UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
MESTRADO EM ENGENHARIA DE PRODUÇÃO

DANIEL DIAS DE OLIVEIRA NETO

IMPROVEMENTS TO A BRANCH-CUT-AND-PRICE ALGORITHM FOR THE EXACT SOLUTION OF PARALLEL MACHINES SCHEDULING PROBLEMS

NITERÓI
2015

DANIEL DIAS DE OLIVEIRA NETO

IMPROVEMENTS TO A BRANCH-CUT-AND-PRICE ALGORITHM FOR THE EXACT SOLUTION OF PARALLEL MACHINES SCHEDULING PROBLEMS

Thesis presented to the Production Engineering graduate program of Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science. Concentration field: Systems, Decision Support and Logistics.

Advisor: Prof. Dr. Artur Alves Pessoa

Niterói

2015

DANIEL DIAS DE OLIVEIRA NETO

IMPROVEMENTS TO A BRANCH-CUT-AND-PRICE ALGORITHM FOR THE EXACT SOLUTION OF PARALLEL MACHINES SCHEDULING PROBLEMS

Thesis presented to the Production Engineering graduate program of Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science. Concentration field: Systems, Decision Support and Logistics.

Approved in December 2015.

JURY

Advisor: Prof. Artur Alves Pessoa , D.Sc.

Universidade Federal Fluminense

Prof. Eduardo Uchoa Barboza , D.Sc.

Universidade Federal Fluminense

Prof. Ruslan Sadykov , D.Sc.

INRIA Bordeaux – Sud-Quest

Niterói

2015

To Paula.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Artur Pessoa, for his support throughout my Master's. For all his efforts to explain me many complex concepts, for revising my writing and helping me with programming. His passion for his research is admirable and inspiring.

I would also like to thank Dr. Eduardo Uchoa, for his invaluable lessons and willingness to help, every time I showed up by his office. And Dr. Ruslan Sadykov, for accepting to be be part of the Jury and for all valuable comments and suggestions to my work.

To my friends Marcos Roboredo, Luiz Aizemberg, Liana Franco, Hugo Krammer, Frederico Galaxe, André Velasco, Luiz Henrique Santanna, José Maurício, Ubiratam, Rafaelli and Antônio Camargo, thanks for making the time we spent at the lab such a pleasure and fun.

To Paula, for all her support, despite many nights and weekends we spent apart during the past two years. Her company gives purpose to all my endeavours. Last, to my family, for all their support and for teaching me the value of knowledge.

ABSTRACT

This work proposes improvements to a Branch-Cut-and-Price algorithm proposed by Pessoa et al (2010) to solve the parallel machine scheduling problem minimizing the weighted tardiness, along with a review of previous works. A new family of cuts was proposed to strengthen the used relaxation, with an efficient genetic separation algorithm. The main result of this work was the projection of a very large formulation into a more tractable one, strengthened by the variable fixation performed in the latter and by cut generation, in a Benders decomposition fashion. The improved algorithm was capable of solving 143 out of the 150 literature instances, being 8 solved for the first time.

Keywords: Operations Research, Scheduling, Integer Programming, Column Generation, Polyhedral Combinatorics, Branch-Cut-and-Price.

RESUMO

Esse trabalho propõe melhorias ao algoritmo de Branch-Cut-and-Price proposto por Pessoa et al (2010) para resolver o problema de escalonamento de tarefas em máquinas paralelas minimizando o atraso ponderado, além de uma revisão de trabalhos passados. Uma nova família de cortes foi proposta para fortalecer a formulação usada, com um eficiente algoritmo genético de separação. O principal resultado desse trabalho foi a projeção de uma formulação extensa para uma formulação mais tratável, fortalecida pela fixação de variáveis realizada e por geração de cortes, como feito numa Decomposição de Benders. O algoritmo aprimorado foi capaz de resolver 143 das 150 instâncias da literatura, sendo 8 resolvidas pela primeira vez.

Palavras-chave: Pesquisa Operacional, Escalonamento de Tarefas, Programação Inteira, Geração de Colunas, Combinatória Poliédrica, Branch-Cut-and-Price.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ATIF	Arc-Time-Indexed Formulation
BCP	Branch-Cut-and-Price
BCP-PWMT	Branch-Cut-and-Price - Parallel Machines Weighted Tardiness
BCP-PWMT-OTI	Branch Cut and Price - Parallel Machines Weighted Tardiness - Overload Time Indexed
ECC	Extended Capacity Cut
LPP	Linear Programming Problem
RHECC	Rounded Homogeneous Extended Capacity Cut
TIF	Time-Indexed Formulation

1 INTRODUCTION

1.1 MOTIVATION

The meaning of the word Scheduling is to organize, distribute. In this work's context, Job Scheduling is the process of assigning and sequencing jobs on machines, in the best possible way. However, the resources and tasks considered by the models, can be other than machines to process jobs, such as computer processors to process programs, airport runways to process landings and take-offs, construction crews to process construction phases, and so on. Scheduling is a well studied class of combinatorial problems, that emerges on diverse real world situations, specially in the manufacturing of goods, where the sequencing of hundreds of tasks may be necessary. An optimal sequence in such context may translate to the savings of a significant share of resources, when compared to the possibly practiced sub optimal solution. There are many variants of the scheduling problem, with varying machine environments, jobs characteristics and objectives to optimize.

This work focus on the parallel machine scheduling problem minimizing the weighted tardiness, where the environment is composed of two or more identical machines, each job has a particular processing time, due date and weight, and the objective is to minimize the total weighted tardiness, where the tardiness is defined as how much each job was past its due date, being zero if completed on, or prior to, the due date. For shortness we refer to this problem as the weighted tardiness parallel machine problem.

1.2 THEME DEFINITION

Combinatorial optimization problems, such as the job scheduling, may be solved by heuristic or exact methods. On the first case, the solution time is modest, but the solution quality is not guaranteed, when comparing to an optimal solution. In the second case, the found solution

is necessarily optimal, but the computational time may grow intractably large. In both cases, it is desired that academic research be carried in order to provide estimates on the solution quality for heuristics and on the computational time, in the case of exact methods. This dissertation is restricted to the study of exact methods for combinatorial optimization problems.

For the problem considered in this dissertation, the best method found in the literature is an Branch-Cut-and-Price (BCP) algorithm proposed by Pessoa et al (2010). In the course of this document, this algorithm is referred to as BCP-PMWT, where PMWT stands for parallel machine with weighted tardiness. The focus of this study is to develop additional techniques to be incorporated to the BCP-PMWT, in order to improved its performance.

1.3 ACHIEVED RESULTS

The BCP algorithm is a variant of the Branch-and-Bound algorithm, where the relaxation optimized to prune the tree is solved using column and cut generation combined. The use of these techniques allows solving very complex relaxations, in terms of the number of variables and constraints, in reduced computational time. For example, the relaxation solved by the BCP-PMWT uses a formulation that defines a binary decision variable for each time period t and job pair (i, j) scheduled consecutively on the same machine. For that, this relaxation is referred to as arc-time-indexed, where each arc refers to the pair of jobs (i, j). The number of variables and constraints generated by this formulation grows rapidly with the size of the instance. To contain this increase, this formulation is substituted for an equivalent, called master, with only one constraint per job, but with an exponential number of variables, where each variable now corresponds to a complete sequence of jobs processed on a single machine. The column generation method consists in adding these variables as needed in the course of the relaxation resolution. Beside that, the arc-time-indexed formulation is strengthened by additional constraints that are satisfied by any feasible solution, but prohibits certain relaxed solutions. These constraints, called cuts, are exponential in number, being also added by demand using a separation algorithm. By solving the relaxation by column generation, all cuts to be inserted in the arc-time-indexed formulation are translated to the master formulation, by a systematic procedure. Another technique employed in the BCP-PMWT algorithm is the fixation of variables by reduced cost. In this case, a procedure is carried out to demonstrate by the reduced costs in the master formulation that certain arc-time-indexed variables can be removed from the formulation without loss of the optimal solution. For each fixed variable, this procedure proves that any feasible solution where it assumes a positive value have a bigger cost than the best known

solution. The elimination of arc-time-indexed variable have the effect of accelerating the column generation procedure, besides improving the relaxation quality. At last, when the number of remaining variables, is below a threshold,the BCP-PMWT method transfer the remaining of the optimization to a commercial solver, using the reduced arc-time-indexed formulation.

In this dissertation, the following techniques were developed in order to improve the BCP-PMWT.

- A new family of cuts was derived, called Overload Elimination Cuts (OEC), for the arc-time-indexed formulation, in a way to improve even more the quality of the relaxation used by the BCP-PMWT. As the number of constraints potentially generated by this family is exponential, they are also added by demand.

- To insert the OECs translated to the master formulation during the BCP-PMWT execution, a genetic algorithm was developed for the cut separation.

- When the BCP-PMWT transfer the remaining optimization to the commercial solver, it was proposed to use job-time indexed, or simply time-indexed for shortness, formulations instead of arc-time-indexed. Experiments using four alternative formulations are presented. The time-indexed formulations are more compact, allowing for the commercial solver to expand the Branch-and-Bound tree much faster. However, the relaxations based on these formulation are weaker, reducing the algorithm ability to prune the tree.

- To strengthen the time-indexed formulation used in the previous item case, cuts derived from the arc-time-indexed formulation are projected over the polyhedron defined by the time-indexed formulation.

The improved method is called as BCP-PMWT-OTI, where OTI stands for overload, referring to the new family of cuts and time-indexed, due to the formulation used to finish the optimization.

As a result of the inclusion of these techniques, the BCP-PMWT-OTI was capable of solving 143 out of the 150 literature instances, being 8 solved for the first time. Besides that, for the instances already solved by the BCP-PMWT, the computational time of the new method was inferior in 45 cases.

1.4 OUTLINE

Chapter 2 presents some theoretical foundations for the development of the proposed improvements. Next, chapter 3 presents a brief literature review of exact methods related to the weighted tardiness scheduling problem, followed by a review of the BCP-PMWT and other existing techniques that are applied to the BCP-PMWT-OTI. Chapter 4 details the novel improvements that also composes the BCP-PMWT-OTI. Following, chapter 5 details how the computational tests were performed and presents its results. Finally, chapter 6 draws some conclusions and proposes aspects that could be pursued in future works.

2 THEORETIC FOUNDATIONS

2.1 SCHEDULING

In this section, we present the standard framework and notation used on the scheduling literature, and three different examples of scheduling problems.

2.1.1 FRAMEWORK AND NOTATION

Scheduling problems consists of a set of jobs $J = \{1, \ldots, n\}$ to be processed on a set of machines $M = \{1, \ldots, m\}$. The index used for jobs is usually j and the index used for machines is usually i. The number of jobs to be processed is n and the number of machines available to process them is m. Each job can be processed for at most a single machine at a time, and each machine can process at most a single job at a time.

As time is discretized, a period of time t refers to the period spawning from instant $t-1$ to t. We usually say that a job started or ended at time t meaning the instant of time t, and that a machine was busy at a time t meaning the time period t. So, a job of processing time one, if starting at time zero, will finish at time one, and its machine was busy during period of time one, or simply, at time one.

Jobs can have many information to characterize them, such as:

- A processing time p_j, or p_{ij} when it depends on the machine processing it.
- A release date r_j, when job j can start to be processed.
- A due date d_j, when job j should be completed.
- A deadline $\bar{d}_j$, when job j must be completed.
- An earliness penalty weight h_j, also called holding costs.

- A tardiness penalty weight w_j.
- A cost function $f_j(t)$ based on time of completion.

Graham et al (1979) established the 3-field notation,

$$\alpha \mid \beta \mid \gamma, \tag{2.1}$$

where α describes the machine environment, β represents the set of tasks characteristics, and γ is the objective to be minimized. We here present a brief overview of it. A more comprehensive review can be found in Pinedo (2012).

The field α can assume one of the following values:

Single machine (1) The simplest environment, consisting of a single machine to process all jobs.

Identical machines in parallel (P_m) Each job j can be processed by any of the m identical machines, unless restricted by M_j, as explained latter.

Machines in parallel with different speeds (Q_m) Like the previous environment, with speed v_i defined for each machine so that the processing time of job j on machine i equals p_j/v_i.

Unrelated machines in parallel (R_m) Like the P_m, with an specific processing time $p_{i,j}$ defined for each job-machine pair.

Flow shop (F_m) Every job has m production stages to be processed always in the same sequence. Each of m available machines can execute only one stage of production.

Job shop (J_m) Like the flow shop, but different jobs can have different production sequences. Jobs not necessarily need to be processed once at each machine.

Open shop (O_m) Each job have a set of operations, that may or may not involve all machines, to be performed following in any sequence.

The field β can be empty or assume one or more of the following values:

Release dates (r_j) Each job has a release date and cannot start to be processed before it.

Preemptions ($prmp$) An operation can stop and be resumed later.

Precedence constraints ($prec$) A set of job pairs (j, k) can be defined so that no job k must be preceded by its pair k.

Sequence dependent setup times ($s_{j,k}$) For each job pair (j, k), a setup time is defined, meaning that, on the same machine, job k can only start being processed $s_{j,k}$ periods of time after job j terminates.

Machine eligibility restrictions (M_j) For multi-machine environments, each job j has a set of machines M_j where it can be processed.

Field γ assumes the objective function formula. The three main penalty functions $f_j(C_j)$, based on the completion time C_j of job j, are:

Lateness (L_j) $L_j = C_j - d_j$
Tardiness (T_j) $T_j = \max\{0,\, C_j - d_j\}$
Unit Penalty (U_j) $U_j = 1$ if $C_j > d_j$ and 0 otherwise

The notation of our focused variant is then $P||\Sigma w_j T_j$, since its machine environment is the identical machines in parallel, its objective function is to minimize the weighted tardiness, and there are no special constraints or characteristics.

2.1.2 SCHEDULE REPRESENTATION

A schedule is fully described by listing for each job, the intervals it is processed and on which machine. In $P||\Sigma w_j T_j$, since a job can be processed on any machine and there is no preemption, the start or completion time of each job suffices to describe the schedule.

One way to graphically represent a schedule is the Gantt chart. The original Gantt chart, as presented in Gantt (1910), represents when each resource is being used, using the x axis to represent time and the y axis to represent resources, with no labels on the bars. Figure 2.1 shows a Gantt chart representing when two machines are busy, while executing 5 jobs of processing times {4, 5, 4, 5 e 2}.

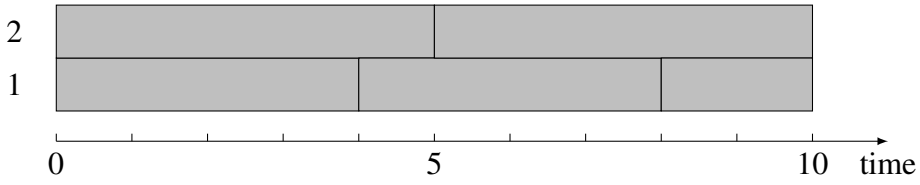


Figure 2.1: Traditional Gantt Chart

Note that on this representation, there is no information about on which machine a job is being processed. For the $P||\Sigma w_j T_j$, this is acceptable. For different schedule problems, a better representation of the same schedule perhaps would be the one of figure 2.2. In it, the machines are represented on the y axis and the jobs are labeled on their respective bars.

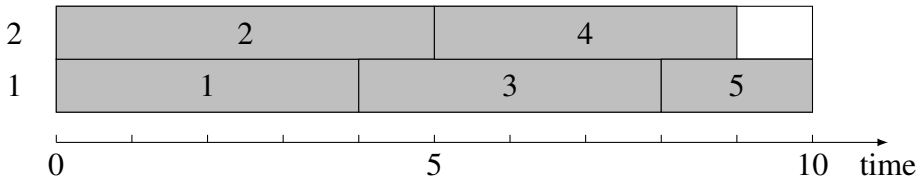


Figure 2.2: Adapted Gantt Chart for Machine Representation

We will refer to any Gantt-like chart simply as Gantt charts throughout the text.

2.1.3 THREE EXAMPLES OF SCHEDULING PROBLEMS

The first example we present is for the single machine with release dates problem minimizing the total , weighted completion time ($1|r_j|\sum w_j C_j$). The problem data is presented in the table below.

Job(j)	Processing time (p_j)	Release Date (r_j)	Weight (w_j)
1	4	5	1
2	1	0	2
3	4	4	1
4	4	3	3
5	5	3	3

Table 2.1: First Example Instance Data

A feasible solution for the problem, with cost 109, is listed in the table below and represented in figure 2.3.

Job(j)	Start	Completion (C_j)
1	7	11
2	0	1
3	11	15
4	3	7
5	15	20

Table 2.2: First Example Instance Solution

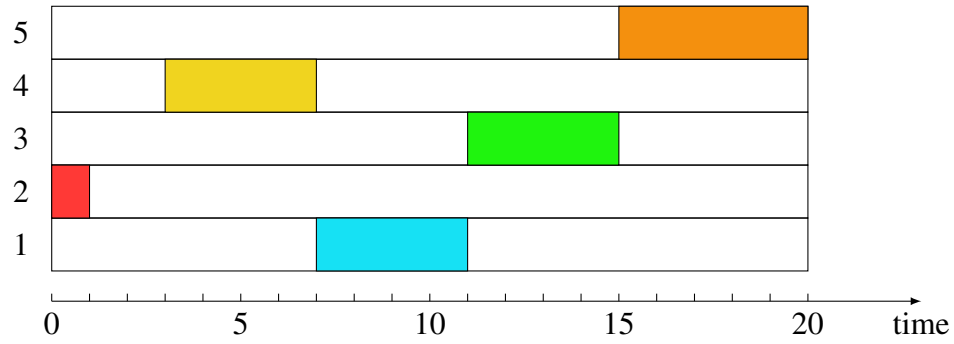


Figure 2.3: Gantt Chart – First Example Instance Solution

The second example we present is the makespan minimization on parallel machines with sequence dependent setup times problem ($P|s_{i,k}|C_{\max}$). The problem data is presented in the table below, except for the number of machines $m = 2$.

Job(j)	Processing time (p_j)
1	5
2	9
3	4
4	4
5	6

Table 2.3: Second Example Instance Data

j \ k	1	2	3	4	5
1		1	1	1	0
2	0		2	3	2
3	0	1		0	1
4	3	1	0		2
5	1	0	3	2	

Table 2.4: Second Example Instance Setup Times

A feasible solution for the problem, with cost 16, is listed in the table below and represented in figure 2.4.

Job(k)	Start	Preceding Job(j)	Completion (C_j)
1	0	-	5
2	5	1	15
3	0	-	4
4	4	3	8
5	8	4	16

Table 2.5: First Example Instance Solution

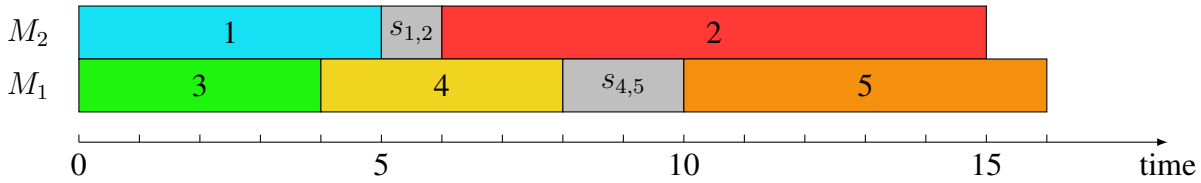


Figure 2.4: Gantt Chart – Second Example Instance Solution

The third example we present is the makespan minimization on the flow shop environment ($F||C_{\max}$). Three machines composes the flow shop, namely M_1, M_2 and M_3. Each job has to be processed following the sequence $M_1 - M_2 - M_3$. The processing time p_{ij} of job j in machine i is given by the table below.

Job(j)	p_{1j}	p_{2j}	p_{3j}
1	3	1	3
2	2	1	1
3	2	2	1

Table 2.6: Third Example Instance Data

A feasible solution for the problem, with cost 10, is represented in figure 2.5.

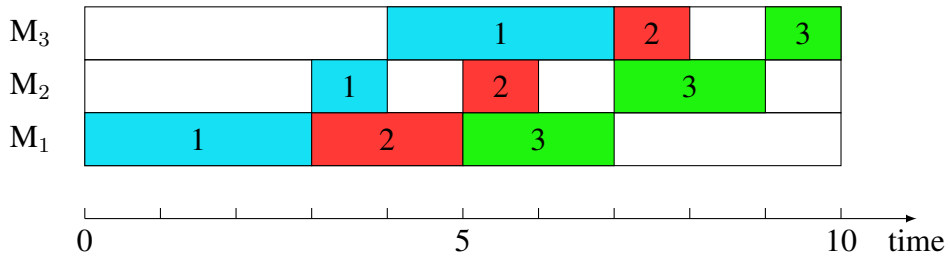


Figure 2.5: Gantt Chart – Third Example Instance Solution

2.2 METAHEURISTICS

Heuristics are optimization methods that seek for good solutions, not guaranteeing its optimality. According to Glover and Kochenberger (2003), metaheuristics are heuristics that incorporates higher level procedures for escaping from local optima in order to better search the solution space.

