_{\max}$	$O((n+1)^{3m})$	DP

Table 6 continued from previous page

$Pm dep, p_j = p \sum L_j$	$O((n+1)^{3m})$	DP
----------------------------	-----------------	----

3.4.2 Number of Tardy Jobs

(LAWLER; MOORE, 1969) also extend the properties of the functional DP formulation for uniform and identical machines. Despite being presented as a natural extension this analysis performs calculations with exponential growth even for cases where machines are identical.

(LI, 1995) proposes a heuristic method as solution for the $Pm||\sum U_j$, considering agreeable due dates in the process and in order to compare its results, the author develops a bounding scheme derived from a DP formulation. The experiments are applied in job sets from 50 to 200 jobs and machine sets from 2 to 5 machines.

(CHEN; POWELL, 1999b) explore a branch and bound algorithm by deriving bounds from a decomposition method in order to solve the $Pm||\sum w_j U_j$, $Qm||\sum w_j U_j$ and $Rm||\sum w_j U_j$. This decomposition method is solved by a column generation algorithm which, in turn, is subdivided into two single machine sub-problems, being each solved by a DP formulation with $O(n^2 \sum p_{jk})$ complexity.

(HALL; LESAOANA; POTTS, 2001) investigate the application of the DP formulations on $Pm|d_s|\sum U_j$ and $Pm|d_s, s = \bar{s}|\sum w_j U_j$ with with complexity bounded by $O(n(\sum p_j)^m)$.

(CHEN; POWELL, 2003) resort to a B&B method to solve the $Pm|SI_f|\sum w_j U_j$ and $Pm|SD_f|\sum w_j U_j$ when a given time window (t_{1j}, t_{2j}) is considered. The branching process is partitioned via decomposition and column generation algorithms until the m -machine problem can be converted into single machines sub-problems, which are solved by a DP recursion of $O(n^2 \max\{t_{2j} - t_{1j} + 1\})$ complexity.

(CHEN, 2004) elaborates a similar algorithmic structure to previous works on B&B solved via partition, decomposition, column generation and single machine problems. The last portion of this division generates two DP algorithms, one of which is employed on finding the solution for the $Pm|p(x_j)|\sum w_j U_j + \sum G(x_j)$ problem with $O(n^2 t_{\max} \max\{b_j - a_j\})$ complexity.

Table 7: Number of tardy jobs related problems for parallel machines

Authors	Problem Notation	Complexity	Method
(LAWLER; MOORE, 1969)	$Pm \sum U_j$	–	DP
	$Qm \sum U_j$	–	DP
(LI, 1995)	$Pm \sum U_j$	–	DP and Heuristic
(CHEN; POWELL, 1999b)	$Pm \sum w_j U_j$	$O(n^2 \sum p_{jk})$	B&B with DP
	$Qm \sum w_j U_j$	$O(n^2 \sum p_{jk})$	B&B with DP
	$Rm \sum w_j U_j$	$O(n^2 \sum p_{jk})$	B&B with DP
(HALL; LESAOANA; POTTS, 2001)	$Pm d_s \sum U_j$	$O(n(\sum p_j)^m)$	DP
	$Pm d_s \sum w_j U_j$	$O(n(\sum p_j)^m)$	DP
(CHEN; POWELL, 2003)	$Pm SI_f \sum w_j U_j$	$O(n^2 \max\{t_{2j} - t_{2j} + 1\})$	DP
	$Pm SD_f \sum w_j U_j$	$O(n^2 \max\{t_{2j} - t_{2j} + 1\})$	DP
(CHEN, 2004)	$Pm p(x_j) \sum w_j U_j + \sum G(x_j)$	$O(n^2 t_{\max} \max\{b_j - a_j\})$	B&B with DP

3.4.3 Completion time related criteria

(BAKER; MERTEN, 1973) report a variety of results concerning the mean weighted flow time minimization, which are divided in practical and theoretical approaches. The latter states, among a series of findings, that the DP formulation proposed by Held and Karp can be used to reduce the state space for optimal solution in identical and uniform machines, however it is only fit for small and medium sized sets.

(HOROWITZ; SAHNI, 1976) provide algorithms to solve the $Q2||\sum w_j \overline{F_j}$ problem and, among those, a DP algorithm is presented. Such algorithm is used to determine the partition that produces the optimal value for the objective function and subsequently, it is extended to the m -machine case with $O(\min\{nUB, 2^n\})$ complexity. Approximation algorithms are also presented as solution methods.

(DESSOUKY et al., 1990) present algorithms of different types to solve several problems in uniform machines. Among those, a DP formulation is designed to compute recursively the completion times in the $Qm|r_j, p_{ij} = 1|\sum C_j$ problem. The algorithm is bounded by an $O(mn^{2m+1})$ time complexity.

(LEE; LIMAN, 1993) prove that the $P2|m_2 \leq K|\sum C_j$ problem is NP-complete, where K represents a capacity constraint imposed on the second machine, and develop a pseudo-polynomial DP algorithm to compute the completion time recursively at each stage and a heuristic, which is proven to have a 50% error bound assuming a worst-case scenario.

(ALIDAEI, 1993) establishes similarity patterns between $P2||\sum w_j \overline{F_j}$ and the single machine problem investigated by (HALL; POSNER, 1991). The author incorporates properties previously known such as V-shaped structure and proposes an adapted DP formulation with $O(n \sum w_j)$ complexity for solving the identical parallel machines problem.

(WEBSTER, 1994) describes the DP formulation proposed by (ALIDAEI, 1993) as a specific case of the one proposed by (ROTHKOPF, 1966) for an arbitrary number of machines and a more general function, in which discounts and dependencies between jobs might be allowed.

(CHENG; DIAMOND, 1995) study the problem for minimizing the $Pm|J_f|\sum C_j$, where J_f denotes job families and no job in the second family is allowed to precede jobs in the first one. The authors investigate a two-job family case and a general DP recursive formulation with $O(n^2m)$ overall complexity is devised by adding the contribution of each scheduled job in a family to the objective function separately.

(LEE, 1996) addresses two problems involving unavailability for the second machine on $P2|r - a|\sum w_j C_j$ and $P2|nr - a|\sum w_j C_j$. Initially, a re-indexation of jobs is made through the WSPT priority rule and a DP formulation is provided to compute the objective function in each case, yielding $O(n \sum p_{jk} s_2 p_{\max})$ and $O(n \sum p_{jk} s_2)$ complexities.

(ALON et al., 1998) present an ILP and a DP approach as alternative methods to constitute the formulation of a PTAS to solve the $Pm || \sum f(C_j)$. Although the DP formulation is estimated by a pseudo-polynomial bound, the ILP displays more effective results by producing a polynomial time complexity.

(AKKER; HOOGEVEEN; VELDE, 1999) describe a pricing algorithm as a forward DP formulation to solve a column generation approach to a linear programming relaxation. A B&B algorithm is also provided and the authors prove that, when combined with the linear programming relaxation, it outperforms the previous algorithms used for solving $Pm || \sum w_j C_j$ regarding number of jobs and CPU time.

(CHEN; POWELL, 1999b) devise DP formulations, which are modifications from those used for solving the weighted number of tardy jobs, to minimize the weighted completion time for the same environments. These formulations solve each single machine sub-problem with the same $O(n^2 \sum p_{jk})$ complexity.

(BAPTISTE, 2000) also verifies that the premises to apply the decomposition algorithm hold for the $Pm | p_j = p, r_j | \sum w_j C_j$ problem. Since the algorithm has no modification from the total tardiness version, the complexity is also computed at $O(n^{2m+2})$ bound.

(LEE; CHEN, 2000) analyze the NP-hardness related to the $Pm | nr - a, ind | \sum w_j C_j$ and $Pm | nr - a, dep | \sum w_j C_j$ problem, where *dep* and *ind* define dependent and independent availability, respectively. A column generation method is also proposed in order to provide optimal solutions for these problems working similarly to that seen in (CHEN; POWELL, 1999b). The inbuilt DP formulations are defined with $O(n^2 R_2)$ and $O(n^2 R_3 + n R_4 t_2)$ for the independent and dependent case, respectively. In this notation $R_2 = [\sum p_j - p_{\max}] / m + (t_2 - t_1) + p_{\max}$, $R_3 = [\sum p_j + (m - 1)(t_2 - t_1) + (m - 2)p_{\max}] / m + p_{\max}$ and $R_4 = [\sum p_j + (m - 1)p_{\max}] / m + (t_2 - t_1) + p_{\max}$. Moreover, computational experiments are performed in order to test the algorithm's efficiency for medium sized jobs (up to 60) and m ranging from 2 to 8 machines.

(WEBSTER; AZIZOGLU, 2001) propose a forward and a backward DP algorithm to solve the $Pm | SI_f | \sum w_j C_j$ problem, where SI_f denotes family independent setup times. The problem is classified as NP-hard and the algorithms are pseudo-polynomial with $O(m f^{m+1-f} n^f (\sum w_j)^{m-1})$ complexity for the backward DP and $O(m f^{m+2-f} n^f (\sum w_j + \sum p_j)^m)$ complexity for the forward DP.

(HALL; LESAOANA; POTTS, 2001) also present the DP algorithms in the last two sections to furnish optimal solutions for the $Pm | d_s | \sum C_j$, $Pm | d_s, s = \bar{s} | \sum C_j$ and $Pm | d_s, s = \bar{s} | \sum w_j C_j$. The algorithm used to solve the first problem produces an optimal schedule within a $O(n(\sum p_j)^{m-1})$ complexity while the other one, which solves the remaining problems, shows an $O(n(\sum p_j)^{m\bar{s}-1})$ complexity.

(CHEN; POWELL, 2003) use a network-like structure for $Pm | SI_f | \sum w_j C_j$ and

$Pm|SD_f|\sum w_j C_j$ problems in association with decomposition and column generation algorithms, reaching single-machine sub-problems, which are solved by a DP formulation bounded by $O(n^2 \max\{t_{2j} - t_{1j} + 1\})$ complexity.

(CHEN, 2004) applies the second DP algorithm for the single machine generated via decomposition to find the solution for the $Pm|p(x_j)|\sum w_j C_j + \sum G(x_j)$ problem with $O(n^2 \sum p_j)$ complexity and based on the principle of optimality.

(RAMACHANDRA; ELMAGHRABY, 2006) offer a BIP method and a DP algorithm to find optimal solutions for $P2|prec|\sum w_j C_j$. The DP presents better results for a larger number of jobs and processing times range, even though the DP can generate a smaller number of solutions for reduced number of jobs and processing times. A genetic algorithm (GA) is also proposed for larger number of jobs and the 2 and 3-machine cases and experimental results are registered.

(MELLOULI et al., 2009) evaluate the performance of exact algorithms based on MILP, B&B and DP for the $Pm|h(\mu_k)|\sum C_j$. Experiments are performed on job sets with up to 200 jobs and machine sets ranging from 2 to 4 machines. DP shows the best results for $m = 2$, being able to solve 200-job instances and reasonable results for $m \geq 2$, however, for this case, its efficiency is compromised as jobs as job sets become higher than 25.

(KIM; SUNG; LEE, 2009) investigate the minimization of $Pm|s - prec|\sum C_j$, where $s - prec$ indicates that s precedence relationships are allowed into the scenario. The authors propose a LP-based heuristic, whose core solution is given by an $O(n \sum p_j)$ pseudo-polynomial DP approach to a separation problem. Experiments are performed for at most 300 jobs and 10 machines and the heuristic is compared to a SPT-based heuristic, proving the former obtains better results in terms of (n/m) ratio and CPU times.

(ZHAO; TANG; CHENG, 2009) investigate a special case of maintenance, which is the rate-modifying (denoted by α_j) activity, for two identical parallel machines and $\sum w_j C_j$ criterion. The authors also develop a DP algorithm for cases where jobs are agreeable according to the WSPT-ratio and α_j -WSPT-ratio and it presents $O(n(\sum \max\{p_j, \alpha_j p_j\})^6)$ complexity and show that the approach can be extended to a m -machine scenario with $O(n(\sum \max\{p_j, \alpha_j p_j\})^{3m})$.

(TANG; ZHANG, 2011) devise a Lagrangian relaxation algorithm in order to minimize the $Rm|r_j|\sum C_j$ problem. Such algorithm can be divided in two parts, which are connected. A relaxation decomposition is performed and it originates machine-level sub-problems, whose solvability is given by a combination of a forward DP recursion and WSPT priority rule. Experiments are performed in order to verify the efficiency of the method and it furnishes high quality solutions in reasonable CPU time for instances with up to 100 jobs and 10 machines.

(ZHAO; TANG, 2014) develop a DP approach for the $Pm|nr - a, p_j(s_j)|\sum w_j C_j$

problem. This algorithm utilizes the premises of the Principle of Optimality and indexation of jobs follows the weighted shortest deteriorating rate (WSDR) priority rule with complexity bounded by $O(n \prod (t_{1k} - t_0) \prod (t_{2k}(\prod 1 + b_j)^m))$.

([YOO; LEE, 2016](#)) also derive DP algorithms for the $Pm|ind|\sum C_j$, $Pm|ind|\sum w_j C_j$ problems, their dependent cases counterparts and the dependent cases with equal processing times for all jobs and their complexities are equivalent to those shown for the total lateness and maximum lateness.

Table 8: Completion Time related problems for parallel machines

Authors	Problem Notation	Complexity	Method
(BAKER; MERTEN, 1973)	$Pm \sum w_j \bar{F}_j$	–	DP and Heuristics
(HOROWITZ; SAHNI, 1976)	$Qm \sum w_j \bar{F}_j$	–	DP
(DESSOUKY et al., 1990)	$Qm \sum w_j \bar{F}_j$	$O(\min\{nUB, 2^n\})$	DP and Approximation algorithm
(LEE; LIMAN, 1993)	$Qm r_j, p_{ij} = 1 \sum C_j$	$O(mn^{2m+1})$	DP
(ALIDAEI, 1993)	$P2 m_2 \leq K \sum C_j$	–	DP and Heuristic
(WEBSTER, 1994)	$P2 \sum w_j \bar{F}_j$	$O(n \sum w_j)$	DP
(CHENG; DIAMOND, 1995)	$Qm \sum w_j \bar{F}_j$	–	DP
(LEE, 1996)	$Pm J_f \sum C_j$	$O(n^2 m)$	DP
(ALON et al., 1998)	$P2 r - a \sum w_j C_j$	$O(n \sum p_{jk} s_2 p_{\max})$	DP
(AKKER; HOOGEVEEN; VELDE, 1999)	$P2 nr - a \sum w_j C_j$	$O(n \sum p_{jk} s_2)$	DP
(CHEN; POWELL, 1999b)	$Pm \sum f(C_j)$	–	DP, ILP and PTAS
(BAPTISTE, 2000)	$Pm \sum w_j C_j$	$O(n \sum p_j)$	Linear Relaxation with DP
(LEE; CHEN, 2000)	$Pm \sum w_j C_j$	$O(n^2 \sum p_{jk})$	B&B with DP
(WEBSTER; AZIZOGLU, 2001)	$Qm \sum w_j C_j$	$O(n^2 \sum p_{jk})$	B&B with DP
(HALL; LESAOANA; POTTS, 2001)	$Rm \sum w_j C_j$	$O(n^2 \sum p_{jk})$	B&B with DP
(CHEN; POWELL, 2003)	$Pm p_j = p, r_j \sum w_j C_j$	$O(n^{2m+2})$	DP and Decomposition algorithm
(CHEN, 2004)	$Pm nr - a, \text{ind} \sum w_j C_j$	$O(n^2 R_2)$	B&B with DP
(RAMACHANDRA; ELMAGHRABY, 2006)	$Pm nr - a, \text{dep} \sum w_j C_j$	$O(n^2 R_3 + n R_4 t_2)$	B&B with DP
(MELLOULI et al., 2009)	$Pm SI_f \sum w_j C_j$	$O(m f^{m+1} - f n^f (\sum w_j)^{m-1})$	DP
(KIM; SUNG; LEE, 2009)	$Pm SI_f \sum w_j C_j$	$O(m f^{m+2} - f n^f (\sum w_j + \sum p_j)^m)$	DP
(ZHAO; TANG; CHENG, 2009)	$Pm ds, s = \bar{s} \sum C_j$	$O(n (\sum p_j)^{m-1})$	DP
(TANG; ZHANG, 2011)	$Pm ds, s = \bar{s} \sum w_j C_j$	$O(n (\sum p_j)^{m-1})$	DP
(ZHAO; TANG, 2014)	$Pm SI_f \sum w_j U_j$	$O(n (\sum p_j)^{m-1})$	DP
	$Pm SD_f \sum w_j U_j$	$O(n (\sum p_j)^{m-1})$	DP
	$Pm p(x_j) \sum w_j C_j + \sum G(x_j)$	$O(n^2 \max\{t_{2j} - t_{1j} + 1\})$	DP
	$P2 \text{prec} \sum w_j C_j$	$O(n^2 \max\{t_{2j} - t_{1j} + 1\})$	B&B with DP
	$Pm h(\mu_k) \sum C_j$	$O(n^2 \sum p_j)$	DP, BIP and GA
	$Pm s - \text{prec} \sum C_j$	–	DP, B&B and MILP
	$P2 p_j(\alpha_j) \sum w_j C_j$	$O(n \sum p_j)$	LP-based Heuristic with DP
	$Pm p_j(\alpha_j) \sum w_j C_j$	$O(n (\sum \max\{p_j, \alpha_j p_j\})^6)$	DP
	$Rm r_j \sum C_j$	$O(n (\sum \max\{p_j, \alpha_j p_j\})^{3m})$	DP
	$Pm nr - a, p_j(s_j) \sum w_j C_j$	$O(n \sum p_{jk})$	Lagrangian Relaxation with DP
		$O(n \prod (t_{1k} - t_0) \prod (t_{2k} - t_0) \prod (1 + b_j)^m)$	DP

Table 8 continued from previous page

$Pm ind \sum w_j C_j$	$O(nm(R_1 + R_2)^m)$	DP
$Pm ind \sum C_j$	$O(nR_1^{2m}r_2^m)$	DP
$Pm dep \sum C_j$	$O(nmR_3^m R_4^m T^m)$	DP
$Pm dep \sum w_j C_j$	$O(nmR_3^m R_4^m T^m)$	DP
$Pm dep, p_j = p \sum C_j$	$O((n+1)^{3m})$	DP
$Pm dep, p_j = p \sum w_j C_j$	$O((n+1)^{3m})$	DP

3.4.4 Makespan

(HOROWITZ; SAHNI, 1976) also investigate solution methods to minimize makespan in parallel machines. In order to do so, a DP algorithm is devised for the $R2||C_{\max}$ problem, in which the optimal schedule can be found using an SPT rule and the partition of jobs can be determined through the DP. This algorithm presents $O(\min\{n \min\{\sum C_{ik}\}, 2^n\})$ complexity and, can be naturally extended for the $Qm||C_{\max}$ problem with $O(\min\{n \min\{\sum C_{ik}\}, m^n\})$ complexity.

(JANSEN; PORKOLAB, 2001) develop an approximation algorithm to solve the $Rm||C_{\max}$ problem. This method contains several steps and among them, a DP algorithm is devised in order to find an optimal schedule for jobs with long processing times, which are grouped according similarities tested against a given criterion. Such algorithm is bound with $O(lm(lm/\delta)^m)$, with l being the number of jobs grouped accordingly and δ a partition parameter.

(HALL; LESAOANA; POTTS, 2001) applies their DP algorithms to in order to analyze the behaviour for makespan. Both of them are suitable for such criterion and the complexities associated are also $O(n(\sum p_j)^{m-1})$ and $O(n(\sum p_j)^{m\bar{s}-1})$.

(GHIRARDI; POTTS, 2005) propose a beam search algorithm to solve the $Rm||C_{\max}$ problem. Such algorithm presents similarities to a B&B algorithm, however a DP-based pruning criterion is used in order to select which vertices should compose the solution. The theoretical results are computationally tested and compared to an approximation algorithm for jobs ranging from 100 to 1000 and machines ranging from 10 to 100. Results show that for a majority of cases the beam search shows superior results in terms of CPU times and solution quality.

(TANG; LUO, 2006) develop a powerful iterated local search (ILS) algorithm for solving instances with up to 1000 and 40 machines regarding the $Pm||C_{\max}$ problem. An approximately DP algorithm is integrated to its constitution to furnish the minimum cost to a cyclic exchange in an associated graph so as to design an initial solution for the algorithm.

(GLASS; KELLERER, 2007) consider a job assignment restriction for the $P2||C_{\max}$, which means that there might occur impostions on which machines jobs are allocated to. They propose a FPTAS and an $O(np_{\max})$ complexity DP formulation to tackle the problem, with the latter being a a knapsack adapted formulation through a Subset Sum Problem analysis.

(KELLERER; STRUSEVICH, 2008) evaluate solutions to solve the job assignment restriction problem denoted by $P2|p_j(x_j), x_j = 1, rnw, Bi|C_{\max}$, where rnw represents a renewable resource and Bi reinforces that consumption occurs while this resource is being used as a "speed-up" mechanism for the processing times. A DP algorithm is proposed to

solve the problem, which is partitioned into two knapsack problems, however the DP does not contribute efficiently to this objective and a DP-based FPTAS is provided instead. The complexity of the DP algorithm is bounded by $O(nR)$, where $R = \max\{A, B\}$, $A = \sum_{j \in N_1} (a_{j1} - b_{j1})$ and $B = \sum_{j \in N_2} (a_{j2} - b_{j2})$.

(LI; WANG, 2010) conduct a research regarding the $Pm|r_j|C_{\max}$ considering the job assignment restriction constraint with a DP-based PTAS as solution method for an arbitrary number of machines and a DP-based FPTAS for a fixed number of machines. The algorithms are bounded by $O(m|X_m||Y_m|)$ (with $|X_m|, |Y_m|$ being assignment subsets for the last machine) and $O(n^{m+1}(1/\varepsilon)^m)$, respectively.

(HANED et al., 2012) consider the problem of minimizing the makespan taking into account delay transportation and preemption constraints, which is denoted by $P2|delay, prmp|C_{\max}$. Despite the paper focusing on a FPTAS, the authors also present a DP algorithm with $O(n^2 \sum p_j)$ complexity to solve the problem, which is derived from a DP formulation proposed for the $P2||C_{\max}$ in this paper.

(YOO; LEE, 2016) develop DP procedures, which have been also applied to lateness and total completion times functions, and convey them to solve the $Pm|ind|C_{\max}$, $Pm|dep|C_{\max}$ and $Pm|dep, p_j = p|C_{\max}$ problems with the same complexity achieved by the other criteria.

(RUDEK, 2017) discusses the influence on learning or aging effects regarding arbitrary functions of processing times to minimize the makespan. The author includes preliminary topics on continuous functions for the single machine environment and devises a pseudo-polynomial DP algorithm for integer-valued functions, which is bounded by $O(n(\sum p_j)^m(\sum p_j f(j))^m)$ as well as the real-valued.

(GHALAMI; GROSU, 2019) attempt a more sophisticated approach to finding solutions for the $Pm||C_{\max}$ problem, in which they resort to a parallel DP method to construct a PTAS. First a DP approach is designed in order to select all the machines where $C_{\max}$ does not surpass a given threshold and then the DP formulation is associated to a graph-like structure in order to determine how to proceed with the adaptation to the parallel DP version with $O((n/\varepsilon)^{(1/\varepsilon^2)} / \sum p_j)$.

