UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

JOÃO MARCOS PEREIRA SILVA

ABORDAGENS EXATA E HEURÍSTICA PARA O PROBLEMA DE ESCALONAMENTO EM MÁQUINAS PARALELAS IDÊNTICAS COM SERVIDOR ÚNICO DE SETUP

JOÃO PESSOA
2019

JOÃO MARCOS PEREIRA SILVA

ABORDAGENS EXATA E HEURÍSTICA PARA O PROBLEMA DE ESCALONAMENTO EM MÁQUINAS PARALELAS IDÊNTICAS COM SERVIDOR ÚNICO DE SETUP

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal da Paraíba, como requisito parcial para a obtenção do Grau de Mestre em Engenharia de Produção. Área: Gestão da produção. Linha de Pesquisa: Gestão e otimização dos sistemas de produção

Orientador: Prof. Dr. Anand Subramanian

JOÃO PESSOA
2019

Catalogação na publicação
Seção de Catalogação e Classificação

S586a Silva, João Marcos Pereira.
Abordagens exata e heurística para o problema de escalonamento em máquinas paralelas idênticas com servidor único de setup / João Marcos Pereira Silva. - João Pessoa, 2019.
78 f. : il.

Orientação: Anand Subramanian.
Dissertação (Mestrado) - UFPB/CT.

1. Sequenciamento da produção. 2. Servidor único de setup. 3. Programação linear inteira. 4. Meta-heurística. I. Subramanian, Anand. II. Título.

UFPB/BC

JOÃO MARCOS PEREIRA SILVA

ABORDAGENS EXATA E HEURÍSTICA PARA O PROBLEMA DE ESCALONAMENTO EM MÁQUINAS PARALELAS IDÊNTICAS COM SERVIDOR ÚNICO DE *SETUP*

Dissertação apresentada como requisito à obtenção do grau de Mestre em Engenharia de Produção do Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal da Paraíba, pela comissão formada pelos seguintes membros.

BANCA EXAMINADORA

Prof. D.Sc. Anand Subramanian — Orientador
Departamento de Engenharia de Produção — UFPB

Prof. D.Sc. Hugo Harry Frederico Ribeiro Kramer — Examinador Interno
Departamento de Engenharia de Produção — UFPB

Prof. D.Sc. Eduardo Uchoa Barboza— Examinador Externo
Departamento de Engenharia de Produção — UFF

João Pessoa, 20 de Agosto de 2019.

À minha família, aos meus amigos.

Agradecimentos

Aos meus pais e irmãos, pelo apoio incondicional.

Ao meu orientador, Anand Subramanian por todos os ensinamentos e conselhos sobre pesquisa e a vida e também por toda a confiança que me foi conferida.

Aos amigos e colegas do laboratório: Bia, Geraldo, Eduardo, Bruno Bruck, Teobaldo, Rubão, Carlos, Igor, Bruno War, Rodrigo, Luan, Felipe e Anderson, pelos auxílios e momentos de descontração. E em especial, a Ewerton pela grande ajuda nesse trabalho.

Aos amigos do mestrado, especialmente Ialy, George e Luciano pelos momentos de companheirismo e descontração.

À Capes pela bolsa concedida.

Á todos que de alguma forma contribuíram para a realização deste trabalho.

Resumo

As expectativas dos clientes, em termos de qualidade, custo e prazos de entrega de produtos e/ou serviços, elevam cada vez mais a competitividade no ambiente empresarial atual. Por isso, se faz importante a utilização de conceitos modernos de gerenciamento do processo de produção com intuito de decidir a melhor forma de alocação e utilização dos recursos disponíveis. O gerenciamento adequado de recursos é ainda mais necessário em ambientes de produção em que a produtividade é significativamente afetada pelos tempos de preparação e pelos recursos associados as equipes de *setup*. Este trabalho lida com o problema de escalonamento de máquinas paralelas com servidor único e tempos de preparação (*setup*) dependentes e independentes da sequência, cujo objetivo é minimizar o *makespan*. Nesse caso, o servidor de *setup*, que pode ser uma máquina, indivíduo ou equipe, é responsável por executar as operações de configuração. Portanto, não pode haver conflitos entre essas operações. São propostas duas formulações indexadas no tempo para o problema, bem como dois algoritmos baseados na meta-heurística de busca local iterada (ILS). Experimentos computacionais foram realizados em 350 instâncias, envolvendo até 100 tarefas, variando de 2 a 10 máquinas. Os resultados obtidos sugerem que os métodos desenvolvidos foram capazes de encontrar soluções altamente competitivas quando comparados aos obtidos pelas abordagens existentes na literatura.

Palavras-chave: Sequenciamento da produção, Servidor Único de *Setup*, Programação Linear Inteira, Meta-heurística.

Abstract

Customer expectations, in terms of quality, cost and delivery times of products and/or services, increasingly raise competitiveness in today's business environment. Therefore, it is important to use modern concepts of production process management in order to decide the best way to allocate and use available resources. Proper resource management is even more necessary in production environments where productivity is significantly affected by the preparation times and the resources associated with the setup teams. This work addresses the problem of scheduling parallel machines with a common server and sequence-dependent and independent setup times, whose objective is to minimize the makespan. In this case, the common server, which can be a machine, individual or a team, is responsible for performing the setup operations. Therefore, there must be no conflicts while conducting them. Two time-indexed formulations are proposed for the problem, as well as two algorithms based on the metaheuristic iterated local search. Computational experiments were carried out on 150 instances involving up to 100 jobs ranging from 2 to 10 machines. The results obtained suggest that the methods developed were capable of finding highly competitive solutions when compared to those achieved by existing approaches.