In this section, we present the general guidelines of one of the most popular metaheuristics, the Genetic Algorithm.

2.2.1 GENETIC ALGORITHM

According to Goldberg (1989), genetic algorithms perform its search for better solutions based on the mechanics of natural selection and genetics. They are based on operators to combine solutions into new ones, to randomize a few characteristics of the solutions and to select the best solutions.

Usually, a local search is also performed, which results in a memetic algorithm. But it is often referred to simply as genetic or evolutionary algorithm. The literature borrows some of its terms from the Genetics field, some of them are summarized in table 2.7.

Table 2.7: Glossary of a few Genetic Algorithm's terms

Term	Meaning
Fitness	indicator of how good a solution is, relative to the solutions set
Chromosome	string, a codified solution
Crossover	combination of 2 solutions
Gene	feature, character
Alelle	feature value
Individual	solution
Mutation	random modification of one or more features of a solution
Population	set of solutions

The classical codification of a chromosome is binary, with each bit representing a feature. The traditional operators of a genetic algorithm are the crossover, mutation and selection, they are called operators since they are applied to single individuals or a population, to change them in some way.

A crossover operator is applied to a population, and consists of selecting 2 individuals, called parents, and generating a new individual, called child, by replicating parts of the parent chromosomes. Figure 2.6 illustrates a crossover operation, in which the child is formed by the first half of its father chromosome, and the second half of its mother chromosome. Crossover operators can vary in the way the parents are selected, and how they are combined to form the child.

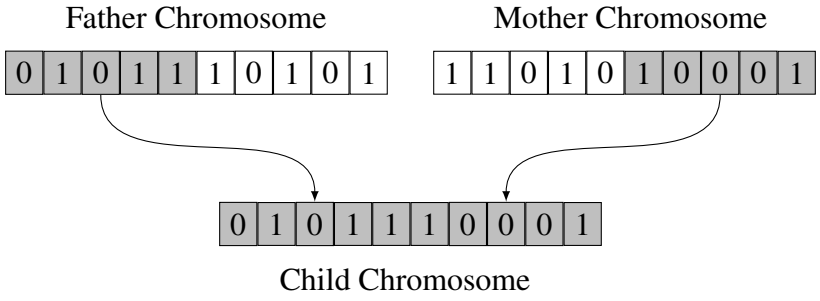


Figure 2.6: Classic Crossover

A mutation operator is applied to an individual, and consists on randomly switching a set of bits, that is, changing bits from 0 to 1, or vice versa. Figure (2.7) illustrates a mutation operation in which the eighth bit was switched. Mutation operators can vary in the size of the set of bits switched, and in how the set is chosen.

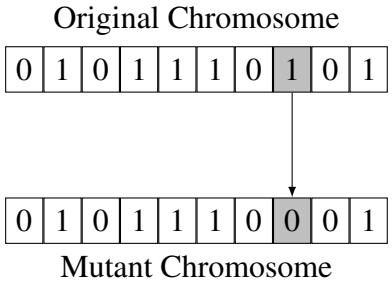


Figure 2.7: Classic Mutation

A selection operator is applied to a population, and consists of eliminating a number of the worst individuals, based on a fitness function.

The algorithm 1 represent the steps of a simple genetic algorithm. This model makes use of binary codification and generic crossover, mutation and selection operators.

Algorithm 1: Classic Genetic Algorithm

Input: $nPopulation, rateCrossover, probMutation$
Output: Best found solution
$Population \leftarrow generateInitialPopulation(nPopulation)$
while $not\ stopCriteria(\ )$ **do**
  for $i \leftarrow 1$ **to** $nPopulation \times rateCrossover$ **do**
    $father \leftarrow random\ individual\ from\ Population$
    $mother \leftarrow random\ individual\ from\ (Population - father)$
    $child \leftarrow crossover(father, mother)$
    $child \leftarrow mutation(child)\ (probMutation\%\ chance\ of\ ocurring)$
    if $Fitness(child) > Fitness(bestIndividual)$ **then**
      $bestIndividual \leftarrow child$
    end
    $Population \leftarrow Population + child$
  end
  $Population \leftarrow Selection(Population)$
end
return $bestIndividual$

Each iteration of the algorithm is called a generation, since a portion of the population dies and the remaining portion will reproduce. The algorithm is very flexible, allowing for any sequence and frequency of operations. Also, custom codification and operators more suitable to any class of problem can be developed.

According to Whitley (1994), genetic algorithms belongs to the class of methods known as "weak methods", because it makes few assumptions about the problems it solves. From the practical standpoint, this is good because the implementation of a genetic algorithm will then require little knowledge about the problem at hand.

2.3 COLUMN GENERATION

Column generation is a method for optimizing linear programming problems (LPPS) with a huge number of variables. Consider the LPP (2.2) below.

$$\begin{aligned} \text{Minimize} \quad & c^{\intercal}x \\ \text{Subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.2}$$

Each variable has a correspondent column in the matrix A. In general lines, the algorithm consists of solving a restricted LPP, with only a subset of matrix A columns. Then, it is verified if all columns not in A have non-negative reduced costs. If so, the algorithm terminates and the

restricted LPP solution is optimal, if not, a negative reduced cost column is included in A and the algorithm iterates. The term column generation comes from this iterative insertion of columns.

Since the number of columns is very large, the computation of the reduced cost for every column is not feasible in practice, as it would be too computationally expensive. The applicability of the method relies in the pricing sub-problem (2.3) having a special structure, for it to be solved efficiently.

$$\min_{i \in \text{set of columns}} \quad \bar{c}_i \tag{2.3}$$

Some problems are naturally fit to the use of column generation, such as the Cutting Stock. One way to use column generation is by applying Dantzig-Wolfe Decomposition to problems presenting a special structure.

2.3.1 DANTZIG-WOLFE DECOMPOSITION

The Dantzig-Wolfe Decomposition can be applied to problems presenting the following structure,

$$\begin{aligned} \text{Minimize} \quad & c^{\intercal} x \\ \text{Subject to} \quad & Ax \leq b \\ & x \in P. \end{aligned} \tag{2.4}$$

Let x^j, $j \in J$ and θ^k, $k \in K$, be respectively the sets of extreme points and extreme rays of the polyhedron P. Any element $x \in P$ can then be represented in the form

$$x = \sum_{j \in J} x^j \lambda_j + \sum_{k \in K} \theta^k w_k \tag{2.5}$$

where $\sum_{j \in J} \lambda_j = 1$, $\lambda_j \geq 0$, $j \in J$ and $w_k \geq 0$, $k \in K$.

Replacing x in (2.4), we obtain the so-called Dantig-Wolfe Master ($\mathcal{DWM}$) problem

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j\in J}(c^\intercal x^j)\lambda_j + \sum_{k\in K}(c^\intercal \theta^k)w_k \\
\text{Subject to} \quad & \sum_{j\in J}(Ax^j)\lambda_j + \sum_{k\in K}(A\theta^k)w_k \leq b \\
& \sum_{j\in J}\lambda_j = 1 \\
& \lambda_j \geq 0 \qquad\qquad (j \in J) \\
& \theta_k \geq 0 \qquad\qquad (k \in K).
\end{aligned} \tag{2.6}$$

Each column of the $\mathcal{DWM}$ problem correspond to one extreme point or extreme ray of P. In order to solve it by column generation, we restrict it to $m+1$ columns, forming the basis, solve it, and check if there are any column(s) that have negative reduced costs to be included. This restricted version of the $\mathcal{DWM}$ problem is intuitively referred to as the restricted master problem.

The reduced cost of a variable λ_j is given by

$$\bar{c}_j = (c^\intercal - \pi^\intercal A)x^j - \alpha, \tag{2.7}$$

where (π, α) is the dual solution of the Dantig-Wolfe master problem. And the reduced cost of variable w_k is given by

$$\bar{c}_k = (c^\intercal - \pi^\intercal A)w^k. \tag{2.8}$$

The main idea of the decomposition algorithm is that, to find a variable with negative reduced cost, we don't need to check every variable λ_j and w_k. Since P is a polyhedron, the variable with lowest reduced cost can be found via the LPP

$$\min_{x\in P}[(c^\intercal - \pi^\intercal A)x - \alpha]. \tag{2.9}$$

If the found minimum reduced cost is negative, the LPP provide us with an extreme point, or an extreme ray of P, for which we include its corresponding column in the restricted master problem, repeating the procedure until no variable's reduced cost remains negative.

It is not uncommon for the CG algorithm to suffer from slow convergence, in order to alleviate it, several stabilization techniques were proposed. Pessoa et al (2010) used in its BCP the Weighted Dantzig-Wolfe decomposition, introduced by Wentges (1997).

2.4 BENDERS DECOMPOSITION

Like the column generation approach, when a MILP have a large number of constraints, the Benders Decomposition is a procedure to generate constraints as needed. Consider the following MILP,

$$\begin{aligned} \text{Minimize} \quad & c^\intercal x + g^\intercal y && \text{(2.10a)} \\ \text{Subject to} \quad & Ax \geq b && \text{(2.10b)} \\ & Dx + Ey \geq f && \text{(2.10c)} \\ & x, y \geq 0. && \text{(2.10d)} \end{aligned}$$

By relaxing constraint (2.10c), replacing $g^\intercal y$ by z, and solving (2.10), we obtain the solution $\tilde{x}$. But $\tilde{x}$ can only be a solution of (2.10) if, and only if, $\exists\, y \mid D\tilde{x} + Ey \geq f$. Consider the LPP (2.11), and its dual (2.12).

$$\begin{aligned} \text{Minimize} \quad & 0 && \text{(2.11a)} \\ \text{Subject to} \quad & Ey \geq f - D\tilde{x} && \text{(2.11b)} \\ & y \geq 0 && \text{(2.11c)} \end{aligned}$$

$$\begin{aligned} \text{Maximize} \quad & \pi^\intercal(f - D\tilde{x}) && \text{(2.12a)} \\ \text{Subject to} \quad & \pi^\intercal E \leq 0 && \text{(2.12b)} \\ & \pi \geqslant 0 && \text{(2.12c)} \end{aligned}$$

By the weak duality theorem, $\pi^\intercal(f - D\tilde{x}) \leq 0$. If $\nexists\, y \mid Dx + Ey \geq f$, then (2.11) is infeasible, (2.12) is unbounded, and $\pi^\intercal(f - D\tilde{x}) \leq 0$ is violated by an extreme ray $\bar{\pi}$ of (2.12). A so called Bender's feasibility cut $\bar{\pi}^\intercal(f - Dx) \leq 0$ is then generated and added to (2.10). This procedure iterates until $\tilde{x}$ is feasible for (2.10) (with some y). Let Π be the set of all $\bar{\pi}$ associated

to the added feasibility cuts. The current reduced master problem is then

$$\begin{aligned}
\text{Minimize} \quad & c^\intercal x + z && \text{(2.13a)}\\
\text{Subject to} \quad & Ax \geq b && \text{(2.13b)}\\
& \pi^\intercal x \geq \pi^\intercal f \quad (\forall\, \bar{\pi} \in \Pi) && \text{(2.13c)}\\
& x, y \geq 0. && \text{(2.13d)}
\end{aligned}$$

Now, consider the LPP (2.14), and its dual (2.15).

$$\begin{aligned}
\text{Minimize} \quad & g^\intercal y && \text{(2.14a)}\\
\text{Subject to} \quad & Ey \geq f - D\tilde{x} && \text{(2.14b)}\\
& y \geq 0 && \text{(2.14c)}
\end{aligned}$$

$$\begin{aligned}
\text{Maximize} \quad & \pi^\intercal (f - D\tilde{x}) && \text{(2.15a)}\\
\text{Subject to} \quad & \pi^\intercal E \leq g^\intercal && \text{(2.15b)}\\
& \pi \geqslant 0 && \text{(2.15c)}
\end{aligned}$$

Also by the weak duality theorem, $\pi^\intercal(f - D\tilde{x}) \leq g^\intercal y$. If $\nexists\, y \mid g^\intercal y \leq z,\, Dx + Ey \geq f$, then $\tilde{\pi}^\intercal(f - D\tilde{x}) \leq \tilde{z}$ is violated, where $\tilde{\pi}$ is the optimal solution to (2.15). A so called Bender's optimality cut $\tilde{\pi}^\intercal(f - Dx) \leq z$ is then generated and added to (2.13). This procedure iterates until $\tilde{z} \geq g^\intercal y$ for some feasible y. At this point the pair ($\tilde{x}$, $\tilde{z}$) represents an optimal solution to the original problem whose associated value of $\tilde{y}$ can be found by solving (2.14).

3 PREVIOUS WORKS

This chapter presents a brief literature review on the weighted tardiness parallel machine problem, an overview of the BCP-PMWT algorithm, and techniques from previous works that were incorporated into BCP-PMWT-OTI.

The outline of this chapter is as follows.

Section 3.2 presents a literature review of exact algorithms for related problems. Section 3.3 presents the time-indexed formulation and a small example of the $P||\sum w_j T_j$ problem. Section 3.4 presents the BCP-PMWT algorithm. Section 3.5 presents the triangle clique cuts, with a simple and exact separation procedure.

3.1 THE PROBLEM

Let J be a set of n jobs to be processed in a set M of m identical parallel machines. Each task has a processing time p_j and is associated with a cost $f_j(C_j)$, based on its completion time C_j. Each machine can only process one job at a time and each job has to be processed by a single machine, with no pauses or preemption. The goal is to find a schedule that minimizes the sum of individual costs. Here, we assume that $f_j(C_j) = w_j T_j$, where $w_j$is the weight of job j and T_j is its tardiness in the schedule, calculated as $\max 0, C_j - d_j$ where d_j is the job due date.

3.2 LITERATURE REVIEW

In this section, we present a review of exact algorithms for scheduling problems closely related to $P||\sum f_j(C_j)$ found in the literature.

Figure 3.1 shows the considered problems, each box representing a variant, with a list of works that presents an exact algorithm or a review of exact algorithms for it. Each arrow connects one variant to its immediately more specialized case. It is important to remember

that, for any combinatorial optimization problem, a general algorithm can be applied to special cases but not vice versa. For example, an algorithm for $P||\sum f_j(C_j)$ can be used to solve $1||\sum f_j(C_j)$, but an algorithm for $1||\sum w_jT_j$ can't be used to solve $1|s_{i,j}|\sum w_jT_j$.

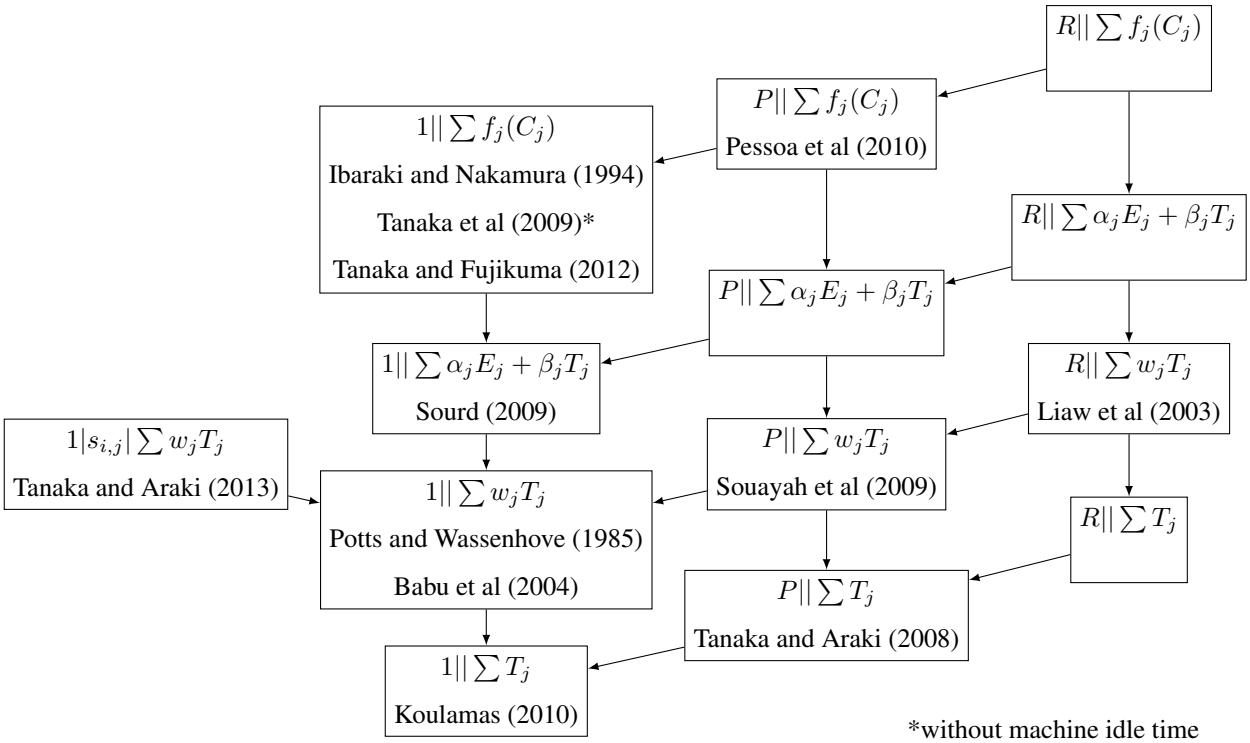


Figure 3.1: Hierarchy of related Scheduling Problems

Potts and Wassenhove (1985) proposed a branch-and-bound algorithm for the single machine total weighted tardiness problem ($1||\sum w_jT_j$) capable of solving instances of up to 40 jobs. Its lower bound was based on Lagrangian relaxation, solved by a multiplier adjustment method. A review of 6 exact algorithms for $1||\sum w_jT_j$ was presented by Abdul-Razaq et al (1990), with computational tests that highlighted the superiority of Potts and Wassenhove (1985) algorithm at the time.

Tanaka et al (2009) presented an efficient algorithm for solving the $1||\sum f_j(C_j)$ when machine idle times are not permitted. Their algorithm is based on the work of Ibaraki and Nakamura (1994), which used the Successive Sublimation Dynamic Programming (SSDP) to solve $1||\sum f_j(C_j)$.

The SSDP, proposed by Ibaraki (1987), is based on a set of dynamic programming relaxations for a problem. It consists of solving a sequence of dynamic programming relaxations, stepping from the weaker to the stronger, tightening the gap in the process. Also, when stepping from one model to the next, states that can be proved not to be part of an optimal solution are

eliminated to alleviate memory use and speed up computations.

The dynamic programming relaxations can be derived in many different ways. The ones used by Ibaraki and Nakamura (1994) were the state-space relaxations presented by Abdul-Razaq and Potts (1988).

It is interesting to note that the SSDP method uses dynamic programming in a different way than most exact algorithms, which usually employ the Lagrangian bounds on a Branch and Bound framework. As Tanaka et al (2009) stated, their algorithm is based fully on dynamic programming, and they tried a Branch and Bound algorithm first (Tanaka and Araki, 2006), not obtaining the same efficiency of the SSDP method.

Tanaka and Fujikuma (2012) extended Tanaka et al (2009) to permit machine idle times. Following the same SSDP framework, Tanaka and Araki (2013) presented an algorithm for the single machine total weighted tardiness problem with sequence-dependent setup times, $1|s_{i,j}|\sum w_j T_j$.

Pessoa et al (2010) was the first to explicitly present an integer programming model for scheduling problems using arc-time-indexed variables. The arc-time-indexed formulation (ATIF), uses variables $x_{i,j}^t$ to indicate job i finishes and job j starts at time t, on the same machine. These variables are an extension of the time-indexed variables of Dyer and Wolsey (1990), y_j^t, which indicates job j finishes at time t.

3.3 THE TIME-INDEXED FORMULATION

Dyer and Wolsey (1990) proposed a MIP formulation for scheduling problems using one binary variable for each job-time to indicate that a job j finishes at time t. A time period t is defined as spawning from instant $t-1$ to instant t. This formulation is referred to as the time-indexed formulation (TIF). For the single machine case, given p_j as the processing time of job j, $f_j(t)$ as the cost of job j finishing on instant t, and T as the last time period an optimal schedule may finish, the TIF is defined as

$$\text{Minimize} \quad \sum_{j\in J}\sum_{t=p_j}^{T} f_j(t)\, y_j^t \tag{3.1a}$$

$$\text{Subject to} \quad \sum_{t=p_j}^{T} y_j^t = 1 \qquad (j \in J) \tag{3.1b}$$

$$\sum_{j\in J}\sum_{s=\max\{p_j,t\}}^{\min\{t+p_j-1,T\}} y_j^s \leq 1 \qquad (t = 1,\ldots,T) \tag{3.1c}$$

$$y_j^t \in \{0,1\} \qquad (j \in J;\ t = 0,\ldots,T-p_j). \tag{3.1d}$$

Many particularities can be easily incorporated into this formulation, for example release dates (r_j) are modeled by simply eliminating any variable y_j^t with $t - p_j < r_j$. The multi-machine environment is modeled by replacing the right side of (3.1c) by the number of available machines, m. For the weighted tardiness parallel machine problem, $f_j(t) = \max\{0, d_j - t\}$.