Table 9: Makespan problems for parallel machines

Authors	Problem Notation	Complexity	Method
(HOROWITZ; SAHNI, 1976)	$R2 C_{\max}$	$O(\min\{n \min\{\sum C_{ik}\}, 2^n\})$	DP and Approximation algorithm
(JANSEN; PORKOLAB, 2001)	$Qm C_{\max}$	$O(\min\{n \min\{\sum C_{ik}\}, m^n\})$	DP and Approximation algorithm
(HALL; LESAOANA; POTTS, 2001)	$Rm C_{\max}$	$O(lm(lm/\delta)^m)$	Approximation algorithm with DP
	$Pm d_s C_{\max}$	$O(n(\sum p_j)^{m-1})$	DP
	$Pm d_s, s = \bar{s} C_{\max}$	$O(n(\sum p_j)^{m\bar{s}-1})$	DP
(GHIRARDI; POTTS, 2005)	$Rm C_{\max}$	–	Beam Search with DP
(TANG; LUO, 2006)	$Pm C_{\max}$	–	Heuristic with DP
(GLASS; KELLERER, 2007)	$P2 C_{\max}$	$O(np_{\max})$	DP and FPTAS
(KELLERER; STRUSEVICH, 2008)	$P2 p_j(x_j), x_j = 1, rnw, Bi C_{\max}$	$O(nR)$	FPTAS with DP
(LI; WANG, 2010)	$Pm r_j C_{\max}$	$O(m X_m Y_m)$	PTAS with DP
	$Pm r_j C_{\max}$	$O(n^{m+1}(1/\varepsilon)^m)$	FPTAS with DP
(HANED et al., 2012)	$P2 delay, prmp C_{\max}$	$O(n^2 \sum p_j)$	DP and FPTAS
(YOO; LEE, 2016)	$Pm ind C_{\max}$	$O(nR_1^{2m}r_2^m)$	DP
	$Pm dep C_{\max}$	$O(nmR_3^mR_4^mT^m)$	DP
	$Pm dep, p_j = p C_{\max}$	$O((n+1)^{3m})$	DP
(RUDEK, 2017)	$Pm Ae C_{\max}$	$O(n(\sum p_j)^m(\sum p_j f(j))^m)$	DP
	$Pm Le C_{\max}$	$O(n(\sum p_j)^m(\sum p_j f(j))^m)$	DP
(GHALAMI; GROSU, 2019)	$Pm C_{\max}$	$O((n/\varepsilon)^{(1/\varepsilon^2)} / \sum p_j)$	PTAS with Parallel DP

3.4.5 Multi-criteria and Additional Objective Functions

(ROTHKOPF, 1966) describes a DP formulation to solve two problems related to the minimization of a linear waiting cost function in m uniform machines and a non-linear waiting cost function in m identical machines. This method is able to create $n - 1$ independent state variables, however the calculations reach exponential order.

(GUPTA; MAYKUT, 1973) develop a DP formulation for three waiting cost functions. For the linear and exponential cases, the authors prove that the allocation problem is reduced to a solving one sequencing problem per stage, however for non-decreasing monotonic general functions the same cannot be stated and exponential growth of $O(2^n)$ order is reached.

(SO, 1990) derives three heuristic procedures to solve the $Pm|S_{if}|\sum Rew_j$ problem in order to maximize the reward by scheduling composed by I job families and J job types. The first and second heuristics employ DP formulations with by finding an optimal schedule for each machine as a single machine problem (Sequential Heuristic) and assigning each machine to a set of jobs (Decomposition Heuristic). The algorithms present $O(JmK^2)$ and $O(Jm^2K^2 + Jm^2K \ln(mK))$, respectively.

(KRÄMER; LEE, 1994) study optimality properties and complexity analysis for the $P2|d_w|\sum(E_j + T_j)$ problem and proves it to be NP-hard. The optimality properties are applied to the proposed algorithm that is partitioned into an $O(n^2d_w)$ complexity DP algorithm to schedule the jobs that fit in the due window interval and an absolute error bound heuristic method, which reduces the overall complexity of the algorithm.

(DE; GHOSH; WELLS, 1994) outline the complexity analysis for the $Pm|d_j = d|\sum(\alpha_j E_j + \beta_j T_j)$ and $Pm|d_j = d|\sum \alpha_j E_j + \beta_j T_j + \gamma_j L(d)$ problems and prove their NP-hardness in the strong sense and their particular cases involving two machines are NP-hard. A pseudo-polynomial DP algorithm is provided for each problem based on the optimality properties and computational experiments are performed, indicating that the proposed DP formulation yields better results in terms of CPU times for the $Pm|d_j = d|\sum(\alpha_j E_j + \beta_j T_j)$ problem.

(CHEN; POWELL, 1999a) develop a complex scheme to analyze the $Pm||\sum(\alpha_j E_j + \beta_j T_j)$. A branch and bound algorithm is proposed with a decomposition process via column generation process until each node of the tree can be examined as two single machines sub-problems which, in turn, rely on two DP algorithms of $O(n^2 \sum p_j)$ complexity to be solved optimally.

(BARTAL et al., 2000) apply the rounding DP method seen in (HOROWITZ; SAHNI, 1976) to design a FPTAS to solve the $Pm||C_{\max} + \sum \theta_j^{rej}$ problem. The DP formulation denotes optimal solutions for each instance and it is used to verify the approximation ratio of the FPTAS.

(SUN; WANG, 2003) elaborate a DP algorithm based on the LPT priority rule to establish a initial indexation of jobs and define a recursive relation to compute the objective function for the $Pm|w_j = a_j p_j, d_j = d|\sum w_j|C_j - d|$ problem. Since the DP is classified as a pseudo-polynomial algorithm with $O(nmd^m(\sum p_j)^{m-1})$ complexity, the problem is NP-hard in the ordinary sense. Furthermore, an additional study on heuristics is shown with calculation of error bounds.

(JANSEN; MASTROLILLI, 2004) incorporate PTASs to solve three problems related to makespan, in which three bi-criteria involving a cost function with respect to resource allocation. Among these, one of the PTASs, which is applied to the $Pm|p_j(x_j)|(G(x_j), C_{\max} \leq K)$ uses a DP algorithm to reduce to a polynomial number the state space of the problem

(SUNG; VLACH, 2005) investigate the minimization the $Rm||\sum (D_j + U_j)$ problem by analyzing its complexity and classifying it as NP-hard in the strong sense. In addition to this analysis, the authors design an $O(mn^{m+1})$ DP polynomial-time algorithm that consists of three steps and incorporates V-bounded structure properties.

(LI et al., 2005) work with a criterion that computes the weighted square sum of the difference between the real length of time on a given machine (y_k) and the predefined length of time on a given machine (z_k). The problem is modelled as a 0-1 quadratic model and a DP formulation is provided in order to solve it.

(RIOS-SOLIS; SOURD, 2008) evaluate theoretical and practical aspects of an exponential search neighbourhood algorithm for the $Pm|d_j = d|\sum E_j + T_j$ problem. Optimality properties are presented and a DP approach, which is based on interchange operations of early and tardy jobs in the schedule, is given. This formulation is the main element to optimize the search of feasible sequences and, therefore, the algorithm's complexity is mostly defined by the DP's, which is dependent on number of allocations. Experiments are performed in a set of jobs with up to 150 and applied in $m = \{4, 8\}$ with satisfactory results given the CPU times and the gap between upper and lower bounds.

(KEDAD-SIDHOUM; SOLIS; SOURD, 2008) adapt their own approach of the single-machine problem with earliness-tardiness criterion and devise bounds for the $Pm||\sum (E_j + T_j)$ problem. The DP-based heuristic and the Lagrangian bound show the best results when compared to the remaining bounds and the heuristic yields the smallest gap with regard to CPU times that are close to those registered for the Lagrangian bound.

(AGNETIS; ALFIERI; NICOSIA, 2009) resort to a bounding scheme to assess the quality of given heuristic to solve general problems whose functions are defined as $\max\{f(C_j)\}$. Therefore, a bounding scheme is developed in order to provide a comparison with the proposed heuristics. Such scheme relies on the column generation technique, which, in turn, is divided to the level of a pricing algorithm that might be solved by a

DP formulation or a DP-based heuristic. Experimental results are presented for $T_{\max}$ and $C_{\max}$ but we omit them since they are not part of the scope of our research.

(LEYVAND et al., 2010) analyze a variety of problems involving resource consumption and earliness-tardiness criteria with controllable processing times. They develop a $O(mn^{m+1})$ time DP algorithm to solve the $Rm|p_j(x_j)|\sum(E_j + T_j) + \sum G(x_j)$ and a FPTAS is derived from a DP formulation for the remaining problems.

(LI; YUAN, 2010) design a FPTAS and a polynomial time DP algorithm to solve $Pm|rej, p_j(t)|\sum w_j C_j + \sum \theta_j^{rej}$ and $Pm|rej, p_j(t)|\sum C_j + \sum \theta_j^{rej}$ problems, respectively. The FPTAS depends on a DP recursion, which can be computed in $O(n(\sum n \log(1 + b_j)/\varepsilon)^m)$ complexity. The polynomial DP algorithm is an extension of the algorithm devised by (CHENG; SUN, 2009) for single machine environment and yields $O(n^2)$ complexity.

(ZHAO; LU, 2013) analyze two distinct models involving two-agent scheduling criterion. The first model is related to the $Pm||C_{\max}^A : C_{\max}^B \leq K$ and a DP algorithm is developed with $O(mn_A n_B (\sum p_j)_A^m (\sum p_j)_B^m)$ as well as an FPTAS derived from it. An akin procedure is devised for the $Pm||\sum C_j^A : C_{\max}^B \leq K$ with the the previously mentioned complexity.

(DONG, 2013) aims at minimizing functions that are related to machine setup costs, job processing and energy consumption considering that the maintenance intervals are set within the problem rather than being prespecified. A B&B algorithm is proposed and its inner mechanism falls under the set partition scenario, which is ultimately solved by a DP algorithm in conjunction with a column generation method. The algorithm is tested via computational experiments, which proves the method is efficient for medium sized problems.

(REBAI; KACEM; ADJALLAH, 2013) evaluate solutions for the $Pm|h(\mu)|\sum w_j C_j + \sum H(t_1, t_2)$ where $H(t_1, t_2)$ represents the preventive maintenance cost. The solutions proposed rely on a B&B algorithm and a GA, which can be reinforced by applying a DP formulation in order to obtain better initial solutions for the heuristic. The authors show that for problems with up to 300 jobs and 5 machines, both algorithms produce high quality solutions within reasonable time. In addition, the use of DP enhances the GA performance regarding solution quality.

(ZHANG; LU, 2016) design a theoretical analysis on the $Pm|rej, r_j|C_{\max} + \sum \theta_j^{rej}$ problem. The authors propose a pseudo-polynomial DP approach in conjunction with earliest release date (ERD) priority rule for indexation and also apply it to a special case denoted by $Pm|rej, r_j, p_j = p|C_{\max} + \sum \theta_j^{rej}$. Each problem presents, respectively, $O(mn(r_{\max} + \sum p_j)^m)$ and $O(mn^{2m+1})$ complexities. Furthermore, a FPTAS is also offered as solution method for the first problem.

(YIN et al., 2016) present a theoretical assessment for designing algorithms to solve

two-competing agents $Rm||(\sum (\alpha_j D_j + \beta_j U_j)_A, \min (\alpha_j D_j + \beta_j U_j)_B)$ and $Rm||(\sum (\alpha_j D_j + \beta_j U_j)_A, \sum (\alpha_j D_j + \beta_j U_j)_B)$ problems. The authors develop a pseudo-polynomial DP algorithm for each case, including elimination properties to reduce the solutions state space and their complexities are bounded by $O(mn_A^{m+1}n_B)$ and $O(m^2n^{m+1} \min\{\sum (\alpha_j + \beta_j)_A, \sum (\alpha_j + \beta_j)_B\})$, respectively. In addition, and FPTAS is derived from the DP approach for the last problem.

(CHEN et al., 2016) evaluate online and offline scheduling problems stemming from the $P2|d_j = d|\sum Y_j$. The authors prove the given problem is NP-hard, however a simple pseudo-polynomial DP algorithm with $O(nd^2)$ complexity is provided. In addition, they also state how close this problem is to a knapsack problem/bin packing problem as well as how it can relate to the $P2||C_{\max}$ problem.

(STERNA; CZERNIACHOWSKA, 2017) develop a PTAS in order to solve the $Pm||\sum V_j$. Its construction derives from adaptations made in a series of list algorithms and the DP formulation designed by (CHEN et al., 2016). Dominance properties are concocted from relative improvements performed by observing the algorithms' behaviour and the PTAS is generated upon them so as to work in an optimal manner.

(YIN et al., 2019) devise a B&P algorithm to determine solutions for the two-agent $Pm||\sum C_{j_A} : \sum w_j U_{j_B}$ problem. The solution method is composed of a three-step algorithm and an $O(n_A n_B \max\{n_A, n_B\} \sum p_j)$ DP forward recursion is used in order to provide solutions for the pricing sub-problem generated from a column generation method in each node. Furthermore, a dominance rule is also coupled to the DP in order to reduce the state space via fathoming. Experiments are performed for at most 80 jobs and 10 machines.

Table 10: Multicriteria and Additional Functions problems for parallel machines

Authors	Problem Notation	Complexity	Method
(ROTHKOPF, 1966)	$Pm \sum c_j t$	—	DP
(GUPTA; MAYKUT, 1973)	$Qm \sum c_j (1 - e^{-rt})$	—	DP
	$Pm \sum c_j t$	—	DP
	$Pm \sum c_j (1 - e^{-rt})$	—	DP
(SO, 1990)	$Pm \sum G(t)$	$O(2^n)$	DP
	$Pm S_{if} \sum Rew_j$	$O(JmK^2)$	Heuristic with DP
(KRÄMER; LEE, 1994)	$Pm S_{if} \sum Rew_j$	$O(Jm^2 K^2 + Jm^2 K \ln(mK))$	Heuristic with DP
(DE; GHOSH; WELLS, 1994)	$P2 d_w \sum E_j + T_j$	$O(n^2 d_w)$	DP and Heuristic
	$Pm d_j = d \sum \alpha_j E_j + \beta_j T_j$	$O(nm(2p_1)^m)$	DP
(CHEN; POWELL, 1999a)	$Pm d_j = d \sum \alpha_j E_j + \beta_j T_j$	$O(nm[(n/m) + 1]p_1)^{2m}$	DP
(BARTAL et al., 2000)	$Pm d_j = d \sum \alpha_j E_j + \beta_j T_j + \gamma_j L(d)$	$O(nm \sum p_j)$	B&B with DP
(SUN; WANG, 2003)	$Pm \sum \alpha_j E_j + \beta_j T_j$	—	FPTAS with DP
(SUNG; VLACH, 2005)	$Pm C_{\max} + \sum \theta_j^{rej}$	$O(nmd^m (\sum p_j)^{m-1})$	DP and Heuristic
(LI et al., 2005)	$Pm w_j = a_j p_j, d_j = d \sum w_j C_j - d $	$O(mn^{m+1})$	DP
(RIOS-SOLIS; SOURD, 2008)	$Rm \sum D_j + U_j$	—	DP and Heuristic
(KEDAD-SIDHOUM; SOLIS; SOURD, 2008)	$Rm \sum (y_k - z_k)^2$	—	DP with Heuristic and LBs
(AGNETIS; ALFIERI; NICOSIA, 2009)	$Pm \sum (E_j + T_j)$	$O(n^2)$	Bounding scheme with DP
(LEYVAND et al., 2010)	$Pm \max\{f(C_j)\}$	—	DP and FPTAS
(LI; YUAN, 2010)	$Rm p_j(x_j) \sum (E_j + T_j) + \sum G(x_j)$	$O(mn^{m+1})$	FPTAS with DP
	$Pm rej, p_j(t) \sum w_j C_j + \sum \theta_j^{rej}$	$O(n(\sum n \log(1 + b_j)/\varepsilon)^m)$	DP
(ZHAO; LU, 2013)	$Pm rej, p_j(t) \sum C_j + \sum \theta_j^{rej}$	$O(n^2)$	DP and FPTAS
	$Pm C_{\max}^A : C_{\max}^B \leq K$	$O(mn_A n_B (\sum p_j)_A^m (\sum p_j)_B^m)$	DP and FPTAS
(DONG, 2013)	$Pm \sum C_j^A : C_{\max}^B \leq K$	$O(mn_A n_B (\sum p_j)_A^m (\sum p_j)_B^m)$	B&B with DP
(REBAI; KACEM; ADJALLAH, 2013)	$Pm h(\mu) \sum SC(t) + \sum EC(t) + \sum JPC(t)$	—	GA with DP and B&B
(ZHANG; LU, 2016)	$Pm h(\mu) \sum w_j C_j + \sum H(t_1, t_2)$	$O(mn(\tau_{\max} + \sum p_j)^m)$	DP and FPTAS
(YIN et al., 2016)	$Pm rej, r_j C_{\max} + \sum \theta_j^{rej}$	$O(mn^{2m+1})$	DP
	$Pm rej, r_j, p_j = p C_{\max} + \sum \theta_j^{rej}$	$O(mn^{m+1} n_B)$	DP
(CHEN et al., 2016)	$Rm (\sum (\alpha_j D_j + \beta_j U_j)_A, \min(\alpha_j D_j + \beta_j U_j)_B)$	$O(m^2 n^{m+1} \min\{\sum (\alpha_j + \beta_j)_A, \sum (\alpha_j + \beta_j)_B\})$	DP and FPTAS
(STERNA; CZERNIACHOWSKA, 2017)	$Rm (\sum (\alpha_j D_j + \beta_j U_j)_A, \sum (\alpha_j D_j + \beta_j U_j)_B)$	$O(nd^2)$	DP
	$P2 d_j = d \sum Y_j$	—	FPTAS with DP
	$Pm d_j = d \sum V_j$	—	

(YIN et al., 2019)

Table 10 continued from previous page

$Pm \sum C_{j\ A}:\sum w_jU_{j\ B}$	$O(n_A n_B \max\{n_A, n_B\} \sum p_j)$	B&P with DP
--------------------------------------	--	-------------

3.5 Dynamic Programming approach for the Flow Shop Environment

Flow shop scheduling has been an environment that, despite being far more complex compared to the ones previously outlined in this research, has received nearly undivided attention from most authors whose scheduling is a core theme and this interest can be traced back nearly to 70 years. Nevertheless, when seeking information on the contribution of dynamic programming in this environment, one can notice that only few authors have actually employed it as a primary or secondary method to solve several problems that compose the literature in flow shop scheduling and those are presented below. One may notice that, in this section, some of the criteria have been grouped in the same subsection due to a decrease in the number of articles involving DP and a specific criterion.

3.5.1 Classic due dates-related criteria

(JÓZEFOWSKA; JURISCH; KUBIAK, 1994) address the $F2|d_j = d|\sum U_j$ problem among a series of shop environments and develop a study on the complexity proving it is ordinary NP-hard by association with a partition problem. In addition a $O(nd^2)$ pseudopolynomial DP formalism is introduced as solution for the given problem.

(SONMEZ; BAYKASOGLU, 1998) incorporate the theoretical analysis of the $Fm|s_{ijk}|\sum w_j T_j$ problem into a study case in a pipe manufacturing industry. The first part of the article is dedicated to introducing a DP formulation for the problem, which seems to be quite simple compared to some robust exact methods found in literature, although no complexity analysis is displayed. Sequentially, the authors employ the method to the pipe scenario and, due to the shop floor structure, they are able to convert the problem into a single machine and apply the DP formulation to solve the problem efficiently.

(NISHI; HIRANAKA, 2013) explore features of a Lagrangian relaxation to solve the $Fm|s_{ijk}|\sum w_j T_j$ problem. The algorithm undergoes a decomposition via machine capacity relaxation and a DP subroutine is proposed for its solution. Furthermore, the authors improve the lower bound by applying a cut generation method that also resorts to a DP subroutine. Experiments show that the algorithm is quite efficient for medium sized instances and they suggest the investigation of other methods to carry out more accurate cuts.

(KOULAMAS, 2020) analyzes solutions for the $Fm|p_{jk} = p_j|\sum T_j$ problem by developing a DP formalism, which is actually an extension from the one presented by (LAWLER, 1977) by expanding the recursion method to m machines. The author proves the algorithm runs in $O(n^4 \sum p_j)$ and performs experiments for problems with up to 50 jobs and 20 machines.

Table 11: Due date related problems for flow shop

Authors	Problem	Notation	Complexity	Method
(JÓZEFOWSKA; JURISCH; KUBIAK, 1994)	$F2 d_j = d \sum U_j$	$O(nd^2)$	DP	
(SONMEZ; BAYKASOGLU, 1998)	$Fm s_{ijk} \sum T_j$	–	DP	
(NISHI; HIRANAKA, 2013)	$Fm s_{ijk} \sum w_j T_j$	–	Lagrangian Relaxation with DP	
(KOULAMAS, 2020)	$Fm p_{jk} = p_j \sum T_j$	$O(n^4 \sum p_j)$	DP and FPTAS	

3.5.2 Makespan and Completion Time-related criteria

(CORWIN; ESOGBUE, 1974) address the $F2|s_{ij}|C_{\max}$ problem by dividing it into two different problems that are characterized by the the absence of sequence dependent setup times. When assuming the first one is allowed to present setup times, a backward DP formulation is proposed. Conversely, the authors offer a forward DP formulation for the problem where setup times are allowed on the second machine. Furthermore, in order to reduce storage in each case, some dominance rules are also included in both cases and the authors are able to find optimal solutions for 15 job-instances.

(DUTTA; CUNNINGHAM, 1975) develop a theoretical study on mathematical formulations to describe the $F2|b_{j,j+1}|C_{\max}$ with $b_{j,j+1}$ denoting a limited buffer scenario. First, the authors introduce a DP formulation in order to find optimal solutions for the problem and, although it might be considered a fairly realistic model, it is an expensive one from the computational requirements standpoint. Therefore, a reduction in its complexity is performed and a sub-optimal DP is presented.

(VELDE, 1990) devise a B&B algorithm that essentially converges to a Lagrangian Relaxation which, in turn, results in a linear ordering problem. In order to create a more efficient bound, a DP dominance rule in association with two other based on Johnson's rule is applied to reduce the search tree.

(KOVALYOV; WERNER, 1997) analyze theoretical aspects of the $F2|r_j|C_{\max}$ problem and devise a PTAS to compare with preliminary studies. The approach is based on a DP formulation that bear resemblance to those proposed by (LAWLER; MOORE, 1969) and (ROTHKOPF, 1966) and has $O(ni(2R)^i R^{2i-2})$ complexity, being i a subset of J and $R = \max\{\sum p_{j1}, \sum p_{j2}\}$.

(LEE, 1997) evaluates the $F2|r - a(M_1)|C_{\max}$ and $F2|r - a(M_2)|C_{\max}$ problems and a complexity analysis proves both problems are NP-hard. Furthermore, the author design a pseudo-polynomial DP algorithm for the problem with unavailability placed on machine 1. Although the method is deemed fairly strong, its complexity renders it not viable and Johnson-based heuristics are offered as alternative solutions.

(LEE, 1999) also extends the analysis in (LEE, 1997) for the semi-resumable case problem. A complexity analysis is stated about the NP-hardness of the problem, since it is considered a generalized problem of the non-resumable scenario. In addition, the results are the same found in its previous work regarding the DP formulation.