Keywords: Production Scheduling, Commom Server, Integer Linear Programming, Metaheuristic.

Sumário

Lista de Siglas e Abreviaturas

AG	Algoritmo Genético
ATIF	*Arc-Time-Indexed Formulation*
B&B	*Branch-and-Bound*
B&P	*Branch-and-Price*
CS	*Commom Server*
FIFO	*First In First Out*
GRASP	*Greedy Randomized Adaptive Search Procedure*
ILS	*Iterated Local Search*
LPT	*Longest Processing Time*
LB	*Lower Bound*
LS	*List Scheduling*
MDA	*Modified Decoding Algorithm*
MILP	*Mixed Integer Linear Programming*
MIP	*Mixed Integer Programming*
OC	Otimização Combinatória
ODA	*Original Decoding Algorithm*
PO	Pesquisa Operacional
RC	Resultados de Complexidade
SA	*Simulated Annealing*
SPT	*Shortest Processing Time*
TIF	*Time-Indexed Formulation*
UB	*Upper Bound*
VE	Vizinhança entre Máquinas
VI	Vizinhança intra Máquinas

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Definição do tema

Problemas de *scheduling* referem-se à alocação de recursos para a realização de tarefas de modo a garantir a conclusão dessas em um dado período de tempo (PINEDO, 2008, p. 1). Devido a sua grande importância e aplicabilidade, pode-se afirmar que tais problemas compõem uma das mais importantes e estudadas áreas no campo da Pesquisa Operacional (PO). Tal afirmação é corroborada pela *survey* desenvolvida por Potts e Strusevich (2009).

O sequenciamento de produção (*production scheduling*) está entre os problemas de *scheduling*. A relevância e o potencial de pesquisa e aplicação nessa área é enorme tanto para as empresas manufatureiras quanto para as de serviços, o que levou pesquisadores a abordar os problemas de sequenciamento de produção sob várias perspectivas nas últimas décadas (FUCHIGAMI; RANGEL, 2018).

Dentre essas perspectivas de abordagem, existem aquelas que consideram a existência de um único dispositivo, equipe ou indivíduo responsável pela realização de todas as atividades de preparação, ou *setup*, dos recursos disponíveis. Na literatura é chamado de servidor único de *setup* (do inglês, *commom server* (CS) ou *single server*). Aliado a esse, outras características são comumente encontradas, tais como a presença de múltiplas máquinas paralelas idênticas e tempos de *setup* dependentes, ou não, da sequência.

Assim como na maioria dos problemas clássicos de Otimização Combinatória (OC), esse também pode ser resolvido através da aplicação de algoritmos exatos e/ou heurísticos. O primeiro tem por objetivo encontrar soluções ótimas, enquanto que o segundo busca obter soluções aproximadas ou sub-ótimas.

Além disso, os métodos heurísticos buscam prover essas soluções em um tempo com-

putacional aceitável (SOUZA, 2011). Dentre tais, destacam-se aqueles conhecidos como "meta-heurísticas", que Osman e Laporte (1996) definem como sendo um processo iterativo responsável por guiar uma heurística subordinada, combinando de forma inteligente diferentes conceitos com o intuito de explorar e aproveitar o espaço de busca das soluções.

Diante o exposto, esse trabalho apresentará abordagens exatas e heurísticas para o problema de escalonamento em máquinas paralelas com servidor único e tempos de *setup* dependentes e independentes da sequência. A seguir, são apresentados a justificativa do estudo, bem como os objetivos a serem alcançados e a estrutura da dissertação.

1.2 Justificativa

O escalonamento da produção é uma das atividades mais importantes de uma empresa no nível operacional para se manter competitiva em mercados consumidores exigentes. A mesma está relacionada à otimização de diversas medidas de desempenho, todas voltadas para o bom funcionamento do sistema e a satisfação do cliente, tais como: uso eficiente de recursos, entrega de produtos em prazos e redução de custos de produção (FUCHIGAMI; RANGEL, 2018).

Por isso, se faz importante a utilização de conceitos modernos de gerenciamento do processo de produção com intuito de decidir a melhor forma de alocação e utilização dos recursos disponíveis. Porém, o fato de muitos desses problemas pertencerem à classe $\mathcal{NP}$-difícil torna complicada a tarefa de resolvê-los na prática.

Em ambientes produtivos caracterizados pela presença de múltiplas máquinas idênticas, a presença do servidor único para realização das atividades de preparação indica que apenas uma máquina pode passar por tal preparação por vez. Caso mais de uma máquina necessite de *setup*, enquanto uma o recebe, as demais permanecem ociosas. Esse fato reflete no aumento do custo, direto ou indireto, de processamento das tarefas, como por exemplo, no aumento do tempo máximo necessário para o processamento de todas elas, cuja terminologia em inglês é *makespan*.

Uma vez que o problema de sequenciamento de produção com duas máquinas paralelas com servidor único e tempos de *setup* dependentes da sequência, com o objetivo de minimizar o *makespan*, é $\mathcal{NP}$-difícil, então o mesmo problema para um número arbitrário de máquinas paralelas também é $\mathcal{NP}$-difícil (HAMZADAYI; YILDIZ, 2017). Portanto não é possível garantir a solução de menor custo em tempo polinomial.