In order to analyse the linear relaxation of (3.1), we adapt the Gantt chart for the visualization of these "fractional schedules", using the height of each bar to represent how much of a job is being executed.

Figures 3.2 and 3.3 show the same 5-job schedule on 2 machines, with job 1 starting 70% at time 0 and 30% at time 2, job 2 starting 30% at time 0 and 70% at time 2, job 3 starting 70% at time 4 and 30% at time 6, job 4 starting 30% at time 5 and 70% at time 7 and job 5 starting fully at time 0. For the case of the second figure, when a bar is too thin to accommodate a label, colors can be used instead of labels.

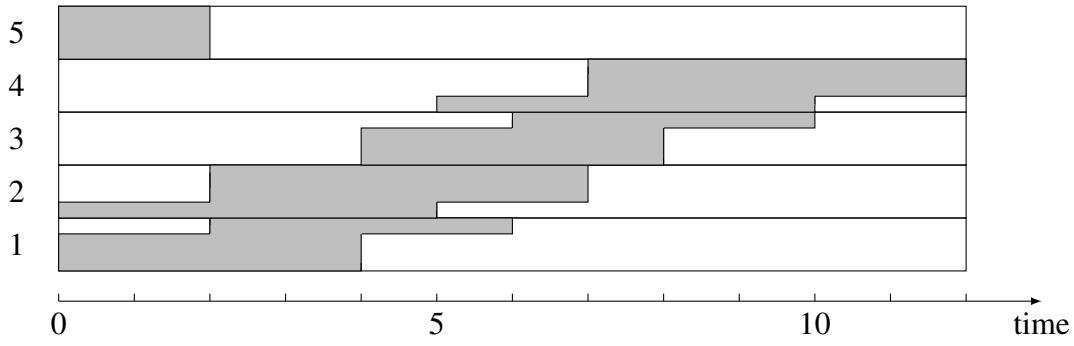


Figure 3.2: Fractional Gantt Chart

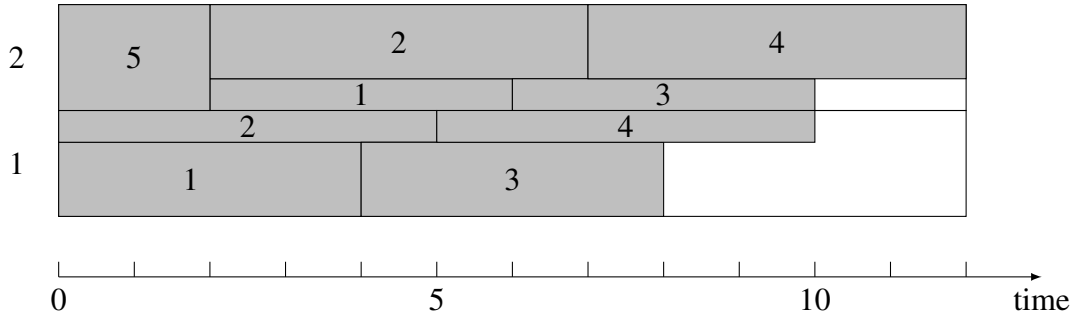


Figure 3.3: Another Fractional Gantt Chart

Now, a small instance of $P||\sum w_j T_j$ with 8 jobs and 2 machines is presented as an example. Its data is in table (3.1).

Job(j)	Processing time (p_j)	Weight (w_j)	Due Date (d_j)
1	6	2	5
2	4	2	10
3	3	3	11
4	6	1	7
5	5	4	10
6	6	2	5
7	4	2	10
8	8	3	11

Table 3.1: Example Instance data

An optimal solution to the linear relaxation of this example is given by the set of variables listed in (3.2), being null every non listed variable.

$$\begin{array}{lllll} x^{0}_{0,1} = 1 & x^{6}_{1,2} = 0,5 & x^{10}_{2,8} = 0,5 & x^{13}_{3,6} = 0,5 & x^{19}_{8,0} = 0,5 \\ x^{0}_{0,5} = 0,5 & x^{6}_{1,5} = 0,5 & x^{10}_{7,3} = 0,5 & x^{16}_{2,4} = 0,5 & x^{22}_{4,0} = 0,5 \\ x^{0}_{0,6} = 0,5 & x^{6}_{6,7} = 0,5 & x^{11}_{5,8} = 0,5 & x^{18}_{8,4} = 0,5 & x^{24}_{4,0} = 0,5 \\ x^{5}_{5,7} = 0,5 & x^{9}_{7,3} = 0,5 & x^{12}_{3,2} = 0,5 & x^{19}_{6,0} = 0,5 & \end{array} \tag{3.2}$$

Figure (3.4) is a Gantt chart for the fractional solution listed in (3.2).

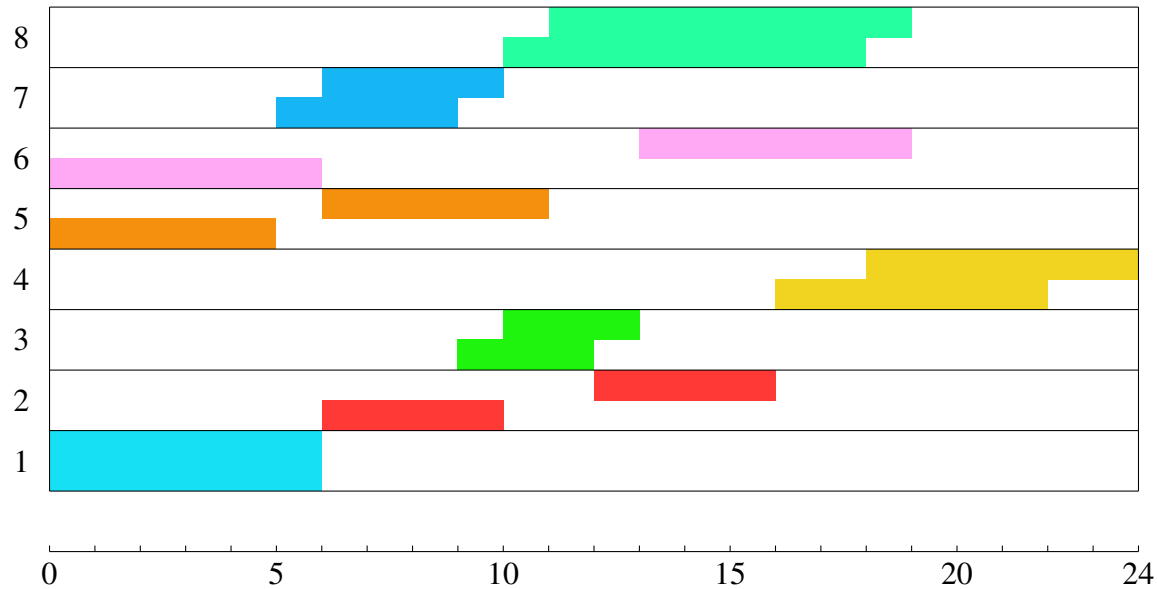


Figure 3.4: Gantt chart of a fractional solution

3.4 THE BCP-PMWT ALGORITHM

Pessoa et al (2010) proposed a Branch-Cut-and-Price algorithm for the weighted tardiness parallel machine problem, here referred to as the BCP-PMWT. The algorithm considers a generic cost function over the completion time, $\sum_{j \in J} f_j(C_j)$, but the reported experiments consider only the parallel machines weighted tardiness problem ($P||\sum w_j T_j$). The formulation used was the arc-time-indexed formulation (ATIF), which uses a binary variable for each job pair (i, j) and time period t to indicate that job i finishes and job j starts at time t, on the same machine. Considering T as the latest time a job can finish in an optimal schedule, defining $J_+ = J \cup \{0\}$ and assuming that $p_0 = 0$, the ATIF follows:

$$\text{Minimize} \quad \sum_{j \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t=p_i}^{T-p_j} f_j\left(t + p_j\right) x_{i,j}^t \tag{3.3a}$$

$$\text{Subject to} \quad \sum_{j \in J_+ \setminus \{j\}} \sum_{t=p_i}^{T-p_j} x_{i,j}^t = 1 \qquad (j \in J) \tag{3.3b}$$

$$\sum_{\substack{j \in J_+ \setminus \{i\}, \\ t-p_j \geq 0}} x_{j,i}^t - \sum_{\substack{j \in J_+ \setminus \{i\}, \\ t+p_i+p_j \leq T}} x_{i,j}^{t+p_i} = 0 \quad (\forall i \in J;\, t = 0, \ldots, T - p_i) \tag{3.3c}$$

$$\sum_{\substack{j \in J_+, \\ t-p_j \geq 0}} x_{j,0}^t - \sum_{\substack{j \in J_+, \\ t+p_j+1 \leq T}} x_{0,j}^{t+1} = 0 \qquad (t = 0, \ldots, T-1) \tag{3.3d}$$

$$\sum_{j \in J_+} x_{0,j}^0 = m \tag{3.3e}$$

$$x_{i,j}^t \in Z^+ \qquad (\forall\, i \in J_+;\, \forall\, j \in J_+ \setminus \{i\};\, t = p_i, \ldots, T - p_j) \tag{3.3f}$$

$$x_{0,0}^t \in Z^+ \qquad (t = 0, \ldots, T-1) \tag{3.3g}$$

For non-decreasing cost function, variables $x_{0,j}^t$ $(t > 0)$ may be removed since there is always an optimal solution with no idle time. In this case, Pessoa et al (2010) proved that $T = \lfloor (\sum_{j=1}^n p_j - p_{max})/m \rfloor + p_{max}$. Moreover, variables $x_{0,j}^0$ indicate that job j start at time 0 and variables $x_{j,0}^t$ indicate that job j is the last job processed on its machine, that is, there is no job succeeding it.

The ATIF solution can be seen as a set of paths in a graph $G = (V, A)$, where $V = \{(0,0), (0,T)\} \cup \{(i,t) \mid i \in J_+,\, t \in \{1, 2 \ldots, T-1\}\}$ and each arc $((i, t-p_i), (j, t))$ correspond to a $x_{i,j}^t$ variable. Figure 3.5 represents solution (3.2) as such graph. This shows that the linear

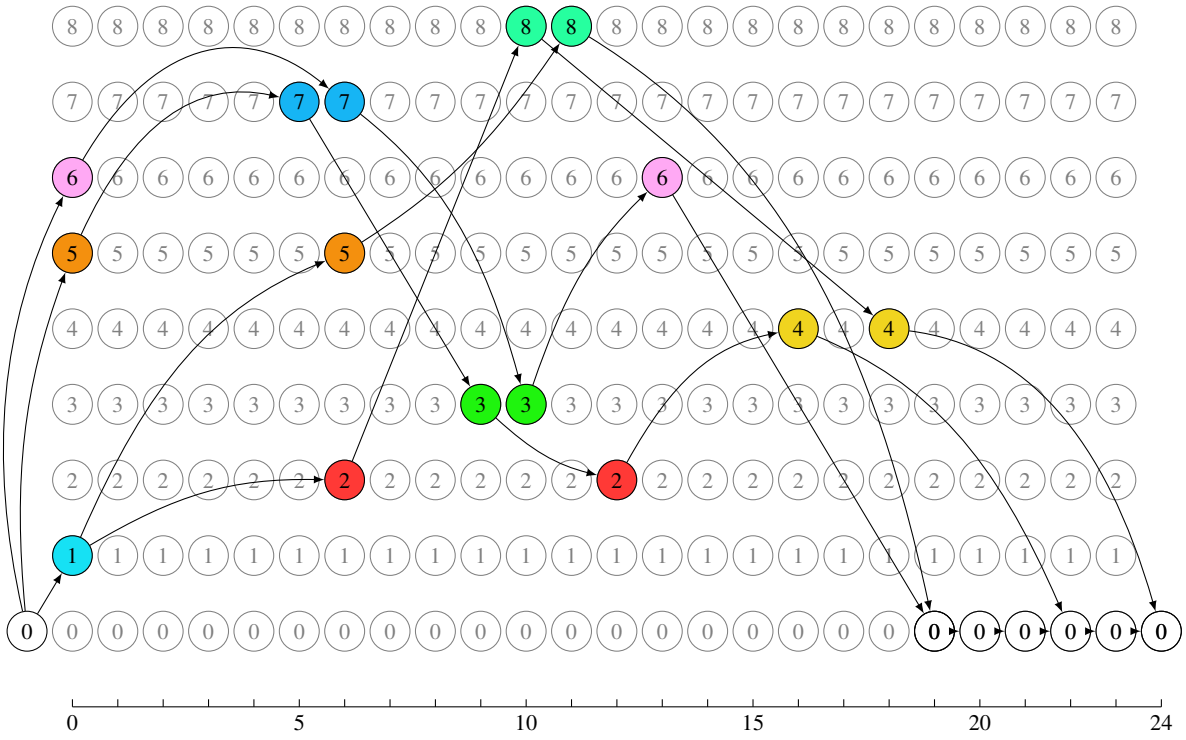


Figure 3.5: Graph of a fractional solution

relaxation of both TIF and ATIF are the same for this instance.

Pessoa et al (2010) proved that the ATIF strictly dominates the TIF and proposed a few simple ways to eliminate some variables to strengthen the formulation. For the ATIF to be just as good as the TIF, it would only need to have the variables $x^t_{j,j}$ included. However, ATIF can be further strengthened in a way that TIF cannot, through the elimination of some variables. Such variable eliminations come from observing that in any schedule, two consecutive jobs on the same machine can be swapped without affecting the rest of the schedule.

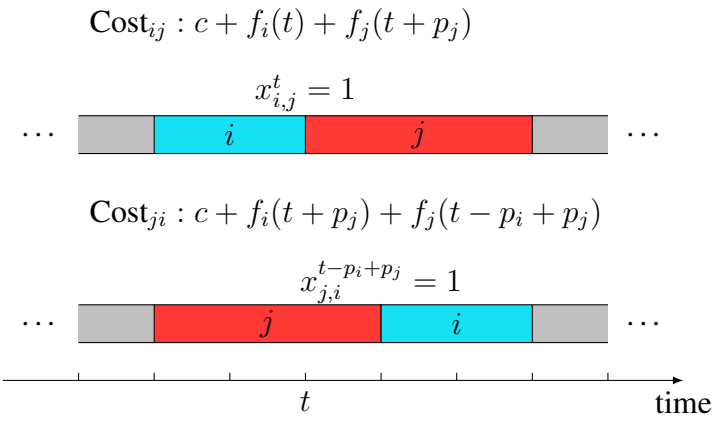


Figure 3.6: Swap of Two Consecutive Jobs

Figure 3.6 shows the effect that the swap have on the overall cost of the schedule, being c the cost of all jobs but i and j. When Cost$_{ij}$ > Cost$_{ji}$, $x^{t+p_i-p_j}_{j,i}$ can be eliminated, and when

$\text{Cost}_{ij} < \text{Cost}_{ji}$, $x^t_{i,j}$ can be eliminated. When both configuration have the same cost, any of the two variables can be eliminated without losing the optimal solution (one must ensure that such eliminations follow a consistent tie-breaking criterion).

For example, the variable $x^{10}_{7,3}$ can be eliminated in solution (3.2) (arc ((7,6),(3,10)) in Figure 3.5) since cost $\text{cost}_{7,3} = \text{cost}_{3,7}$ in this case. This elimination, together with the elimination of $x^{10}_{2,3}$ would improve the linear relaxation of ATIF.

The second elimination approach does not have effect for non decreasing cost functions, which is the case for the weighted tardiness problem. It considers if the last job on a machine can be postponed without increasing the overall cost, creating some machine idle time before it begins. Figure 3.7 shows the effect of the postponement.

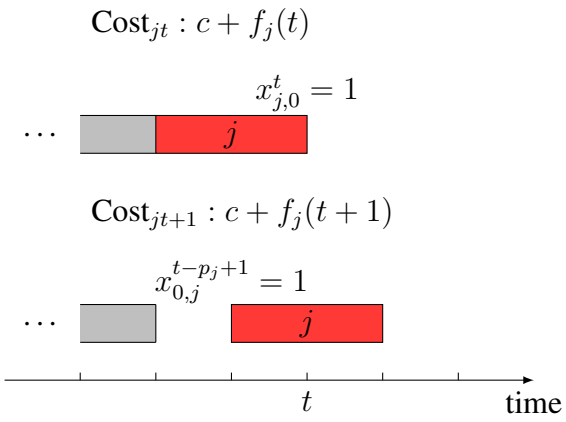


Figure 3.7: Postponement of the Last Job

When $\text{Cost}_{jt} > \text{Cost}_{jt+1}$, $x^t_{j,0}$ can be eliminated, and when $\text{Cost}_{jt} < \text{Cost}_{jt+1}$, $x^{t-p_j+1}_{0,j}$ can be eliminated. When both configuration have the same cost, any of the two variables can be eliminated without losing the optimal solution (consistent tie-breaking is also required here).

Following the idea represented in figure 3.5, we define a pseudo-schedule as a path from $(0, 0)$ to $(0, T)$ in G. In it, four pseudo-schedules compose the fractional solution. Each pseudo-schedule represents the part of the schedule to be processed on one machine.

Let P be the set of all possible pseudo-schedules. For every pseudo-schedule $p \in P$, define a variable λ_p and a set of constants $\{q^{tp}_a | a^t \in A\}$ to indicate if an arc a^t appears in p. Define $f_0(t)$ as zero for any t. The ATIF formulation can then be reformulated as

$$\text{Minimize} \quad \sum_{(i,j)^t \in A} f_j(t + p_j) x_{ij}^t \tag{3.4a}$$

$$\text{Subject to} \quad \sum_{p \in P} q_a^{tp} \lambda_p - x_a^t = 0 \qquad (\forall\, a^t \in A) \tag{3.4b}$$

$$\sum_{(j,i)^t \in A} x_{ji}^t = 1 \qquad (\forall\, i \in J) \tag{3.4c}$$

$$\sum_{(0,j)^0 \in A} x_{0j}^0 = m \tag{3.4d}$$

$$\lambda_p \geq 0 \qquad (\forall\, p \in P) \tag{3.4e}$$

$$x_a^t \in Z_+ \qquad (\forall\, a^t \in A). \tag{3.4f}$$

Formulation (3.4), containing both λ and x variables, is said to be in the explicit format, as defined by Marcus Poggi de Aragão (2003). Using the redundant equations (3.4b) to eliminate x, and relaxing the integrality, the Dantzig-Wolfe Master (DWM) LP is written as:

$$\text{Minimize} \quad \sum_{p \in P} \Big(\sum_{(i,j)^t \in A} q_{ij}^{tp} f_j(t + p_j) \Big) \lambda_p \tag{3.5a}$$

$$\text{Subject to} \quad \sum_{p \in P} \Big(\sum_{(j,i)^t \in A} q_{ji}^{tp} \Big) \lambda_p = 1 \qquad (\forall\, i \in J) \tag{3.5b}$$

$$\sum_{p \in P} \Big(\sum_{(0,j)^0 \in A} q_{0j}^{0p} \Big) \lambda_p = m \tag{3.5c}$$

$$\lambda_p \geq 0 \qquad (\forall\, p \in P) \tag{3.5d}$$

$$\tag{3.5e}$$

Any generic constraint l of format $\sum_{a^t \in A} \alpha_{al}^t x_a^t \geq b_l$ can be included in the DWM by converting the x variables to λ by using the same substitution (3.4b). This is needed for all the cuts used, since they were defined in terms of x.

Since the number of λ variables is exponential on n, in order to solve DWM, these variables are generated on demand. For that, given an optimal solution to (3.5) restricted to a subset of the λ variables, and its dual, the pricing subproblem consists of finding the variable λ_p with minimum reduced cost. If such a reduced cost is negative, then λ_p is added to the restricted master problem and the process continues. Otherwise, the current solution is also optimal for

the complete DWM.

To efficiently compute the optimal λ_p variable, its reduced cost is expressed as the sum of the reduced costs of the arcs of p, which are defined in the following.