(HOU; HOOGEVEEN, 2003) investigate the $F3|p_{jk} = p_j/s_k|C_{\max}$ problems under the assumption that all machines operate under different speeds and machine 2 is the slowest one. An $O(ns_1s_2^2s_3 \max\{p_j\}^2)$ DP algorithm is proposed by incorporating V-shaped optimality properties as well as a recursive function that computes the makespan for sub-sequences. Furthermore, a combination of optimality properties and a series of lemmas

prove that the problem can be classified as ordinary NP-hard.

(BOUQUARD et al., 2005) develop a diversified research on problems involving number of regular and no-wait jobs and methods are provided so as to solve them. Among them, the $F2|reg + nwt, pmru, n_{nwt} = q|C_{\max}$ can be solved via a pseudo-polynomial time DP algorithm and a subsequent FPTAS (by converting the former approach). The complexity associated with the DP is of the form $O(n(\tilde{M})^{3q+1})$, where $\tilde{M}$ represents the maximum value among states generated when scheduling a given job.

(ALLAOUI et al., 2006) investigate the effects of heuristics derived from Johnson's algorithm and a DP model applied to the $F2|r - a(M_1)|C_{\max}$ and $F2|nr - a(M_1)|C_{\max}$ problems. A complexity analysis is carried out for the pseudo-polynomial DP model and, despite containing an exponential component and the number of subsets in J , it performs significantly better than the formulation proposed by (LEE, 1997), since it does not depend on processing times.

(KUBZIN; STRUSEVICH, 2006) investigate some shop problems considering unavailability for either one or two machines. One of the problems of main interest was the $F2|h(\mu)|C_{\max}$ problem and the authors develop a theoretical analysis on its complexity, which has been categorized as binary NP-hard with the manufacturing of a DP formulation that delivers the makespan in $O(nM^4)$, with M being the number of generated state variables dependent on the maintenance interval. Furthermore, a FPTAS can be derived from this approach to solve the same problem using the DP as a subroutine.

(BAUTISTA et al., 2012) delineate properties related to a DP-based heuristic, which is commonly known as Bounded Dynamic Programming (BDP), for solving the $Fm|block|C_{\max}$ problem. It considers features of both DP (transition through states of given stage and dominance properties) and B&B (bounding scheme) to generate solutions and given some conditions, the method can actually be viewed as an exact method. The method is applied the Taillard benchmark (up to 500 jobs) and it actually improves some of the instances for the given problem.

(SHANG et al., 2018) apply a DP algorithm to the $F3||C_{\max}$ in order to generate a Pareto frontier whose composition are vectors of non-dominated solutions based on the completion times of permutations π and π' . By employing these dominance rules, the DP is bounded by an $O(3^n)$ complexity.

(OZOLINS, 2019b) improves the technique developed by (BAUTISTA et al., 2012) in order to fully guarantee the optimality of the BDP approach. The author devises three adapted lower bounds and uses a recursive procedure in order to consider as many solutions as possible without memory overreaching. Optimality is proved for 20 jobs and 10 machines in reasonable time and some cases with 20 machines.

(MOR; MOSHEIOV; SHAPIRA, 2019) investigate three proportionate flow shop

problems on two machines taking into account a threshold for the penalty incurred on rejected jobs and learning effect, aiming at minimizing $C_{\max}$, $\sum C_j$ and $\sum C_{\max}$ functions. Three DP algorithms are designed and, despite differing according to the objective function, their functioning mechanism is fairly similar since the main recursive equation determines which jobs are selected or rejected. Furthermore, the three problems are classified as NP-hard and the algorithms are bounded by an $O(n^2K)$ complexity. The authors also report their computational results for different learning factors and problems containing at most 50 jobs with efficient outcomes for medium-sized instances.

Table 12: Makespan and Completion Time related problems for flow shop

Authors	Problem Notation	Complexity	Method
(CORWIN; ESGBUE, 1974)	$F2 s_{ij} C_{\max}$	–	DP
(DUTTA; CUNNINGHAM, 1975)	$F2 b_{j,j+1} C_{\max}$	–	DP and Sub-optimal DP
(VELDE, 1990)	$F2 \sum C_j$	–	DP
(KOVALYOV; WERNER, 1997)	$F2 r_j C_{\max}$	$O(ni(2R)^i R^{2i-2})$	PTAS with DP
(LEE, 1997)	$F2 r - a(M_1) C_{\max}$	$O(nt_{11}(\sum p_{jk})^2 p_{\max 1})$	DP and Heuristic
(LEE, 1999)	$F2 sr - a(M_1) C_{\max}$	$O(nt_{11}(\sum p_{jk})^2 p_{\max 1})$	DP and Heuristic
(HOU; HOOGEVEEN, 2003)	$F3 p_{ik} = p_j/s_k C_{\max}$	$O(ns_1 s_2^2 s_3 \max\{p_j\}^2)$	DP
(BOUQUARD et al., 2005)	$F2 reg + nwt, pmru, n_{mut} = q C_{\max}$	$O(n(\tilde{M})^{3q+1})$	DP and FPTAS
(ALLAOUI et al., 2006)	$F2 r - a(M_1) C_{\max}$	$O(2^{n_1} n \log n)$	DP and Heuristics
(KUBZIN; STRUSEVICH, 2006)	$F2 h(\mu) C_{\max}$	$O(nM^4)$	DP and FPTAS
(BAUTISTA et al., 2012)	$Fm block C_{\max}$	–	BDP
(SHANG et al., 2018)	$F3 C_{\max}$	$O(3^n)$	DP
(OZOLINS, 2019b)	$Fm block C_{\max}$	–	BDP
(MOR; MOSHEIOV; SHAPIRA, 2019)	$F2 Le, \theta_j^{rej} \leq K C_{\max}$	$O(n^2 K)$	DP
	$F2 Le, \theta_j^{rej} \leq K \sum C_j$	$O(n^2 K)$	DP
	$F2 Le, \theta_j^{rej} \leq K \sum C_{\max}$	$O(n^2 K)$	DP

3.5.3 Multi-criteria and Additional Objective Functions

(LAWLER; MOORE, 1969) also take interest in applying their DP knapsack functional equation to problems considering two machines with the intent of minimizing a general loss time function. This application of the recursion takes into account a given common due date and the formulation is bounded by a $O(nd^2)$ complexity.

(T'KINDT; GUPTA; BILLAUT, 2003) furnish a variety of solution methods in order to solve the lexicographic bi-criteria problem denoted by $F2||Lex(C_{\max}, \sum C_j)$, which are divided into mathematical programming formulations, exact methods and heuristic. Among, those a DP formulation is created and its complexity is bounded by $O(UBn2^n)$. Although the method is mathematically structured, the computational requirements are high enough to make it the worst option among the others, being able to solve optimally only problems with up to 17 jobs.

(YEUNG; OĞUZ; CHENG, 2004) propose a B&B and a heuristic to solve the $F2|d_w|\sum(E_j + T_j)$ problem. The B&B structure is connected to an $O(n'(C(S) + p_{n/2} - s_{\min}))$ DP formulation via a bounding scheme and computational experiments are performed in order to verify efficiency. Although optimal solutions are found for 15-job instances in about 5 minutes, the heuristic is able to find near-optimal solutions for instances as ten times as large in 20 seconds.

(BŁAŻEWICZ et al., 2005) outline the proof for the NP-hardness associated to the $F2|d_j = d|\sum w_j Y_j$ problem by transforming the partition problem into their scheduling problem. They also develop a general DP formulation with $O(n^2 d^4)$ time complexity, which is able to prove the problem can actually be classified as binary NP-hard. Furthermore, a comparison among the DP, an enumeration method and list algorithms can be seen in (BLAZEWCZ et al., 2005) and the authors claim that, even though DP cannot be discarded because it was used for classifying the problem, it might not be the best method compared to the other ones in terms of efficiency.

(DAWANDE; GAVIRNENI; RACHAMADUGU, 2006) compile a variety of algorithms with the intent of maximizing a reward function under a makespan constraint for a two-machine scenario. Similarities between the knapsack DP functional approach seen in (LAWLER; MOORE, 1969) and the proposed DP algorithm are established, being the latter an $O(n^2 K^2 \log n)$ pseudo-polynomial time algorithm. Moreover comparisons are made among the DP, MIP and heuristics and due to its complexity, the DP algorithm is the one with the worst performance.

(T'KINDT; CROCE; BOUQUARD, 2007) address the problem of minimizing the number of tardy jobs and finding the best common due date in order to optimize it according to a Pareto-optimum curve. The authors design a ε -exact method and an ILP in order to verify their efficiency according to the experimental results. Furthermore,

the problem is defined as ordinary NP-hard and a pseudo-polynomial DP of $O(nd^2)$ complexity is also formulated by merging Johnson's rule with the method proposed by (JÓZEFOWSKA; JURISCH; KUBIAK, 1994).

(YEUNG; OĞUZ; CHENG, 2009) branch their research on the JIT framework and develop two DP algorithms to solve $F2|d_j = d|\sum(\alpha_j D_j + \beta_j U_j)$ and $F2|d_j = d|\sum(\alpha_j D_j + \beta_j U_j) + F(d)$ problems. First they prove both problems are NP-hard through a partition problem analogy and the optimality properties regarding the jobs positioning are presented. Both algorithms are pseudo-polynomial with $O(n^2(C(S))^4 K^2)$ and $O(n^2(C(S))^4 K^2 \sum p_{jk})$ complexities, respectively.

(LUO; CHEN; ZHANG, 2012) outline a theoretical approach to the $F2||C_{\max}^A + \theta C_{\max}^B$ and $F2||C_{\max}^A : C_{\max}^B \leq K$ problems, respectively. Approximation algorithms and a DP that is suitable for both situations are also presented and, due to the latter, the problems can be classified as weakly NP-hard. It might be valid to comment that the complexity of the DP approach relies on variables denoted here by q_j^A and q_j^B , which are valued according to allocation in analogy with the partition problem.

(SHABTAY; BENSOUSSAN, 2012) develop a serial analysis on shop problems for the earliness-tardiness minimization. One of the problems of interest is the $F2||\sum(\alpha_j E_j + \beta_j T_j)$. Its solution relies on an algorithm whose DP is mainly used for updating the recursive function and partitioning the jobs in groups of early and tardy based on a dominance rule and bounded by $O(n^2 \sum p_{j1})$ complexity. The authors also provide an FPTAS with the DP applied as a fundamental element for its functioning.

(FAN; CHENG, 2016) consider the $F2||C_{\max}^A + \alpha C_{\max}^B$ problem, taking into account a pseudo-polynomial DP algorithm that generates a partition of subsets depending on their start time to update the objective function, which is a four-dimension vector containing partial sequences and start times of both agents. The complexity of the problem is proven to be ordinary NP-hard and the algorithm's complexity is given by $O(n \sum p_{j1} (\sum p_{j1} + \sum p_{j2})^2)$. Ultimately an approximation algorithm based on Johnson's rule is also presented.

(SHANG et al., 2018) extend their approach to the case with general functions $f_{\max}$ and $\sum f_j$, which can display similar behaviour to the makespan criterion. For the first class of problems, the dominance rule adds the objective function as an element to discard partial sequences and the complexity is associated with an exponential form of total feasible solutions of the Pareto frontier ($O(4^{|S^*|})$). The second group of functions is solved analogously and is bounded by $O(5^n)$.

(KOVALEV et al., 2019) design a polynomial time DP algorithm of $O(n^3)$ complexity to minimize the total absolute deviation of completion times for a no-wait proportionate flow shop, also known as TADC, denoted by $F2|nwt, p_{jk} = p_j \sum |C_j - C_i|$. The DP is based on computing the objective function for the partial sequences and find an optimal

schedule based on a V -shaped structure and in case it cannot be found, the algorithm shifts into a heuristic. Experiments are performed for at most 300 jobs and the algorithm is able to find V -shaped schedules within few seconds on average.

Table 13: Multicriteria and Additional Functions problems for flow shop

Authors	Problem Notation	Complexity	Method
(LAWLER; MOORE, 1969)	$F2 d_j = d \sum F(t)$	$O(nd^2)$	DP
(T'KINDT; GUPTA; BILLAUT, 2003)	$F2 Lex(C_{\max}, \sum C_j)$	$O(nUB2^n)$	DP, MIP and Heuristic
(YEUNG; OĞUZ; CHENG, 2004)	$F2 d_w \sum (E_j + T_j)$	$O(n'(C(S) + p_{n/2} - st_{\min}))$	B&B with DP and Heuristic
(BŁAŻEWICZ et al., 2005)	$F2 d_j = d \sum w_j Y_j$	$O(n^2 d^4)$	DP
(BLAZEWICZ et al., 2005)	$F2 d_j = d \sum w_j Y_j$	$O(n^2 d^4)$	DP, Enumeration method and Heuristic
(DAWANDE; GAVIRNENI; RACHAMADUGU, 2006)	$F2 (\sum Rew_j, C_{\max} \leq K)$	$O(n^2 K^2 \log n)$	DP, MIP and Heuristics
(T'KINDT; CROCE; BOUQUARD, 2007)	$F2 d_j = d P(\sum U_j, d)$	$O(nd^2)$	DP and Enumeration Methods
(YEUNG; OĞUZ; CHENG, 2009)	$F2 d_j = d \sum (\alpha_j D_j + \beta_j U_j)$	$O(n^2 (C(S))^4 K^2)$	DP
(LUO; CHEN; ZHANG, 2012)	$F2 d_j = d \sum (\alpha_j D_j + \beta_j U_j) + F(d)$	$O(n^2 (C(S))^4 K^2 \sum p_{jk})$	DP
	$F2 C_{\max}^A + \theta C_{\max}^B$	$O((n_A + n_B)(\sum (p_j^A + q_j^A) + \sum (p_j^A + q_j^A))^4)$	DP and Approximation Algorithm
(SHABTAY; BENSOUSSAN, 2012)	$F2 C_{\max}^A : C_{\max}^B \leq K$	$O((n_A + n_B)(\sum (p_j^A + q_j^A) + \sum (p_j^A + q_j^A))^4)$	DP and FPTAS
(FAN; CHENG, 2016)	$F2 \sum (\alpha_j E_j + \beta_j T_j)$	$O(n^2 \sum p_{j1})$	DP and FPTAS
(SHANG et al., 2018)	$F2 C_{\max}^A : \alpha C_{\max}^B$	$O(n \sum p_{j1} (\sum p_{j1} + \sum p_{j2})^2)$	DP and Approximation algorithm
	$F3 f_{\max}$	$O(4^{ S'' })$	DP
	$F3 \sum f_j$	$O(5^n)$	DP
(KOVALEV et al., 2019)	$F2 nwt, p_{jk} = p_j \sum C_j - C_i $	$O(n^3)$	DP

3.6 Dynamic Programming Approach for the Job-Shop Environment

Job shop scheduling has also been an environment which has been a target for researchers due to the establishment of different routes on distinct machines. Some studies have been developed since the sixties but contribution has grown quite moderately only in the eighties. Hence, when comparing the use of dynamic programming to other methods, there might not be numerous articles tying this method with the environment. In this section we outline these papers and, due to a shortage in the number of references, all the criteria will be summarized in Table 15.

3.6.1 Classic due dates-related criteria

(JÓZEFOWSKA; JURISCH; KUBIAK, 1994) describe a DP formulation for the $J2|d_j = d|\sum w_j U_j$ in which the computation of the tardy jobs depends on a conditional formalism that groups the jobs according to their selection for being early or tardy. It is a backward recursive relation bounded by an $O(nd^3)$ complexity.

3.6.2 Makespan and Completion Time-related criteria

(ARTIGUES; FEILLET, 2008) investigate the $Jm|s_{ijk}|C_{\max}$ by incorporating varied concepts such as graph disjunction and TSP with time windows (TSPTW) and a heuristic to compose a B&B algorithm. The authors collaborate with a DP formulation to solve the TSPTW relaxations and they compare the effectiveness of such method to propagation methods that have also been provided. They develop experiments for small, medium and large sized problems and notice that it could perform well for small instances and suggest that such method can be improved for larger sets.

(BALAS; SIMONETTI; VAZACOPOULOS, 2008) also develop an adapted Bottleneck Heuristic for the problem $Jm|s_{ijk}, \bar{d}_j|C_{\max}$ by reformulating it as an asymmetric TSPTW problem in order to solve a general problem with fixed deadlines and as consequence, this problem, with addition of precedence constraints, can be solved via a DP subroutine combined with feasibility check properties in order to reduce the state space for accepted solutions. An overall comparison indicates the approach is promising even for harder instances, hence it might be useful in further studies.

(GROMICHO et al., 2012) furnish a DP algorithm to solve the $Jm||C_{\max}$ problem since no other exact method had been provided as solution for it. The paper focuses on deriving a dominance rule in order to supplement the algorithm with the adequate formulation via Bellman's principle and the authors, in order to optimize the DP due to the curse of dimensionality, consider optimality conditions that avert a full set of feasible solutions. Its complexity is bounded by $O(p_{\max}^{2n}(m+1)^n)$ complexity and empirical results show that the proposed algorithm performs exponentially better than brute force.

(OZOLINS, 2018a) proposes a BDP algorithm, which can be considered an improvement of the DP algorithm presented by (GROMICHO et al., 2012), however including sequence dependent setup times. A tight bound is provided and also the selection of feasible solutions is based on the solution for the TSPTW seen in (ARTIGUES; FEILLET, 2008). Computational results show that the BDP solves to optimality and even achieve better solutions than the previous best known solutions in several benchmarks with at most 20 jobs and 10 machines.

(OZOLINS, 2018b) conducts a robust study on the $Jm|nwt|C_{\max}$ problem. Two remarks must be made concerning the DP structure of the solution method. The first one is that the DP algorithm presents a dominance rule that is part of the dynamic programming graph-like definition. Additionally, a recursive formalism is also presented as well as a prominent bound. The results are proven to achieve optimality for moderate size of jobs and machines (10 x 10) from several benchmarks in seconds.

3.6.3 Multi-criteria and Additional Objective Functions

(WANG et al., 1997) examine the $Jm||\sum(\alpha_j E_j^2 + \beta_j T_j^2)$ problem considering that some machines can be grouped into a set H_{ij} according to their capabilities and a given horizon L . The authors design a Lagrangian relaxation method to solve the problem and it is split into sub-problems that are relative to each part of the objective function. A backward DP formulation is proposed to furnish a solution to the weighted tardiness and yields an $O(L \sum |H_{ij}|)$.

(LUH et al., 1998) also extend the complexity of the problem analyzed in (WANG et al., 1997) with the inclusion of limited buffer, family dependent setup times and a long horizon. Their solution also relies on decomposing the problem and the relaxation occurs on the family constraints and machine capacity. In addition, a backward DP formulation is presented to solve the decomposed part of the Lagrangian relaxation and its concept can be extended to the constraints without further adjustments. Since similarities can be shared with Wang's DP formalism, the complexity remains $O(L \sum |H_{ij}|)$.

(BLAZEWICZ et al., 2007) design a DP approach to find the optimal schedule for the $J2|d_j = d|\sum w_j Y_j$ problem in which the recursive equation is based on the positioning of jobs throughout the sequence depending on separation of subgroups of early and tardy jobs. The overall complexity is given by $O(n^3 d^{11})$ and the problem is classified as binary NP-hard due to this construction of optimal solutions based on DP.

(BAPTISTE; FLAMINI; SOURD, 2008) bounds are based on precedence constraints relaxation and resource constraint relaxation for the $Jm||\sum(\alpha_j E_j + \beta_j T_j)$. The second type, which relies on a time-indexed formulation of the problem with release dates on a single machine, can be solved by a DP approach and when experiments are performed, it can present advantage over the precedence type even for large sets of machines. The

complexity for this relaxation is given by $O(n_i(\max\{d_j + \sum p_{jk}\}))$, where n_i is a subset of J considering jobs performing a given number of operations.

Table 14: Problems in the job-shop environment

Authors	Problem Notation	Complexity	Method
(JÓZEFOWSKA; JURISCH; KUBIAK, 1994)	$J2 d_j = d \sum w_j U_j$	$O(nd^3)$	DP
(WANG et al., 1997)	$Jm \sum (\alpha_j E_j^2 + \beta_j T_j^2)$	$O(L \sum H_{i_j})$	Lagrangian Relaxation with DP
(LUH et al., 1998)	$Jm SD_f, b_{j,j+1} \sum (\alpha_j E_j^2 + \beta_j T_j^2)$	$O(L \sum H_{i_j})$	Lagrangian Relaxation with DP
(BLAZEWICZ et al., 2007)	$J2 d_j = d \sum w_j Y_j$	$O(n^3 d^{11})$	DP
(ARTIGUES; FEILLET, 2008)	$Jm s_{i,jk} C_{\max}$	–	B&B with DP
(BALAS; SIMONETTI; VAZACOPOULOS, 2008)	$Jm s_{i,jk}, \bar{d}_j C_{\max}$	–	Heuristic with DP
(BAPTISTE; FLAMINI; SOURD, 2008)	$Jm \sum (\alpha_j E_j + \beta_j T_j)$	$O(n_i(\max\{d_j + \sum p_{jk}\}))$	Lagrangian relaxation with DP
(GROMICHO et al., 2012)	$Jm C_{\max}$	$O(p_{\max}^{2n} (m+1)^n)$	DP
(OZOLINS, 2018a)	$Jm s_{i,jk} C_{\max}$	–	BDP
(OZOLINS, 2018b)	$Jm nwt C_{\max}$	–	DP

3.7 Dynamic Programming approach for the Open Shop Environment

The randomness of the open shop environment has also turned this problem one of the most interesting settings in machine scheduling, since its behaviour can be considered an ordeal. Due to this fact, the use of exact methods in solving such method has been scarce or just theoretically conceived. Nevertheless, when verifying the application of dynamic programming in this scenario one can notice that an unexpected amount of papers concerning this topic might appear, which is surprising considering the storage issues readily associated with the DP algorithms. These articles have been organized as follows and, as in the previous sections, metaheuristics have not been included nor have been articles involving the dynamic and flexible scenarios.

3.7.1 Classic due dates-related criteria

(BRUCKER; JURISCH; JURISCH, 1993) address problems in which unit processing times are considered in the open shop environment. First several problems are proved to be solved under the assumption that some of the open shop problems can be transformed into identical parallel machine problems. Then, further results are presented, in which an $O(n^2)$ DP formulation is developed to construct an optimal schedule of all early jobs that can be assembled in block structures to solve the $O2|p_{jk} = 1|\sum w_j U_j$.

(BRUCKER et al., 1993) investigate the $Om|p_{jk} = 1|\sum w_j U_j$ problem by incorporating the findings regarding the open shop being transformed in parallel machine seen in (BRUCKER; JURISCH; JURISCH, 1993). This needs to be done in order to apply the DP formulation, which is of $O(n^2 m^{m+1})$ complexity, for the construction of an optimal scheduling according to the sequencing of early jobs in optimal manner and tardy jobs arbitrarily.

3.7.2 Makespan and Completion Times-related criteria

(STRUSEVICH; HALL, 1997) consider the minimization of makespan setting one of the machines as non-bottleneck (NB). The problem is denoted by $O2|NB|C_{\max}$ and the authors prove it to be ordinary NP-hard by developing a forward DP formulation that computes the makespan at each stage, which may either keep the value of the previous stage or calculate the completion time of a block of jobs on the second machine. Moreover, its complexity is given by $O(n \sum p_{j2})$ and a FPTAS is also derived from it.