O escalonamento da produção em sistemas produtivos caracterizados pela presença

de servidor único de *setup* também se insere no contexto anteriormente descrito. Sendo assim, escalonar a produção sob essas condições é uma tarefa difícil e requer a aplicação de métodos eficientes para sua resolução. Contudo, a maioria das pesquisas de escalonamento de maquinas paralelas envolvendo considerações de *setup* ignora a disponibilidade de equipamentos e/ou trabalhadores qualificados que são necessários para realizar tais operações (HUANG et al., 2010).

Estudos que considerem esses aspectos são necessários, pois é um problema que surge frequentemente em ambientes de produção em que a produtividade é significativamente afetada pelos tempos de preparação e pelos recursos associados as equipes de *setup* (HUANG et al., 2010). Por exemplo, em indústrias gráficas e também naquelas que possuem operações de envase ao longo do seu processo produtivo, tais como na produção de bebidas, saneantes e outros, a sequência da produção pode influenciar drasticamente nos tempos de preparação das máquinas.

Enumera-se também os casos de unidades produtivas de pequeno porte, que possuem equipes de trabalho reduzidas, como também aquelas que sofrem redução de tais equipes em virtude de períodos de crise. Pode-se ainda encontrar aplicações, desse problema, em sistemas de manufatura flexíveis, onde um robô é compartilhado entre várias máquinas para troca de ferramentas e/ou *setup* das peças produzidas (KOULAMAS, 1996).

Por se tratar de um problema de OC, resolver tal problema de maneira ótima através da aplicação de métodos exatos, a depender da quantidade de tarefas e máquinas, torna-se uma tarefa inviável dado o tempo proibitivo de se encontrar a solução ótima. Na literatura, as formulações exatas para o problema de sequenciamento com servidor único de *setup*, tanto para os casos com tempos *setup* independentes da sequência (GAN et al., 2012; KIM; LEE, 2012; HASANI et al., 2014a) quanto àqueles dependentes da sequência (HAMZADAYI; YILDIZ, 2017) limitam-se a resolução de problemas de pequeno/médio porte.

Devido a limitação dos métodos exatos, uma alternativa para sua resolução dá-se pela aplicação de métodos (meta-)heurísticos. A aplicação desses para resolução de problemas de sequenciamento de produção é ampla, como pode ser visto no trabalho de Allahverdi (2015). No tocante a sua utilização para os problema com servidor único de *setup*, têm destaque as meta-heurísticas: Algoritmos Genéticos (AG) (ABDEKHODAEE et al., 2006; HUANG et al., 2010), e *Simulated Annealing* (SA) (HASANI et al., 2014c; HAMZADAYI; YILDIZ, 2016, 2017).

Os primeiros estudos para a classe de problemas CS surgiram no início da década

de 90, porém constata-se na literatura a pouca diversificação tanto dos métodos exatos quanto daqueles heurísticos propostos para sua resolução. Em meio a essa constatação, este trabalho tem como foco o desenvolvimento de novos métodos exato e heurístico para a dada classe.

Em relação à classe do problema, serão consideradas aquelas que apresentam um número arbitrário de máquinas, com tempos de *setup* dependentes e independentes da sequência, cujo o objetivo é a minimização do *makespan*. Os métodos exatos desenvolvidos são baseados nas formulações de tarefas indexadas no tempo propostas por Dyer e Wolsey (1990) e de arcos indexados no tempo de Silva et al. (2018). O métodos heurísticos propostos são baseados na meta-heurística *Iterated Local Search* (ILS).

1.3 Objetivos

1.3.1 Objetivo geral

Desenvolver abordagens exata e heurística para a resolução de problemas da classe CS com máquinas paralelas idênticas e tempos de *setup* dependentes, ou não, da sequência.

1.3.2 Objetivos específicos

- Realizar uma ampla revisão acerca dos modelos e métodos abordados na literatura.
- Propor novos modelos de programação linear inteira para resolução do problema.
- Desenvolver abordagens heurísticas para a resolução do problema.
- Gerar instâncias para aplicação dos modelos desenvolvidos, devido a ausência dessas na literatura.
- Comparar os resultados obtidos com os das principais abordagens da literatura.

1.4 Estrutura do trabalho

O trabalho está organizado da seguinte maneira: o Capítulo 2 pontua alguns conceitos referentes ao problema de sequenciamento da produção, apresenta a classe do problema abordado e a revisão da literatura relacionada. No Capítulo 3 são expostos os métodos propostos para a resolução do problema em questão. Resultados obtidos são apresentados no Capítulo 4 e, por fim, as conclusões do trabalho são elencadas no Capítulo 5.

Capítulo 2

Fundamentação Teórica e Revisão da Literatura

Nesta seção são apresentadas algumas definições e conceitos relacionados aos problemas de sequenciamento da produção. Em seguida, a classe de problemas abordados no trabalho é apresentada, assim como a bibliografia dos trabalhos da literatura relacionada. Por fim, são mostrados os principais métodos sugeridos na literatura para o problema abordado.

2.1 Conceitos sobre sequenciamento da produção

Os sistemas de manufatura podem ser caracterizados por uma variedade de fatores, tais como o número de recursos ou máquinas, suas características e configurações, entre outros. Em modelos de sequenciamento de produção, os recursos são comumente referidos como *máquinas* e as operações a serem realizadas são conhecidas como *tarefas*.