Suppose that, at a given instant, there are $r+1$ constraints in the DWM. Let π_0 be the dual variable of constraint 3.5c, π_i be the dual variable of constraints (3.5b) for $i \in J$, and π_l, $n < l \leq r$, be the dual variable of any additional constraint. Being α_{al}^t the coefficient of variable x_a^t in constraint l, the reduced cost of arc $a^t = (i,j)^t$ is defined using the α as:

$$\bar{c}_a^t = f_j(t + p_j) - \sum_{l=0}^{r} \alpha_{al}^t \pi_l. \tag{3.6}$$

3.4.1 EXTENDED CAPACITY CUTS

The BCP-PMWT uses two families of cuts to strengthen the ATIF. The first one, called the Extended Capacity Cuts (ECC) is defined as follows. Let $S \subseteq J$ be a set of jobs, and define $p(S) = \sum_{j \in S} p_j$ as the total processing time of S, $\delta^-(S) = \{(i,j)^t \in A : i \notin S, j \in S\}$ and $\delta^+(S) = \{(i,j)^t \in A : i \in S, j \notin S\}$, being A the set of arcs $(i,j)^t$. Equation (3.7) below is valid.

$$\sum_{a^t \in \delta^+(S)} t x_a^t - \sum_{a^t \in \delta^-(S)} t x_a^t = p(S) \tag{3.7}$$

Uchoa et al (2006) call it the Capacity-Balance Equation over S and define an Extended Capacity Cut (ECC) over S as any inequality valid for $P(S)$, the polyhedron given by the convex hull of the 0-1 solutions of (3.7). Furthermore, an Homogeneous Extended Capacity Cut (HECC) over S is an ECC where all variables with the same time index have the same coefficients.

Dash et al (2010) present a deep study of the polythedron, which they call the Master Equality Polyhedron (MEP), defined as:

$$K(n,q) = conv\left\{(x,y) \in \mathbb{Z}_+^n \times \mathbb{Z}_+^n : \sum_{i=1}^{n} i x_i - \sum_{i=1}^{n} i y_i = q\right\} \tag{3.8}$$

where $n, q \in \mathbb{Z}$ and $n > 0$. In our case, $n = |J|$ and $q = p(S)$.

Another family of cuts, the Rounded Homogeneous Extended Capacity Cuts (RHECC), inequality (3.9), is obtained by multiplying a HECC by a value $r \in [0,1]$ and applying integer rounding.

$$\sum_{a^t \in \delta^+(S)} \lceil rt \rceil x_a^t - \sum_{a^t \in \delta^-(S)} \lfloor rt \rfloor x_a^t \geq \lceil rp(S) \rceil \tag{3.9}$$

Uchoa et al (2006) proved that for any multiplier $r' \in \mathbb{R}$, there exists a dominating multiplier $r = a/b$, where $0 \leq a \leq b \leq T$. The sequence of all irreducible rational numbers $a/b, 0 \leq a \leq b \leq n$ arranged in increasing order is called a Farey Sequence of order n (F_n). Then, any violated RHECC will have $r \in F_T$.

Each cut is defined by the set S and a multiplier $r = a/b$. Figure 3.8 represents one choice of S over a possible solution for a 9 jobs instance. To illustrate how separating RHECC's can be difficult, consider one of the tackled instances, namely *wt100-2m-1*, which has $T = 2.806$, leading to $|F_T| = 2.393.846$ ($\approx 10^{21}$). Combining it to $2^{100} - 1$ ($\approx 10^{30}$) possible choices of an S set over 100 jobs, we have a search space of approximately 10^{51} RHECC's.

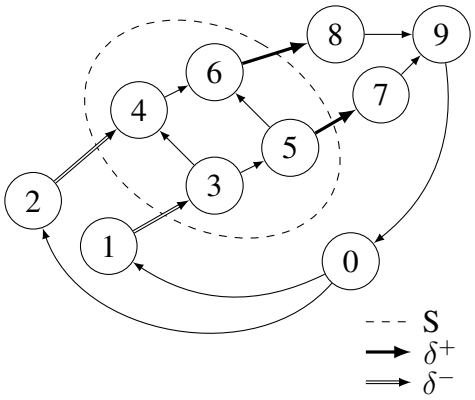


Figure 3.8: RHECC Support Graph

To separate RHECCs, the BCP-PMWT used a simple depth-first search, as explained in Uchoa et al (2006).

As noted by Uchoa et al (2006), instead of integer rounding, other subadditive functions can be used to get stronger cuts. An interesting alternative is to add a scalar $s \in [0, 1]$ to each left hand side coefficient, as in 3.10. This approach is not explored in that, nor this work.

$$\sum_{a^t \in \delta^+(S)} \lceil rt + s \rceil x_a^t - \sum_{a^t \in \delta^-(S)} \lfloor rt + s \rfloor x_a^t \geq \lceil rp(S) \rceil \tag{3.10}$$

3.4.2 BRANCH-CUT-AND-PRICE

The pricing subproblem in the BCP-PMWT algorithm consists of finding the pseudo-schedule p with the minimum reduced cost for its corresponding variable λ_p. This can be done

by finding the shortest path from $(0,0)$ to $(0,T)$ in the previously defined graph $G = (V, A)$, setting each arc $a = ((i, i - p_i), (j, t))$ length to $\bar{c}^t_a$.

Being $F(j,t)$ the minimum reduced cost subpath that starts at $(0,0)$ and finishes at (j,t), the following dynamic programming recursion was then used to find the lowest reduced cost column:

$$F(j,t) = \begin{cases} 0, & \text{if } j = 0 \text{ and } t = 0; \\ \infty, & \text{if } \nexists\, i : (i,j)^t \in A; \\ \min_{i:(i,j)^t \in A}\{F(i, t - p_i) + \bar{c}^t_{ij}\}, & \text{otherwise.} \end{cases}$$

If $F(0,T) < 0$, the corresponding pseudo-schedule λ_p column can be added to the Dantzig-Wolfe Restricted master problem.

After every 5 column generation iterations, variable fixing by reduced costs was performed. It consists of checking, for each variable strictly less than one, if setting it to one will take the solution cost past the best know integer solution. This helps to reduce the number of arcs in A, therefore speeding the pricing and improving convergence. The relaxation bounds may also be improved.

For the best know solution, used for variable fixation and for pruning the tree, when solving $P||\sum w_j T_j$, the BCP-PMWT used the values of Rodrigues et al (2008) heuristic.

Branching was performed by choosing a vertex $j \in J$ and partitioning the variables x^t_{ij} (those entering j) into two sets S_1 and S_2 such that $\sum_{(i,j,t)\in S_1} \bar{x}^t_{ij}$ and $\sum_{(i,j,t)\in S_2} \bar{x}^t_{ij}$ are close to 0.5. The variables in S_1 are fixed to zero in the left child node; variables in S_2 are fixed to zero in the right child node. Strong branching was performed by testing eight possible choices of sets before each branch.

When the number of arcs in A after solving the root node was below 200,000, additionally to the BCP, the current reduced ATIF was fed to a commercial MIP solver (CPLEX 11.1). Such approach had good results, since the formulation was reasonably tractable and modern MIP solvers have many advanced features for better exploring the search tree.

3.5 TRIANGLE CLIQUE CUTS

Following the approach of Pessoa et al (2009) on a BCP algorithm for the Heterogeneous Fleet Vehicle Routing Problem, the Triangle Clique cuts were used to improve the ATIF relaxation in the BCP-PMWT-OTI.

Given a set $S \subset J$, with exactly three elements, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the compatibility graph where each vertex in $\mathcal{V}$ represents an arc $a^t = (i,j)^t$, $i,\ j \in S$ and each edge $e = (a_1^{t_1}, a_2^{t_2})$ belongs to $\mathcal{E}$ if and only if $a_1^{t_1}$ and $a_2^{t_2}$ are compatible. For each $i, j, k \in S$, there are 4 cases:

Case 1: if $e = ((i,j)^{t_1}, (i,k)^{t_2})$, then $e \notin \mathcal{E}$

Case 2: if $e = ((i,j)^{t_1}, (k,j)^{t_2})$, then $e \notin \mathcal{E}$

Case 3: if $e = ((i,j)^{t_1}, (j,k)^{t_2})$ and $t_2 \neq t_1 + p_j$, then $e \notin \mathcal{E}$

Case 4: if $e = ((i,j)^{t_1}, (j,k)^{t_2})$ and $t_2 = t_1 + p_j$, then $e \in \mathcal{E}$

In general terms, what each case represents is:

Case 1: a job cannot end more than once

Case 2: a job cannot start more than once

Case 3: a job cannot be processed for less or more time than its processing time p_j

Case 4: a job can be processed exactly for its processing time p_j

For each independent set $I \subset V$, a triangle clique is defined by inequality (3.11).

$$\sum_{a^t \in I} x_a^t \leq 1 \tag{3.11}$$

3.5.1 SEPARATING ALGORITHM

A simple algorithm is used for finding all trianqle clique for a given solution. As noted by Pessoa et al (2009), G is a set of chains. Since by preliminary computation it was observed that chains of four or more elements don't occur frequently, we only consider chains with up to three elements to use a faster separating algorithm. Figure 3.9 illustrates the 3 configurations of chains that may be present in the graph.

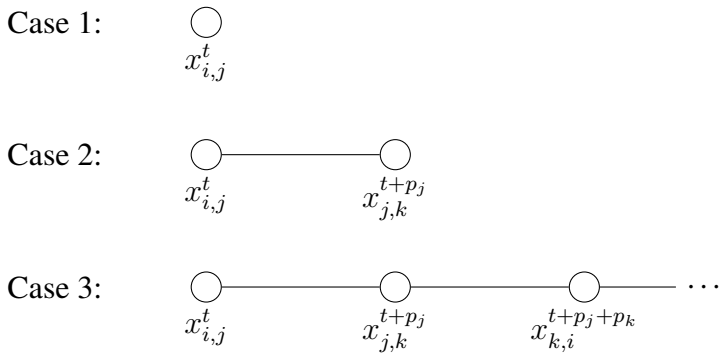


Figure 3.9: Triangle Clique Compatibility Graph ($\mathcal{G}$)

The algorithm consists in building the graph for every triple of jobs (i, j, k) and searching the maximum-weight independent set by enumeration. Since each chain is independent from all others, the maximum-weight independent set will be the union of each single chain maximum-weight independent set. Considering figure 3.9, the maximum weight for each possible chain case is:

Case 1: $x^t_{i,j}$;

Case 2: $\max\{x^t_{i,j}, x^{t+p_j}_{j,k}\}$;

Case 3: $\max\{x^t_{i,j} + x^{t+p_j+p_k}_{k,i}, x^{t+p_j}_{j,k}\}$.

Figure 3.10 represents the compatibility graph of a fractional solution where $i = 2, j = 4, k = 23$ and $p_i = 46, p_j = 16, p_k = 93$. A violated TCC for this example is

$$x^{270}_{2,4} + x^{288}_{4,23} + x^{268}_{2,4} + x^{279}_{4,23} + x^{264}_{2,4} + x^{288}_{2,4} + x^{262}_{2,4} + \\ + x^{287}_{2,4} + x^{287}_{2,4} + x^{258}_{2,4} + x^{283}_{2,4} + x^{214}_{23,2} + x^{276}_{4,23} < 1.$$

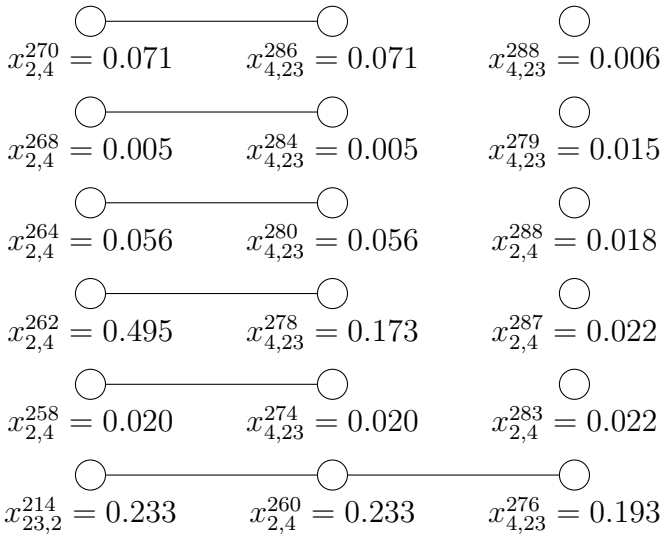


Figure 3.10: Example Compatibility Graph ($\mathcal{G}$)

4 PROPOSED IMPROVEMENTS

This chapter presents the new improvements incorporated into the BCP-PMWT algorithm, resulting in the BCP-PMWT-OTI. The outline of this chapter is as follows.

Section 4.1 presents a polyhedron used to derive the new family of cuts. Section 4.2 presents the new family of cuts, the Overload Elimination Cuts (OECs). Section 4.3 outlines the genetic algorithm used to separate OECs. Section 4.4 shows how the time-indexed formulation can be used to finish the optimization once the number of fixed variables allows for a reasonable sized projection of the ATIF onto the TIF.

4.1 THE CAPACITY PATH POLYHEDRON

In order to derive a new family of cuts, in this section, a specialization of the Capacity Balance Polyhedron is presented. Let $S \subseteq J$ be a set of jobs, $p(S) = \sum_{j \in S} p_j$ be the total processing time of S and $J_+ = J \cup \{0\}$. We define the aggregated variables y^t and z^t for a given S as follows:

$$y^t = \sum_{j \in S} \sum_{i \in (J_0 \setminus S)} x^t_{j,i} \tag{4.1}$$

$$z^t = \sum_{j \in S} \sum_{i \in (J_0 \setminus S)} x^t_{i,j} \tag{4.2}$$

The capacity path polyhedron, $P(S, m)$ is the convex hull of all integer solution of (4.3).

$$\sum_{t=1}^{T} y^t - \sum_{t=0}^{T-1} z^t = 0 \tag{4.3a}$$

$$\sum_{t=1}^{T} t\, y^t - \sum_{t=0}^{T-1} t\, z^t = p(S) \tag{4.3b}$$

$$\sum_{t=1}^{q} y^t - \sum_{t=0}^{q-1} z^t \leq 0 \qquad (q = 1, \ldots, T-1) \tag{4.3c}$$

$$\sum_{t=0}^{q} z^t - \sum_{t=1}^{q-1} y^t \leq m \qquad (q = 1, \ldots, T-1) \tag{4.3d}$$

$$0 \leq y^t \leq m \qquad (t = 1, \ldots, T) \tag{4.3e}$$

$$0 \leq z^t \leq m \qquad (t = 0, \ldots, T-1) \tag{4.3f}$$

What $P(S, m)$ differs from $P(S)$ is that solutions for the first configures multiple paths entering and leaving the set S to satisfy the demand $p(S)$, with the number of paths inside S not exceeding m at any moment, while solutions for the second configures multiple paths entering and/or leaving the set S to satisfy the demand $p(S)$, with no restrictions on the number of paths and no association among entering and leaving paths.

4.2 OVERLOAD ELIMINATION CUTS

For $m \geq 2$ and $t \in \left\{1, \ldots, \left\lfloor \frac{p(S)-1}{m} \right\rfloor + 1 \right\}$ we have the following inequality, that we call overload elimination constraint (OEC):

$$\sum_{q=t}^{t_1} y^q + \sum_{q=t_1+1}^{T} 2\, y^q - \sum_{\substack{q=\max\{t_1,\\ T-p(S)+m(t-1)+1\}}}^{T-1} z^q \geq 2, \quad t_1 = p(S) - t - (m-2)(t-1) \tag{4.4}$$

For didactic purposes, we first prove that the OEC is valid for $m = 2$. The proof for the general case is similar.

Theorem 4.1. *The OEC inequality is valid for* $m = 2$*,* $S \subseteq J$ *and* $t \in \{1, \ldots, \lfloor p(S)/2 \rfloor\}$.

Proof As $m = 2$, at least one and at most two paths crosses the set, each entering and exiting it maybe more than once. Let t_z^i and t_y^i be the last time path i enters and exits the set S, respectively. We can assume, w.l.o.g., that $t_y^1 \leq t_y^2$. Since all jobs in S must be processed by the two machines,

$t_y^1 + t_y^2 \geq p(S)$. Considering paths with $t_z^1 = t_z^2 = 0$, we have 2 possible sets of solutions (illustrated in figures 4.1 and 4.2):

Case 1: $t_y^1 \geq t$

As $t_y^1 + t_y^2 \geq p(S)$, we can infer that $t_y^2 \leq p(S) - t$. So, both paths would exit S in the interval $[t, p(S) - t]$. Resulting in a left hand side of 2 for (4.4).

Case 2: $t_y^1 < t$

As $t_y^1 + t_y^2 \geq p(S)$, we can infer that $t_y^2 > p(S) - t$ which is the same as $t_y^2 \geq p(S) - t + 1$. So, one path would exit S in the interval $[0, t[$ and the other in the interval $[p(S) - t + 1, T]$. Resulting again in a left hand side of 2 for (4.4).

Note that cases $t_y^1 > p(S) - t$ and $t_y^2 < t$ are impossible when $t_z^1 = t_z^2 = 0$. The smallest t_y^2 and greatest t_y^1 values occur when $t_y^2 = t_y^1 = p(S)/2$, for an even value of $p(S)$.

Define C_1 and C_2 as the amount of $p(S)$ processed by machines 1 and 2 before t_z^1 and t_z^2, respectively. Cases 1 and 2 have $C_1 = C_2 = 0$.

To complete the proof we need to show that the inequality still is not violated when $t_z^1 > 0$ or $t_z^2 > 0$. It is true that for each possible solution considered in cases 1 and 2, if we increase t_z^1 or t_z^2, its corresponding t_y^i also increases, resulting in equal or greater coefficients for the y portion of the inequality. So, when both t_z^1 and t_z^2 are less than $\max\{p(S) - t,\ T - p(S) + 2t - 1\}$, the inequalities are valid. For simplicity, let us assume now that $t_z^1 \leq t_z^2$ (and not necessarily $t_y^1 \leq t_y^2$).

Case 1: $t_z^2 < \max\{p(S) - t,\ T - p(S) + 2t - 1\}$

Valid, as stated above.