(TAUTENHAHN; WOEGINGER, 1997) study the $Om|p_{jk} = 1, r_j|\sum C_j$ by designing an $O(n^2)$ DP algorithm to solve it. First, some optimality properties regarding the ordering of jobs is presented according to a increasing monotonic behaviour of completion times and a recursive relation is introduced with real and bounded values for cases where the sequence follows the optimality conditions and it is infinite, otherwise.

([SHAFRANSKY; STRUSEVICH, 1998](#)) analyze the problem of minimizing the makespan by imposing that the jobs have to be scheduled according a given sequence on the last machine. According to Graham's notation, the problem is denoted by $O2|GS(1)|C_{\max}$. Cases of preemption are considered, however, the DP algorithm is developed for the non-preemptive case. The algorithm allows partitions and deletion of some of these partitions a_λ that might correspond to partial infeasible sequences. The algorithm's complexity is given by $O(n \sum p_{j1} a_\lambda)$ and a FPTAS is provided considering a DP subroutine based on the previous formulation.

([GUÉRET; PRINS, 1999](#)) develop a lower bound for the problem $Om||C_{\max}$ in which one of the steps that consist the B&B algorithm is to calculate values that work as lower and upper bounds for the completion times. By doing that, the authors create an analogy with the subset-sum problem and apply a DP algorithm, which solves it quickly, and therefore, the DP is viewed as a subroutine.

([LI, 2011](#)) considers a non-bottleneck machine under jobs deterioration regime. The author delineates optimality properties based on preliminary sources for obtaining an optimal schedule and apply those in order to devise a DP algorithm that is unfolded into two formulations depending on the route for scheduling a given job. Furthermore, the algorithm can be transformed in an FPTAS by reusing some optimality concepts.

([ZHANG; BAI, 2014](#)) construct a lower bound by using Lagrangian relaxation for the $Om||\sum C_j^2$ problem. The problem is actually decomposed in two parts, being the first one relative to sequencing the operations and the second one to calculating the Lagrangian multipliers (referred to as assignment costs) via a forward DP formulation. The authors prove that the DP complexity is estimated as $O(m \max\{m, n\} p_{\max})$ and it is appropriate for small-sized instances.

([OZOLINS, 2019a](#)) apply a DP algorithm by relying on dominance rules and a lower bound to find optimal solutions for the $Om||C_{\max}$ problem. These features are incorporated in order to reduce the state space compared to an enumeration method, which considers all possible schedules. In addition, the algorithm has its performance tested for problems with up to 7 jobs and 7 machines and, despite showing good results only for small-sized instances, the dominance rule can be useful in further research and it can assist on reducing a variety of solutions that can be disregarded.

3.7.3 Multi-criteria and Additional Objective Functions

([BŁAŻEWICZ et al., 2004](#)) present some problems characterized by minimization of late work criterion and variations, among which a DP formulation is provided as solution for the $O2|d_j = d|\sum w_j Y_j$ problem. The algorithm is able to calculate the minimum total weighted late work based on parts of jobs that are processed earlier according to a common due date. Its complexity is bounded by $O(nd^2 \min\{p_{\max}, d\})$ and due to the

pseudo-polynomial nature of the algorithm, the problem is proven to be binary NP-hard.

(ZHANG; LU; YUAN, 2016) investigate the $O2|rej|C_{\max} + \sum \theta_j^{rej}$ problem by studying its complexity and proposing algorithms for solving it. Firstly, the problem is actually proved to be NP-hard even for unit processing times and equal rejection penalties. Then a DP algorithm is presented and its computation of the makespan depends on a recursive formulation that separates rejected from accepted jobs. Therefore, the problem and its aforementioned special cases can be classified as ordinary NP-hard due to this algorithm. Also, the complexity of the algorithm is estimated as $O(n^2(\sum p_{j1})(\sum p_{j2}))$ and, thereafter the approximation scheme is provided.

(JIANG et al., 2018) develop a scheme similar to the one presented in (FAN; CHENG, 2016) so as to provide solutions for the $O2||C_{\max}^A + \alpha C_{\max}^B$ problem. A DP algorithm and an approximation algorithm are devised and the first one relies on optimal properties that include cases for idle and no-idle machines in order to apply the recursive relation and also a longest alternate processing times (LAPT) rule, producing an overall $O(n(\sum p_{j1} + \sum p_{j2})^3)$ complexity. Moreover, since a DP pseudo-polynomial algorithm can be constructed for such problem, it is categorized as ordinary NP-hard.

Table 15: Problems in the open-shop environment

Authors	Problem Notation	Complexity	Method
(BRUCKER; JURISCH; JURISCH, 1993)	$O2 p_{jk} = 1 \sum w_j U_j$	$O(n^2)$	DP
(BRUCKER et al., 1993)	$O2 p_{jk} = 1 \sum w_j U_j$	$O(n^2 m^{m+1})$	DP
(STRUSEVICH; HALL, 1997)	$O2 NB C_{\max}$	$O(n \sum p_{j2})$	DP and FPTAS
(TAUTENHAHN; WOEGINGER, 1997)	$Om p_{jk} = 1, r_j \sum C_j$	$O(n^2)$	DP
(SHAFRANSKY; STRUSEVICH, 1998)	$O2 GS(1) C_{\max}$	$O(n \sum p_{j1} a_\lambda)$	DP and FPTAS
(GUÉRET; PRINS, 1999)	$Om C_{\max}$	–	B&B with DP
(BŁAŻEWICZ et al., 2004)	$O2 d_j = d \sum w_j Y_j$	$O(nd^2 \min\{p_{\max}, d\})$	DP
(LI, 2011)	$O2 NB, p_{jk}(t) C_{\max}$	–	DP
(ZHANG; BAI, 2014)	$Om \sum C_j^2$	$O(m \max\{m, n\} p_{\max})$	Lagrangian Relaxation with DP
(ZHANG; LU; YUAN, 2016)	$O2 rej C_{\max} + \sum \theta_j^{rej}$	$O(n^2 (\sum p_{j1})(\sum p_{j2}))$	DP and Approximation algorithm
(JIANG et al., 2018)	$O2 C_{\max}^A + \alpha C_{\max}^B$	$O(n(\sum p_{j1} + \sum p_{j2})^3)$	DP and Approximation algorithm
(OZOLINS, 2019a)	$Om C_{\max}$	–	DP

4 BOUNDED DYNAMIC PROGRAMMING APPLIED TO THE BLOCKING FLOW SHOP PROBLEM WITH SEQUENCE DEPENDENT SETUP TIMES

Blocking flow shop scheduling (BFSP) is a problem that often arises in several segments of manufacturing. Applications regarding its occurrence have been reported in iron and steel industries (see(GONG; TANG; DUIN, 2010)), industrial waste, metallic parts manufacturing (see(MARTINEZ et al., 2006)), chemical and pharmaceutical industries (see(MERCHAN; MARAVELIAS, 2016)) and robotic cells (see(RIBAS; COMPANYS; TORT-MARTORELL, 2015)), among others. Blocking is a technological constraint imposed usually on flow shops whenever the system does not present an intermediate buffer. Therefore, if a subsequent machine is yet not available, the current job remains blocked on the current machine until it can be transferred to the next.

Another technological constraint that commonly emerges in the industrial context is the inclusion of activities in which setup times are added to the process execution. These times are relevant because they represent the preparation for receiving a given job and, since it may be a significant amount in the manufacturing environment, they might as well be added. (ALLAHVERDI et al., 2008) collected over 300 articles regarding the subject (also including setup costs) and demonstrated that, throughout the years, not only have setup times drawn the attention of many researchers but also it has helped develop knowledge for a whole new set of problems that are often recurrent in industry (e.g. job family setup times).

Nevertheless, some of the combinations of blocking and setup times are frequently neglected in research, even though they might have substantial participation in realistic manufacturing systems. For instance, in a recent review paper, (MIYATA; NAGANO, 2019) showed that, among 140 papers regarding the blocking flow shop, only six of them would reckon setup times in single objective functions. Hence, one can notice that just few methods have been designed to solve this class of problems, which might also be associated with the complexity of this problem.

The interest in this work is to present an alternative solution for the $Fm|s_{ijk}, block|C_{\max}$ (BFSP-SDST) problem. It can be evidenced that some solutions for such problem have been relying on metaheuristics (e.g.(SHAO; PI; SHAO, 2018)), which have produced outstanding results for large-scale jobs for Taillard benchmark, and constructive heuristics in (TAKANO; NAGANO, 2019), which showed that a profile fitting heuristic adapted to the setup times, whether combined with NEH heuristic or not, was the most efficient one compared to other 13 heuristics. Apart from that fact, a MILP and a B&B have been proposed by (TAKANO; NAGANO, 2017) as exact methods and instances with up to 20 jobs and a combination of machines that would consider 10 machines.

From the previous chapter, it can be stated that some authors have described dynamic programming methods to solve problems in a variety of environments as well as problems where setup times were among the main technological constraints within the scheduling problem configuration. Combining that information with the fact that no DP-based algorithm has been explored for this specific formulation, a Bounded Dynamic Programming algorithm has been designed for this purpose. Hence, this chapter is dedicated to the fundamentals of the problem and the algorithmic structure of the proposed BDP.

4.1 Problem Statement and Notations

Let $j \in J$ be a given job to be processed on a given machine $k \in M$ in a serial fashion, such that the sequencing of jobs must be exactly the same in every machine. In addition, let p_{jk} denote the time taken by job j to be processed on machine k and consider a setup time s_{ijk} between the preceding job $i \in J$ and j to indicate that a preparation must take place before job j starts its processing. For this case, the setup times are classified as anticipatory, i.e. a given machine k is allowed to be prepared while machine $k - 1$ is still processing job j . Furthermore, whether a given job i is yet to finish its processing on machine $k + 1$ or setup is being carried out and job j has already completed its processing on machine k , the latter must be held until machine $k + 1$ is ready to begin its processing. C_{jk} is the completion time, which is defined as the point in time job j finishes its processing on machine k , and D_{jk} represents the departure time, i.e. the point in time job j leaves machine k . Lastly, let $C_{\max}$ denote the maximum completion time for the last job n on the last machine k , i.e. makespan.

The problem described above is defined as $Fm|s_{ijk}, block|C_{\max}$ and aims at finding the optimal sequence for n jobs that minimizes $C_{\max}$. It can be modelled as follows:

$$\min C_{\max} = C_{[n]m} \quad (4.1)$$

$$D_{[1]k} = \max\{s_{[0][1]k+1}, D_{[1]k-1} + p_{[1]k}\}; \quad k = 2, \dots, m - 1 \quad (4.2)$$

$$D_{j1} = \max\{D_{i2} + s_{ij2}, D_{i1} + s_{ij1} + p_{j1}\}; \quad j = 1, \dots, n \quad (4.3)$$

$$D_{jk} = \max\{D_{i(k+1)} + s_{ij(k+1)}, D_{j(k-1)} + p_{jk}\}; \quad i \neq j, \forall k \neq 1, m \quad (4.4)$$

$$D_{jm} = D_{j(m-1)} + p_{jm}; \quad \forall j \in J. \quad (4.5)$$

Eq. 4.2 refers to the allocation of the first job ($[l]$ stands for the job occupying the l -th position) for every machine and Eq.4.3 plays the same role, however by varying jobs

rather than machines. Note that, since i precedes j , when j is set as the first job, i would be the zeroth job, and because it does not exist, $D_{i1} = 0$. Eq. 4.4 is the general recursive relation to obtain the departure times on every machine except for the last one, which, in turn, is calculated via Eq. 4.5.

The next section introduces the idea of a Bounded Dynamic Programming applied to a version of the problem of interest without considering sequence dependent setup times. Understanding it is crucial since the fundamentals for the BFSP-SDST are similar to the one proposed for the BFSP.

4.2 Bounded Dynamic Programming for the $Fm|block|C_{\max}$ problem

Although several methods have been designed for solving a variety of problems in machine scheduling, one may notice that the amount of heuristics and metaheuristics that researchers resort to in the present days is, by far, greater than the ones that produced by exact methods. Clearly, the advantage of heuristics and metaheuristics relies on the fact that these methods are able to obtain good quality solutions in a very small CPU time and storage requirements are usually shorter than the ones observed in enumeration methods, for instance. However, these methods might also be tentative, since their functioning might depend on other heuristics or metaheuristics that, in turn, might not result in solutions as satisfactory as one could hope. Furthermore, another factor that is also relevant is that improvements on exact formulations usually generate optimality properties that might help describe a whole set of problems and classify them by providing exact-based algorithms, pseudo-polynomial and approximation algorithms, thus enlarging the set of useful mechanisms to solve problems of a more realistic framework.

In the previous chapter, some of the solutions methods that have been discussed were, rather than pure DP formulations, branches of it. These methods might be considered hybrid or just relaxations that resulted in applications of DP at some point. Among those, Bounded Dynamic Programming (BDP) has shown significant and efficient outcomes for the blocking flow shop problem.

According to (BAUTISTA; PEREIRA, 2009), BDP is an exact algorithm that combines features of DP and B&B that, under certain circumstances, can be employed as a heuristic. One of the advantages of such method is that lower bounds and either heuristic or exact dominance rules are embedded in its structure so as to reduce the state space intrinsic to the problem, thus allowing results that would possibly not be achieved by pure DP in an acceptable time or computational requirements. It is worth mentioning that DP-based approaches are defined as a graph-like structure $G(L, T)$, where L represents the states generated once a given vertex begins branching and T is referred to the decision making process of transforming one state into another, which is denominated transition. Consequently, the problem is solved once there are no further transitions to take place and

the graph constitutes a path. Hence, it is equivalent to solving a shortest path problem.

BDP generally consists of four elements as follows:

- Generation of state space: This step is characterized by an initial stage $l = 0$, which represents an empty set and a subsequent stage $l = 1$ containing n states, each relative to the assignment of a given task. Computationally, these stages are created as two consecutive lists, one for each stage specifically. These lists are currently updated as the first list is emptied and the second is filled with viable policies;
- Dominance rule: Artifice in which two states that belong to the same stage are compared and the one that presents the largest-valued policy for a partial sequence is dominated, thus being removed from the current stage;
- Bounding scheme: Lower bounds are introduced and calculated for each state. Whenever $LB \geq UB$ the state is eliminated from the current stage;
- Window width: This parameter, which is denoted by H , is used as a threshold for the number of states generated at a given stage and therefore, does not allow memory overreach due to the curse of dimensionality of a pure DP approach.

In order to give a detailed explanation on how these parts are assembled, we discuss the algorithm proposed by (BAUTISTA et al., 2012) for the $Fm|block|C_{\max}$. Each list contains a vertex denoted by $X(j, l)$, which is equivalent to a given state at stage l . In addition, a vertex is composed of three elements, such that $X(j, l) = [\vec{q}(j, l); \vec{D}(j, l); C_{\max}]$, where $\vec{q}(j, l)$ is defined as a binary vector for the assignment of job j , $\vec{D}(j, l)$ denotes the vector of departure times and $C_{\max}$ corresponds to the makespan of the a partial sequence at stage l . Note that $\vec{q}(j, l)$ is an actual representation of a partial sequence and therefore

$$\sum_{p=1}^n q_p(j, l) = l \quad (4.6)$$

The reason to define these components are that they are strictly related to the dominance rule established for reducing the state space. Consider j and j' jobs to be allocated at a given stage l . Then

$$X(j, l) \prec X(j', l) \iff [\vec{q}(j, l) = \vec{q}(j', l)] \wedge [\vec{D}(j, l) < \vec{D}(j', l)] \quad (4.7)$$

and

$$X(j, l) \equiv X(j', l) \iff [\vec{q}(j, l) = \vec{q}(j', l)] \wedge [\vec{D}(j, l) = \vec{D}(j', l)]. \quad (4.8)$$

Eq.4.7 refers to a situation where vertex $X(j, l)$ dominates $X(j', l)$ since they contain the same jobs allocated in the partial sequence, however, the former yields a

Figure 1: Schematics of dominance rule and transitional state in the graph structure

smaller vector of completion times. Similarly Eq. 4.8 stands for an equivalence of vertices since the vector of completion times is strictly equal.

By applying these relations, a smaller state space is already being built, since when comparing vertices the most promising one will undergo transition to the next stage while the other will be removed, thus avoiding a complete enumeration of all sequences (see Fig. 1).

Although this rather "simple" configuration, in fact, diminishes the state space involved and delivers a shrunk graph structure, for practical purpose, this might still not be as efficient as it is expected from an improved method. Therefore, the authors also devise a job-based lower bound and a machine-based lower bound. Since the latter attains better solutions, it is the one we focus on and is formulated as follows:

$$LB_k(j, l) = D_k(j, l) + \sum_{i \in \bar{S}} p_{ik} + \min_{i \in \bar{S}} \left\{ \sum_{h=k+1}^m p_{ih} \right\} \quad (4.9)$$

and, consequently, the lower bound for the vertex, is given by:

$$LB(j, l) = \max_{k \in M} \{LB_k(j, l)\}. \quad (4.10)$$

Feasible policies in BDP are commonly bounded by an upper bound and a lower bound. The latter is frequently obtained by some auxiliary method such as heuristics in the form of an initial solution. If the $LB \geq UB$ for a given vertex, then it can be disregarded, otherwise the algorithm will allow solutions whose values are worse than the initial one.

Before considering the window width parameter, one must be aware that in order to soothe the computational burden of seeking vertices randomly for comparison when using the dominance rule, it is recommended to sort the vertices. This sorting is carried out taking into account a priority rule vector $(LB(j, l); C_{\max})$, which can follow two variants. The first one sorts the vertices in an increasing order of $LB(j, l)$ and, in case ties occur, the $C_{\max}$ is selected to determine the sorting. The second variant operates in an analogous manner by shifting the $C_{\max}$ and $LB(j, l)$.

Ultimately, the window width works as a regulator for the scheduler. The bigger its size, the larger the number of solutions that can be admitted in the state space, i.e. the number of feasible solutions at a given stage l is bounded by H . Although it is a very effective measure in terms of computational effort, the drawback of utilizing is that it turns the exact approach into a heuristic. It can be explained by the fact that a intermediate vertex of the graph, one which provides the shortest path, may be discarded, as shown in Fig. 2. In addition, the authors identify the exact behaviour of the BDP if the following two conditions are satisfied

$$\max_{1 \leq l \leq L} \{H(l)\} < H \quad (4.11)$$

$$(\max_{1 \leq l \leq L} \{H(l)\} = H) \wedge (D_m(j, L) < LBZ_{\min}) \quad (4.12)$$

where $LBZ_{\min}$ represents the lowest LB among all the vertices that have been disregarded throughout the process, otherwise the procedure is a heuristic. Note that, since only two lists are allowed for storage, the first list should be empty when all the jobs in it have been branched and the feasible policies remain stored in the second list. Thus, in order to reuse them, the vertices in the second list are transferred to the first one, stage $l + 1$ is updated to become stage l and, once again the process of branching starts in order to fill the second list, which is now empty.

Once the the algorithm's profile has been outlined for this specific problem, the pseudo-code will be presented only in the next section, however a final remark should be made about BDP. One can notice due to the similarities to the B&B and also Beam Search methods, some misconceptions might occur regarding the functioning of BDP, so a simple description on the differences might be needed. First, when comparing the BDP and B&B (with Breadth First Search algorithm, for example), the BDP relies on a graph rather than a tree to perform the search of the optimal shortest path, which in fact, makes it more simple. Also, the B&B passes through every possible state in the tree, whereas this is not required in the BDP. When compared to the Beam search method, despite the fact that either methods contain a window width in its formulation, BDP makes mandatory use of bounds, while the Beam Search does not require them. Additionally, modern approaches of Beam Search have been including some randomness in their formulations, whereas BDP,

Figure 2: Representation of BDP and its perspective as a heuristic procedure considering the criteria for space state reduction.

as of date, is purely deterministic. Moreover, Beam Search also relies on a tree-based structure.

4.3 Bounded Dynamic Programming extended for the $Fm|s_{ijk}, block|C_{max}$ problem

This section is not supposed to be extensive because the major portion of the algorithm's functioning has been detailed in the previous section. Hereafter, the adaptation will be carried out taking into account two factors: slight modifications in the BDP algorithm proposed by (BAUTISTA et al., 2012) and a major one, which is providing a lower bound that performs in an advantageous manner, i.e. finding the closest vector to $\vec{v} = (0, 0)^T$ in the Pareto frontier analysis regarding solution quality and computational effort.

As stated in (TOMAZELLA; NAGANO, 2020) recent review on B&B, problems regarding the permutation flow shop scheduling with sequence dependent setup times are frequently solved via heuristics and metaheuristics, once these methods obtain high quality solutions within a small time frame. In case of a B&B approach for the same problem, the tightness of the bound is a crucial factor in determining the performance of the algorithm and it becomes even more costly when setup times are included because they are reckoned

the most complex part of the LB formulation. Therefore, either improvements on lower bounds formulations are often required or their employment might be more adequate in other methods.

(TAKANO; NAGANO, 2017) proposed a B&B method by designing four lower bounds for solving the $Fm|s_{ijk}, block|C_{\max}$ problem. Compared to bounds previously proposed for the $Fm|s_{ijk}|C_{\max}$ and $Fm|block|C_{\max}$ separately, these bounds can be gauged a less complex structure and not being a tight bound, despite calculating solutions more quickly, may generally demand more time to achieve high quality solutions. Nevertheless, the authors present an upper bound formulation for the idle times and a lower bound for the blocking, which are relatively simple to be computed and given respectively by

$$\delta_j^{(k)} = \max\{0, (\delta_j^{(k-1)} + s_{ij(k-1)} + p_{j(k-1)}) - (s_{ijk} + p_{ik})\} \quad \forall i \neq j \quad (4.13)$$

with

$$\delta_j^{(1)} = 0 \quad (4.14)$$

and

$$LBB_j^{(k)} = \max\{0, (s_{ij(k+1)} + p_{i(k+1)}) - (\delta_j^k + s_{ijk} + p_{jk})\} \quad \forall i \neq j \quad (4.15)$$

with

$$LBB_j^{(m)} = 0. \quad (4.16)$$

Unless the sequence to which the makespan obtained is previously known, the idle times and the the blocking time cannot be calculated due to the fact the next scheduled job is a combinatorial problem. Therefore, due to some equivalences, it is possible to obtain the equations above regarding bounded values. Additionally, their initial conditions are overt because there exists no blocking on the last machine and no idle time on the first machine.