Seja $J = \{1, \ldots, n\}$ o conjunto das tarefas que devem ser processadas em um conjunto $M = \{1, \ldots, m\}$ de máquinas. Assumindo que cada máquina pode processar apenas uma tarefa por vez, que cada tarefa pode ser processada apenas por uma máquina por vez e que a tarefa $j \in J$ e a máquina $k \in M$, os seguintes dados podem ser especificados:

- Tempo de processamento p_j necessário para a execução da tarefa j, ou p_j^k, caso o processamento esteja vinculado a máquina k;
- Data de liberação r_j a partir da qual j pode começar a ser processada, não sendo permitida iniciar antes dessa;
- Data de entrega d_j, na qual j deve, idealmente, ser finalizada. Caso o processamento

termine após a data de entrega, uma penalidade pode ser incorrida;

- Peso w_j indica a importância relativa da tarefa j em relação as demais.

Esses quatro dados listados são ditos dados estáticos, pois não dependem do sequenciamento. Por outro lado, os dados que não são fixados antecipadamente e que dependem do sequenciamento são chamados de dados dinâmicos (PINEDO, 2009, p. 21). Dentre esses, podemos destacar o tempo de término C_j que indica o instante de tempo em que a tarefa j terminou seu processamento, ou C_j^k, caso o término seja na máquina k.

2.1.1 Nomenclatura ($\alpha|\beta|\gamma$)

As várias características, relacionadas às máquinas, tarefas e ao processo de atribuição de tarefas às máquinas, podem conferir aos problemas de *scheduling* inúmeras particularidades, que quando combinadas geram um número extenso de variantes. Por isso, Graham et al. (1979) propuseram a classificação composta por três campos ($\alpha|\beta|\gamma$). Tal classificação tem como objetivo representar de maneira simplificada cada variante. A seguir são apresentadas as definições e algumas das principais entradas para cada campo:

- No campo α é especificado a configuração das máquinas (ver Tabela 2.1).

Tabela 2.1: Configurações das Máquinas - α

$\boldsymbol{\alpha}$	**Termo**	**Definição**
1	Máquina única	Todas as tarefas devem ser processadas em apenas uma máquina.
P	Máquinas paralelas idênticas	m máquinas paralelas idênticas estão disponíveis para processar as n tarefas, com tempos de processamentos análogos em todas as máquinas.
Q	Máquinas paralelas uniformes	Existem m máquinas em paralelo, porém, com diferentes fatores de velocidade q_k para cada máquina k. Assim, o tempo de processamento da tarefa j na máquina k é dado por $p_j^k = q_k \times p_j$.
O	*Open shop*	As n tarefas j devem ser processadas em todas as m máquinas k por p_j^k unidades de tempo. Não há restrição quanto à sequência de execução das tarefas.
J	*Job shop*	Cada tarefa possui uma rota predefinida a ser seguida, geralmente distinta das demais.
F	*Flow shop*	Cada tarefa deve ser processada em cada uma das m máquinas, dispostas em série, de modo que todas as tarefas seguem a mesma sequência.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2008, p. 14–15).

- Em β são detalhadas algumas características relativas ao processamento das tarefas (ver Tabela 2.2) e pode ser vazio, conter uma ou múltiplas entradas.

Tabela 2.2: Características das tarefas - β

β	**Termo**	**Definição**
pmtn	Preempção	Uma tarefa pode ser dividida em várias etapas, ou seja, seu processamento pode ser interrompido e retomado posteriormente na mesma, ou em outra máquina.
prec	Precedência	As restrições de precedência podem aparecer em ambientes de máquina única ou máquinas paralelas, exigindo que uma ou mais tarefas tenham que ser concluídas antes que outra possa iniciar seu processamento.
r_j	Datas de liberação	A tarefa j não pode ter seu processamento iniciado antes de sua data de liberação r_j.
nwt	*No-wait*	O requisito de não-espera ocorrem em ambientes *Flow shop*. As tarefas não podem esperar entre duas máquinas sucessivas.
s_j	Tempos de *setup*	Apenas a tarefa a ser processada define o tempo de *setup*, ou seja, independem da ordem de processamento.
s_{ij}	Tempos de *setup* dependentes da sequência	Ao contrário de s_j, indica que o tempo de *setup* da tarefa j depende da sua tarefa predecessora i.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2008, p. 16–17).

- O campo γ denota a função objetivo do problema, ou critério de otimalidade, na maioria dos casos a ser minimizada (ver Tabela 2.3). Os mais comumente escolhidos estão diretamente relacionados a C_j.

Tabela 2.3: Função objetivo - γ

γ	**Definição**
C_{max}	O *makespan*, definido como $max(C_1, \ldots, C_n)$, é equivalente ao tempo de término C_j da última tarefa j a deixar o sistema.
L_{max}	O atraso máximo é definido como $max(L_1, \ldots, L_n)$. O atraso L_j da tarefa j é obtido pela diferença entre o tempo real e o tempo desejado de seu término.
$\sum w_j C_j$	A soma ponderada dos tempos de término das tarefas. Neste tipo de objetivo cada tarefa j possui um peso w_j associado. Fornece uma indicação dos custos totais de manutenção ou estoque incorridos pelo sequenciamento.
$\sum w_j T_j$	A soma dos atrasos ponderados das tarefas. $T_j = max\{0, L_j\}$.
$\sum w_j U_j$	O número ponderado de tarefas atrasadas.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2008, p. 18–19).