Case 2: $t_z^2 \geq \max\{p(S) - t, T - p(S) + 2t - 1\}$

As $t_z^2 \geq p(S) - t$, t_y^2 will be strictly greater than $p(S) - t$. And as $t_z^2 \geq T - p(S) + 2t - 1$, we have $t_y^2 - t_z^2 \leq p(S) - 2t + 1$, this means that at least $2t - 1$ have to be processed by machine 1 or the first portion of machine's 2 path. By that, we can infer that at least one arc will exit S at time t or later, as illustrated in figure 4.3. Resulting in a left hand side of at least 2 for (4.4). For the case where the paths enter more than once at time greater than or equal to t_1, one more exit is required for each additional entrance increasing the y side of the inequality by 2, resulting in a net increase of 1 per entrance

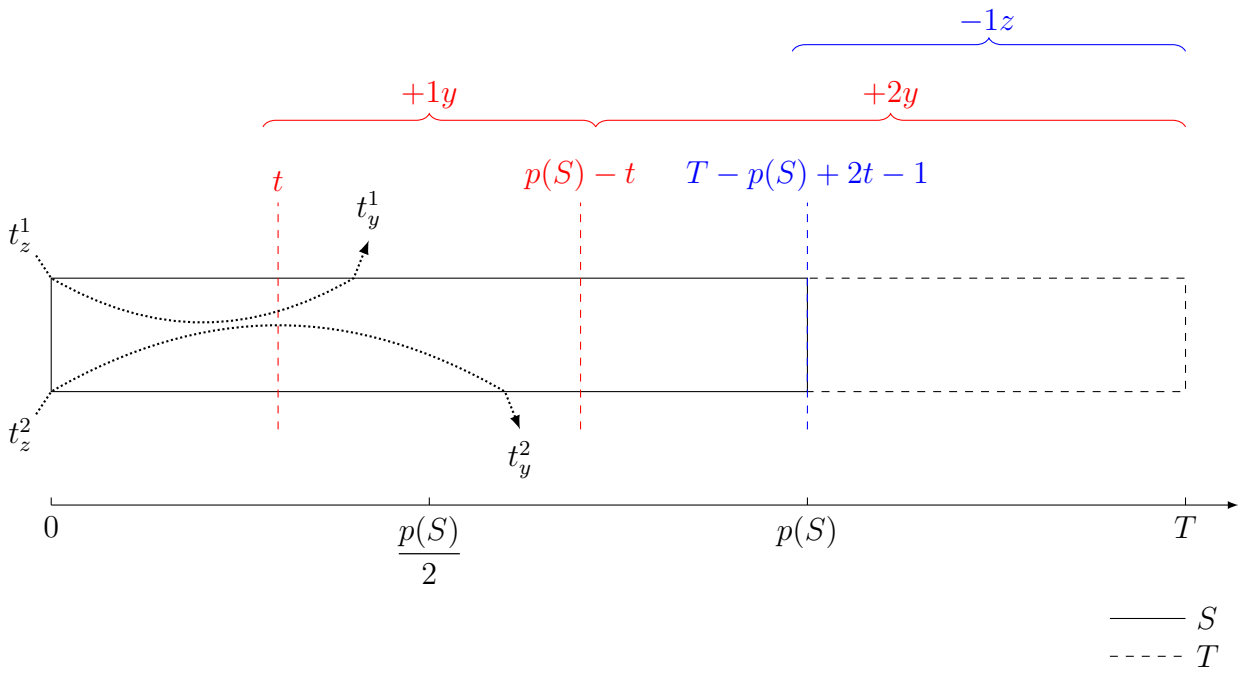


Figure 4.1: OEC's Validity Proof - Case 1 when $m = 2$ and $t_z^i = 0$

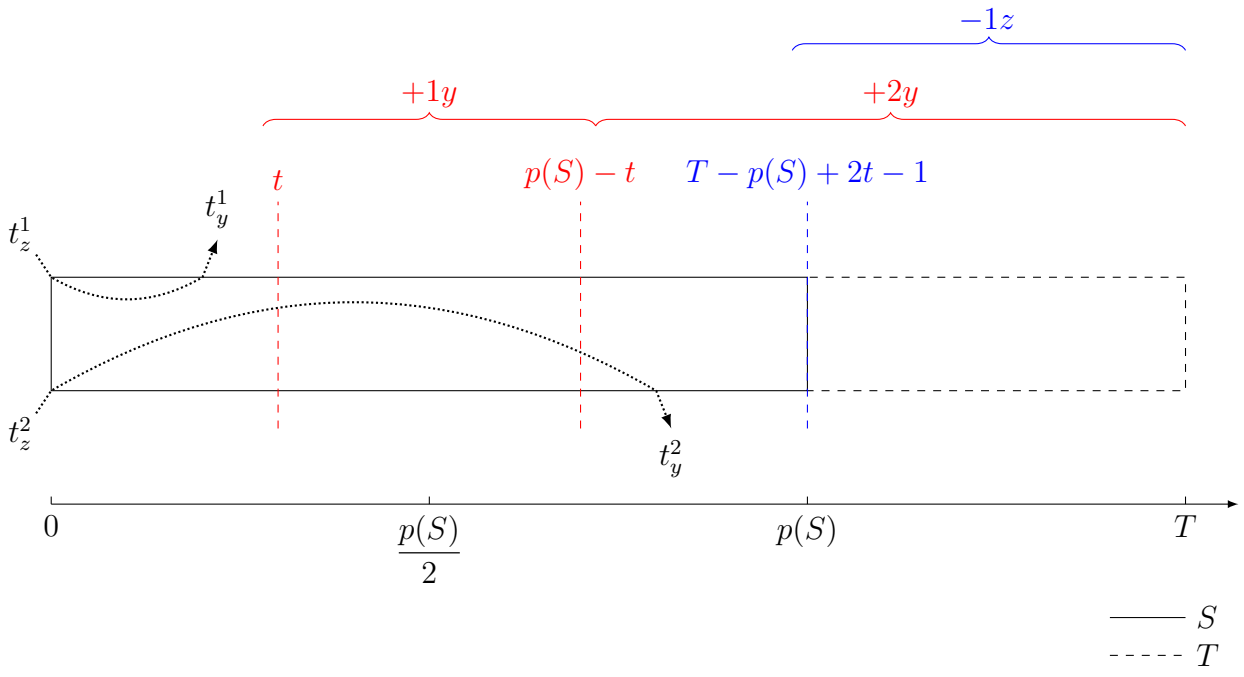


Figure 4.2: OEC's Validity Proof - Case 2 when $m = 2$ and $t_z^i = 0$

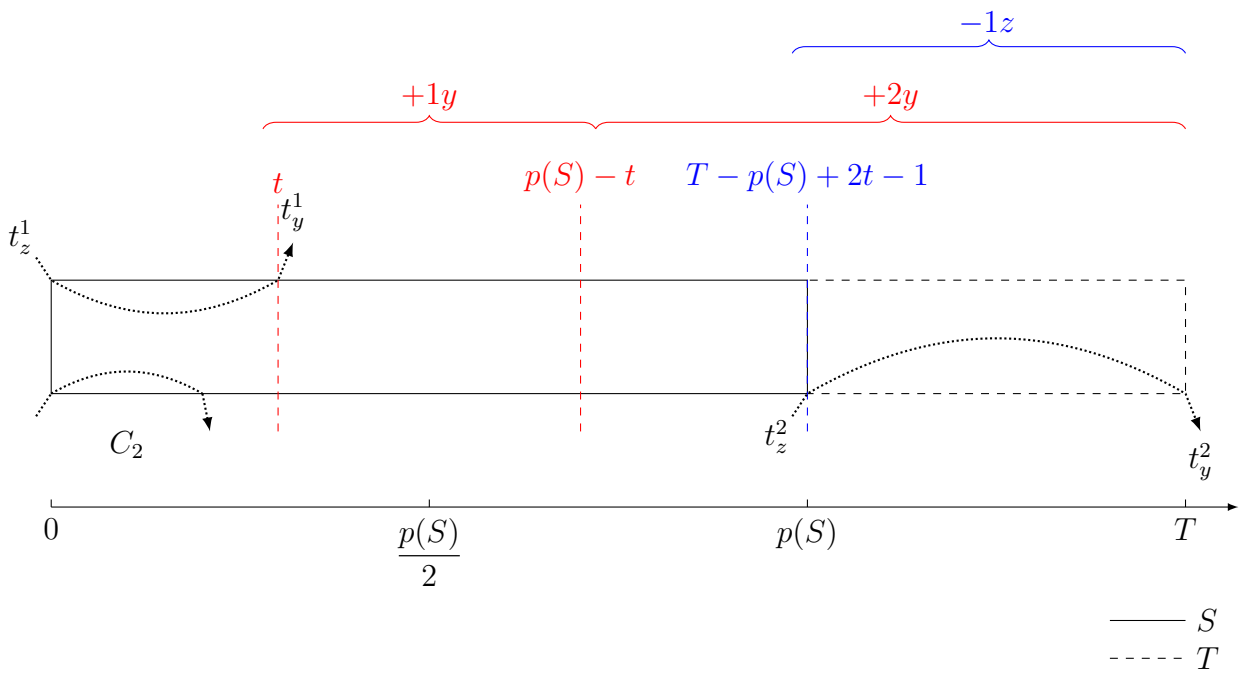


Figure 4.3: OEC's Validity Proof - Case 2 when $m = 2$ and $t_z^i \geq 0$

Now, we give the proof of the validity of OEC for the general case.

Theorem 4.2. *The OEC inequality is valid in the general case for* $S \subseteq J$ *and* $t \in \left\{1, \ldots, \left\lfloor \frac{p(S)-1}{m} \right\rfloor + 1\right\}$.

Proof At least one and at most m paths crossing the set, each entering and exiting it maybe more than once. Let t_z^i and t_y^i be the last time path i enters and exits the set S, respectively. We can assume, w.l.o.g., that $t_y^{i-1} \leq t_y^i$, $i = 2, \ldots, m$. Since all jobs in S must be processed by the m machines, $\sum_{i=1}^{m} t_y^i \geq p(S)$. Considering paths with $t_z^1 = t_z^2 = \ldots = t_z^m = 0$, we have 2 possible set of solutions:

Case 1: $t_y^{m-1} < t$

In this case, $\sum_{i=1}^{m-1} t_y^i \leq (m-1)(t-1)$, so $t_y^m \geq p(S) - (m-1)(t-1) = t_1 + 1$ and the cut is valid since $y^{t_y^m}$ has coefficient 2.

Case 2: $t_y^{m-1} \geq t$

In this case, the cut is valid since both $y^{t_y^{m-1}}$ and $y^{t_y^m}$ have coefficient greater than or equal to 1.

Define C_i as the amount of $p(S)$ processed by machines M_i before time t_z^i. Cases 1 and 2 have $C_1 = C_2 = \ldots = C_m = 0$.

To complete the proof we need to show that the inequality still is not violated when $t_z^i > 0$, for $i = 1, \ldots, m$. It is true that for each possible solution considered in cases 1 and 2, if we increase at least on t_z^i, its corresponding t_y^i also increases, resulting in equal or greater coefficients for the y portion of the inequality. So, when t_z^i is less than $\max\{t_1, T - p(S) + m(t-1)+1\}$, for $i = 1, \ldots, m$ the inequalities are valid. For simplicity, let us assume now that $t_z^{i-1} \leq t_z^i$, $i = 2, \ldots, m$ (and not necessarily $t_y^{i-1} \leq t_y^i$, $i = 2, \ldots, m$).

Case 1: $t_z^m < \max\{t_1, T - p(S) + m(t-1) + 1\}$

Valid, as stated above.

Case 2: $t_z^m \geq \max\{t_1, T - p(S) + m(t-1) + 1\}$

As $t_z^m \geq t_1$, t_y^m will will be strictly greater than t_1. And as $t_z^m \geq T - p(S) + m(t-1) + 1$, we have $t_y^m - t_z^m \leq p(S) - m(t-1) - 1$, this means that at least $m(t-1)+1$ have to be processed by machine $1, \ldots, m-1$ or the first portion of machine's m path. By that, we can infer that at least one arc will exit S at time t or later. Resulting in a left hand side of at least 2 for (4.4). For the case where the paths enter more than once at time greater than or equal to t_1, one more exit is required for each additional entrance increasing the y side of the inequality by 2, resulting in a net increase of 1 per entrance.

The term Overload Elimination comes from the fact that the inequality could only be violated by an integer solution if, hypothetically, some machine was overloaded to allow one or more jobs of S to finish prematurely before t. In light of fractional solutions, such premature finish occurs when a fraction of a job finishes before t. Figure 4.4 represents an OEC ($S = \{1, 2, 3, 5, 6, 7\}$ and $t = 13$) for a fractional solution of the 8 jobs instance presented previously, and figure 4.5 represents the same OEC for an integer solution of the same instance. Such integer solution was found after applying the violated OEC to solve the linear relaxation.

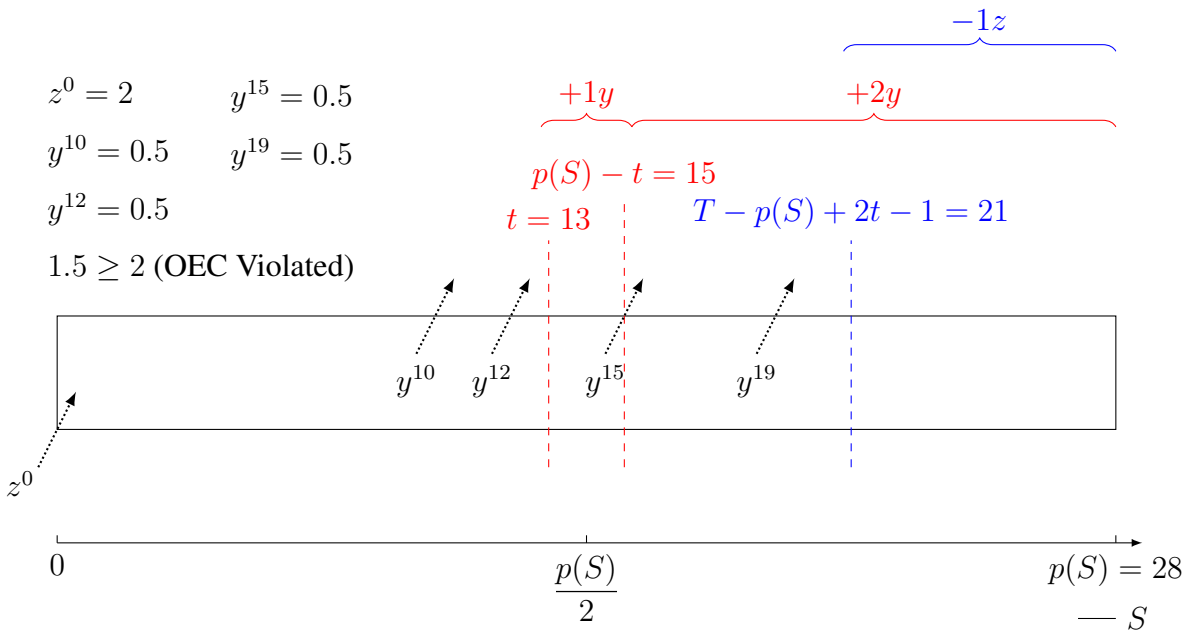


Figure 4.4: Example OEC for a Fractional Solution

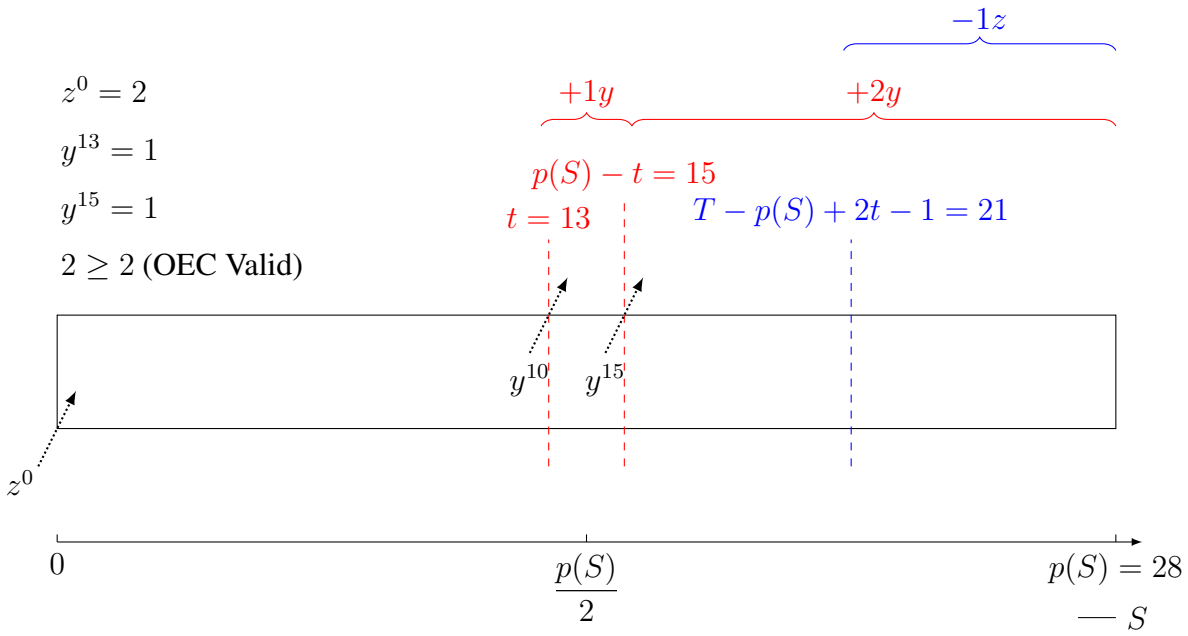


Figure 4.5: Example OEC for an Integer Solution

4.3 THE GENETIC ALGORITHM FOR CUT SEPARATION

Here we present the details of the genetic algorihtm used to separate OEC's. In this section, the term solution and individual will be used interchangeably to denote a cut, being it violated or not.

Each chromosome encodes a single OEC. To codify it, a sequence of n bit for representing S, followed by an integer representing t is used. Figure 4.6 shows the chromosome for a 10

jobs instance, being $S = \{1, 3, 10\}$ and $t = 13$.

1	0	1	0	0	0	0	0	0	1	13

Figure 4.6: Example OEC Chromosome Codification

Define $\bar{G} = (\bar{V}, \bar{E})$ as an (undirected) support graph for the fractional solution $\bar{x}$, such as $\bar{V} = J$ and $\bar{E} = \{(i, j) : \exists\, t \mid \bar{x}^t_{i,j} > 0 \text{ or } \bar{x}^t_{j,i} > 0\}$. Also, define $\bar{C}$ as the average completion time for jobs $j \in J$, calculated as

$$\bar{C}_j = \sum_{(i,t)|\bar{x}^t_{j,i}>0} t\, \bar{x}^t_{j,i}, \tag{4.5}$$

$\bar{F}(k)$ as the set of jobs with the k smallest values of $\bar{C}$, and $\bar{F}_j$ as the job with the j-th smallest $\bar{C}$ job. The set $\bar{F}(k)$ can be interpreted as the first k jobs to finish, and $\bar{F}_j$ as the j-th job to finish, in a fractional schedule.

The genetic algorithm generates an initial population of n solutions and selects the $nPopulation$ best ones (assuming $nPopulation \leq n$). Then, a number of crossover and selection operations are performed iteratively until some stopping criteria is met. For every new individual, being generated either by the initial population or the crossover operation, local search is performed to improved it. Algorithm 2 outlines the genetic algorithm, where the input parameters are the size of the population to be generated initially and to be carried from one generation to the next ($nPopulation$), the crossover rate ($crossoverRate$), which is the percentage of the population to be generated at each generation by crossover, and the criteria to stop the algorithm ($stopCriteria$).

Preliminary tests showed that most of the violated cuts had its set S inducing a connected subgraph in $\bar{G}$. Due to that, each individual in the initial population is generated by calling algorithm 3, which randomly generates sets with this property. For shortness, in this section, we refer to such sets as connected S sets.

The selection operator consists of eliminating the worst individuals so that the population is back to its initial size. However, a pool of violated cuts is kept apart from the current population. No cut is removed from this pool.

Algorithm 2: Separation Genetic Algorithm

Input: $nPopulation, crossoverRate, stopCriteria(\ )$
Output: $Violatedcuts$
$violatedCuts \leftarrow [\ ]$
for $k \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** k **do**
    $Individual \leftarrow generateInitialIndividual(k, j, \bar{C}, supportGraph)$
    if $Violated(Individual)$ **then**
      $violatedCuts \leftarrow violatedCuts + Individual$
    end
    $Population \leftarrow Population + Individual$
  end
end
$Population \leftarrow Selection(Population, nPopulation)$
while *not* $stopCriteria(\ )$ **do**
  for $i \leftarrow 1$ **to** $nPopulation \times crossoverRate$ **do**
    $father \leftarrow random\ individual\ from\ Population$
    $mother \leftarrow random\ individual\ from\ (Population - father)$
    $child \leftarrow crossover(father, mother)$
    if $Violated(child)$ **then**
      $violatedCuts \leftarrow violatedCuts + child$
    end
    $Population \leftarrow Population + child$
  end
  $Population \leftarrow Selection(Population, nPopulation)$
end
return $violatedCuts$

To generate a single initial solution, we search for the largest connected $S \subset \bar{F}(k)$ including $\bar{F}_j$ (root node). We do so by performing breadth-first search on the subgraph of $\bar{G}$ induced by $\bar{F}(k)$, starting from the $\bar{F}_j$. Algorithm 3 represents this procedure, where the inputs parameters are the size k of set $\bar{F}(k)$, the rank j of job $\bar{F}_j$, the average completion times $\bar{C}$, and the support graph.

Algorithm 3: Generation of Initial Individuals

Input: k, j, $\bar{C}$, $supportGraph$
Output: $Individual$
$\tilde{G} \leftarrow supportGraph\ induced\ by\ \bar{F}k$
$Individual.S \leftarrow BreadthFirstSearch(\tilde{G}, \bar{F}_j)$
return $localSearch(Individual)$

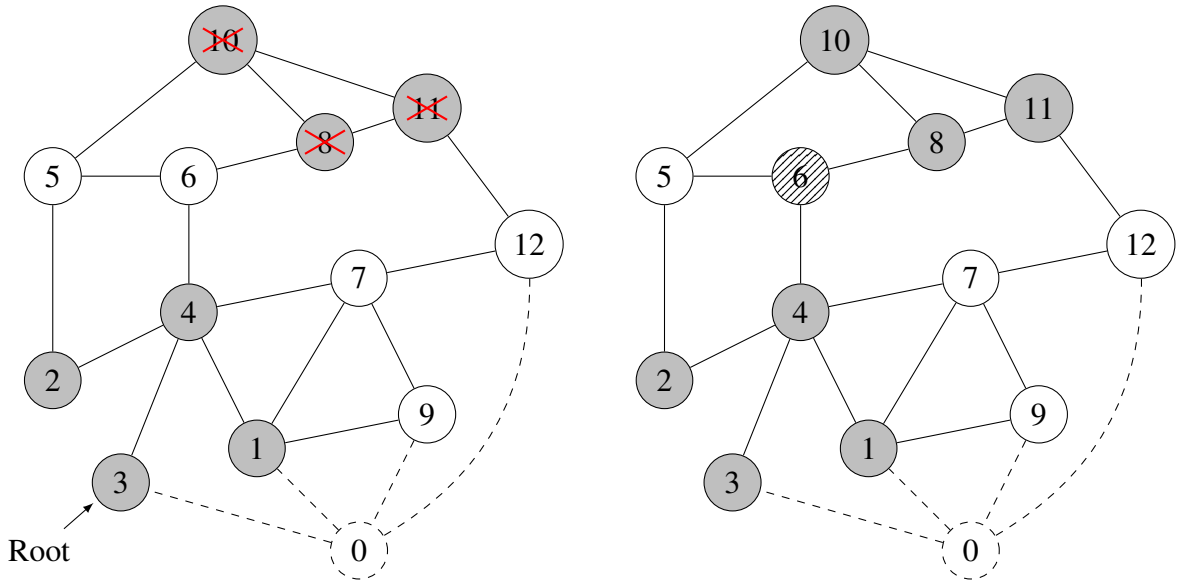
The crossover operator consists in deriving a new solution from two solutions chosen randomly from the population. We call the new solution as child and the existing solutions as parents, or as father and mother. To generate the child, we first select all elements that are in

both parents. If this intersection is empty, a random element is chosen from each parent and based on $\bar{G}$, we select the least amount of nodes that will induce a connected subgraph. Then, some elements contained in the parents are included in S at random. If the resulting S does not induce a connected subgraph in $\bar{G}$, a random element from S, called root, is chosen and all elements of S not reachable in $\bar{G}$ from root are eliminated, so that the remaining S induces a connected subgraph. Algorithm 4 outlines the described crossover procedure.