When comparing the structure of the LB proposed by (BAUTISTA et al., 2012) and the one designed by (TAKANO; NAGANO, 2017), it can be seen that they differ by including terms relative to the setup and the lower bound for blocking. Also, since the former yielded such promising results in the analysis for the $Fm|block|C_{\max}$ using BDP, it seems fitting to apply it to the variant where setup times are included. Additionally, once the efficiency of BDP is directly connected to the strength of the lower bound, selecting a bound that presents similarities with one that has already been successful for the BFSP may induce a better adherence of such LB to the BDP method by including alterations in its structure that are relative to the setup times. Some of the modifications depend on two other variables, which are the sum of minimum setup times and sum of the minimum

lower bounds for blocking and they are defined as follows:

$$SMST_k(j, l) = \sum_{\substack{u \in (\bar{S} \oplus \{j\}) \\ v \in \bar{S} \\ u \neq v}} (\min(s_{uvk})) - \max_{\substack{u \in \bar{S} \\ v \in \bar{S} \\ u \neq v}} (\min(s_{uvk})) \quad (4.17)$$

$$SMLBB_k(j, l) = \sum_{u \in (\bar{S} \oplus \{j\})} (\min(LBB_u^{(k)})) - \max_{u \in \bar{S}} (\min(LBB_u^{(k)})) \quad (4.18)$$

where S stands for the set of scheduled jobs and $\bar{S}$ the set of unscheduled jobs. These sums are necessary because they are related to the possibilities of allocating the unscheduled jobs and they have to be minimum in order not to create values that are often greater than a given upper bound. An additional remark that should be made about the computation in Eq. 4.18 is that the second term refers only to unscheduled jobs. Therefore, when coupling the jobs in Eq. 4.15 and Eq. 4.13, one must be aware that both i and j for that set belongs to $\bar{S}$. In addition, the first equation represents the estimation of the sum of the setup times for the remaining jobs and the second equation indicates the sum of the lower bounds related to the blocking, which is also an estimation of blocking that can occur along the sequencing.

Finally, the lower bound is given by

$$\begin{aligned} LB_k(j, l) = & D_k(j, l) + SMST_k(j, l) + \sum_{u \in \bar{S}} (p_{uk}) + SMLBB_k(j, l) \\ & + \min_{u \in \bar{S}} \left(\sum_{h=k+1}^m (p_{uh}) + \min \left(\sum_{h=k+1}^m (LBB_u^{(h)}) \right) \right), \quad \forall k \neq m. \end{aligned} \quad (4.19)$$

and

$$LB_m(j, l) = D_m(j, l) + SMST_m(j, l) + \sum_{u \in \bar{S}} (p_{um}). \quad (4.20)$$

This LB as well as the others that are not part of the scope of this research have been adapted from the one proposed by (RONCONI, 2005) for the $Fm|block|C_{\max}$. The author based the LB construction on a graph representation of the problem by finding the path that minimizes the distance between the first and last vertices. In addition, one can also perceive similarities between this LB and the one seen in the previous section, which ensures that the latter, under adaptations such as the one presented in this section, is appropriate.

Finally we introduce the pseudo-code for the BDP approach for the $Fm|s_{ijk}, block|C_{\max}$ below. Note that in order to consider the lower bound as a reduction method for the state space of the algorithm, an upper bound must be also calculated. The aforementioned

sources that applied BDP to a given problem suggest either the use of heuristics such as Greedy procedures/local search or BDP with a small window width.

Algoritmo 1: BDP algorithm for the $Fm|s_{ijk}, block|C_{\max}$

Input: $n, m, H, p_{jk}, s_{ijk}, UB$
Output: $C_{\max}, List(L)$ of optimal sequences

```

1 Initialization:  $l = 0, LBZ_{\min} = \infty, S = \emptyset$ 
2 while  $l < n$  do
3    $l \leftarrow (l + 1)$ 
4   if  $l = 1$  then
5     forall  $j \in \bar{S}$  do
6       if  $LB(j, l) \geq UB$  then
7         delete  $X(j, l)$ 
8          $S \leftarrow S$ 
9       else
10         $S \leftarrow (S \oplus \{j\})$ 
11        if  $|List(l)| > H$  then
12          insert  $X(j, l)$  in  $List(l)$ 
13          sort  $List(l)$  according to  $(LB; C_{\max})$  priority
14          remove last vertex in  $List(l)$ 
15        else
16          insert  $X(j, l)$  in  $List(l)$ 
17          sort  $List(l)$  according to  $(LB; C_{\max})$  priority
18   else
19     while  $|List(l)| \neq 0$  do
20       select the first vertex  $X(i, l)$  in  $List(l)$ 
21       forall  $j \in \bar{S}$  do
22         if  $LB(j, l + 1) \geq UB$  then
23           delete  $X(j, l + 1)$ 
24            $S \leftarrow S$ 
25         else
26           forall  $X(j', l + 1)$  do
27             if  $X(j', l + 1) \prec X(j, l + 1)$  then
28               delete  $X(j, l + 1)$ 
29             else
30                $S \leftarrow (S \oplus \{j\})$ 
31               if  $|List(l + 1)| > H$  then
32                 insert  $X(j, l + 1)$  in  $List(l + 1)$ 
33                 sort  $List(l + 1)$  according to  $(LB; C_{\max})$  priority
34                 remove last vertex in  $List(l + 1)$ 
35               else
36                 insert  $X(j, l + 1)$  in  $List(l + 1)$ 
37                 sort  $List(l + 1)$  according to  $(LB; C_{\max})$  priority
38       remove  $X(i, l)$  from  $List(l)$  and make next vertex first vertex
39     while  $|List(l + 1)| \neq 0$  do
40        $List(l) \leftarrow List(l + 1)$ 

```

Note that this algorithm is composed of two main parts. The first one is related to the initial allocation of jobs. First, a null set is programmed to be the first list. When the first jobs are branched into vertices, the second list is created to store them. For them

the fathoming can occur only via the the upper bound analysis or the window width parameter. The list is updated according to these criteria and becomes the first list by emptying the second one. For the remaining levels, the process is similar, however the comparison among the permutations with the same allocated jobs is considered so as to reduce the state space.

5 METHODOLOGY

This chapter is essentially motivated by the methods applied to conduct the research thoroughly and in order to delineate them, it is divided in two subsections. The first section considers the methods employed to carry out the computational tests, i.e. the computer's configuration, algorithms used for comparison and efficiency measures for the next chapter. The second one introduces the statistical tools that will be used in order to evaluate the efficiency of the algorithm.

5.1 Computational Experiments

The computational analysis is divided in two sets of experiments and each set will be characterized by a job set, a machine set, a time limit, the distribution of processing times and the distribution set of setup times. Also, for sake of organization, two subsections are created in order to introduce methods used for comparison as well as to establish a proper configuration of each set. Additionally, before introducing the description of each set, it is relevant to mention that the BDP and B&B methods have been coded in C language in a PC Intel(R) Core(TM) i5-2500K CPU 3.30 GHz with 4 GB RAM under a 64 bits Windows 7 operating system.

5.1.1 First set of problems

The first set of problems is based on the one carried out in (TAKANO; NAGANO, 2017) by considering 540 problems and subdividing them into 27 classes of 20 instances each, where a given class is defined by a combination of machines being assigned to a given number of jobs. For the first 18 classes, the job set is defined as $n = \{10, 12, 14\}$ and the machine set is defined by $m = \{2, 3, 4, 5, 7, 10\}$. For the remaining classes, the combination of is given by $n = \{16, 18, 20\}$ and $m = \{2, 3, 4\}$. In addition, each instance will be terminated when the time limit of 3600 seconds is reached.

Regarding the processing times the authors use the data provided by (RONCONI, 2005) considering an uniform distribution for values in a $[1, 100]$ interval. As for the setup times, the interval also is uniformly distributed in a $[1, 100]$ interval. Therefore, in this first analysis, the database will be the same used by these authors.

The BDP algorithm will be a replicated experiment by varying only the value of the window width, whose set is defined as $H = \{1, 10, 50, 100, 250, 500, 750, 1000, 1250, 2500, 5000, 10000, 20000\}$, thus each replication will be treated as an independent method. In addition, for a given H , the value of the upper bound UB for the current problem, is given by the value of $C_{\max}$ obtained for the same instance in the preceding window width. For $H = 1$, the UB value

is set to infinity, given that no preceding window width exists.

The comparisons for this experimental set rely on the results obtained by programming the BDP and comparing its results with a MILP procedure and a B&B method, both provided by (TAKANO; NAGANO, 2017). Since the functioning of the MILP, which has been coded in Python by using Python 3.6.5 interface and run by the IBM(R) ILOG CPLEX 12.8 solver for this research, has been carefully described, its mathematical formalism will be presented, however the same will not be done for the B&B since its main structure is the LB that has been previously shown in chapter 3.

Let $st_{[t]k}$ be the time a given job in position t starts its processing one machine k . Furthermore, let $z_{[t]ij}$ and $x_{[t]j}$ be binary variables that represent adjacency between job i and job j when the latter is inserted in position t and the assignment of job j in position t , respectively. Then, the MILP formulation is given by

$$\min C[n]m \quad (5.1)$$

$$\sum_{t=1}^n x_{[t]j} = 1; \quad j = 1, \dots, n \quad (5.2)$$

$$\sum_{j=1}^n x_{[t]j} = 1; \quad t = 1, \dots, n \quad (5.3)$$

$$\sum_{i=1}^n \sum_{i \neq j} z_{[0]ij} = 0 \quad (5.4)$$

$$\sum_{i=1}^n \sum_{i \neq j} z_{[t]ij} = 0; \quad t = 2, \dots, n \quad (5.5)$$

$$z_{[t]ij} \geq x_{[t-1]i} + x_{[t]j} - 1; \quad i = 1, \dots, n; j = 1, \dots, n; t = 2, \dots, n; i \neq j \quad (5.6)$$

$$st_{[1]1} = \sum_{j=1}^n x_{[1]j} s_{[0]j1} \quad (5.7)$$

$$st_{[1]k} \geq \sum_{j=1}^n x_{[1]j} s_{[0]jk}; \quad k = 2, \dots, m \quad (5.8)$$

$$st_{[1]k} \geq C_{[1](k-1)}; \quad k = 2, \dots, m \quad (5.9)$$

$$st_{[t]k} \geq st_{[t-1](k+1)} + \sum_{i=1}^n \sum_{i \neq j} z_{[t]ij} s_{ijk}; \quad k = 2, \dots, m-1; t = 2, \dots, n \quad (5.10)$$

$$st_{[t]k} \geq C_{[t]k}; \quad k = 2, \dots, m-1; t = 2, \dots, n \quad (5.11)$$

$$C_{[t]k} = st_{[t]k} + \sum_{j=1}^n x_{[t]j} p_{jk}; \quad k = 1, \dots, m; t = 1, \dots, n \quad (5.12)$$

$$x_{[t]j}, z_{[t]ij} \in \{0, 1\}; \quad i, j = 1, \dots, n; t = 1, \dots, n; i \neq j \quad (5.13)$$

$$C_{[t]k}, st_{[t]k} \geq 0; \quad k = 1, \dots, m; t = 1, \dots, n. \quad (5.14)$$

Eq. 5.1 denotes the minimization of makespan. Eq. 5.2 and Eq. 5.3 guarantee that each job is only assigned to a single position and that the opposite also holds. Eq. 5.4 ensures that the binary variable relative to the setup times is zero, since no $j \in J$ must occupy the zeroth position. Eq. 5.5 shows that the setup times binary variable is only activated when the job i in the $(t - 1) - th$ position precedes job j in the $t - th$ position. Eq. 5.6 may be valued as -1, 0 or 1, if neither job i nor job j are assigned to the $t - 1$ and t positions, if only one of the jobs is assigned to the correct position or both of them are correctly assigned to their positions, respectively. Eq. 5.7 calculates the start time of the first job on the first machine. Eq. 5.8 and Eq. 5.9 make sure that the start time of setup operations are executed after blocking or completion time of a given job. Eq. 5.10 and Eq. 5.11 play the same role for every job, except the first one assigned. Eq. 5.12 computes the completion times of every job on every machine. Finally, Eq. 5.13 and Eq. 5.14 describe the domain of the decision variables.

5.1.2 Second set of problems

This set of problems contains 2880 problems with a job set denoted by $n = \{4, 5, 6, 7, 8, 10, 12, 15, 20\}$ and a machine set defined by $m = \{2, 3, 4, 5, 7, 10, 15, 20\}$. Therefore there are 72 classes containing 40 instances each, which are internally divided into four classes of 10 instances each, according to the setup time classification that will be explained briefly ahead. Also, to maintain the equality of comparison to other methods, the time limit here for every experiment will be set to 3600s.

The processing times for each problem are randomly generated according to an uniform distribution considering the interval for the data to be $[1, 100]$. Regarding the setup times, four classes have been defined and they are denoted as $ST_{(25)}$, $ST_{(50)}$, $ST_{(100)}$ and $ST_{(125)}$. The number within the parentheses represents the maximum value assumed by the setup times data at a given interval, which means that the first class considers setup times in $[1, 25]$, the second one in $[1, 50]$, the third one in $[1, 100]$ and the last one in $[1, 125]$.

The algorithms compared will be the BDP and the MILP with a slight modification from the BDP applied in the previous experimental set. Once this development has been designed for a larger set of machines, the window width will be evaluated with a maximum

value of 10000. Besides that consideration, the premises exhibited in the previous subsection still hold.

5.2 Statistical Analysis

This section is a direct ramification from the previous one because the measurements related to the performance of the proposed model and the ones used for comparison are only verified through statistical evaluation and the three main datasets that can be extracted from the computational analysis are the makespan obtained for each algorithm, the CPU time and the number of nodes generated. Although this data is furnished, it is necessary to transform it into viable information. Therefore, two variables that will be constantly used in our analysis will be the relative percentage deviation (RPD) and the average relative percentage time (ARPT).

The RPD computes the percentage deviation of the solution for a given problem when compared with the best result found among the methods for this same problem. Let ρ denote a given problem, β be a given method among all the B methods used for comparison and $C_{\max}^*$ be the best makespan among all the methods, i.e. $C_{\max}^*(\rho) = \min_{1 \leq \beta \leq B} \{C_{\max}(\rho, \beta)\}$. Then

$$RPD(\rho, \beta) = 100 \left(\frac{C_{\max}(\rho, \beta) - C_{\max}^*(\rho)}{C_{\max}^*(\rho)} \right). \quad (5.15)$$

The RPD measures the solution quality of the method proposed compared to others. Therefore, the closer to zero, the better the solution found by a given method, since there exists a small difference between the makespan for a method and the best one reported.

The ARPT functioning can be similarly described, however some additional calculations need to be performed in order to reach the final value. Let $T(\rho, \beta)$ be the CPU time for a given problem ρ using a given method β . Then

$$AT(\rho) = \frac{\sum_{\beta=1}^B T(\rho, \beta)}{B} \quad (5.16)$$

which can be successfully applied to find the RPT given by

$$RPT(\rho, \beta) = \frac{T(\rho, \beta) - AT(\rho)}{AT(\rho)} \quad (5.17)$$

and yield the ARPT defined by

$$ARPT(\beta) = \frac{\sum_{\rho=1}^R RPT(\rho, \beta)}{R}. \quad (5.18)$$

These two deviation measures are relevant for the two types of experimental sets since they will be used for the most part of the analysis. A third measurement that can be

used is success rate of the algorithm, which estimates the the frequency attributed to a given method in finding the best solution among all the methods. Thus it can be defined as

$$SR(\beta) = \frac{\sum_{\rho=1}^R G_{\rho,\beta}}{R} \quad (5.19)$$

where $G_{\rho,\beta}$ is a binary variable that is set to 1 if the method has been able to achieve the best solution and 0, otherwise.

After performing these calculations, the next step consists in verifying which test is appropriate for the analysis. The objective of this analysis is to determine if statistical differences between a certain indicator can be perceived when comparing different methods. For this research, Analysis of Variance (ANOVA) and Kruskal-Wallis test have been considered. The first one demands some conditions such as normality condition for the data, homoscedasticity and independent samples in order to be applied. In case one of these conditions fails, the analysis will rely on the Kruskal-Wallis test, which is non-parametric. In addition, for the latter it is also relevant to show a pair-wise comparison test in order to show which methods can present a statistical difference. The design of these tests are based on two hypotheses, which are commonly known as H_0 (null hypothesis) and H_1 . The null hypothesis often states that whatever the factor used for comparison is, there exists no statistical difference between the samples and the contrary is stated by H_1 . In order to verify these hypotheses, the p-value provided by the chosen test is provided and in case p-value is less or equal a significance level, denoted by α , the null hypothesis is rejected and the factors differ, otherwise the H_1 hypothesis is rejected and the factors do not present differences statistically. For the context of this research, the null hypothesis for Kruskal-Wallis test states that when comparing algorithms in a pairwise manner, they present no statistical difference.

To compose the statistical analysis, other features will also be considered by displaying graphic behaviour of the *ARPD* measure varying according to the number of jobs, number of machines, ARPT and setup times configurations. Moreover, in order to enrich the study, a boxplot investigation will be presented so as to fully comprehend the *ARPD*'s variability through the perspicive of the algorithms designed in this research.

6 DISCUSSION AND RESULTS

This chapter is divided in two sections that are relative to the experimental sets previously outlined. First, a discussion is carried out considering the analysis of the set containing 540 problems and afterwards a similar investigation is executed for the set with 2880 problems.

6.1 First experimental set

MILP, B&B, and 13 variants of BDP considering window widths H varying from 1 to 20000 have been applied to the set of instances for this first experimental group and due to the authors' signature the BDP methods are denoted by BDP-SN(H) for a general notation. Additionally, the data that are handled for this first analysis are kept as a single group regardless the number of jobs, machines, setup configuration and similar attributes, being only categorized according to a given method.

Table 16: General data for the first experimental set

Method	SR (%)	ARPD (%)	ARPT (s)
BDP-SN(1)	0.740741	10.749	-0.99998
BDP-SN(10)	6.111111	3.58	-0.99983
BDP-SN(50)	16.85185	1.6083	-0.99934
BDP-SN(100)	23.7037	1.1139	-0.99876
BDP-SN(250)	36.85185	0.7033	-0.99712
BDP-SN(500)	43.88889	0.5218	-0.99473
BDP-SN(750)	50.18519	0.4062	-0.9924
BDP-SN(1000)	53.33333	0.3542	-0.99004
BDP-SN(1250)	55.74074	0.3099	-0.98764
BDP-SN(2500)	65	0.2231	-0.97267
BDP-SN(5000)	72.59259	0.1514	-0.92404
BDP-SN(10000)	80	0.1136	-0.766
BDP-SN(20000)	87.03704	0.0822	-0.17259
B&B	58.14815	1.1434	5.121619
MILP	46.2963	1.174	6.673523

As it can be seen in Table 16, the descriptive analysis shows that as the success rate of BDP-SN (H) improves, the values of ARPD decrease and the ones relative to ARPT tend to increase. This statement is obvious in the sense that as window widths become larger, more vertices are allowed in the state space. Therefore, this increases the

probability of finding the graph that contains the shortest path and, consequently, the optimal solution. However, this process also generates a computational burden, mostly for sorting and dominance conditions, causing an expansion regarding the ARPT. Thus, the best results in terms of SR and ARPD are provided by BDP-SN(20000), whereas its ARPT can be considered the worst among the BDP-SN(H). The opposite can also be stated for BDP-SN(1). Furthermore, the values relative to the B&B and the MILP are highlighted. The descriptive comparison shows that these methods may present some statistical difference regarding their solution quality to some of the bounded dynamic programming variations and clearly displayed far higher CPU times, which indicates that some of the BDP algorithms may produce a better set of solutions with less computational effort. In fact, if one were to compare, for instance, BDP-SN(2500), the success rate is superior when tested against both methods and so are the ARPD and ARPT. Additionally, when recording the CPU times of individual samples, for example, the largest portion of complex instances, such as $n = 20$ and $m = 4$, would reach optimum in about 40s while the MILP and B&B would reach time limit without obtaining the best makespan.

Although these results seem promising, for sake of completeness regarding the statements previously made, some analysis might be needed to corroborate them. The data of individual samples have been subjected to Anderson-Darling normality test and Bonferroni homoscedasticity test so as decide whether ANOVA would be an appropriate test when comparing different methods. The first test has null hypothesis for normality of data and the second one constructs a null hypothesis for equal variances. In Table 17 one can notice that the second column provides the p-values for the first test. For the majority of methods, p-value has confirmed that the data is not normally distributed, once the $p\text{-value} < 0.05$ (significance level). Furthermore, the same pattern can be observed for the second test, indicating the conditions for the ANOVA do not hold and therefore, Kruskal-Wallis test is selected to carry out the analysis.

Table 17: Verification for ANOVA

Method	P-value (Normality Test)	P-value (Bonferroni Test)
BDP-SN(1)	0.196	0.000
BDP-SN(10)	<0.005	0.000
BDP-SN(50)	<0.005	0.000
BDP-SN(100)	<0.005	0.000
BDP-SN(250)	<0.005	0.000
BDP-SN(500)	<0.005	0.000
BDP-SN(750)	<0.005	0.000
BDP-SN(1000)	<0.005	0.000
BDP-SN(1250)	<0.005	0.000
BDP-SN(2500)	<0.005	0.000

Table 17 continued from previous page

Method	P-value (Normality Test)	P-value (Bonferroni Test)
BDP-SN(5000)	<0.005	0.000
BDP-SN(10000)	<0.005	0.000
BDP-SN(20000)	<0.005	0.000
B&B	<0.005	0.000
MILP	<0.005	0.000

According to Kruskal-Wallis test performed for RPD, one can conclude that at least one of the median differ significantly, since the overall p-value for groups is less than 0.05. In this case a *post hoc* test, commonly known as Dunn's Test is applied so as to establish pairwise comparisons between groups and define those that, in fact, differ statistically. It is also valid to highlight that, due to corrections inserted in the Dunn's test formulation, the value of the significance level α might be altered to fulfill the test's purpose. For this data set, Minitab yields $\alpha = 0.002$ and those that present significant difference are shown in Table 18.