Em acordo com o esquema de classificação proposto por Hall et al. (2000), a existência do servidor (S) é inserida em α. Portanto, $\alpha = P_m, S$ indica que há um número fixo, m, de

máquinas paralelas idênticas; de outra forma, $\alpha = P, S$ indica que o número de máquinas é arbitrário. Em β, a presença de $p_j = 1$ e $s_j = 1$ assinala que os tempos de processamento e *setup* são unitários, respectivamente, e $s_j = s$ se todos os tempos de *setup* são iguais a um inteiro positivo arbitrário s.

2.2 Problema de escalonamento com servidor de *setup* único

Formalmente, o problema pode ser definido como: seja $J = \{1, \ldots, n\}$ o conjunto de tarefas a serem escalonadas em um conjunto de $M = \{1, \ldots, m\}$ máquinas paralelas idênticas de modo a minimizar o *makespan*. Cada tarefa $j \in J$ possui um tempo de processamento p_j e um tempo de *setup*. Contudo, as atividades de *setup* não podem ser realizadas de forma simultânea. Assim sendo, é possível que haja a ocorrência de tempos ociosos nas máquinas para que uma solução seja viável.

Para os casos do problema com tempo de *setup* independentes da sequência, esse é dado por s_j, do contrário, s_{ij} é considerado uma vez que a tarefa $j \in J$ seja processada imediatamente após a tarefa $i \in J$ na mesma máquina.

2.2.1 Literatura Relacionada

Nessa seção são debatidos os trabalhos da literatura relacionada e sumarizados na Tabela 2.4 em relação ao problema, abordagens e resultados abordados. RC é a abreviação de resultados de complexidade.

Koulamas (1996), com o objetivo de minimizar o tempo ocioso (IT, *idle time*) resultante da indisponibilidade do servidor de *setup*, mostrou que o problema $P_2, S_1|s_j|IT$ é fortemente $\mathcal{NP}$-difícil e propôs uma heurística *beam search* para a resolução do mesmo, comparou-a com uma heurística de busca local e constatou que a primeira obteve melhor desempenho.

Kravchenko e Werner (1997) mostraram que os problemas $P, S_1|s_j = 1|C_{max}$ e $P_2, S_1|s_j = s|IT$ são fortemente $\mathcal{NP}$-difícil e apresentaram um algoritmo pseudo-polinomial para o problema $P_2, S_1|s_j = 1|C_{max}$. Kravchenko e Werner (1999) definiram a complexidade computacional para generalizações de alguns problemas de máquinas paralelas com m máquinas e k servidores, onde $1 \leq k < m$.

Hall et al. (2000) descreveram uma série de resultados que mapeiam a complexi-

dade computacional de alguns problemas de sequenciamento em máquinas paralelas com um servidor de *setup*, e em particular propuseram um algoritmo que resolve o problema $P_2, S_1|s_j = 1|\sum C_j$ otimamente em tempo $O(n \log n)$. Kravchenko e Werner (2001) também propuseram um algoritmo heurístico para a generalização desse problema com m maquinas, $P, S_1|s_j = 1|\sum Cj$.

Glass et al. (2000) provaram que o problema com duas maquinas paralelas dedicadas é $\mathcal{NP}$-difícil, mesmo se todos os tempos de *setup* ou de processamento forem iguais. Para o caso $PD, S|s_j|C_{max}$ eles apresentaram um algoritmo guloso que fornece um *makespan* com no máximo duas vezes o valor ótimo e para o problema $PD_2, S_1|s_j|C_{max}$, uma heurística que garante uma taxa de pior caso apertada de até 3/2 vezes o valor ótimo.

Wang e Cheng (2001) proveram um algoritmo aproximativo baseado em relaxação linear para resolver o problema $P, S_1|s_j = 1|\sum w_j C_j$. Brucker et al. (2002) continuaram os estudos dos aspectos de complexidade dos problemas com servidores de *setup* iniciados por Hall et al. (2000) e Kravchenko e Werner (1997).

Abdekhodaee e Wirth (2002) abordaram o problema $P_2, S_1|s_j|C_{max}$ e provaram que é $\mathcal{NP}$-difícil. Eles apresentaram um modelo de programação inteira, que pode ser utilizado para resolver problemas com até 12 tarefas em tempo computacional aceitável, e duas heurísticas para resolução de dois casos especiais do dado problema: com tempos de processamento curtos e com tarefas de mesma duração.

Abdekhodaee et al. (2004) também estudaram dois casos especiais para o problema $P_2, S_1|s_j|C_{max}$, sendo esses com tempos de processamento iguais e tempos de *setup* iguais, e propuseram heurísticas para resolução dos mesmos. Em um outro trabalho, Abdekhodaee et al. (2006), sugeriram uma heurística gulosa, um AG e um algoritmo de Gilmore-Gomory, um caso especial do problema do caixeiro viajante, para o problema geral $P_2, S_1|s_j|C_{max}$.

Guirchoun et al. (2005) consideraram um problema de escalonamento de máquinas paralelas com um servidor de *setup* e um problema *flow shop* híbrido com a restrição *no-wait*. Eles propuseram uma redução polinomial entre estes dois problemas e forneceram uma visão geral dos resultados de complexidade.

Zhang e Wirth (2009) aplicaram algoritmos LS (*List Scheduling*) em três casos especiais do problema $P_2, S_1|s_j|C_{max}$: tarefas com mesma duração, tempos de processamento iguais e tempos de *setup* iguais. Para esses casos, é considerado um esquema de chegada dinâmica de tarefas, onde cada uma das tarefas são apresentadas uma a uma, de acordo com uma sequencia de entrada, sem qualquer conhecimento das tarefas restantes na lista.