Algorithm 4: Crossover Operator

Input: $father, mother, supportGraph$
Output: $child$
if $| father.S \cap mother.S | > 0$ **then**
  $child.S \leftarrow father.S \cap mother.S$
else
  $child.S \leftarrow (random\ element\ of\ father.S) \cup (random\ element\ of\ mother.S)$
  $child.S \leftarrow connectS(child.S, supportGraph)$
end
foreach $j \in ((father.S \cup mother.S) - child.S)$ **do**
  $with\ 50\%\ chance\ of\ ocurring,\ child.S \leftarrow child.S \cup j$
end
$child.S \leftarrow$ a random connected component of $child.S$
return $localSearch(child)$

Figure 4.7 illustrates the $connectS()$ and $eliminateDisconnection()$ procedures. In figure 4.7(a), elements 8, 10 and 11 were removed from S, remaining 1, 2, 3 and 4. In figure 4.7(b), element 6 was included in S so that edges (4,6) and (6,8) connects the two disconnected subgraphs.



(a) connectS() (b) eliminateDisconnection()

Figure 4.7: Example of Procedures for Generating Connected Subgraphs

The local search operator, represented in algorithm 5, consists of performing every single element insertion and deletion on S until no movement yields a better cut. For every change, the violation is calculated for every valid value of t and we pick the one generating the bigger violation. Function $Violation()$ returns a negative value when the cut is not violated, its absolute value is the constraint slack in such cases.

Algorithm 5: Local Search Operator

Input: $Individual$
Output: $Individual$
foreach $j \in J$ **do**
  if $j \in Individual.S$ **then**
    $S \leftarrow Individual.S - j$
  else
    $S \leftarrow Individual.S + j$
  end
  $t_{best} = 1$
  foreach $t \in validTimePeriods(S, m)$ **do**
    if $Violation(OEC(S,t)) > Violation(OEC(S, t_{best}))$ **then**
      $t_{best} \leftarrow t$
      if $Violation(OEC(S,t))$ *is significant* **then**
        $solutionPool \leftarrow solutionPool + OEC(S,t)$
      end
    end
  end
  if $Violation(OEC(S, t_{best}) > Violation(Individual))$ **then**
    $Individual \leftarrow OEC(S, t_{best})$
  end
end
return $Individual$

No mutation operator was used in the final version of the genetic algorithm.

Based on preliminary tests we set the population size to 20, the rate of crossover of 100%, that is, 20 crossover operations per generation, and the stop criteria to the completion of 100 generations.

Initially, this genetic algorithm was applied to separate RHECC cuts. But since its performance was not better than the greedy algorithm used by Pessoa et al (2010), it was not used.

4.4 TIME-INDEXED FORMULATIONS

When investigating different formulations for a problem, the usual purpose is to find the one that yields the best bounds. In this section, we explore different ways of describing the exactly same polyhedron and discuss how they may influence the MIP solver. We believe that two characteristics of a formulation can improve the MIP solver performance. The first is how sparse is the constraint matrix, due to the faster computation of the simplex iterations. The second is the impact of fixing one variable over the linear relaxation bound, due to a stronger branching performance.

When modeling scheduling problems, there are basically two characteristics to be con-

strained, one is that every job has to be processed, and the other is that no more than m machines are active at any moment. The first two formulations, (4.6) and (4.7), use binary variables y_j^t indicating job j has finished at time t, and differ in how the machine constraints are enforced. The first does it by a network flow, where m flow units leave the source (4.6c) and no flow may leave other nodes (4.6d). The second does it by explicitly restricting how many jobs are being processed at each period t (4.7c), which is also referred to as resource constraints.

$$\text{Minimize} \quad \sum_{j \in J} \sum_{t=p_j}^{T} f_j(t)\, y_j^t \tag{4.6a}$$

$$\text{Subject to} \quad \sum_{t=p_j}^{T} y_j^t = 1 \qquad (j \in J) \tag{4.6b}$$

$$\sum_{j \in J} y_j^{p_j} = m \tag{4.6c}$$

$$\sum_{j \in J | t \geq p_j} y_j^t \geq \sum_{j \in J} y_j^{t+p_j} \qquad (t = 1, \ldots, T) \tag{4.6d}$$

$$y_j^t \in \{0, 1\} \qquad (j \in J;\ t = p_j, \ldots, T) \tag{4.6e}$$

$$\text{Minimize} \quad \sum_{j \in J} \sum_{t=p_j}^{T} f_j(t)\, y_j^t \tag{4.7a}$$

$$\text{Subject to} \quad \sum_{t=p_j}^{T} y_j^{p_j} = 1 \qquad (j \in J) \tag{4.7b}$$

$$\sum_{j \in J} \sum_{t'=t}^{\min\{t+p_j-1, T\}} y_j^{t'} \leq m \qquad (t = 1, \ldots, T) \tag{4.7c}$$

$$y_j^t \in \{0, 1\} \qquad (j \in J;\ t = p_j, \ldots, T) \tag{4.7d}$$

Let z_j^t be a binary variable indicating job j has finished until time t. By using the substitution $y_j^t = z_j^t - z_j^{t-1}$, we transform formulations (4.6) and (4.7) into (4.8) and (4.9), respectively. The main difference is how fixing one variable influences the other variables. For example, by setting $z_j^{t_1} = 1$, every z_j^t with $t > t_1$ is also set to 1, and, by setting $z_j^{t_1} = 0$, every z_j^t with $t < t_1$ is also set to 0. This leads to a more effective branching, since both fixations usually change the relaxed solutions substantially.

$$\text{Minimize} \quad \sum_{j \in J} \sum_{t=p_j}^{T} f_j(t) \left(z_j^t - z_j^{t-1} \right) \tag{4.8a}$$

$$\text{Subject to} \quad \sum_{j \in J} \left(z_j^{p_j} - z_j^{p_j - 1} \right) = m \tag{4.8b}$$

$$\sum_{j \in J | t \geq p_j} \left(z_j^t - z_j^{t-1} \right) \geq \sum_{j \in J} \left(z_j^{t+p_j} - z_j^{t+p_j-1} \right) \quad (t = 1, \ldots, T) \tag{4.8c}$$

$$z_j^{t-1} \leq z_j^t \quad (j \in J;\ t = p_j, \ldots, T) \tag{4.8d}$$

$$z_j^{p_j - 1} = 0 \quad (j \in J) \tag{4.8e}$$

$$z_j^t \in \{0, 1\} \quad (j \in J;\ t = p_j, \ldots, T-1) \tag{4.8f}$$

$$z_j^T = 1 \quad (j \in J) \tag{4.8g}$$

$$\text{Minimize} \quad \sum_{j \in J} \sum_{t=p_j}^{T} f_j(t) \left(z_j^t - z_j^{t-1} \right) \tag{4.9a}$$

$$\text{Subject to} \quad \sum_{j \in J} \left(z_j^{\min\{t+p_j-1, T\}} - z_j^{t-1} \right) \leq m \quad (t = 1, \ldots, T) \tag{4.9b}$$

$$z_j^{t-1} \leq z_j^t \quad (j \in J;\ t = p_j, \ldots, T) \tag{4.9c}$$

$$z_j^{p_j - 1} = 0 \quad (j \in J) \tag{4.9d}$$

$$z_j^t \in \{0, 1\} \quad (j \in J;\ t = p_j, \ldots, T-1) \tag{4.9e}$$

$$z_j^T = 1 \quad (j \in J) \tag{4.9f}$$

In the next chapter, we detail how computational tests will be performed to compare the different formulations in terms of performance.

4.4.1 TIME-INDEXED CUTS BY PROJECTION OF THE ARC-TIME-INDEXED FORMULATION

As proved by Pessoa et al (2010), the TIF is dominated by the ATIF, meaning that the ATIF linear relaxation will always yield a better or equal lower bound when compared to the TIF linear relaxation over the same instance. We then devise a procedure to generate cuts from the projection of the ATIF to the TIF. These cuts take further advantage of the variable fixing performed by BCP-PMWT over the ATIF variables.

We define a fractional solution of the TIF as y^* and the set of remaining arcs A_t from the ATIF, for each time period t, as below.

$$A_t = \left\{(i,j) \in J \times J \mid x_{i,j}^t \text{ is not fixed}\right\} \tag{4.10}$$

From the flow time-indexed formulation (4.6), the following inequality is valid in the y variable space,

$$\sum_{i \in J} y_i^t \geq \sum_{j \in J} y_j^{t+p_j} \qquad (t = 1, \ldots, T). \tag{4.11}$$

This inequality can be interpreted as the association of each variable $y_j^{t+p_j}$ equal to one variable y_i^t also equal to one, meaning that job j is the successor of job i on some machine. Since, by the ATIF (and the variable fixing), not all such successions are allowed, additional cuts are derived from A_t.

$$\sum_{i:\exists j \in S | (i,j) \in A_t} y_i^t \geq \sum_{j \in S} y_j^{t+p_j} \qquad (S \subset J;\ t = 1, \ldots, T) \tag{4.12}$$

For a given time period t, exact separation of (4.12) can be done by finding the minimum cut, or maximum flow, on a digraph built as follows. We lay one source node (s) and one sink node (t), n nodes $\{1, 2, ..., n\}$ and other n nodes $\{1', 2', ..., n'\}$. For each variable $y^{*t}_i \neq 0$ we draw an arc connecting the source node to the node i with capacity $c_{s,i} = y^{*t}_i$. For each variable $y^{*t+p_j}_j \neq 0$ we draw an arc connecting the node j' to the sink with capacity $c_{j',t} = y^{*t+p_j}_j$. Finally, for each non-fixed variable $x^t_{i,j}$ such that $y^{*t}_i \neq 0$, we draw an arc from i to j' with capacity $c_{i,j'} = \infty$. After calculating the minimum cut, the set S will be all nodes i' in the sink side of the cut. Figure 4.8 shows how the graph is constructed.

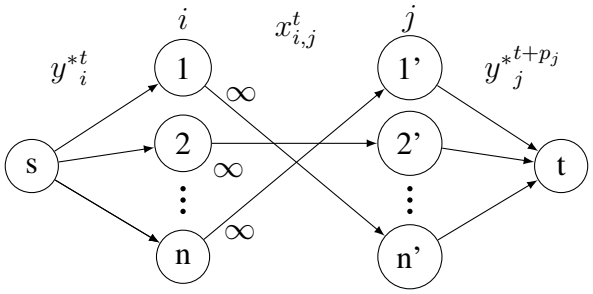


Figure 4.8: Support graph for separation of cuts derived from ATIF

This procedure can by viewed as a way to check if a solution in the y variables space is feasible in the x variables space by searching for a "broken path". Figure 4.9 shows the

separation support graph for the TIF linear relaxation of the example instance of 8 jobs presented in section 3.3. In it, for $t = 8$, we have variables $y_3^8 = 0.5$, $y_3^{11} = 0.5$ ($p_3 = 3$) and in the x variables space, $A_8 = \{(3,1),(3,4),(3,6),(3,8)\ ,(7,3),(7,4),(7,8),(8,4)\}$. It can be seen in the graph that no path exists from the source to the sink going through $i = 3$.

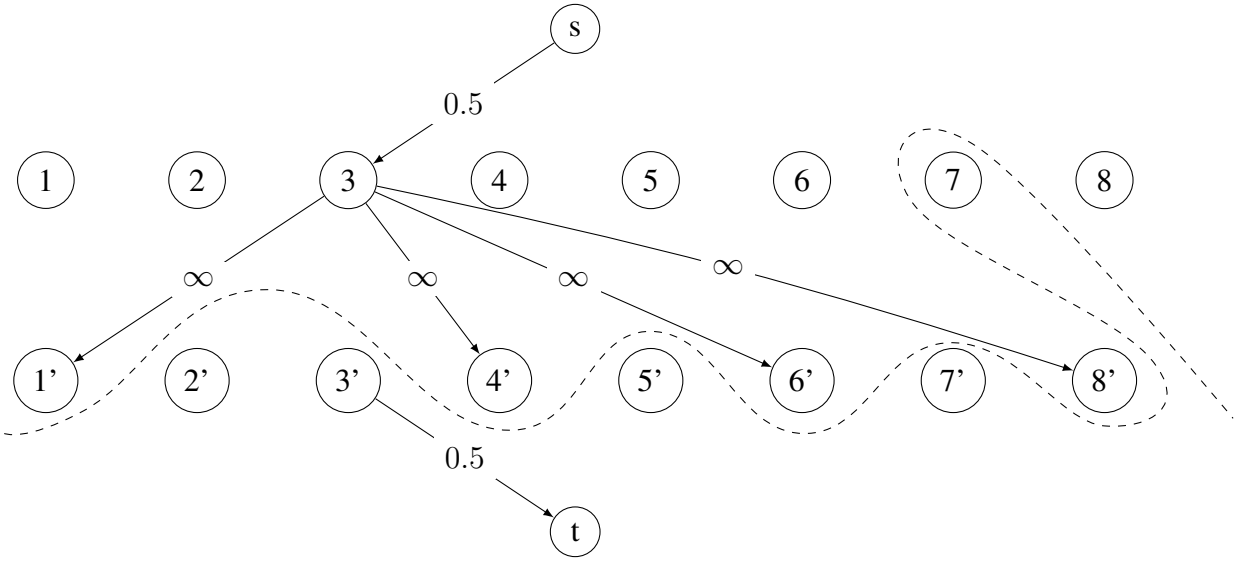


Figure 4.9: Projection of cuts for 8 jobs example instance

The set of variables in (4.13) and figure 4.10 shows the TIF linear relaxation solution for an example instance. We can see in it, that job 3 is preceded by itself, something that is not feasible in the x variables space, since no variable $x_{i,j}^t$ such that $i = j$ exists.

$$\begin{array}{lllll} y_1^6 = 1 & y_3^{11} = 0,5 & y_5^5 = 0,5 & y_6^{20} = 0,5 & y_8^{18} = 0,5 \\ y_2^{10} = 1 & y_4^{21} = 0,5 & y_5^{11} = 0,5 & y_7^{14} = 0,5 & y_8^{19} = 0,5 \\ y_3^8 = 0,5 & y_4^{24} = 0,5 & y_6^6 = 0,5 & y_7^{15} = 0,5 & \end{array} \tag{4.13}$$

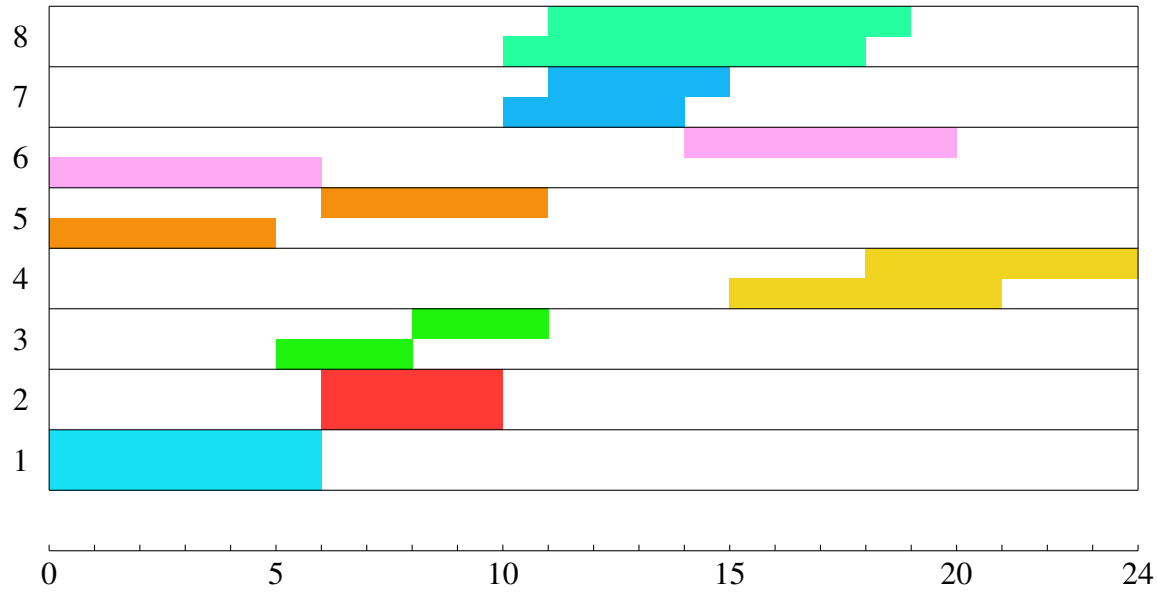


Figure 4.10: Gantt chart of example instance TIF linear relaxation solution

For shortness, in the remaining of the work, we will refer to the cuts proposed here simply as projected cuts.

5 COMPUTATIONAL TESTS AND RESULTS

In this chapter, we detail how computational tests were performed and comment its results. First section evaluate the effectiveness of the Triangle Clique and Overload Elimination Cuts on improving the ATIF bounds. The second section compares the performance of the four time-indexed formulations presented in Section 4.4 in order to choose which will be used to solve the problem. Also, we evaluate the effect of variable fixation and strengthening by projected cuts. The third and last section summarize the results of the full BCP-PWMT-OTI algorithm.

The instances of $P||\Sigma w_j T_j$ used were generated by transforming the $1||\Sigma w_j T_j$ instances of Potts and Wassenhove (1985), found on the OR-Library. As described by Pessoa et al (2010), for $n \in \{40, 50, 100\}$, and $m \in \{2, 4\}$, the first instance for each group of $1||\Sigma w_j T_j$ was picked (instance numbers ending with 1 or 6) and had its due date d_j divided by m.

For the upper bound of each instance, we used the solutions computed by the heuristic procedure of Kramer and Subramanian (2015). For the sake of better evaluating our attempts to solve the $P||\Sigma w_j T_j$ problem, we highlight where those values are better than the ones of Rodrigues et al (2008). Tests are run only on instances not proven optimal by the first ATIF LP relaxation, i.e., the first linear relaxation lower bound is less than the upper bound given by the heuristic procedure.

The LP/MILP solver used was IBM ILOG CPLEX 12.5 All tests were run on a Intel Core i7-3770 PC with a 3.4Ghz clock (using one thread), 12GB of RAM and the Linux operating system.

5.1 SOLVING THE ARC-TIME-INDEXED LINEAR RELAXATION

Table A.1 presents the results for the root relaxation after cut separation for the 106 instances not proven optimal by the primal heuristic and the ATIF LP relaxation. The first three columns identify the instance (n,m and Instance number), the next column (Heu UB) gives the upper bound provided by Kramer and Subramanian (2015), the next column gives the ATIF linear relaxation solution (First LB), then, 2 sets of columns compares the root lower bound, its corresponding integrality gap and the time it took to solve it, with the values of Pessoa et al (2010). The integrality gap is calculated as $\left(\frac{\text{Heu UB}-\text{Root LB}}{\text{Heu UB}}\right)$. By root lower bound, we refer to the objective vaue of the LP after generating all cuts. When column Gap is empty, an optimal solution has been found. The best LB value of each row is highlighted in bold.

Table 5.1 summarizes the results of both algorithms in solving the root node. It can be seen that the increase in lower bound compensates the time expense of separating OEC cuts. Especially for instances with $n = 100$ and $m = 2$, for example, instance (100-2m-81) which was not solved by BCP-PMWT, was solved at the root using the new cuts.