Table 18: Pairwise comparison between methods according to Dunn's Test

Method	P-value
BDP-SN(1) vs. BDP-SN(10)	0.0000
BDP-SN(1) vs. BDP-SN(50)	0.0000
BDP-SN(1) vs. BDP-SN(100)	0.0000
BDP-SN(1) vs. BDP-SN(250)	0.0000
BDP-SN(1) vs. BDP-SN(500)	0.0000
BDP-SN(1) vs. BDP-SN(750)	0.0000
BDP-SN(1) vs. BDP-SN(1000)	0.0000
BDP-SN(1) vs. BDP-SN(1250)	0.0000
BDP-SN(1) vs. BDP-SN(2500)	0.0000
BDP-SN(1) vs. BDP-SN(5000)	0.0000
BDP-SN(1) vs. BDP-SN(10000)	0.0000
BDP-SN(1) vs. BDP-SN(20000)	0.0000
BDP-SN(1) vs. B&B	0.0000
BDP-SN(10) vs. BDP-SN(50)	0.0000
BDP-SN(10) vs. BDP-SN(100)	0.0000
BDP-SN(10) vs. BDP-SN(250)	0.0000
BDP-SN(10) vs. BDP-SN(500)	0.0000
BDP-SN(10) vs. BDP-SN(750)	0.0000
BDP-SN(10) vs. BDP-SN(1000)	0.0000
BDP-SN(10) vs. BDP-SN(1250)	0.0000

Table 18 continued from previous page

Method	P-value
BDP-SN(10) vs. BDP-SN(2500)	0.0000
BDP-SN(10) vs. BDP-SN(5000)	0.0000
BDP-SN(10) vs. BDP-SN(10000)	0.0000
BDP-SN(10) vs. BDP-SN(20000)	0.0000
BDP-SN(10) vs. B&B	0.0000
BDP-SN(50) vs. BDP-SN(100)	0.0000
BDP-SN(50) vs. BDP-SN(250)	0.0000
BDP-SN(50) vs. BDP-SN(500)	0.0000
BDP-SN(50) vs. BDP-SN(750)	0.0000
BDP-SN(50) vs. BDP-SN(1000)	0.0000
BDP-SN(50) vs. BDP-SN(1250)	0.0000
BDP-SN(50) vs. BDP-SN(2500)	0.0000
BDP-SN(50) vs. BDP-SN(5000)	0.0000
BDP-SN(50) vs. BDP-SN(10000)	0.0000
BDP-SN(50) vs. BDP-SN(20000)	0.0000
BDP-SN(50) vs. B&B	0.0000
BDP-SN(100) vs. BDP-SN(250)	0.0000
BDP-SN(100) vs. BDP-SN(500)	0.0000
BDP-SN(100) vs. BDP-SN(750)	0.0000
BDP-SN(100) vs. BDP-SN(1000)	0.0000
BDP-SN(100) vs. BDP-SN(1250)	0.0000
BDP-SN(100) vs. BDP-SN(2500)	0.0000
BDP-SN(100) vs. BDP-SN(5000)	0.0000
BDP-SN(100) vs. BDP-SN(10000)	0.0000
BDP-SN(100) vs. BDP-SN(20000)	0.0000
BDP-SN(100) vs. B&B	0.0000
BDP-SN(250) vs. BDP-SN(750)	0.0000
BDP-SN(250) vs. BDP-SN(1000)	0.0000
BDP-SN(250) vs. BDP-SN(1250)	0.0000
BDP-SN(250) vs. BDP-SN(2500)	0.0000
BDP-SN(250) vs. BDP-SN(5000)	0.0000
BDP-SN(250) vs. BDP-SN(10000)	0.0000
BDP-SN(250) vs. BDP-SN(20000)	0.0000
BDP-SN(500) vs. BDP-SN(1250)	0.0001
BDP-SN(500) vs. BDP-SN(2500)	0.0000
BDP-SN(500) vs. BDP-SN(5000)	0.0000
BDP-SN(500) vs. BDP-SN(10000)	0.0000

Table 18 continued from previous page

Method	P-value
BDP-SN(500) vs. BDP-SN(20000)	0.0000
BDP-SN(750) vs. BDP-SN(2500)	0.0000
BDP-SN(750) vs. BDP-SN(5000)	0.0000
BDP-SN(750) vs. BDP-SN(10000)	0.0000
BDP-SN(750) vs. BDP-SN(20000)	0.0000
BDP-SN(1000) vs. BDP-SN(2500)	0.0015
BDP-SN(1000) vs. BDP-SN(5000)	0.0000
BDP-SN(1000) vs. BDP-SN(10000)	0.0000
BDP-SN(1000) vs. BDP-SN(20000)	0.0000
BDP-SN(1000) vs. B&B	0.0008
BDP-SN(1250) vs. BDP-SN(5000)	0.0000
BDP-SN(1250) vs. BDP-SN(10000)	0.0000
BDP-SN(1250) vs. BDP-SN(20000)	0.0000
BDP-SN(1250) vs. B&B	0.0000
BDP-SN(2500) vs. BDP-SN(10000)	0.0005
BDP-SN(2500) vs. BDP-SN(20000)	0.0000
BDP-SN(2500) vs. B&B	0.0000
BDP-SN(5000) vs. B&B	0.0000
BDP-SN(10000) vs. B&B	0.0000
BDP-SN(20000) vs. B&B	0.0000
MILP vs. BDP-SN(1)	0.0000
MILP vs. BDP-SN(10)	0.0000
MILP vs. BDP-SN(50)	0.0000
MILP vs. BDP-SN(500)	0.0000
MILP vs. BDP-SN(750)	0.0000
MILP vs. BDP-SN(1000)	0.0000
MILP vs. BDP-SN(1250)	0.0000
MILP vs. BDP-SN(2500)	0.0000
MILP vs. BDP-SN(5000)	0.0000
MILP vs. BDP-SN(10000)	0.0000
MILP vs. BDP-SN(20000)	0.0000
MILP vs. B&B	0.0000

One can notice that, out of the 105 possible combinations, 89 presented significant difference according to Dunn's test. Note that BDP-SN(1) and BDP-SN(10) clearly show these differences when tested against the other methods and this fact might be expected since their ARPD values are more scattered throughout the range of data and the medians

Figure 3: Boxplot relative to the RPD data

drastically differ from the others. Furthermore, comparisons developed with the MILP method statistically differ from almost every BDP-SN algorithm and a similar behaviour can be observed for the B&B. Regarding the ARPD measure, both methods are positioned between the BDP-SN algorithms, which means that there might be methods with better and worse ARPD. Methods with worse ARPD tend to exhibit data with higher variability, while those with better ARPD show reduced variability and, depending on how scattered these data are, even with the equal medians, the test acknowledges the existing difference between the paired methods. These claims are also endorsed by the boxplot analysis in Fig. 3.

The boxplot is shown in order to present statements about variability and disposition of data that concerns the RPD. It can be seen that variability reduces as window widths' size increases, once the makespan values of each sample are either improved or kept at the same level of the previous window width. Furthermore, it is perceived that when a comparison with MILP is furnished, BDP-SN(50) produces a RPD range that is significantly shorter, despite a much lower success rate, indicating that even a narrow window width, BDP approach is able to provide smaller gaps based on the best solution among all methods. Similarly, the comparison between medium sized window width BDP and B&B shows that the former presents data range with overall smaller gaps. It can also be recognized that

the concentration of outliers seems more prominent for wider window widths and this can be explained by the reduction in the boxplot for these methods. Once the scheduler allows larger window widths, the number of optimal solutions in the data rises, causing more RPD data to be equal zero and it alters the calculations for the quartile interval, which sets the limits for the boxes and the outliers. When this interval reaches zero, which can be observed for BDP-SN(10000) and BDP-SN(20000), the optimal makespan is ensured for at least 75% of the data associated to a given method.

The analysis of solution quality and average CPU time is imperative for this experimental set once that it helps the scheduler in the decision making process to optimize its policies based on a particular reasonable time frame. According to the aforementioned remarks, BDP-SN(1) and BDP-SN(10) will be neglected for this analysis, once they show discrepancies when contrasted with their counterparts. Results are depicted in Fig. 4 with the best trade-off being defined by the point closest to the leftmost bottom corner of the scatter plot, which is represented by BDP-SN(5000). According to the graphic description, there are points relatively close to BDP-SN(5000) that could also be adequate options to provide a reasonably good trade-off (e.g. BDP-SN(2500) and BDP-SN(10000)) however, the farther the points are from the best trade-off in terms of ARPD, the worse the success rate of a certain method. Analogously, the farther the points are in terms of ARPT, the higher the computational effort. In addition, one can tell that MILP and B&B are among the worst performances, since their location in the plot are not anywhere near those provided by BDP and despite their capacity to find optimal solutions, their average CPU times are significantly higher in comparison to some of the BDP approaches.

In order to complete the investigation for this model, it is also interesting to cover the behaviour of solution quality considering the number of jobs, depicted in Fig. 5. It can be seen that this partitioned analysis follows the behaviour of the general considerations outlined thus far. The Initially, MILP and B&B can be accounted for the best values for ARPD but for $n > 12$, an inversion occurs as ARPD for MILP and B&B increase at a rather elevated rate compared to the other algorithms. Furthermore, BDP-SN algorithms ranging from $H = 50$ to $H = 1250$ also show an increase in their growth rates, however they are far subtler than those obtained by B&B and MILP. Moreover, a decreasing rate can be observed in the BDP-SN(5000), BDP-SN(10000) and BDP-SN(20000) when $n > 16$, which may indicate that those are also more adequate for a larger set of jobs.

A similar analysis can be addressed in terms of number of machines but rather than investigating the a single set of machines, the data was split in two parts because the first group is composed of 120 samples for each machine while the second presents 60 samples. In Fig. 6, it is observed MILP is the method with the highest variation in this group, although a slight reduction is seen from $m = 3$ to $m = 4$. Also, B&B rapidly increases its ARPD value as the number of machines is expanded. The BDP-SN algorithms

Figure 4: Performance analysis with regard to ARPD and ARPT

tend to maintain the behaviour of differing slightly from one another in an increasing rate of ARPD for the small and medium sized window widths and a decreasing rate for larger window widths. The second group of machines is depicted in Fig. 7 and one can notice that for this set MILP and B&B perform better in terms of solution quality, however still varying at higher values than the majority of BDP-SN and once more, the BDP-SN with large window widths presents a slight decrease in the ARPD as the number of machines increases.

6.2 Second experimental set

This second set contains data from 13 methods, which can be divided into 12 variants of the BDP-SN(H) and a MILP procedure. For the former, only window widths that range from $H = 1$ to $H = 10000$ have been considered. BDP-SN(20000) has been excluded from the analysis due to the fact that an expressive amount of samples in the last group of jobs extrapolated the time limit established for the experiment.

The first analysis furnished for this set categorizes the general information according to the success rate, ARPD and ARPT of each method and it is summarized on Table 19. Note that, as expected, the larger the window width, the better the success rate related to an extension of BDP algorithm. Additionally, one can notice that, even though the success

Figure 5: ARPD variation with respect to the number of jobs

rates presented by BDP-SN(1) and BDP-SN(10) are inferior compared to the other ones, a remarkable improvement can be observed when comparing to the first experimental set and this might indicate that the algorithm is susceptible to fast refinement of solutions, once the difference between their ARPTs is fairly small. It is also noticeable that solutions are upgraded for the medium-sized window widths, however it occurs at a smaller step between two consecutive BDP-SN methods. Regarding the MILP method, its results have produced a far superior success rate as well as ARPD than those presented by the first set of experiments, which is also seen in the other methods when compared to their counterparts in the first experimental set. This may also be explained due to the fact that the number of samples have grown considerably and therefore, so has the likelihood of problems that could achieve the optimal makespan. Furthermore, some intermediate BDP-SN algorithms such as BDP-SN(1000) and BDP-SN(1250), as it shall be shown later, might be comparable to the MILP performance, once a trade-off between solution quality and CPU time is developed. BDP-SN(2500), BDP-SN(5000) and BDP-SN(10000) have presented the best performances in terms of success rate and consequently, ARPD. It is also valid to point out that these methods have also been able to yield these results under a much smaller CPU time than the one imposed as a stopping criterion for the algorithms, while MILP has reached time limit for a variety of problems and has not been able to obtain the optimum for 615 of them.

Table 19: General data for the second experimental set

Method	SR (%)	ARPD (%)	ARPT (s)
BDP-SN(1)	8.715278	8.21940	-0.9929
BDP-SN(10)	37.84722	2.26220	-0.9767
BDP-SN(50)	54.61806	1.01650	-0.9601
BDP-SN(100)	60.34722	0.71440	-0.952298
BDP-SN(250)	66.07639	0.47730	-0.938084
BDP-SN(500)	70.9375	0.33860	-0.942152
BDP-SN(750)	73.88889	0.28010	-0.936395
BDP-SN(1000)	75.38194	0.24010	-0.933759
BDP-SN(1250)	76.28472	0.21590	-0.930479
BDP-SN(2500)	80.90278	0.14473	-0.910878
BDP-SN(5000)	86.04167	0.09276	-0.845928
BDP-SN(10000)	94.0625	0.05031	-0.643232
MILP	78.64583	0.58730	10.9629

Once again, it is paramount to show which algorithms differ statistically and prior to that, determine whether an ANOVA is adequate to define a comparison among the algorithms. In this case, it can be verified in Table 20 that the data does not follow normal distribution (p-value < 0.05 for Anderson-Darling Test) and the homoscedasticity cannot be confirmed (p-value < 0.05 for Bonferroni Test) and therefore, ANOVA is not compatible with the analysis. Thus, a Kruskal-Wallis test is performed with a *post hoc* investigation in order to establish an appropriate discussion and, as it has been previously stated for the first experimental set, the test is composed of a series of pairwise comparisons that rely on a different value of α , due to some corrections that need to be performed to validate the analysis. Hence, according to the calculations of Minitab, the value of α is set to 0.003 and therefore, the p-value is analyzed with respect to this value.

Table 20: General data for the first experimental set

Method	P-value (Normality Test)	P-value (Bonferroni Test)
BDP-SN(1)	<0.005	0.000
BDP-SN(10)	<0.005	0.000
BDP-SN(50)	<0.005	0.000
BDP-SN(100)	<0.005	0.000
BDP-SN(250)	<0.005	0.000
BDP-SN(500)	<0.005	0.000
BDP-SN(750)	<0.005	0.000
BDP-SN(1000)	<0.005	0.000
BDP-SN(1250)	<0.005	0.000

Table 20 continued from previous page

Method	P-value (Normality Test)	P-value (Bonferroni Test)
BDP-SN(2500)	<0.005	0.000
BDP-SN(5000)	<0.005	0.000
BDP-SN(10000)	<0.005	0.000
MILP	<0.005	0.000

The pairwise comparison shown in Table 21 is composed of 70 samples, out of 78, that present statistical difference regarding the RPD measure. Although the comparison among the several BDP-SN methods is displayed, the interest here is to contrast them with MILP, once it has a different nature from the others. One can notice that, as expected, MILP can be distinguished from small and large-sized window widths BDP-SN methods. For the small ones, one can clearly state that the high variability generates values for the median that are greater than zero and RPD data is more scattered than those belonging to MILP. For the large window widths, the median are indeed equal to the that yielded by MILP, however the data is much more compressed and this factor is also accounted by the test, hence demonstrating that a statistical difference occurs. Conversely, no difference can be accounted when referring to MILP and medium-sized window widths BDP-SN algorithms and this could be explained by the fact that the slight rate of change between two consecutive BDP-SN methods may not be perceived drastically by Dunn's Test. This subtle change is actually an indication that only few improvements have taken place when switching to the next window width (for some of the medium-sized window widths) and these improvements might be closer to that obtained by MILP.

Table 21: Pairwise comparison between methods according to Dunn's Test for the second set

Method	P-value
BDP-SN(1) vs. BDP-SN(10)	0.0000
BDP-SN(1) vs. BDP-SN(50)	0.0000
BDP-SN(1) vs. BDP-SN(100)	0.0000
BDP-SN(1) vs. BDP-SN(250)	0.0000
BDP-SN(1) vs. BDP-SN(500)	0.0000
BDP-SN(1) vs. BDP-SN(750)	0.0000
BDP-SN(1) vs. BDP-SN(1000)	0.0000
BDP-SN(1) vs. BDP-SN(1250)	0.0000
BDP-SN(1) vs. BDP-SN(2500)	0.0000
BDP-SN(1) vs. BDP-SN(5000)	0.0000
BDP-SN(1) vs. BDP-SN(10000)	0.0000
BDP-SN(10) vs. BDP-SN(50)	0.0000

Table 21 continued from previous page

Method	P-value
BDP-SN(10) vs. BDP-SN(100)	0.0000
BDP-SN(10) vs. BDP-SN(250)	0.0000
BDP-SN(10) vs. BDP-SN(500)	0.0000
BDP-SN(10) vs. BDP-SN(750)	0.0000
BDP-SN(10) vs. BDP-SN(1000)	0.0000
BDP-SN(10) vs. BDP-SN(1250)	0.0000
BDP-SN(10) vs. BDP-SN(2500)	0.0000
BDP-SN(10) vs. BDP-SN(5000)	0.0000
BDP-SN(10) vs. BDP-SN(10000)	0.0000
BDP-SN(50) vs. BDP-SN(100)	0.0000
BDP-SN(50) vs. BDP-SN(250)	0.0000
BDP-SN(50) vs. BDP-SN(500)	0.0000
BDP-SN(50) vs. BDP-SN(750)	0.0000
BDP-SN(50) vs. BDP-SN(1000)	0.0000
BDP-SN(50) vs. BDP-SN(1250)	0.0000
BDP-SN(50) vs. BDP-SN(2500)	0.0000
BDP-SN(50) vs. BDP-SN(5000)	0.0000
BDP-SN(50) vs. BDP-SN(10000)	0.0000
BDP-SN(100) vs. BDP-SN(250)	0.0000
BDP-SN(100) vs. BDP-SN(500)	0.0000
BDP-SN(100) vs. BDP-SN(750)	0.0000
BDP-SN(100) vs. BDP-SN(1000)	0.0000
BDP-SN(100) vs. BDP-SN(1250)	0.0000
BDP-SN(100) vs. BDP-SN(2500)	0.0000
BDP-SN(100) vs. BDP-SN(5000)	0.0000
BDP-SN(100) vs. BDP-SN(10000)	0.0000
BDP-SN(250) vs. BDP-SN(500)	0.0000
BDP-SN(250) vs. BDP-SN(750)	0.0000
BDP-SN(250) vs. BDP-SN(1000)	0.0000
BDP-SN(250) vs. BDP-SN(1250)	0.0000
BDP-SN(250) vs. BDP-SN(2500)	0.0000
BDP-SN(250) vs. BDP-SN(5000)	0.0000
BDP-SN(250) vs. BDP-SN(10000)	0.0000
BDP-SN(500) vs. BDP-SN(1000)	0.0003
BDP-SN(500) vs. BDP-SN(1250)	0.0000
BDP-SN(500) vs. BDP-SN(2500)	0.0000
BDP-SN(500) vs. BDP-SN(5000)	0.0000

Table 21 continued from previous page

Method	P-value
BDP-SN(500) vs. BDP-SN(10000)	0.0000
BDP-SN(750) vs. BDP-SN(2500)	0.0000
BDP-SN(750) vs. BDP-SN(5000)	0.0000
BDP-SN(750) vs. BDP-SN(10000)	0.0000
BDP-SN(1000) vs. BDP-SN(2500)	0.0000
BDP-SN(1000) vs. BDP-SN(5000)	0.0000
BDP-SN(1000) vs. BDP-SN(10000)	0.0000
BDP-SN(1250) vs. BDP-SN(2500)	0.0005
BDP-SN(1250) vs. BDP-SN(5000)	0.0000
BDP-SN(1250) vs. BDP-SN(10000)	0.0000
BDP-SN(2500) vs. BDP-SN(5000)	0.0004
BDP-SN(2500) vs. BDP-SN(10000)	0.0005
BDP-SN(5000) vs BDP-SN(10000)	0.0000
MILP vs. BDP-SN(1)	0.0000
MILP vs. BDP-SN(10)	0.0000
MILP vs. BDP-SN(50)	0.0000
MILP vs. BDP-SN(100)	0.0000
MILP vs. BDP-SN(250)	0.0000
MILP vs. BDP-SN(2500)	0.0000
MILP vs. BDP-SN(5000)	0.0000
MILP vs. BDP-SN(10000)	0.0000

The analysis made via Kruskal-Wallis test can actually be corroborated by the Boxplot depicted in Fig. 8. It can be noticed that BDP-SN(1) and BDP-SN(10) are the algorithms that show the highest variability among all methods and consequently, deliver the worst results. This variability reduces drastically when $H = 50$ as a large portion of solutions are rapidly enhanced or reach the optimum. For $H \geq 1000$, the interquartile difference is zero, which is also the same for the MILP, however for the former one can notice that the number of outliers tend to diminish as H becomes larger and this phenomenon specifies the increase in the number of problems whose optimal solutions have been found. Furthermore, the outliers for the MILP method vary on a larger scale than those observed for BDP-SN whose $H \geq 1000$.

For this set of experiments it is also relevant to establish a Pareto trade-off analysis in order to investigate which of the methods that have been developed for this problem may be the best choice for the scheduler. It is relevant to mention that the points relative to BDP-SN(1) and BDP-SN(10) have been neglected since their values show discrepancies when compared to the other algorithms. Fig. 9 represents the scatter plot adjusted according to

the values of ARPD and ARPT seen in Table 19. The objective is to find the vector closest to $P(-1, 0)$, once the values are asymptotic to $ARPD = 0$ and $ARPT = -1$. Therefore, by calculating the distance between a point relative to a given method and $P(-1, 0)$, one can conclude that BDP-SN(5000) is the most efficient method among those that have been tested in this experimental set, since it is the closest to the reference point. Although MILP might not produce the farthest point, it can be observed that some BDP-SN algorithms (e.g. BDP-SN(1000), BDP-SN(1250), BDP-SN(2500), BDP-SN(10000)) are indeed better choices when compared to it because their ARPD is, in fact, smaller and the ARPT is much closer to the reference axis. In addition, some of these methods are comparable with MILP's performance, once they produce a much smaller ARPD whereas their success rate is gauged to be worse, yet relatively close.

The last analysis comprises the evaluation of the ARPD with respect to the variation in the number of jobs, variation in the number of machines and variation in the setup rates. The first situation is depicted in Fig. 10, and one can notice that for $n = 4$ and $n = 5$, every algorithm is able to maintain their ARPD levels. The MILP method is able to keep its ARPD at zero up to $n = 10$, which means that it responds well to variations in a small-sized job set and the same can be stated about the BDP-SN algorithms with large window widths. However, this scenario tends to change once a medium-sized job set is analyzed and MILP presents a strong leap when varying jobs in this class of jobs while large window width BDP-SN algorithms tend to vary subtly when compared to the previous n analyzed. Note that for BDP-SN(10000), the ARPD values tend to reduce as the number of jobs increases. It is worth mentioning that, despite presenting continuous lines in the graph, there exists no guarantee that the behaviour displayed by those lines is in fact the behaviour of the system for jobs in-between and hence, the reader should only draw conclusions from the markers relative to the specific job set that has been used in the computational experiments.

Fig. 11 depicts the behaviour of ARPD regarding the variation in the number of machines. It is noticeable that for small window widths up to $H = 500$ the ARPD tends to increase as number of machines in the given set becomes larger. Indeed, when undergoing the computational experiments, the BDP-SN algorithms that comprised these window widths did not present significant changes for this set of machines. For the MILP method, it has been shown that it tends to increase for the small-sized portion of the machine set, however for $m > 5$ it responds quite well, even if slight changes are bound to occur for $m > 10$. For large window widths, one can observe that the algorithms tend to show a decreasing rate for the small-sized machines while they tend to increase for the medium-sized machines. Nevertheless, these increasing rates are not severe, which leads to satisfactory values of ARPD and consequently, to an excellent performance of these methods when compared to MILP and their smaller window widths counterparts.

In Fig. 12 it is shown how ARPD responds to the variation in setup rates. It is observed that the behaviour is clearly the same for the BDP-SN algorithms, regardless the window width and this is an expected outcome, once the nature of it is the same for the BDP-SN. Furthermore, the range of the ARPD is not drastically affected by altering the setup rates and it is noticed that BDP-SN tends to present better solution for $ST_{(125)}$, once the ARPDs reach the minimum with regard to their respective window width. For MILP, the variation has a different effect, since as the setup rates grow larger, the values of ARPD increase rapidly and the changes can be clearly perceived.

A final remark that needs to be done is a comparison between the two experimental sets that have been outlined in this chapter. When analyzing the graphs, one may realize that the plots with the same characteristics may differ and it is relevant to mention that this is bound to happen because there are some factor that have influence on the results obtained experimentally. The first one is the number of samples for each algorithm. which are greater for the second set. Furthermore, when analyzing the solution quality regarding the number of machines there exists a rupture on the first experimental set due to the number of samples used for a given set. The number of machine is also another determinant factor to establish a difference between the two sets because the first one is bounded by at most 10 machines and the second comprises 20 machines at most. Finally, the first set is restricted to a single type of setup rates and the second set presents a variation of those and as it was previously shown, the alteration in these rates have an effect upon the solution quality.