Werner e Kravchenko (2010) consideraram o problema com m máquinas paralelas idênticas e múltiplos servidores. Forneceram resultados de complexidade de alguns desses problemas para minimização do *makespan* e do atraso máximo. Propuseram um algoritmo polinomial para o problema com tempos de processamento iguais e tempos de *setup* iguais, e um algoritmo pseudo–polinomial para $P_m, S_{m-1}|s_j = 1|C_{max}$. Além disso, realizaram uma análise de pior caso de dois algoritmos LS para minimização do *makespan*.

Ou et al. (2010) desenvolveram algoritmos heurísticos e com estratégias de *branch-and-bound* (B&B), com o objetivo de minimizar o $\sum C_j$ no problema de sequenciamento de máquinas paralelas com múltiplos servidores de descarregamento. Huang et al. (2010) consideraram o problema $PD, S_1|s_{ij}|C_{max}$ e formularam um modelo de programação inteira e um AG híbrido.

Gan et al. (2012), ampliando a análise iniciada por Abdekhodaee et al. (2006), apresentaram dois modelos de programação inteira mista (MIP, *Mixed Integer Programming*) e duas variantes do esquema *branch-and-price* (B&P) para o problema $P_2, S_1|s_j|C_{max}$. Os resultados experimentais mostraram que uma das formulações MIP alcançou desempenho superior somente para instâncias menores, ao passo que as variantes do esquema *branch-and-price* conseguiram lidar com instâncias maiores de forma mais favorável.

Kim e Lee (2012) desenvolveram dois MIPs e um algoritmo heurístico híbrido, que combina as meta-heurísticas busca tabu e SA, para resolver o problema $P, S_1|s_j|C_{max}$. Jiang et al. (2013), para o problema $P_2, S_1|pmtn, s_j|C_{max}$, apresentaram um algoritmo que pode produzir escalonamentos ótimos para dois casos especiais: tempos de processamento iguais e tempos de *setup* iguais.

Su (2013) considerou o problema $P_2, S_1|r_j, s_j|C_{max}$, onde as tarefas chegam ao longo do tempo, ou seja, chegada dinâmica de tarefas. Para o problema foi proposto uma heurística LPT (*Longest Processing Time*), que se baseia em alocar, primeiro, a tarefa cujo tempo de processamento seja o maior dentre as tarefas disponíveis.

Hasani et al. (2014a) abordaram o problema $P_2, S_1|s_j|C_{max}$. Para este, propuseram dois MIPs baseados na ideia de decompor o escalonamento das tarefas em um conjunto de blocos. Os resultados computacionais mostraram que ambos os modelos obtiveram melhor desempenho que todas as heurísticas propostas em Gan et al. (2012). Hasani et al. (2014c), para o mesmo problema, sugeriram um AG e um SA avaliados para instâncias com até 1000 tarefas. Em estudos posteriores, Hasani et al. (2014b), propuseram um algoritmo aproximativo para o problema $P, S_1|s_j|\sum w_j C_j$, e Hasani et al. (2015) apresentaram dois algoritmos construtivos para a configuração $P_2, S_1|s_j|C_{max}$ com até 10000 tarefas.

Tabela 2.4: Literatura relacionada ao problema CS.

Publicações	$\boldsymbol{\alpha\|\beta\|\gamma}$	**Abordagens/ Resultados**
Koulamas (1996)	$P_2, S_1\|s_j\|IT$	RC, heurística *beam search*
Kravchenko e Werner (1997)	$P, S_1\|s_j{=}1\|C_{max}$	RC, heurísticas
	$P_2, S_1\|s_j{=}s\|IT$	
	$P_2, S_1\|s_j{=}1\|C_{max}$	
Kravchenko e Werner (1999)	$P, S_{m-1}\|s_j\|C_{max}$	RC
Hall et al. (2000)	$P, S_1\|s_j\|\gamma*$	RC, algoritmo pseudo-polinomial
Glass et al. (2000)	$PD, S\|s_j\|C_{max}$,	RC, algoritmo guloso
	$PD_2, S_1\|s_j\|C_{max}$	
Kravchenko e Werner (2001)	$P, S_1\|s_j{=}1\|\sum Cj$	Heurística
Wang e Cheng (2001)	$P, S_1\|s_j{=}1\|\sum w_jC_j$	Algoritmo aproximativo
Brucker et al. (2002)	$P, S_1\|s_j\|\gamma*$	RC
Abdekhodaee e Wirth (2002)	$P_2, S_1\|s_j\|C_{max}$	RC, MIP, heurísticas
Abdekhodaee et al. (2004)	$P_2, S_1\|s_j\|C_{max}$	RC, heurísticas
Guirchoun et al. (2005)	$FH(P), S_1\|s_j\|\gamma*$	RC
Abdekhodaee et al. (2006)	$P_2, S_1\|s_j\|C_{max}$	Heurística gulosa e AG
Zhang e Wirth (2009)	$P_2, S_1\|s_j\|C_{max}$	Algoritmo LS
Werner e Kravchenko (2010)	$P, S\|s_j\|C_{max}$,	RC, algoritmo LS, polinomial e
	L_{max}, $\sum w_jC_j$	pseudo-polinomial
Ou et al. (2010)	$P, S\|s_j\|\sum C_j$	Algoritmos heurísticos, B&B
Huang et al. (2010)	$PD, S_1\|s_{ij}\|C_{max}$	AG híbrido, modelo matemático
Gan et al. (2012)	$P_2, S_1\|s_j\|C_{max}$	MIPs, variantes B&P
Kim e Lee (2012)	$P, S_1\|s_j\|C_{max}$,	MIPs, heurística híbrida
	$P, S_1\|s_j\|\sum W$	
Jiang et al. (2013)	$P_2, S_1\|pmtn, s_j\|C_{max}$	Algoritmo com taxa de pior caso
Su (2013)	$P_2, S_1\|r_j, s_j\|C_{max}$	Algoritmo LPT
Hasani et al. (2014a)	$P_2, S_1\|s_j\|C_{max}$	MIP baseado em blocos
Hasani et al. (2014b)	$P, S_1\|s_j\|\sum w_jC_j$	Algoritmo aproximativo
Hasani et al. (2014c)	$P_2, S_1\|s_j\|C_{max}$	AG e SA
Hasani et al. (2015)	$P_2, S_1\|s_j\|C_{max}$	Algoritomos construtivos
Hamzadayi e Yildiz (2016)	$P, S_1\|pmtn, s_{ij}\|C_{max}$	SA, regras de despacho
Hamzadayi e Yildiz (2017)	$P, S_1\|s_{ij}\|C_{max}$	MIP, SA, AG