Table 5.1: Root relaxation and cut separation results

		BCP-PMWT		BCP-PMWT-OTI	
n	m	Average Gap	Average Time	Average Gap	Average Time
40	2	0.525%	77.98	0.234%	141.56
40	4	0.456%	23.35	0.458%	144.24
50	2	0.379%	256.76	0.330%	403.71
50	4	0.571%	67.81	0.584%	165.06
100	2	0.878%	6297.01	0.121%	10135.56
100	4	0.494%	984.02	0.404%	753.57

5.2 COMPARISON OF DIFFERENT TIME-INDEXED FORMULATIONS

We ran tests for each of the four time-indexed formulations presented in Section 4.4, in order to compare them. To save time, we constrained the tests to 40 and 50 jobs instances. But we believe that any conclusions drawn can be assumed true for bigger instances.

To perform the tests, we use a list of unfixed arc-time-indexed variables, generated after solving the root node. All time-indexed models are then formulated by taking into consideration

the fixed arc-time-indexed variables. Each time-indexed variable y_i^t is fixed to zero when all variables $x_{i,j}^t$, $j = \{0, \ldots, n\}$ are also fixed to zero.

Table B.1 shows the test results for 29 instances of 40 and 50 jobs. In it, we refer to formulations (4.6), (4.7), (4.8) and (4.9) as Fy, My, Fz and Mz respectively. The first three columns (n,m and Instance number) identify the instance, and the two sets of columns gives the time for solving the linear relaxation plus all cuts generations (LP Time), and the time the MIP solver took to solve the TIF formulation strengthened by projected cuts (MIP Time).

Table 5.2 summarizes Table B.1 by giving the average values for the two instance sets, as well as the number of instances solved in up to 3,600 seconds (# Solved). The averages consider only the 10 instances solved by all four formulation. It can be observed that the best choice is formulation Mz, even though solving the linear relaxation sometimes may be faster for a different TIF, the branching performed by the MIP solver is more effective for Mz.

Table 5.2: Comparison of Alternative Time-Indexed Formulations – Summary

	Average LP Time (s)				Average MIP Time (s)				# Solved			
n	Fy	My	Fz	Mz	Fy	My	Fz	Mz	Fy	My	Fz	Mz
40	0.72	0.84	7.17	0.97	63.17	351.97	122.92	58.28	12	10	12	12
50	1.77	1.98	47.08	2.43	53.46	150.26	70.56	16.47	13	11	14	16

Table B.2 shows, for the same instances of the previous test, the effect of fixing variables in the TIF, based on the state achieved in the ATIF. It can be seen that both the LP Time and MIP time decreases dramatically. Table 5.3 summarizes B.2. Again, the averages consider only the 19 instances solved in up to 3,600 seconds by all four formulation.

Table 5.3: Effect of Variable Fixation in the Mz Time-Index Formulation – Summary

	Average LP Time (s)		Average MIP Time (s)		# Solved	
n	Fix.	w/ Fix.	Fix.	w/ Fix.	Fix.	w/ Fix.
40	0.74	22.54	11.11	561.59	12	10
50	2.04	105.84	11.63	496.61	17	9

Table B.3 shows the projected cuts improvement on the TIF linear relaxation bounds. There is a huge loss of lower bound when projecting the ATIF into the TIF, even with variable

fixation because the RHECC, Triangle Click Cuts and OEC are not translated to the time-indexed variables. To alleviate this loss, the projected cuts plays an important role. For example, on instance (100-4m-86), it closes 27.36% of the integrality gap. Without such improvement, the optimal solution would possibly not be achieved. Table 5.4 summarizes table B.3.

Table 5.4: Effect of Projected Cuts in the Mz Time-Indexed Formulation – Summary

n	m	Avg. ATIF Root Gap	Avg. 1st TIF LP Gap	Avg. Root Gap	Avg. Gap Improv.
100	2	0.121%	0.340%	0.306%	13.55%
100	4	0.404%	0.660%	0.646%	11.20%

It can be seen that both the generation of cuts and the fixing of TIF variables are very effective, being the latter the most significant. For some instances, the generation of cuts worsen the Branch and Bound performance due to the LP increase in complexity. We tried to balance the generation of cuts and this increase of complexity by implementing a rollback procedure, in which the cut generation stops and the cuts inserted in the last iteration are removed if the time for solving the strengthened relaxation more than doubles. Also, we try to avoid tailing off by stopping the cut generation if the objective increases less than 10^{-3} on 5 sequential iterations.

5.3 SOLVING THE PROBLEM TO INTEGRALITY

Tables C.1 to C.6 present the results of solving the 106 instances by the BCP-PMWT and the BCP-PMWT-OTI. Each table correspond to a set (m,n) of instances, where the first column identify the instance. The results for the two algorithms are separated into two sets of columns, where column Heu UB gives the upper bound used by both the BCP and the MIP solver to explore the search tree, column Root LB and Root Time gives the lower bound for the ATIF strengthened with the algorithm's family of cuts and the time to solve the first LP and all subsequent cut insertions by CG, columns BCP Time and ATIF/TIF MIP Time gives the time it took to achieve the integral optimal solution, by BCP from root and by feeding the ATIF/TIF to the MIP solver. Column Best gives which part of the algorithm achieved the optimal solution faster and column Overall Time give the overall time (Root Time, Root+BCP Time or Root + ATIF/TIF MIP Time). We highlight (by bold printed numbers) in column Inst the instances solved for the first time. In column Heu UB for the BCP-PMWT-OTI, we highlight where the improved bounds were used, and, in columns Root LB, we highlight the best lower bound for

the instance.

For the TIF, we used model Mz (4.9), which proved to be the best choice. When feeding the MIP solver, the ATIF carries the cuts separated in the root node, and the TIF do not. Each set (m,n) have 25 instances, the instances proven optimal by the first ATIF LP relaxation are omitted from the table, but are accounted in the number of Solved instances. Tables 5.5 summarizes the six tables. All 40 and 50 jobs instances were already solved, but the better lower bounds and the TIF improved running times. Three instances of the 100-2m set and five instances of the 100-4m set were solved for the first time. Also, there was an improvement of running time.

Table 5.5: Summary of Results

		BCP-PMWT		BCP-PMWT-OTI	
n	m	# Solved	Avg. Time (s)	# Solved	Avg. Time (s)
40		50	357.88	50	158.17
50		50	5734.94	50	461.07
100	2	18	22523.81	21	14678.39
100	4	17	34243.36	22	1617.56

Table 5.6 compares the BCP approach to MIP solver approach, for both the BCP-PMWT and the BCP-PMWT-OTI. It can be noted that in both algorithms, the best approach was the MIP Solver for most of the instances.

Table 5.6: Summary of Results – BCP-PMWT-OTI Best Procedure

		BCP-PMWT				BCP-PMWT-OTI			
n	m	Root	BCP	ATIF MIP	Unsolved	Root	BCP	TIF MIP	Unsolved
40		38	2	10	0	38	0	12	0
50		33	4	13	0	33	2	15	0
100	2	13	2	3	7	15	2	4	4
100	4	7	5	5	8	7	1	14	3

6 CONCLUSION

This work proposed a set of improvements to the BCP algorithm of (Pessoa et al, 2010), referred as BCP-PMWT throughout the text. The resulting algorithm, referred to as BCP-PMWT-OTI, included a new family of cuts, proposed and proved valid here, along with a genetic algorithm for separation. Also, we projected the Arc-Time-Indexed formulation to the Time-Indexed formulation for solving with a MIP Solver, which we consider the main contribution of this work. For the 40 instances that could be solved both by feeding the ATIF and the TIF to the MIP solver, the average time required for solving the TIF decreases 93.74% from the time required to solve the ATIF, even with the TIF bounds being worse than the ATIF bounds.

The improvements proposed then greatly improved the time to solve the $P||\sum w_j T_j$ problem, and solved 8 new instances, of 150, leaving 7 instances still not solved.

For future works, a few suggestions are:

- Develop a faster separation procedure for the OECs.
- Explore if this framework of variable fixation on a extended formulation and projection onto a more tractable one could be applied to other combinatorial optimization problems.
- Explore new branching schemes for the BCP algorithm.
- Translate the RHECCs, Triangle Clique Cuts and OECs to the Time-Indexed Formulation.

BIBLIOGRAPHY

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

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

DASH, S.; FUKASAWA, R.; GÜNLÜK, O. On a generalization of the master cyclic group polyhedron. *Mathematical Programming*, v. 125, p. 1–30, 2010.

DYER, M. E.; WOLSEY, L. A. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, v. 26, n. 2, p. 255–270, 1990.

GANTT, H. *Work, Wages, and Profits: Their Influence on the Cost of Living*. Library of American civilization. Engineering magazine, 1910.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer US, 2003.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, v. 5, p. 287–326, 1979.

IBARAKI, T. Successive sublimation methods for dynamic programming computation. *Annals of Operations Research*, v. 11, n. 1, p. 398–439i, 1987.

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

KRAMER, A.; SUBRAMANIAN, A. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems. *CoRR*, v. abs/1509.02384. URL `http://arxiv.org/abs/1509.02384`, 2015.

MARCUS POGGI DE ARAGÃO, E. U. Integer Program Reformulation for Robust Branch-and-Cut-and-Price Algorithms. *In Proceedings of the Conference Mathematical Program in Rio: A Conference in Honour of Nelson Maculan*, v. , p. 56–61 p., 2003.

PESSOA, A.; UCHOA, E.; DE ARAGÃO, M.; RODRIGUES, R. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, v. 2, n. 3-4, p. 259–290, 2010.

PESSOA, A.; UCHOA, E.; DE ARAGÃO, M. P. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, v. , 2009.

PINEDO, M. L. *Scheduling: theory, algorithms, and systems*. Springer, 2012.

POTTS, C. N.; WASSENHOVE, L. N. V. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, v. , 1985.

RODRIGUES, R.; PESSOA, A.; UCHOA, E.; POGGI DE ARAGÃO, M. Heuristic algorithm for the parallel machine total weighted tardiness scheduling problem. v. , 2008.

TANAKA, S.; ARAKI, M. A branch-and bound algorithm based on lagrangian relaxation for single machine scheduling. *Proceedings of International Symposium on Scheduling*, volume 18–20, p. 148–153, 2006.

TANAKA, S.; ARAKI, M. An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Computers & Operations Research*, v. 40, n. 1, p. 344–352, 2013.

TANAKA, S.; FUJIKUMA, S. A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *Journal of Scheduling*, 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*, v. 12, n. 6, p. 575–593, 2009.

UCHOA, E.; FUKASAWA, R.; LYSGAARD, J.; PESSOA, A.; DE ARAGÃO, M. P.; ANDRADE, D. Robust branch-cut-and-price for the Capacitated Minimum Spanning Tree problem over a large extended formulation. *Mathematical Programming*, v. 112, n. 2, p. 443–472, 2006.

WENTGES, P. Weighted dantzig–wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, v. 4, n. 2, p. 151–162, 1997.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing*, v. 4, n. 2, p. 65–85, 1994.

Appendices

A ROOT NODE RESULTS

Table A.1: Root relaxation and cut separation results

					BCP-PMWT-OTI			BCP-PMWT		
n	m	Inst	Heu UB	First LB	Root LB	Gap	Time	Root LB	Gap	Time
40	2	1	606	583.14	606		126.10	606		63.50
40	2	6	3886	3874.25	3886		56.90	3886		11.80
40	2	11	9617	9591.26	9617		32.60	9617		6.40
40	2	16	38356	38278.10	38356		128.40	38356		41.80
40	2	31	3812	3757.50	3812		89.90	3812		29.20
40	2	36	10713	10661.60	10713		152.00	10713		50.70
40	2	56	1279	1271.86	1279		22.50	1279		8.30
40	2	61	11488	11310.50	**11459**	0.252%	569.40	11394	0.818%	684.80
40	2	66	35279	35129.40	**35203**	0.215%	428.50	35197	0.232%	200.60
40	2	71	47952	47934.30	47952		25.50	47952		0.90
40	2	81	571	451.68	571		203.70	571		67.60
40	2	86	6048	5995.50	6048		230.00	6048		31.90
40	2	96	66116	66110.00	66116		15.40	66116		1.20
40	2	111	17936	17897.20	17936		83.80	17936		3.50
40	2	116	25870	25785.30	25870		88.20	25870		44.40
40	2	121	64516	64506.90	64516		12.00	64516		1.00
40	4	1	439	437.60	439		24.70	439		1.70
40	4	6	2374	2371.85	2374		33.00	2374		2.30
40	4	11	5737	5734.78	5737		9.40	5737		0.50
40	4	16	21493	21483.60	**21493**		368.10	21490	0.014%	19.90
40	4	31	2525	2495.27	**2507**	0.713%	147.80	2500	0.990%	39.30
40	4	36	6420	6354.94	**6368**	0.810%	351.80	6364	0.872%	26.50
40	4	41	17685	17633.30	**17639**	0.260%	175.60	17637	0.271%	24.30

Table A.1: continued

					BCP-PMWT-OTI			BCP-PMWT		
n	m	Inst	Heu UB	First LB	Root LB	Gap	Time	Root LB	Gap	Time
40	4	56	826	797.20	817	1.090%	105.50	817	1.090%	45.50
40	4	61	7357	7315.47	**7326**	0.421%	304.00	7322	0.476%	22.80
40	4	66	20251	20247.00	20251		32.50	20251		1.70
40	4	81	564	549.46	560	0.709%	162.80	560	0.709%	37.90
40	4	86	4725	4719.00	4725		79.50	4725		6.80
40	4	91	15569	15556.10	15562	0.045%	138.90	15562	0.045%	27.00
40	4	111	11263	11211.20	**11221**	0.373%	175.80	11219	0.391%	58.60
40	4	116	15566	15538.50	**15546**	0.128%	109.90	15545	0.135%	19.70
40	4	121	35751	35739.00	35740	0.031%	88.60	**35741**	0.028%	39.10
50	2	1	1268	1231.97	1268		204.30	1268		296.60
50	2	6	14272	14260.60	14272		156.40	14272		63.10
50	2	11	23028	22999.90	23028		100.70	23028		21.30
50	2	16	46072	46010.30	46072		139.60	46072		25.10
50	2	21	111069	111066.00	111069		24.00	111069		0.70
50	2	31	5378	5288.13	5342	0.669%	413.60	**5349**	0.539%	1060.20
50	2	36	18956	18894.80	18956		141.20	18956		26.80
50	2	41	38058	37967.40	38041	0.045%	704.50	**38050**	0.021%	341.60
50	2	46	82105	82084.60	82105		121.20	82105		31.50
50	2	56	761	729.27	761		532.40	761		232.70
50	2	61	13682	13588.60	**13682**		2300.20	13619	0.460%	614.00
50	2	66	40907	40903.10	40907		24.60	40907		0.60
50	2	81	542	537.66	542		26.80	542		5.30
50	2	86	12557	12276.20	**12461**	0.765%	844.80	12427	1.035%	946.70
50	2	91	47349	47293.80	**47338**	0.023%	636.70	47330	0.040%	492.30
50	2	96	92822	92802.00	92822		49.10	92822		6.30
50	2	111	15564	15543.20	15564		20.40	15564		3.50
50	2	116	19608	19523.80	**19579**	0.148%	826.20	19573	0.178%	453.30
50	4	1	785	776.87	785		112.10	785		24.80
50	4	6	8317	8297.16	8304	0.156%	129.10	8304	0.156%	110.80
50	4	11	12879	12870.90	**12876**	0.023%	171.80	12875	0.031%	71.20
50	4	16	25376	25374.70	25376		54.80	25376		8.00
50	4	36	10796	10793.20	10796		25.40	10796		5.00
50	4	41	21806	21782.60	**21786**	0.092%	305.10	21785	0.096%	44.90
50	4	46	44455	44451.80	44453	0.004%	131.90	**44455**		15.10

Table A.1: continued

					BCP-PMWT-OTI			BCP-PMWT		
n	m	Inst	Heu UB	First LB	Root LB	Gap	Time	Root LB	Gap	Time
50	4	56	570	537.64	540	5.263%	58.20	**541**	5.088%	45.80
50	4	61	7898	7849.71	7857	0.519%	260.50	7857	0.519%	81.80
50	4	71	42645	42624.80	42628	0.040%	245.10	42628	0.040%	138.40
50	4	81	495	477.91	495		108.20	495		64.90
50	4	86	8369	8329.27	8336	0.394%	313.90	8336	0.394%	163.30
50	4	91	26551	26546.00	**26551**		69.70	26548	0.011%	111.60
50	4	96	50326	50311.70	50317	0.018%	256.90	50317	0.018%	91.00
50	4	111	10069	10048.80	10051	0.179%	196.10	10051	0.179%	63.20
50	4	116	11552	11519.40	11523	0.251%	147.60	11523	0.251%	54.40
50	4	121	23792	23768.20	**23776**	0.067%	219.70	23775	0.071%	58.50
100	2	1	3339	3313.40	**3333**	0.180%	3538.80	3322	0.509%	814.60
100	2	6	30665	30644.00	30665		449.30	30665		328.10
100	2	11	93894	111977.00	93894		0.00	93894		17.20
100	2	16	209100	209062.00	209100		122.90	209100		5086.70
100	2	21	457836	457814.00	457836		229.30	457836		24.30
100	2	31	12729	12725.00	12729		146.20	12729		210.50
100	2	36	56671	56575.00	**56620**	0.090%	4281.40	56590	0.143%	804.70
100	2	41	237964	237772.00	**237881**	0.035%	6148.40	237841	0.052%	19685.20
100	2	46	422831	422804.00	**422831**		451.30	422830	0.000%	1525.30
100	2	56	5047	4982.07	5047		819.00	5047		796.90
100	2	61	45573	45423.10	**45488**	0.187%	4809.20	45481	0.202%	6818.20
100	2	66	126513	126408.00	**126492**	0.017%	4567.90	126477	0.028%	6969.70
100	2	71	327305	327300.00	327305		71.40	327305		3.10
100	2	81	908	791.69	**908**		1204.00	830	8.590%	968.90
100	2	86	36581	36217.20	**36402**	0.489%	3790.00	36322	0.708%	3234.00
100	2	91	129929	129619.00	**129795**	0.103%	10037.90	129752	0.136%	3617.10
100	2	96	254194	254140.00	254194		440.30	254194		400.20
100	2	111	84220	84004.10	**84138**	0.097%	6828.30	84097	0.146%	6578.00
100	2	116	191186	191085.00	**191178**	0.004%	5317.00	191173	0.007%	11118.60
100	2	121	242018	241953.00	**241998**	0.008%	149458.60	241997	0.009%	56938.90
100	4	1	2001	1989.28	2001		626.70	2001		668.10
100	4	11	50232	50198.00	50203	0.058%	625.00	50203	0.058%	951.30
100	4	16	110219	110114.00	**110120**	0.090%	395.60	110118	0.092%	950.60
100	4	21	237392	237388.00	237388	0.002%	457.20	**237390**	0.001%	1047.80

Table A.1: continued

					BCP-PMWT-OTI			BCP-PMWT		
n	m	Inst	Heu UB	First LB	Root LB	Gap	Time	Root LB	Gap	Time
100	4	31	7130	7080.00	7082	0.673%	500.80	7082	0.673%	1156.70
100	4	36	30791	30772.10	30782	0.029%	1097.70	**30783**	0.026%	1654.20
100	4	41	126185	126130.00	**126146**	0.031%	1229.30	126144	0.032%	923.40
100	4	46	219536	219526.00	219529	0.003%	567.00	219529	0.003%	910.00
100	4	56	3076	3020.93	3030	1.495%	867.40	3030	1.495%	1102.50
100	4	61	24856	24805.30	**24824**	0.129%	496.70	24821	0.141%	1094.90
100	4	66	67970	67947.00	**67955**	0.022%	973.10	67954	0.024%	819.70
100	4	71	170691	170673.00	170675	0.009%	766.30	170675	0.009%	398.80
100	4	81	819	753.51	**785**	4.151%	849.90	772	5.739%	1242.50
100	4	86	21286	21208.30	**21220**	0.310%	614.40	21218	0.319%	798.60
100	4	91	70608	70581.50	70587	0.030%	941.30	**70589**	0.027%	1640.70
100	4	96	133587	133572.00	133574	0.010%	588.20	**133575**	0.009%	443.80
100	4	111	46719	46615.20	**46633**	0.184%	1025.70	46630	0.191%	1094.30
100	4	116	101551	101513.00	101520	0.031%	505.10	101520	0.031%	871.70
100	4	121	127619	127592.00	127597	0.017%	1190.40	127597	0.017%	926.70