Figure 6: ARPD variation with respect to the first group of machines

Figure 7: ARPD variation with respect to the second group of machines

Figure 8: Boxplot relative to the RPD data for the second experimental set

Figure 9: Tradeoff with respect to ARPD and ARPT for the second experimental set

Figure 10: ARPD variation with respect to the number of jobs for the second experimental set

Figure 11: ARPD variation with respect to the number of machines for the second experimental set

Figure 12: ARPD variation with respect to the setup rates for the second experimental set

7 CONCLUSION

Scheduling problems have been one of the pillars with regard to general problems in Operations Research. Since it is a combinatorial problem, it is expected that any slight variation might affect majorly the complexity associated to a given ramification of a classic approach. Despite this fact, some trends in terms of objective functions and technological constraints have been placed in more contemporary problems integrated with other portions of the supply chain over the last ten years and consequently, the classic problems in scheduling have received little attention from the researchers.

Therefore, this research focuses on working on a classic problem that still plays a relevant role in the industrial scenario, which is the blocking flow-shop problem with sequence dependent setup times. Due to little occurrences in literature for this problem and therefore, scarce solution methods to deal with it, we decided to investigate some alternative options that could be fitting for the problem of interest. By researching some methods, it was noticeable that dynamic programming has been resurfacing and been aggregated to other algorithms in order to produce more efficient ones. Hence, after carrying out a thorough research, Bounded Dynamic Programming has been selected due to its results regarding blocking flow-shop and in more complex systems such as job-shops and open shops with the inclusion of sequence dependent setup times.

The mathematical analysis has been outlined by using the concepts regarding the BDP preliminaries with a lower bound formalism and the algorithmic structure has been delineated according to the the junction of both sources. Moreover, some adjustments have been made in the initial algorithm, which is similar to the one in (BAUTISTA et al., 2012), so as to increase the algorithm's performance, once the setup inclusion generates a greater computational burden. These modifications are simply the ordering and the disregard of some functions that have been considered by the original authors and are related with sorting and state space reduction.

The experiments are divided into two sets. The first one is composed of 540 problems, whose division takes place according to $n = \{10, 12, 14, 16, 18, 20\}$ and $m = \{2, 3, 4, 5, 7, 10\}$. The BDP-SN, which is the method based on Bounded Dynamic Programming is basically divided into 13 replicas, each containing a specific window width, and a comparison is established among them, a B&B algorithm and a MILP. The second set is defined analogously but each method is composed of 2880 problems and the comparison is established only with MILP.

The statistics involved in the experiments show that for the first set, the success rate can reach 87%, approximately and for the scenario involved, some of the BDP-SN

algorithms would outperform the MILP and B&B in terms of success rate, ARPD and ARPT measures. Also, statistical difference, which is expected to happen, is confirmed for most methods and this is an initial indication of the best performance achieved by the some of the BDP-SN methods when compared to the others. In addition, the boxplot analysis shows the evolution of the method by increasing the window widths and how the makespan is, in fact, improved and how the variability tends to diminish with the enlargement of window widths. In addition, the trade-off analysis evidences that BDP-SN(5000) is the best choice of algorithm among the ones proposed since the ARPD and ARPT present the shortest distance from the from the reference point, meaning that it can yield a small variation regarding the optimal solutions within a small amount of CPU time.

The statistical analysis is consonant with the one for the first set regarding the advantage of using some of the BDP-SN methods. Since the samples are larger, it is expected that the success rates increase and better values of ARPD and ARPT are obtained when comparing the first and second sets. The MILP presents a quite good performance, being able to reach optimality for 78.65% of the samples, even though the average time associated to it is much larger when compared to the BDP-SN methods. The best one regarding BDP-SN is BDP-SN(10000), which is able to accurately reach the optimal solution for 2709 samples out of 2880 (approximately 94%). The Kruskal-Wallis analysis shows that 70 pairwise comparisons present statistical difference, which means that some BDP-SN might perform better than the MILP or even be a reasonable substitute in a trade-off analysis. Therefore, these assumptions are both confirmed in the boxplot evaluation and the trade-off analysis. The first one shows that good results in terms of variability can be obtained by using algorithms with window widths larger than 1000, once they have the same interquartile difference of MILP and their variability is way more reduced. Furthermore, the trade-off assessment shows that BDP-SN(5000) is the one with the best performance when analyzing both ARPD and ARPT.

No further analysis has been made, however some considerations must be made, once the method worked efficiently for the samples evaluated here. The first one is to expand the number of jobs and verify the strength of the algorithm for larger instances. Taking into account the work done by (BAUTISTA et al., 2012), the BDP for the blocking flow shop has been applied to all the Taillard instances and this might be an indication that the BDP-SN can also obtain fairly good solutions for larger instances. Secondly, the lower bound is also another pathway that must be considered, since the BDP-SN strength can be measured by the tightness of the bound and further research might be developed in constructing a more efficient bound that can show the best trade-off between tightness and exactness. The last consideration is with regard to the upper bound, which here has relied on $H = 1$ for the BDP, however the development of more appropriate heuristics might increase the efficiency of the algorithm and enhance the number of optimal solutions for our benchmark and ones with larger number of jobs.

Since BDP-SN has yielded high quality solutions for the $Fm|s_{ijk}, block|C_{\max}$ it might be also suitable for other regular objective functions such as total tardiness, number of tardy jobs and total flow time. Additionally, once those have been proven efficient, BDP could be extended to more complex objective functions that are either multi-criteria or non-regular. In terms of refinements regarding the BDP structure, we have seen that BDP is dependent on efficient dominance rules so as to reduce the state space and therefore, research on developing new dominance rules might be useful in the algorithm's performance. Additionally, the window width has been defined according to previous sources and rather than setting it from static values, devising a dynamic procedure that is able to widen or shrink its size might help improve the solutions as well as the computational effort.

BIBLIOGRAPHY

ABDUL-RAZAQ, T.; POTTS, C. Dynamic programming state-space relaxation for single-machine scheduling. **Journal of the Operational Research Society**, Taylor & Francis, v. 39, n. 2, p. 141–152, 1988.

ABDUL-RAZAQ, T.; POTTS, C. N.; WASSENHOVE, L. N. V. A survey of algorithms for the single machine total weighted tardiness scheduling problem. **Discrete Applied Mathematics**, Elsevier, v. 26, n. 2-3, p. 235–253, 1990.

ABEDINNIA, H.; GLOCK, C. H.; SCHNEIDER, M. D. Machine scheduling in production: a content analysis. **Applied Mathematical Modelling**, Elsevier, v. 50, p. 279–299, 2017.

AGNETIS, A.; ALFIERI, A.; NICOSIA, G. Assessing the quality of heuristic solutions to parallel machines min–max scheduling problems. **International Journal of Production Economics**, Elsevier, v. 122, n. 2, p. 755–762, 2009.

AKKER, J. M. van D.; HOOGEVEEN, J. A.; VELDE, S. L. van de. Parallel machine scheduling by column generation. **Operations Research**, INFORMS, v. 47, n. 6, p. 862–872, 1999.

ALIDAEE, B. Schedule of n jobs on two identical machines to minimize weighted mean flow time. **Computers & industrial engineering**, Elsevier, v. 24, n. 1, p. 53–55, 1993.

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. **European journal of operational research**, Elsevier, v. 187, n. 3, p. 985–1032, 2008.

ALLAOUI, H. et al. Scheduling of a two-machine flowshop with availability constraints on the first machine. **International Journal of Production Economics**, Elsevier, v. 99, n. 1-2, p. 16–27, 2006.

ALON, N. et al. Approximation schemes for scheduling on parallel machines. **Journal of Scheduling**, Wiley Online Library, v. 1, n. 1, p. 55–66, 1998.

ARTIGUES, C.; FEILLET, D. A branch and bound method for the job-shop problem with sequence-dependent setup times. **Annals of Operations Research**, Springer, v. 159, n. 1, p. 135–159, 2008.

BAKER, K. R. Computational experience with a sequencing algorithm adapted to the tardiness problem. **AIIE Transactions**, Taylor & Francis, v. 9, n. 1, p. 32–35, 1977.

BAKER, K. R.; MERTEN, A. G. Scheduling with parallel processors and linear delay costs. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 20, n. 4, p. 793–804, 1973.

BALAS, E.; SIMONETTI, N.; VAZACOPOULOS, A. Job shop scheduling with setup times, deadlines and precedence constraints. **Journal of Scheduling**, Springer, v. 11, n. 4, p. 253–262, 2008.

BAPTISTE, P. An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. **Operations Research Letters**, Elsevier, v. 24, n. 4, p. 175–180, 1999.

_____. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. **Journal of Scheduling**, Wiley Online Library, v. 2, n. 6, p. 245–252, 1999.

_____. Scheduling equal-length jobs on identical parallel machines. **Discrete Applied Mathematics**, Elsevier, v. 103, n. 1-3, p. 21–32, 2000.

BAPTISTE, P.; FLAMINI, M.; SOURD, F. Lagrangian bounds for just-in-time job-shop scheduling. **Computers & Operations Research**, Elsevier, v. 35, n. 3, p. 906–915, 2008.

BARD, J. F.; VENKATRAMAN, K.; FEO, T. A. Single machine scheduling with flow time and earliness penalties. **Journal of Global Optimization**, Springer, v. 3, n. 3, p. 289–309, 1993.

BARNES, J. W.; VANSTON, L. K. Scheduling jobs with linear delay penalties and sequence dependent setup costs. **Operations Research**, INFORMS, v. 29, n. 1, p. 146–160, 1981.

BARTAL, Y. et al. Multiprocessor scheduling with rejection. **SIAM Journal on Discrete Mathematics**, SIAM, v. 13, n. 1, p. 64–78, 2000.

BAUTISTA, J. et al. Solving the fm/ block/ cmax problem using bounded dynamic programming. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 25, n. 6, p. 1235–1245, 2012.

BAUTISTA, J.; PEREIRA, J. A dynamic programming based heuristic for the assembly line balancing problem. **European Journal of Operational Research**, Elsevier, v. 194, n. 3, p. 787–794, 2009.

BELLMAN, R. On the theory of dynamic programming. **Proceedings of the National Academy of Sciences of the United States of America**, National Academy of Sciences, v. 38, n. 8, p. 716, 1952.

BELLMAN, R. et al. The theory of dynamic programming. **Bulletin of the American Mathematical Society**, American Mathematical Society, v. 60, n. 6, p. 503–515, 1954.

BEN-YEHOSHUA, Y.; MOSHEIOV, G. A single machine scheduling problem to minimize total early work. **Computers & Operations Research**, Elsevier, v. 73, p. 115–118, 2016.

BIANCO, L.; DELL'OLMO, P.; GIORDANI, S. Minimizing total completion time subject to release dates and sequence-dependent processing times. **Annals of Operations Research**, Springer, v. 86, p. 393–415, 1999.

BLAZEWICZ, J.; DROR, M.; WEGLARZ, J. Mathematical programming formulations for machine scheduling: A survey. **European Journal of Operational Research**, Elsevier, v. 51, n. 3, p. 283–300, 1991.

BŁAŻEWICZ, J. et al. New trends in machine scheduling. **European Journal of Operational Research**, Elsevier, v. 37, n. 3, p. 303–317, 1988.

_____. Open shop scheduling problems with late work criteria. **Discrete Applied Mathematics**, Elsevier, v. 134, n. 1-3, p. 1–24, 2004.

BLAZEWCZ, J. et al. A comparison of solution procedures for two-machine flow shop scheduling with late work criterion. **Computers & Industrial Engineering**, Elsevier, v. 49, n. 4, p. 611–624, 2005.

BŁAŻEWICZ, J. et al. The two-machine flow-shop problem with weighted late work criterion and common due date. **European Journal of Operational Research**, Elsevier, v. 165, n. 2, p. 408–415, 2005.

BLAZEWCZ, J. et al. A note on the two machine job shop with the weighted late work criterion. **Journal of Scheduling**, Springer, v. 10, n. 2, p. 87–95, 2007.

BOSIO, A.; RIGHINI, G. A dynamic programming algorithm for the single-machine scheduling problem with release dates and deteriorating processing times. **Mathematical methods of operations research**, Springer, v. 69, n. 2, p. 271, 2009.

BOUQUARD, J.-L. et al. Two-machine flow shop scheduling problems with no-wait jobs. **Operations Research Letters**, Elsevier, v. 33, n. 3, p. 255–262, 2005.

BRUCKER, P.; JURISCH, B.; JURISCH, M. Open shop problems with unit time operations. **Zeitschrift für Operations Research**, Springer, v. 37, n. 1, p. 59–73, 1993.

BRUCKER, P. et al. Scheduling unit time open shops to minimize the weighted number of late jobs. **Operations research letters**, Elsevier, v. 14, n. 5, p. 245–250, 1993.

BÜLBÜL, K.; KEDAD-SIDHOUM, S.; ŞEN, H. Single-machine common due date total earliness/tardiness scheduling with machine unavailability. **Journal of Scheduling**, Springer, v. 22, n. 5, p. 543–565, 2019.

CARRAWAY, R. L. et al. Single machine sequencing with nonlinear multicriteria cost functions: an application of generalized dynamic programming. **Computers & operations research**, Elsevier, v. 19, n. 1, p. 69–77, 1992.

CHAND, S.; CHHAJED, D.; TRAUB, R. A single-machine scheduling model with fixed-interval deliveries. **Production and Operations Management**, Wiley Online Library, v. 3, n. 4, p. 296–307, 1994.

CHAND, S.; SCHNEEBERGER, H. Single machine scheduling to minimize weighted earliness subject to no tardy jobs. **European Journal of Operational Research**, Elsevier, v. 34, n. 2, p. 221–230, 1988.

CHEN, X. et al. Scheduling on parallel identical machines with late work criterion: Offline and online cases. **Journal of Scheduling**, Springer, v. 19, n. 6, p. 729–736, 2016.

CHEN, Z.-L. Simultaneous job scheduling and resource allocation on parallel machines. **Annals of Operations Research**, Springer, v. 129, n. 1-4, p. 135–153, 2004.

CHEN, Z.-L.; POWELL, W. B. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. **European Journal of Operational Research**, Elsevier, v. 116, n. 1, p. 220–232, 1999.

_____. Solving parallel machine scheduling problems by column generation. **INFORMS Journal on Computing**, INFORMS, v. 11, n. 1, p. 78–94, 1999.

_____. Exact algorithms for scheduling multiple families of jobs on parallel machines. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 50, n. 7, p. 823–840, 2003.

CHENG, T.; CHEN, Z.-L.; LI, C.-L. Single-machine scheduling with trade-off between number of tardy jobs and resource allocation. **Operations Research Letters**, Elsevier, v. 19, n. 5, p. 237–242, 1996.

CHENG, T.; DIAMOND, J. Scheduling two job classes on parallel machines. **IIE transactions**, Taylor & Francis, v. 27, n. 5, p. 689–693, 1995.

CHENG, T.; KRAVCHENKO, S. A.; LIN, B. M. Scheduling step-deteriorating jobs to minimize the total completion time. **Computers & Industrial Engineering**, Elsevier, p. 106329, 2020.

CHENG, T. E.; DING, Q. Single machine scheduling with deadlines and increasing rates of processing times. **Acta Informatica**, Springer, v. 36, n. 9-10, p. 673–692, 2000.

CHENG, T. E.; JANIYAK, A.; KOVALYOV, M. Y. Bicriterion single machine scheduling with resource dependent processing times. **SIAM Journal on Optimization**, SIAM, v. 8, n. 2, p. 617–630, 1998.

CHENG, Y.; SUN, S. Scheduling linear deteriorating jobs with rejection on a single machine. **European Journal of Operational Research**, Elsevier, v. 194, n. 1, p. 18–27, 2009.

CORWIN, B. D.; ESOGBUE, A. O. Two machine flow shop scheduling problems with sequence dependent setup times: A dynamic programming approach. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 21, n. 3, p. 515–524, 1974.

DAVARI, M. et al. Minimizing makespan on a single machine with release dates and inventory constraints. **European Journal of Operational Research**, Elsevier, 2020.

DAWANDE, M.; GAVIRNENI, S.; RACHAMADUGU, R. Scheduling a two-stage flowshop under makespan constraint. **Mathematical and computer modelling**, Elsevier, v. 44, n. 1-2, p. 73–84, 2006.

DE, P.; GHOSH, J. B.; WELLS, C. E. Scheduling to minimize weighted earliness and tardiness about a common due-date. **Computers & operations research**, Elsevier, v. 18, n. 5, p. 465–475, 1991.

_____. On the minimization of completion time variance with a bicriteria extension. **Operations Research**, INFORMS, v. 40, n. 6, p. 1148–1155, 1992.

_____. Due-date assignment and early/tardy scheduling on identical parallel machines. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 41, n. 1, p. 17–32, 1994.

DESSOUKY, M. I. et al. Scheduling identical jobs on uniform parallel machines. **Statistica Neerlandica**, Wiley Online Library, v. 44, n. 3, p. 115–123, 1990.

DONDETI, V. R.; MOHANTY, B. B. Impact of learning and fatigue factors on single machine scheduling with penalties for tardy jobs. **European Journal of Operational Research**, Elsevier, v. 105, n. 3, p. 509–524, 1998.

- DONG, M. Parallel machine scheduling with limited controllable machine availability. **International Journal of Production Research**, Taylor & Francis, v. 51, n. 8, p. 2240–2252, 2013.
- DUTTA, S. K.; CUNNINGHAM, A. A. Sequencing two-machine flow-shops with finite intermediate storage. **Management Science**, INFORMS, v. 21, n. 9, p. 989–996, 1975.
- FAN, B.; CHENG, T. Two-agent scheduling in a flowshop. **European Journal of Operational Research**, Elsevier, v. 252, n. 2, p. 376–384, 2016.
- FAN, B. et al. Scheduling resumable deteriorating jobs on a single machine with non-availability constraints. **Theoretical Computer Science**, Elsevier, v. 412, n. 4-5, p. 275–280, 2011.
- FUCHIGAMI, H. Y.; RANGEL, S. A survey of case studies in production scheduling: Analysis and perspectives. **Journal of Computational Science**, Elsevier, v. 25, p. 425–436, 2018.
- GÉLINAS, S.; SOUMIS, F. A dynamic programming algorithm for single machine scheduling with ready times. **Annals of Operations Research**, Springer, v. 69, p. 135–156, 1997.
- GENS, G. V.; LEVNER, E. V. Fast approximation algorithm for job sequencing with deadlines. **Discrete Applied Mathematics**, Elsevier, v. 3, n. 4, p. 313–318, 1981.
- GERSTL, E.; MOSHEIOV, G. The single machine con problem with unavailability period. **International Journal of Production Research**, Taylor & Francis, p. 1–15, 2020.
- GHALAMI, L.; GROSU, D. Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm. **Journal of Parallel and Distributed Computing**, Elsevier, v. 133, p. 221–231, 2019.
- GHIRARDI, M.; POTTS, C. N. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. **European Journal of Operational Research**, Elsevier, v. 165, n. 2, p. 457–467, 2005.
- GLASS, C. A.; KELLERER, H. Parallel machine scheduling with job assignment restrictions. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 54, n. 3, p. 250–257, 2007.
- GONG, H.; TANG, L.; DUIN, C. A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. **Computers & Operations Research**, Elsevier, v. 37, n. 5, p. 960–969, 2010.
- GRAHAM, R. L. et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: **Annals of discrete mathematics**. [S.l.]: Elsevier, 1979. v. 5, p. 287–326.
- GROMICHO, J. A. et al. Solving the job-shop scheduling problem optimally by dynamic programming. **Computers & Operations Research**, Elsevier, v. 39, n. 12, p. 2968–2977, 2012.

GU, M. et al. Single-machine scheduling problems with machine aging effect and an optional maintenance activity. **Applied Mathematical Modelling**, Elsevier, v. 40, n. 21-22, p. 8862–8871, 2016.

GUÉRET, C.; PRINS, C. A new lower bound for the open-shop problem. **Annals of Operations Research**, Springer, v. 92, p. 165–183, 1999.

GUPTA, J. N.; MAYKUT, A. R. Scheduling jobs on parallel processors with dynamic programming. **Decision Sciences**, Wiley Online Library, v. 4, n. 4, p. 447–457, 1973.

HALL, N. G.; KUBIAK, W.; SETHI, S. P. Earliness–tardiness scheduling problems, ii: Deviation of completion times about a restrictive common due date. **Operations Research**, INFORMS, v. 39, n. 5, p. 847–856, 1991.

HALL, N. G.; LESAOANA, M.; POTTS, C. N. Scheduling with fixed delivery dates. **Operations Research**, INFORMS, v. 49, n. 1, p. 134–144, 2001.

HALL, N. G.; POSNER, M. E. Earliness-tardiness scheduling problems, i: weighted deviation of completion times about a common due date. **Operations Research**, INFORMS, v. 39, n. 5, p. 836–846, 1991.

HANED, A. et al. Scheduling on parallel machines with preemption and transportation delays. **Computers & operations research**, Elsevier, v. 39, n. 2, p. 374–381, 2012.

HARIRI, A.; POTTS, C. Single machine scheduling with deadlines to minimize the weighted number of tardy jobs. **Management Science**, INFORMS, v. 40, n. 12, p. 1712–1719, 1994.

HARIRI, A. M.; POTTS, C. N.; WASSENHOVE, L. N. V. Single machine scheduling to minimize total weighted late work. **ORSA Journal on Computing**, INFORMS, v. 7, n. 2, p. 232–242, 1995.

HENDEL, Y.; SOURD, F. Efficient neighborhood search for just-in-time scheduling problems. **European Journal of Operational Research**, 2005.

HOOGEVEEN, J.; VELDE, S. Van de. Scheduling around a small common due date. **European Journal of Operational Research**, Elsevier, v. 55, n. 2, p. 237–242, 1991.

HOROWITZ, E.; SAHNI, S. Exact and approximate algorithms for scheduling nonidentical processors. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 23, n. 2, p. 317–327, 1976.

HOU, S.; HOOGEVEEN, H. The three-machine proportionate flow shop problem with unequal machine speeds. **Operations Research Letters**, Elsevier, v. 31, n. 3, p. 225–231, 2003.

IBARAKI, T.; NAKAMURA, Y. A dynamic programming method for single machine scheduling. **European Journal of Operational Research**, Elsevier, v. 76, n. 1, p. 72–82, 1994.

JANSEN, K.; MASTROLILLI, M. Approximation schemes for parallel machine scheduling problems with controllable processing times. **Computers & Operations Research**, Elsevier, v. 31, n. 10, p. 1565–1581, 2004.

JANSEN, K.; PORKOLAB, L. Improved approximation schemes for scheduling unrelated parallel machines. **Mathematics of Operations Research**, INFORMS, v. 26, n. 2, p. 324–338, 2001.

JENG, A. A.-K.; LIN, B. M. Makespan minimization in single-machine scheduling with step-deterioration of processing times. **Journal of the Operational Research Society**, Springer, v. 55, n. 3, p. 247–256, 2004.

JI, M.; HE, Y.; CHENG, T. E. Scheduling linear deteriorating jobs with an availability constraint on a single machine. **Theoretical Computer Science**, Elsevier, 2006.

JIANG, F. et al. Competitive two-agent scheduling problems to minimize the weighted combination of makespans in a two-machine open shop. **Engineering Optimization**, Taylor & Francis, v. 50, n. 4, p. 684–697, 2018.