$\gamma* \in \{C_{max}, \sum C_j, L_{max}, \sum w_jC_j, \sum T_j, \sum w_jT_j, \sum U_j, \sum w_jU_j\}$

Fonte: Adaptado de Hamzadayi e Yildiz (2017)

Hamzadayi e Yildiz (2016) abordaram o problema $P, S_1|pmtn, s_{ij}|C_{max}$ dinâmico. Desenvolveram uma heurística SA e regras de despacho (LPT, SPT (*Shortest Processing Time*) e FIFO (*First In First Out*)) com base em abordagens de reprogramação completa. O SA obteve melhor performance. Hamzadayi e Yildiz (2017) propuseram, para o problema $P, S_1|s_{ij}|C_{max}$, um MIP, um SA e um AG. De acordo com os resultados reportados, o MIP conseguiu resolver apenas 4 das 30 instâncias propostas, dentre as heurísticas, o GA obteve o melhor desempenho.

2.3 Modelos da literatura

Nessa seção são descritos os principais modelos da literatura para a classe de problemas CS considerados nessa dissertação. Na Tabela 2.5 são apresentadas as características desses, onde a coluna SI indica se o tempo de *setup* da primeira tarefa alocada em uma máquina é considerado. Alguns modelos foram desconsiderados por motivos de não se adequarem as variantes abordadas ou por não possuírem genericidade quanto ao formato das instâncias adotadas.

Tabela 2.5: Formulações da literatura

Formulação	Publicação	Objetivo	Máquinas	s_j	s_{ij}	SI
F1	Kim e Lee (2012)	C_{max}	m	✓		✓
F2	Kim e Lee (2012)	$\sum W$	m	✓		✓
F3	Gan et al. (2012)	C_{max}	2	✓		✓
F4	Hasani et al. (2014a)	C_{max}	2	✓		✓
F5	Hamzadayi e Yildiz (2017)	C_{max}	m		✓	

Nas formulações a seguir, por conveniência, os índices das variáveis e dados serão padronizados na seguinte forma: i,j e k denotam tarefas; h e l referem-se a máquinas; b,d e g inidicam uma posição em uma dada sequência; e por último o índice f está associado a um dado nível. As informações sobre as varíaveis e dados aos quais estão associados serão mostradas em cada formulação.

2.3.1 Formulação F1 de Kim e Lee (2012)

Na formulação F1 as tarefas são escalonadas de modo sequencial, assim temos que E_b indica o tempo de início do *setup* da b-ésima tarefa alocada e C_b o tempo de término de seu processamento. As variáveis de decisão são:

$$v_{bj} = \begin{cases} 1, & \text{se a tarefa } j \text{ é alocada na } b\text{-ésima posição da sequência,} \\ 0, & \text{caso contrário.} \end{cases}$$

$$\sigma_b^h = \begin{cases} 1, & \text{se a } b\text{-ésima tarefa alocada é processada na máquina } h, \\ 0, & \text{caso contrário.} \end{cases}$$

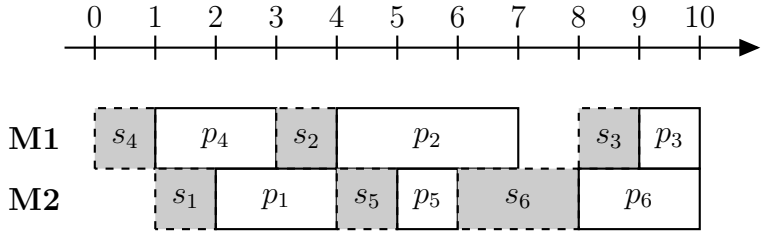

Figura 2.1: Sequência de tarefas alocadas.