B DETAILED TIME-INDEXED FORMULATIONS PERFORMANCE

Table B.1: Comparison of Alternative Time-Indexed Formulations

			LP Time(s)				MIP Time (s)			
n	m	Inst	Fy	My	Fz	Mz	Fy	My	Fz	Mz
40	2	61	0.78	0.57	7.05	1.15	235.6	1379.7	712.1	238.9
40	2	66	2.72	2.75	43.87	3.01	218.4	1350.1	460.3	316.7
40	4	31	0.47	0.93	10.30	0.82	996.8	≥3600	57.4	32.0
40	4	36	0.38	0.37	4.25	0.40	140.8	676.4	15.7	7.1
40	4	41	0.44	0.51	4.47	0.58	13.3	34.9	4.5	4.7
40	4	56	0.46	1.21	8.33	1.54	28.0	≥3600	340.0	52.9
40	4	61	0.18	0.22	1.32	0.39	12.3	44.3	13.8	5.8
40	4	81	0.05	0.06	0.27	0.08	0.1	0.7	0.2	0.6
40	4	91	0.01	0.04	0.03	0.02	0.1	0.3	0.3	0.2
40	4	111	0.19	0.23	1.78	0.46	5.6	19.3	11.1	3.7
40	4	116	0.14	0.19	0.99	0.22	4.7	9.3	7.9	3.1
40	4	121	0.12	0.34	0.70	0.19	1.0	4.6	3.3	2.0
50	2	31	10.56	9.29	515.22	13.62	≥3600	≥3600	≥3600	≥3600
50	2	41	1.54	2.33	18.56	1.80	3.9	18.0	30.3	12.9
50	2	61	3.44	2.30	94.61	3.39	≥3600	≥3600	296.3	26.4
50	2	86	2.28	1.42	38.53	4.04	2631.5	≥3600	≥3600	2109.8
50	2	91	1.09	1.31	7.24	1.71	6.1	30.8	50.8	23.6
50	2	116	1.20	1.28	20.39	1.70	≥3600	≥3600	≥3600	3066.9
50	4	6	0.86	2.08	51.40	1.42	1001.5	≥3600	9.2	16.9

Table B.1: continued

			LP Time(s)				MIP Time (s)			
n	m	Inst	Fy	My	Fz	Mz	Fy	My	Fz	Mz
50	4	11	0.52	0.63	4.61	0.57	4.7	15.4	49.5	11.4
50	4	41	0.56	1.10	7.67	0.67	4.3	29.8	25.9	15.1
50	4	46	0.01	0.05	0.04	0.03	0.1	0.1	0.1	0.1
50	4	56	1.24	3.20	93.93	5.61	$\geq$3600	$\geq$3600	2352.9	317.9
50	4	61	0.98	0.76	16.16	0.79	$\geq$3600	$\geq$3600	260.4	804.5
50	4	71	0.62	0.88	5.30	0.46	3.1	14.1	26.2	10.7
50	4	86	0.21	0.31	1.67	0.34	27.4	46.9	9.6	3.1
50	4	91	0.05	0.18	0.22	0.10	0.2	0.3	0.7	0.2
50	4	96	0.39	0.71	3.37	0.31	8.8	23.2	13.3	17.0
50	4	111	0.15	0.19	0.96	0.44	1.8	3.6	3.6	3.1
50	4	116	0.43	0.47	3.74	0.53	489.8	1415.7	558.1	76.0
50	4	121	0.49	0.94	5.01	0.48	38.2	55.3	8.8	8.2

Table B.2: Effect of Variable Fixation in the Mz Time-Index Formulation

			First LP Time (s)		MIP Time (s)	
n	m	Inst	Fix.	w/ Fix.	Fix.	w/ Fix.
40	2	61	1.16	83.16	232.8	$\geq$3600
40	2	66	3.03	68.65	389.4	$\geq$3600
40	4	31	0.81	15.28	11.4	698.4
40	4	36	0.40	12.70	16.2	2562.7
40	4	41	0.57	15.05	6.1	70.0
40	4	56	1.54	10.72	59.8	1528.7
40	4	61	0.39	11.36	6.8	157.2
40	4	81	0.08	18.15	0.4	44.6
40	4	91	0.02	9.06	0.2	50.3
40	4	111	0.45	14.55	6.0	296.1
40	4	116	0.22	8.22	2.0	106.4

Table B.2: continued

			First LP Time (s)		MIP Time (s)	
n	m	Inst	Fix.	w/ Fix.	Fix.	w/ Fix.
40	4	121	0.19	3.56	2.1	101.5
50	2	31	13.71	717.17	3010.9	≥3600
50	2	41	1.80	193.44	20.3	≥3600
50	2	86	4.03	257.36	2616.2	≥3600
50	2	91	1.71	172.52	31.0	≥3600
50	2	116	1.70	120.13	3081.3	≥3600
50	4	6	1.43	71.88	12.8	1921.7
50	4	11	0.56	17.88	15.4	394.7
50	4	41	0.67	19.62	18.7	193.5
50	4	46	0.03	3.56	0.1	9.3
50	4	56	5.62	20.79	252.5	≥3600
50	4	61	0.79	70.42	97.1	≥3600
50	4	71	0.46	10.51	17.7	157.9
50	4	86	0.35	37.98	5.3	475.7
50	4	96	0.32	6.87	14.1	162.7
50	4	111	0.45	50.12	8.4	907.0
50	4	116	0.53	13.87	395.6	≥3600
50	4	121	0.48	15.13	12.2	246.9

Table B.3: Effect of Projected Cuts in the Mz Time-Indexed Formulation

n	m	Inst	Heu UB	ATIF Root LB	ATIF Root Gap	1st TIF LP LB	1st TIF LP Gap	Root LB	Root Gap	Gap Improv.
100	2	1	3339	3333	0.180%	3314	0.749%	3314	0.749%	0.00%
100	2	36	56671	56620	0.090%	56557	0.201%	56575	0.169%	15.79%
100	2	41	237964	237881	0.035%	237734	0.097%	237771	0.081%	16.09%
100	2	61	45573	45488	0.187%	45370	0.445%	45390	0.402%	9.85%
100	2	66	126513	126492	0.017%	126374	0.110%	126405	0.085%	22.30%
100	2	86	36581	36402	0.489%	36171	1.121%	36217	0.995%	11.22%
100	2	91	129929	129795	0.103%	129557	0.286%	129618	0.239%	16.40%

Table B.3: continued

n	m	Inst	Heu UB	ATIF Root LB	ATIF Root Gap	1st TIF LP LB	1st TIF LP Gap	Root LB	Root Gap	Gap Improv.
100	2	111	84220	84138	0.097%	83970	0.297%	84004	0.256%	13.60%
100	2	116	191186	191178	0.004%	191072	0.060%	191084	0.053%	10.53%
100	2	121	242018	241998	0.008%	241937	0.033%	241953	0.027%	19.75%
100	4	11	50232	50203	0.058%	50194	0.076%	50198	0.068%	10.53%
100	4	16	110219	110120	0.090%	110102	0.106%	110111	0.098%	7.69%
100	4	21	237392	237388	0.002%	237388	0.002%	237388	0.002%	0.00%
100	4	31	7130	7082	0.673%	7080	0.701%	7080	0.701%	0.00%
100	4	36	30791	30782	0.029%	30773	0.058%	30773	0.058%	0.00%
100	4	41	126185	126146	0.031%	126126	0.047%	126130	0.044%	6.78%
100	4	46	219536	219529	0.003%	219525	0.005%	219526	0.005%	9.09%
100	4	56	3076	3030	1.495%	3021	1.788%	3021	1.788%	0.00%
100	4	61	24856	24824	0.129%	24795	0.245%	24806	0.201%	18.03%
100	4	66	67970	67955	0.022%	67938	0.047%	67947	0.034%	28.13%
100	4	71	170691	170675	0.009%	170671	0.012%	170673	0.011%	10.00%
100	4	81	819	785	4.151%	754	7.937%	754	7.937%	0.00%
100	4	86	21286	21220	0.310%	21180	0.498%	21209	0.362%	27.36%
100	4	91	70608	70587	0.030%	70575	0.047%	70582	0.037%	21.21%
100	4	96	133587	133574	0.010%	133561	0.019%	133572	0.011%	42.31%
100	4	111	46719	46633	0.184%	46608	0.238%	46615	0.223%	6.31%
100	4	116	101551	101520	0.031%	101510	0.040%	101513	0.037%	7.32%
100	4	121	127619	127597	0.017%	127590	0.023%	127592	0.021%	6.90%

Table C.1: Detailed Results for $m = 2$ and $n = 40$ instances

	BCP-PMWT							BCP-PMWT-OTI						
Inst	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	606	606	80.90			Root	80.9	606	606	128.30			Root	128.30
6	3886	3886	27.70			Root	27.7	3886	3886	59.60			Root	59.60
11	9617	9617	22.30			Root	22.3	9617	9617	34.80			Root	34.80
16	38356	38356	59.10			Root	59.1	38356	38356	131.70			Root	131.70
31	3812	3812	49.20			Root	49.2	3812	3812	92.40			Root	92.40
36	10713	10713	65.70			Root	65.7	10713	10713	155.00			Root	155.00
56	1279	1279	24.30			Root	24.3	1279	1279	24.70			Root	24.70
61	11488	11394	705.70	2398.50	1994.00	MIP	2699.7	11488	**11459**	573.20	531.30	237.67	MIP	810.87
66	35279	35197	220.40	307.80	1776.00	BCP	528.2	35279	**35203**	431.30	774.40	86.47	MIP	517.77
71	47952	47952	15.60			Root	15.6	47952	47952	28.20			Root	28.20
81	573	571	83.70			Root	83.7	**571**	571	207.40			Root	207.40
86	6048	6048	46.20			Root	46.2	6048	6048	233.10			Root	233.10
96	66116	66116	22.80			Root	22.8	66116	66116	18.30			Root	18.30
111	17936	17936	28.10			Root	28.1	17936	17936	86.70			Root	86.70
116	25874	25870	66.60			Root	66.6	**25870**	25870	91.40			Root	91.40
121	64516	64516	24.30			Root	24.3	64516	64516	14.80			Root	14.80

Table C.2: Detailed Results for $m = 4$ and $n = 40$ instances

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	439	439	11.40			Root	11.4	439	439	26.30			Root	26.30
6	2374	2374	11.10			Root	11.1	2374	2374	35.10			Root	35.10
11	5737	5737	10.30			Root	10.3	5737	5737	11.40			Root	11.40
16	21493	21490	30.40			Root	30.4	21493	**21493**	370.10			Root	370.10
31	2525	2500	50.10	2932.60	12172.30	BCP	2982.7	2525	**2507**	149.60	100.00	12.32	MIP	161.92
36	6420	6364	37.30	18799.50	2829.70	MIP	2867	6420	**6368**	354.30	31068.00	5.96	MIP	360.26
41	17685	17637	36.10	1856.00	128.00	MIP	164.1	17685	**17639**	178.10	1293.30	3.23	MIP	181.33
56	826	817	54.60	>86400	878.00	MIP	932.6	826	817	107.30	1120.60	49.65	MIP	156.95
61	7357	7322	33.80	230.20	143.00	MIP	176.8	7357	**7326**	306.40	380.50	4.70	MIP	311.10
66	20251	20251	12.90			Root	12.9	20251	20251	34.10			Root	34.10
81	565	560	46.40	9.70	0.90	MIP	47.3	**564**	560	165.20	6.10	0.04	MIP	165.24
86	4725	4725	15.30			Root	15.3	4725	4725	81.50			Root	81.50
91	15569	15562	39.40	9.60	0.70	MIP	40.1	15569	15562	141.00	0.50	0.07	MIP	141.07
111	11263	11219	72.00	303.90	105.30	MIP	177.3	11263	**11221**	177.60	220.70	5.54	MIP	183.14
116	15566	15545	31.30	101.40	40.40	MIP	71.7	15566	**15546**	112.20	197.70	2.93	MIP	115.13
121	35751	**35741**	52.00	42.60	4.70	MIP	56.7	35751	35740	90.60	60.60	1.28	MIP	91.88

Table C.3: Detailed Results for $m = 2$ and $n = 50$ instances

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	1268	1268	345.80			Root	345.8	1268	1268	209.30			Root	209.30
6	14272	14272	122.80			Root	122.8	14272	14272	162.10			Root	162.10
11	23028	23028	67.10			Root	67.1	23028	23028	106.50			Root	106.50
16	46072	46072	59.40			Root	59.4	46072	46072	144.90			Root	144.90
21	111069	111069	36.10			Root	36.1	111069	111069	29.30			Root	29.30
31	5378	**5349**	1110.20	11608.90	11997.50	BCP	12719.1	5378	5342	419.50	1064.60	4698.26	BCP	1484.10
36	18956	18956	73.30			Root	73.3	18956	18956	146.70			Root	146.70
41	38058	**38050**	381.50	16.70	43.50	BCP	398.2	38058	38041	710.80	194.20	15.99	MIP	726.79
46	82105	82105	66.00			Root	66	82105	82105	126.00			Root	126.00
56	761	761	267.40			Root	267.4	761	761	536.70			Root	536.70
61	13682	13619	661.40	83122.80	36094.60	MIP	36756	13682	**13682**	2306.40			Root	2306.40
66	40907	40907	37.50			Root	37.5	40907	40907	29.10			Root	29.10
81	542	542	40.50			Root	40.5	542	542	31.60			Root	31.60
86	12557	12427	999.30	13817.70	23070.00	BCP	14817	12557	**12461**	850.60	4137.30	1894.57	MIP	2745.17
91	47349	47330	552.60	210.60	204.50	MIP	757.1	47349	**47338**	641.30	126.80	26.55	MIP	667.85
96	92822	92822	60.90			Root	60.9	92822	92822	54.00			Root	54.00
111	15564	15564	66.60			Root	66.6	15564	15564	25.80			Root	25.80
116	19609	19573	508.80	2390.70	8539.50	BCP	2899.5	**19608**	**19579**	831.20	1196.00	2917.71	BCP	2027.20

Table C.4: Detailed Results for $m = 4$ and $n = 50$ instances

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	785	785	56.40			Root	56.4	785	785	115.50			Root	115.50
6	8317	8304	147.10	>86400	78140.00	MIP	78287.1	8317	8304	132.50	2361.90	21.42	MIP	153.92
11	12879	12875	98.90	5691.60	43.80	MIP	142.7	12879	**12876**	175.40	207.50	9.77	MIP	185.17
16	25376	25376	31.20			Root	31.2	25376	25376	58.40			Root	58.40
36	10796	10796	35.10			Root	35.1	10796	10796	28.90			Root	28.90
41	21806	21785	72.70	232.40	68.10	MIP	140.8	21806	**21786**	308.30	844.80	21.58	MIP	329.88
46	44455	**44455**	40.50			Root	40.5	44455	44453	135.40	0.10	0.05	MIP	135.45
56	570	**541**	67.80	>86400	18126.00	MIP	18193.8	570	540	61.80	>14406.0	449.66	MIP	511.46
61	7898	7857	113.70	>86400	27997.40	MIP	28111.1	7898	7857	264.20	5413.20	840.28	MIP	1104.48
71	42645	42628	218.80	336.60	42.30	MIP	261.1	42645	42628	247.90	317.90	17.08	MIP	264.98
81	495	495	118.90			Root	118.9	495	495	111.80			Root	111.80
86	8369	8336	241.30	575.20	355.60	MIP	596.9	8369	8336	317.60	188.00	4.82	MIP	322.42
91	26552	26548	187.60			Root	187.6	**26551**	**26551**	73.20			Root	73.20
96	50326	50317	183.20	261.10	32.40	MIP	215.6	50326	50317	260.10	279.50	10.37	MIP	270.47
111	10069	10051	134.60	52.00	43.80	MIP	178.4	10069	10051	199.80	89.40	4.85	MIP	204.65
116	11552	11523	114.30	45693.50	4087.20	MIP	4201.5	11552	11523	151.30	30046.20	325.91	MIP	477.21
121	23792	23775	85.30	438.50	248.50	MIP	333.8	23792	**23776**	223.40	314.40	6.72	MIP	230.12

Table C.5: Detailed Results for $m = 2$ and $n = 100$ instances

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	3339	3322	1606.00	>14400				3339	**3333**	3593.70	2493.10	≥3600	BCP	6086.80
6	30665	30665	1001.50			Root	1001.5	30665	30665	493.40			Root	493.40
11	93894	93894	613.00			Root	613	93894	93894	40.20			Root	40.20
16	209100	209100	5603.60			Root	5603.6	209100	209100	177.10			Root	177.10
21	457836	457836	1248.70			Root	1248.7	457836	457836	391.20			Root	391.20
31	12729	12729	927.70			Root	927.7	12729	12729	186.10			Root	186.10
36	56671	56590	1435.70	>14400				56671	**56620**	4324.90	>9324.2	22506.30	MIP	26831.20
41	237964	237841	20436.30	>20436	120721.70	MIP	141158	237964	**237881**	6193.20	13499.10	32047.52	BCP	19692.30
46	422831	422830	2077.90	44.10	51.90	BCP	2122	422831	**422831**	559.20			Root	559.20
56	5047	5047	1582.30			Root	1582.3	5047	5047	865.50			Root	865.50
61	45573	45481	7353.80	>14400	>86400			45573	**45488**	4852.30	>8859.2	≥172800		
66	126522	126477	7526.50	23425.60	>86400	BCP	30952.1	**126513**	**126492**	4613.90	7398.40	5413.94	MIP	10027.84
71	327305	327305	630.20			Root	630.2	327305	327305	118.30			Root	118.30
81	908	830	1299.20	>14400				908	**908**	1263.00			Root	1263.00
86	36581	36322	3905.20	>14400	>86400			36581	**36402**	3845.00	>6458.4	≥172800		
91	129931	129752	4259.40	>14400	>86400			**129929**	**129795**	10093.70	>2932.0	≥172800		
96	254194	254194	1030.50			Root	1030.5	254194	254194	478.30			Root	478.30
111	84274	84097	7424.30	>14400	>86400			**84220**	**84138**	6880.00	>7323.3	≥172800		
116	191198	191173	11740.20	>14400	32567.70	MIP	44307.9	**191186**	**191178**	5350.80	1983.00	1540.90	MIP	6891.70

Table C.5: continued

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
121	242022	241997	57559.60	>86400	4072.40	MIP	61632	**242018**	**241998**	149499.40	>0.3	1398.56	MIP	150897.96

Table C.6: Detailed Results for $m = 4$ and $n = 100$ instances

Inst	BCP-PMWT							BCP-PMWT-OTI						
	UB	Root LB	Root Time	BCP Time	ATIF MIP Time	Best	Overall Time	UB	Root LB	Root Time	BCP Time	TIF MIP Time	Best	Overall Time
1	2001	2001	1337.30			Root	1337.3	2001	2001	679.90			Root	679.90
11	50236	50203	1439.40	>14400				**50232**	50203	659.50	>13290.5	3571.20	MIP	4230.70
16	110222	110118	1449.20	>14400	>86400			**110219**	**110120**	429.20	>13858.9	≥172800		
21	237392	**237390**	1493.80	5.50	29.00	BCP	1499.3	237392	237388	479.50	7.80	0.55	MIP	480.05
31	7130	7082	1901.50	>14400				7130	7082	548.30	>15316.0	≥3600		
36	30791	**30783**	2201.30	>14400				30791	30782	1127.90	>13459.9	190.75	MIP	1318.65
41	126193	126144	1493.60	11041.30	41682.90	BCP	12534.9	**126185**	**126146**	1251.30	6303.50	435.71	MIP	1687.01
46	219537	219529	1372.00	256.30	800.90	BCP	1628.3	**219536**	219529	590.30	298.10	25.27	MIP	615.57
56	3076	3030	1727.30	>14400				3076	3030	914.50	>13807.4	≥3600		
61	24868	24821	1682.50	>14400	>86400			**24856**	**24824**	525.00	>85392.2	751.04	MIP	1276.04
66	67979	67954	1267.70	2885.70	3738.20	BCP	4153.4	**67970**	**67955**	997.40	907.40	232.73	MIP	1230.13
71	170699	170675	862.90	>14400	81028.10	MIP	81891	**170691**	170675	790.10	84914.00	1299.60	MIP	2089.70
81	819	772	1594.40	>14400		MIP	0	819	**785**	900.70	916.80	1233.73	BCP	1817.50
86	21299	21218	1379.00	>14400	>86400			**21286**	**21220**	646.50	>13445.9	11857.18	MIP	12503.68
91	70612	**70589**	2119.40	1436.30	7318.00	BCP	3555.7	**70608**	70587	966.10	2009.20	210.80	MIP	1176.90
96	133591	**133575**	878.60	3156.30	1930.30	MIP	2808.9	**133587**	133574	614.70	313.10	58.09	MIP	672.79
111	46763	46630	1704.60	>14400	>86400			**46719**	**46633**	1051.10	>13512.4	57566.09	MIP	58617.19
116	101563	101520	1374.60	>14400	156440.10	MIP	157814.7	**101551**	101520	527.40	>85299.3	2612.53	MIP	3139.93
121	127639	127597	1389.60	>14400	108063.90	MIP	109453.5	**127619**	127597	1213.00	18756.60	2990.62	MIP	4203.62