JOHNSON, S. M. Optimal two-and three-stage production schedules with setup times included. **Naval research logistics quarterly**, Wiley Online Library, v. 1, n. 1, p. 61–68, 1954.

JÓZEFOWSKA, J.; JURISCH, B.; KUBIAK, W. Scheduling shops to minimize the weighted number of late jobs. **Operations Research Letters**, Elsevier, v. 16, n. 5, p. 277–283, 1994.

KACEM, I.; CHU, C.; SOUISSI, A. Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times. **Computers & operations research**, Elsevier, v. 35, n. 3, p. 827–844, 2008.

KÁDÁROVÁ, J. et al. Education in industrial engineering in slovakia. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 143, p. 157–162, 2014.

KEDAD-SIDHOUM, S.; SOLIS, Y. R.; SOURD, F. Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. **European Journal of Operational Research**, Elsevier, v. 189, n. 3, p. 1305–1316, 2008.

KELLERER, H.; STRUSEVICH, V. A. Scheduling parallel dedicated machines with the speeding-up resource. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 55, n. 5, p. 377–389, 2008.

KIM, E.-S.; SUNG, C.-S.; LEE, I.-S. Scheduling of parallel machines to minimize total completion time subject to s-precedence constraints. **Computers & Operations Research**, Elsevier, v. 36, n. 3, p. 698–710, 2009.

KISE, H.; IBARAKI, T.; MINE, H. A solvable case of the one-machine scheduling problem with ready and due times. **Operations Research**, INFORMS, v. 26, n. 1, p. 121–126, 1978.

KLAMROTH, K.; WIECEK, M. M. A time-dependent multiple criteria single-machine scheduling problem. **European Journal of Operational Research**, Elsevier, v. 135, n. 1, p. 17–26, 2001.

KOULAMAS, C. The proportionate flow shop total tardiness problem. **European Journal of Operational Research**, Elsevier, v. 284, n. 2, p. 439–444, 2020.

KOVALEV, S. et al. Semi-v-shape property for two-machine no-wait proportionate flow shop problem with tadc criterion. **International Journal of Production Research**, Taylor & Francis, v. 57, n. 2, p. 560–566, 2019.

KOVALYOV, M. Y.; WERNER, F. A polynomial approximation scheme for problem f2/rj/cmax. **Operations Research Letters**, Elsevier, v. 20, n. 2, p. 75–79, 1997.

KRÄMER, F.-J.; LEE, C.-Y. Due window scheduling for parallel machines. **Mathematical and Computer Modelling**, Elsevier, v. 20, n. 2, p. 69–89, 1994.

KUBIAK, W.; VELDE, S. van de. Scheduling deteriorating jobs to minimize makespan. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 45, n. 5, p. 511–523, 1998.

KUBZIN, M. A.; STRUSEVICH, V. A. Planning machine maintenance in two-machine shop scheduling. **Operations Research**, INFORMS, v. 54, n. 4, p. 789–800, 2006.

KUNNATHUR, A. S.; GUPTA, S. K. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. **European Journal of Operational Research**, Elsevier, v. 47, n. 1, p. 56–64, 1990.

LANN, A.; MOSHEIOV, G. Single machine scheduling to minimize the number of early and tardy jobs. **Computers & Operations Research**, Elsevier, v. 23, n. 8, p. 769–781, 1996.

LAWLER, E. L. A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness. In: **Annals of discrete Mathematics**. [S.l.]: Elsevier, 1977. v. 1, p. 331–342.

_____. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. **Annals of Operations Research**, Springer, v. 26, n. 1-4, p. 125–133, 1990.

LAWLER, E. L.; MOORE, J. M. A functional equation and its application to resource allocation and sequencing problems. **Management Science**, INFORMS, v. 16, n. 1, p. 77–84, 1969.

LEE, C.-Y. Machine scheduling with an availability constraint. **Journal of global optimization**, Springer, v. 9, n. 3-4, p. 395–416, 1996.

_____. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. **Operations research letters**, Elsevier, v. 20, n. 3, p. 129–139, 1997.

_____. Two-machine flowshop scheduling with availability constraints. **European Journal of Operational Research**, Elsevier, v. 114, n. 2, p. 420–429, 1999.

LEE, C.-Y.; CHEN, Z.-L. Scheduling jobs and maintenance activities on parallel machines. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 47, n. 2, p. 145–165, 2000.

LEE, C.-Y.; LIMAN, S. D. Capacitated two-parallel machines scheduling to minimize sum of job completion times. **Discrete Applied Mathematics**, Elsevier, v. 41, n. 3, p. 211–222, 1993.

LEYVAND, Y. et al. Just-in-time scheduling with controllable processing times on parallel machines. **Journal of combinatorial optimization**, Springer, v. 19, n. 3, p. 347–368, 2010.

LI, C.-L. A heuristic for parallel machine scheduling with agreeable due dates to minimize the number of late jobs. **Computers & operations research**, Elsevier, v. 22, n. 3, p. 277–283, 1995.

LI, C.-L.; WANG, X. Scheduling parallel machines with inclusive processing set restrictions and job release times. **European Journal of Operational Research**, Elsevier, v. 200, n. 3, p. 702–710, 2010.

LI, S. Scheduling proportionally deteriorating jobs in two-machine open shop with a non-bottleneck machine. **Asia-Pacific Journal of Operational Research**, World Scientific, v. 28, n. 05, p. 623–631, 2011.

LI, S.; FAN, B. Single-machine scheduling with proportionally deteriorating jobs subject to availability constraints. **Asia-Pacific Journal of Operational Research**, World Scientific, v. 29, n. 04, p. 1250019, 2012.

LI, S.; YUAN, J. Parallel-machine scheduling with deteriorating jobs and rejection. **Theoretical Computer Science**, Elsevier, v. 411, n. 40-42, p. 3642–3650, 2010.

LI, S.-S.; YUAN, J.-J. Single-machine scheduling with multi-agents to minimize total weighted late work. **Journal of Scheduling**, Springer, p. 1–16, 2020.

LI, Y. et al. A desired load distribution model for scheduling of unrelated parallel machines. **International journal of production research**, Taylor & Francis, v. 43, n. 23, p. 5033–5046, 2005.

LIMAN, S. D.; RAMASWAMY, S. Earliness-tardiness scheduling problems with a common delivery window. **Operations Research Letters**, Elsevier, v. 15, n. 4, p. 195–203, 1994.

LOPES, M. J. P.; CARVALHO, J. V. de. A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. **European journal of operational research**, Elsevier, v. 176, n. 3, p. 1508–1527, 2007.

LUH, P. B. et al. Job shop scheduling with group-dependent setups, finite buffers, and long time horizon. **Annals of Operations Research**, Springer, v. 76, p. 233–259, 1998.

LUO, W.; CHEN, L.; ZHANG, G. Approximation schemes for two-machine flow shop scheduling with two agents. **Journal of Combinatorial Optimization**, Springer, v. 24, n. 3, p. 229–239, 2012.

MARTINEZ, S. et al. Complexity of flowshop scheduling problems with a new blocking constraint. **European Journal of Operational Research**, Elsevier, v. 169, n. 3, p. 855–864, 2006.

MELLOULI, R. et al. Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times. **European Journal of Operational Research**, Elsevier, v. 197, n. 3, p. 1150–1165, 2009.

MERCHAN, A. F.; MARAVELIAS, C. T. Preprocessing and tightening methods for time-indexed mip chemical production scheduling models. **Computers & Chemical Engineering**, Elsevier, v. 84, p. 516–535, 2016.

MIYATA, H. H.; NAGANO, M. S. The blocking flow shop scheduling problem: A comprehensive and conceptual review. **Expert Systems with Applications**, Elsevier, v. 137, p. 130–156, 2019.

MOORE, J. M. An n job, one machine sequencing algorithm for minimizing the number of late jobs. **Management science**, INFORMS, v. 15, n. 1, p. 102–109, 1968.

MOR, B.; MOSHEIOV, G.; SHAPIRA, D. Flowshop scheduling with learning effect and job rejection. **Journal of Scheduling**, Springer, p. 1–11, 2019.

NEUFELD, J. S.; GUPTA, J. N.; BUSCHER, U. A comprehensive review of flowshop group scheduling literature. **Computers & Operations Research**, Elsevier, v. 70, p. 56–74, 2016.

NISHI, T.; HIRANAKA, Y. Lagrangian relaxation and cut generation for sequence-dependent setup time flowshop scheduling problems to minimise the total weighted tardiness. **International Journal of Production Research**, Taylor & Francis, v. 51, n. 16, p. 4778–4796, 2013.

OZOLINS, A. Bounded dynamic programming algorithm for the job shop problem with sequence dependent setup times. **Operational Research**, Springer, p. 1–28, 2018.

_____. A new exact algorithm for no-wait job shop problem to minimize makespan. **Operational Research**, Springer, p. 1–31, 2018.

_____. Dynamic programming approach for solving the open shop problem. **Central European Journal of Operations Research**, Springer, p. 1–16, 2019.

_____. Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. **Central European Journal of Operations Research**, Springer, v. 27, n. 1, p. 15–38, 2019.

PESSOA, A. et al. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. **Mathematical Programming Computation**, Springer, v. 2, n. 3-4, p. 259–290, 2010.

PINEDO, M. **Scheduling**. [S.l.]: Springer, 2012. v. 29.

POTTS, C. N.; WASSENHOVE, L. N. V. A decomposition algorithm for the single machine total tardiness problem. **Operations Research Letters**, Elsevier, v. 1, n. 5, p. 177–181, 1982.

_____. Approximation algorithms for scheduling a single machine to minimize total late work. **Operations Research Letters**, Elsevier, v. 11, n. 5, p. 261–266, 1992.

_____. Single machine scheduling to minimize total late work. **Operations Research**, INFORMS, v. 40, n. 3, p. 586–595, 1992.

POTTS, C. N.; WASSENHOVE, L. V. Dynamic programming and decomposition approaches for the single machine total tardiness problem. **European Journal of Operational Research**, Elsevier, v. 32, n. 3, p. 405–414, 1987.

_____. Algorithms for scheduling a single machine to minimize the weighted number of late jobs. **Management Science**, INFORMS, v. 34, n. 7, p. 843–858, 1988.

- RAMACHANDRA, G.; ELMAGHRABY, S. E. Sequencing precedence-related jobs on two machines to minimize the weighted completion time. **International Journal of Production Economics**, Elsevier, v. 100, n. 1, p. 44–58, 2006.
- REBAI, M.; KACEM, I.; ADJALLAH, K. H. Scheduling jobs and maintenance activities on parallel machines. **Operational Research**, Springer, v. 13, n. 3, p. 363–383, 2013.
- RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flowtime minimization. **Expert Systems with Applications**, Elsevier, v. 42, n. 15-16, p. 6155–6167, 2015.
- RIOS-SOLIS, Y. A.; SOURD, F. Exponential neighborhood search for a parallel machine scheduling problem. **Computers & Operations Research**, Elsevier, v. 35, n. 5, p. 1697–1712, 2008.
- RONCONI, D. P. A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. **Annals of Operations Research**, Springer, v. 138, n. 1, p. 53–65, 2005.
- ROSTAMI, S.; CREEMERS, S.; LEUS, R. Precedence theorems and dynamic programming for the single-machine weighted tardiness problem. **European Journal of Operational Research**, Elsevier, v. 272, n. 1, p. 43–49, 2019.
- ROTHKOPF, M. H. Scheduling independent tasks on parallel processors. **Management Science**, INFORMS, v. 12, n. 5, p. 437–447, 1966.
- RUDEK, R. Parallel machine scheduling with general sum of processing time based models. **Journal of Global Optimization**, Springer, v. 68, n. 4, p. 799–814, 2017.
- SAHNI, S. K. Algorithms for scheduling independent tasks. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 23, n. 1, p. 116–127, 1976.
- SCHRAGE, L.; BAKER, K. R. Dynamic programming solution of sequencing problems with precedence constraints. **Operations research**, INFORMS, v. 26, n. 3, p. 444–449, 1978.
- SEN, T.; BORAH, B. N. On the single-machine scheduling problem with tardiness penalties. **Journal of the Operational Research Society**, Springer, v. 42, n. 8, p. 695–702, 1991.
- SHABTAY, D.; BENSOUSSAN, Y. Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems. **Journal of Scheduling**, Springer, v. 15, n. 1, p. 39–47, 2012.
- SHAFRANSKY, Y.; STRUSEVICH, V. The open shop scheduling problem with a given sequence of jobs on one machine. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 45, n. 7, p. 705–731, 1998.
- SHANG, L. et al. Exact exponential algorithms for 3-machine flowshop scheduling problems. **Journal of Scheduling**, Springer, v. 21, n. 2, p. 227–233, 2018.
- SHAO, Z.; PI, D.; SHAO, W. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. **Swarm and Evolutionary Computation**, Elsevier, v. 40, p. 53–75, 2018.

- SMITH, W. E. Various optimizers for single-stage production. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 3, n. 1-2, p. 59–66, 1956.
- SO, K. C. Some heuristics for scheduling jobs on parallel machines with setups. **Management Science**, INFORMS, v. 36, n. 4, p. 467–475, 1990.
- SONMEZ, A.; BAYKASOGLU, A. A new dynamic programming formulation of (nxm) flowshop sequencing problems with due dates. **International Journal of Production Research**, Taylor & Francis, v. 36, n. 8, p. 2269–2283, 1998.
- SRINIVASAN, V. A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 18, n. 3, p. 317–327, 1971.
- STERNA, M.; CZERNIACHOWSKA, K. Polynomial time approximation scheme for two parallel machines scheduling with a common due date to maximize early work. **Journal of Optimization Theory and Applications**, Springer, v. 174, n. 3, p. 927–944, 2017.
- STRUSEVICH, V. A.; HALL, L. A. An open shop scheduling problem with a non-bottleneck machine. **Operations research letters**, Elsevier, v. 21, n. 1, p. 11–18, 1997.
- SUN, H.; WANG, G. Parallel machine earliness and tardiness scheduling with proportional weights. **Computers & Operations Research**, Elsevier, v. 30, n. 5, p. 801–808, 2003.
- SUNG, S. C.; VLACH, M. Maximizing weighted number of just-in-time jobs on unrelated parallel machines. **Journal of Scheduling**, Springer, v. 8, n. 5, p. 453–460, 2005.
- TAKANO, M.; NAGANO, M. Evaluating the performance of constructive heuristics for the blocking flow shop scheduling problem with setup times. **International Journal of Industrial Engineering Computations**, v. 10, n. 1, p. 37–50, 2019.
- TAKANO, M. I.; NAGANO, M. S. A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. **Cogent Engineering**, Cogent OA, v. 4, n. 1, p. 1389638, 2017.
- TANAKA, S.; FUJIKUMA, S. A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. **Journal of Scheduling**, Springer, v. 15, n. 3, p. 347–361, 2012.
- TANAKA, S.; FUJIKUMA, S.; ARAKI, M. An exact algorithm for single-machine scheduling without machine idle time. **Journal of Scheduling**, Springer, v. 12, n. 6, p. 575–593, 2009.
- TANAKA, S.; SATO, S. An exact algorithm for the precedence-constrained single-machine scheduling problem. **European Journal of Operational Research**, Elsevier, v. 229, n. 2, p. 345–352, 2013.
- TANG, L.; LUO, J. A new ils algorithm for parallel machine scheduling problems. **Journal of Intelligent Manufacturing**, Springer, v. 17, n. 5, p. 609–619, 2006.
- TANG, L.; XUAN, H.; LIU, J. Hybrid backward and forward dynamic programming based lagrangian relaxation for single machine scheduling. **Computers & operations research**, Elsevier, v. 34, n. 9, p. 2625–2636, 2007.

- TANG, L.; ZHANG, Y. A new lagrangian relaxation algorithm for scheduling dissimilar parallel machines with release dates. **International journal of systems science**, Taylor & Francis, v. 42, n. 7, p. 1133–1141, 2011.
- TAUTENHAHN, T.; WOEGINGER, G. J. Minimizing the total completion time in a unit-time open shop with release times. **Operations Research Letters**, Elsevier, v. 20, n. 5, p. 207–212, 1997.
- T'KINDT, V.; CROCE, F. D.; BOUQUARD, J.-L. Enumeration of pareto optima for a flowshop scheduling problem with two criteria. **INFORMS Journal on Computing**, INFORMS, v. 19, n. 1, p. 64–72, 2007.
- T'KINDT, V.; GUPTA, J. N.; BILLAUT, J.-C. Two-machine flowshop scheduling with a secondary criterion. **Computers & Operations Research**, Elsevier, v. 30, n. 4, p. 505–526, 2003.
- TOMAZELLA, C. P.; NAGANO, M. S. A comprehensive review of branch-and-bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem. **Expert Systems with Applications**, Elsevier, p. 113556, 2020.
- TUONG, N. H.; SOUKHAL, A.; BILLAUT, J.-C. A new dynamic programming formulation for scheduling independent tasks with common due date on parallel machines. **European Journal of Operational Research**, Elsevier, v. 202, n. 3, p. 646–653, 2010.
- VELDE, S. Van de. Minimizing the sum of the job completion times in the two-machine flow shop by lagrangian relaxation. **Annals of Operations Research**, Springer, v. 26, n. 1-4, p. 257–268, 1990.
- VENTURA, J. A.; KIM, D.; GARRIGA, F. Single machine earliness–tardiness scheduling with resource-dependent release dates. **European Journal of Operational Research**, Elsevier, v. 142, n. 1, p. 52–69, 2002.
- VENTURA, J. A.; WENG, M. X. An improved dynamic programming algorithm for the single-machine mean absolute deviation problem with a restrictive common due date. **Operations Research Letters**, Elsevier, v. 17, n. 3, p. 149–152, 1995.
- WANG, D.-J. et al. A two-agent single-machine scheduling problem with late work criteria. **Soft Computing**, Springer, v. 21, n. 8, p. 2015–2033, 2017.
- WANG, J. et al. An optimization-based algorithm for job shop scheduling. **Sadhana**, Springer, v. 22, n. 2, p. 241–256, 1997.
- WASSENHOVE, L. V.; GELDERS, L. Four solution techniques for a general one machine scheduling problem: A comparative study. **European Journal of operational research**, Elsevier, v. 2, n. 4, p. 281–290, 1978.
- WEBSTER, S. A note on “schedule of n jobs on two identical machines to minimize weighted mean flor time”. **Computers & industrial engineering**, Elsevier, v. 26, n. 4, p. 795–796, 1994.
- WEBSTER, S.; AZIZOGLU, M. Dynamic programming algorithms for scheduling parallel machines with family setup times. **Computers & Operations Research**, Elsevier, v. 28, n. 2, p. 127–137, 2001.

WENG, M. X.; VENTURA, J. A. Single-machine earliness-tardiness scheduling about a common due date with tolerances. **International journal of production economics**, Elsevier, v. 42, n. 3, p. 217–227, 1996.

WENG, X.; VENTURA, J. A. Scheduling about a given common due date to minimize mean squared deviation of completion times. **European journal of operational research**, Elsevier, v. 88, n. 2, p. 328–335, 1996.

WOEGINGER, G. J. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)? **INFORMS Journal on Computing**, INFORMS, v. 12, n. 1, p. 57–74, 2000.

XINGONG, Z.; YONG, W. Single-machine scheduling con/slk due window assignment problems with sum-of-processed times based learning effect. **Applied Mathematics and Computation**, Elsevier, v. 250, p. 628–635, 2015.

YANO, C. A.; KIM, Y.-D. Algorithms for a class of single-machine weighted tardiness and earliness problems. **European Journal of Operational Research**, Elsevier, v. 52, n. 2, p. 167–178, 1991.

YEUNG, W.; OĞUZ, C.; CHENG, T. E. Two-stage flowshop earliness and tardiness machine scheduling involving a common due window. **International Journal of Production Economics**, Elsevier, v. 90, n. 3, p. 421–434, 2004.

YEUNG, W.-K.; OĞUZ, C.; CHENG, T.-C. E. Two-machine flow shop scheduling with common due window to minimize weighted number of early and tardy jobs. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 56, n. 7, p. 593–599, 2009.

YEUNG, W.-K.; OĞUZ, C.; CHENG, T. E. Minimizing weighted number of early and tardy jobs with a common due window involving location penalty. **Annals of Operations Research**, Springer, v. 108, n. 1-4, p. 33–54, 2001.

YEUNG, W.-K.; OGUZ, C.; CHENG, T. E. Single-machine scheduling with a common due window. **Computers & Operations Research**, Elsevier, v. 28, n. 2, p. 157–175, 2001.

YIN, Y. et al. Two-agent scheduling on unrelated parallel machines with total completion time and weighted number of tardy jobs criteria. **Journal of Scheduling**, Springer, v. 22, n. 3, p. 315–333, 2019.

_____. Just-in-time scheduling with two competing agents on unrelated parallel machines. **Omega**, Elsevier, v. 63, p. 41–47, 2016.

_____. Four single-machine scheduling problems involving due date determination decisions. **Information Sciences**, Elsevier, v. 251, p. 164–181, 2013.

_____. Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 63, n. 2, p. 172–183, 2016.

YOO, J.; LEE, I. S. Parallel machine scheduling with maintenance activities. **Computers & Industrial Engineering**, Elsevier, v. 101, p. 361–371, 2016.

ZHANG, L.; LU, L. Parallel-machine scheduling with release dates and rejection. **4OR**, Springer, v. 14, n. 2, p. 165–172, 2016.

ZHANG, L.; LU, L.; YUAN, J. Single machine scheduling with release dates and rejection. **European Journal of Operational Research**, Elsevier, v. 198, n. 3, p. 975–978, 2009.

_____. Single-machine scheduling under the job rejection constraint. **Theoretical Computer Science**, Elsevier, v. 411, n. 16-18, p. 1877–1882, 2010.

_____. Two-machine open-shop scheduling with rejection to minimize the makespan. **Or Spectrum**, Springer, v. 38, n. 2, p. 519–529, 2016.

ZHANG, Z.-H.; BAI, D. An extended study on an open-shop scheduling problem using the minimisation of the sum of quadratic completion times. **Applied Mathematics and Computation**, Elsevier, v. 230, p. 238–247, 2014.

ZHAO, C. et al. Due date assignment and single machine scheduling with deteriorating jobs to minimize the weighted number of tardy jobs. **Applied Mathematics and Computation**, Elsevier, v. 248, p. 503–510, 2014.

ZHAO, C.; TANG, H. Parallel machines scheduling with deteriorating jobs and availability constraints. **Japan Journal of Industrial and Applied Mathematics**, Springer, v. 31, n. 3, p. 501–512, 2014.

ZHAO, C.-L.; TANG, H.-Y. Single machine scheduling with linear deteriorating jobs under predictive disruption. **Asia-Pacific Journal of Operational Research**, World Scientific, v. 28, n. 03, p. 419–429, 2011.

ZHAO, C.-L.; TANG, H.-Y.; CHENG, C.-D. Two-parallel machines scheduling with rate-modifying activities to minimize total completion time. **European Journal of Operational Research**, Elsevier, v. 198, n. 1, p. 354–357, 2009.

ZHAO, K.; LU, X. Approximation schemes for two-agent scheduling on parallel machines. **Theoretical Computer Science**, Elsevier, v. 468, p. 114–121, 2013.

ZOU, J.; YUAN, J. Single-machine scheduling with maintenance activities and rejection. **Discrete Optimization**, Elsevier, v. 38, p. 100609, 2020.