No exemplo da Figura 2.1, a solução é dada por $v_{14}, v_{21}, v_{32}, v_{45}, v_{56}$ e v_{63}. A formulação segue como:

$$\text{Minimizar} \quad C_{max} \tag{2.1a}$$

Sujeito a:

$$C_{max} \geq C_b \qquad b = 1, \ldots, n, \tag{2.1b}$$

$$\sum_{j \in J} v_{bj} = 1 \qquad b = 1, \ldots, n, \tag{2.1c}$$

$$\sum_{b=1}^{n} v_{bj} = 1 \qquad j \in J, \tag{2.1d}$$

$$\sum_{h \in M} \sigma_b^h = 1 \qquad b = 1, \ldots, n, \tag{2.1e}$$

$$E_b \geq E_{b-1} + \sum_{j \in J} v_{b-1,j} s_j \qquad b = 2, \ldots n, \tag{2.1f}$$

$$E_b \geq C_d - BM(2 - \sigma_b^h - \sigma_d^h) \qquad b = 2, \ldots, n; d < b; h \in M \tag{2.1g}$$

$$C_b = E_b + \sum_{j \in J} v_{bj}(s_j + p_j) \qquad b = 1, \ldots, n, \tag{2.1h}$$

$$v_{bj} \in \{0, 1\} \qquad b = 1, \ldots, n; j \in J, \tag{2.1i}$$

$$\sigma_b^h \in \{0, 1\} \qquad b = 1, \ldots, n; h \in M, \tag{2.1j}$$

$$E_b \geq 0 \qquad b = 1, \ldots, n, \tag{2.1k}$$

$$C_b \geq 0 \qquad b = 1, \ldots, n. \tag{2.1l}$$

O conjunto de restrições (2.1b) definem o *makespan*. As restrições (2.1c) e (2.1d)

implicam que apenas uma tarefa pode ser alocada na b-ésima posição da sequência e que cada tarefa só pode ser atribuída a apenas uma posição. As restrições (2.1e) garantem que cada tarefa tenha que ser processada em uma máquina. As restrições (2.1f) e (2.1g) mostram que a tarefa pode ser processada somente se alguma máquina e o servidor estiverem disponíveis simultaneamente. O *setup* da b-ésima tarefa só pode ser iniciado após o *setup* da $(b-1)$-ésima tarefa alocada, conforme as restrições (2.1f). As restrições (2.1g) mostram que a b-ésima tarefa só pode ser iniciada quando a anterior é concluída na mesma máquina. As restrições (2.1h) definem que o tempo de conclusão da b-ésima tarefa alocada é igual à soma do seu tempo de início, tempo de *setup* e tempo de processamento. As restrições (2.1i) e (2.1j) são requisitos de integralidade e as restrições (2.1k) e (2.1l) são restrições de não negatividade.

2.3.2 Formulação F2 de Kim e Lee (2012)

Nessa formulação, W_b indica o b-ésimo tempo de espera do servidor gerado pela alocação da b-ésima tarefa da sequência. A Figura 2.2 mostra um exemplo dos tempos de espera, W_3 e W_5, do servidor gerados pela alocação da 3ª e 5ª tarefas da sequência.

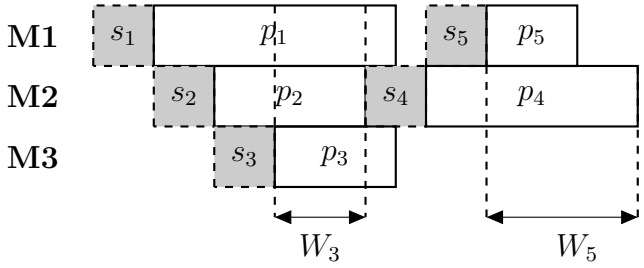

Figura 2.2: Tempo de espera do servidor.
Fonte: Adaptado de Kim e Lee (2012).

Desse modo, minimizar os tempos de espera do servidor equivale a minimizar o *makespan*, que pode ser obtido pela soma dos tempos de *setup* das tarefas e os tempos de espera na sequência, conforme a Equação 2.2.

$$C_{max} = \sum_{j \in J} s_j + \sum_{b=1}^{n} W_b \tag{2.2}$$

As variáveis de decisão são semelhantes àquelas da formulação F1. A formulação segue como:

$$\text{Minimizar} \quad \sum_{b=1}^{n} W_b \tag{2.3a}$$

Sujeito a:

$$\sum_{j \in J} v_{bj} = 1 \qquad b = 1, \ldots, n, \tag{2.3b}$$

$$\sum_{b=1}^{n} v_{bj} = 1 \qquad j \in J, \tag{2.3c}$$

$$\sum_{h \in M} \sigma_b^h = 1 \qquad b = 1, \ldots, n, \tag{2.3d}$$

$$W_b \geq \sum_{j \in J} v_{gj} p_j - \sum_{d=g+1}^{b} \sum_{j \in J} v_{dj} s_j - \sum_{d=g}^{b-1} W_d - BM(2 - \sigma_{b+1,h} - \sigma_{gh}) \qquad b, g = 1, \ldots, n-1; g \leq b; h \in M, \tag{2.3e}$$

$$W_n \geq \sum_{j \in J} v_{gj} p_j - \sum_{d=g+1}^{n} \sum_{j \in J} v_{dj} s_j - \sum_{d=g}^{n-1} W_d \qquad g = 1, \ldots, n, \tag{2.3f}$$

$$v_{bj} \in \{0, 1\} \qquad b = 1, \ldots, n; j \in J, \tag{2.3g}$$

$$\sigma_b^h \in \{0, 1\} \qquad b = 1, \ldots, n; h \in M, \tag{2.3h}$$

$$W_b \geq 0 \qquad b = 1, \ldots, n. \tag{2.3i}$$

Os conjuntos de restrições (2.3b), (2.3c) e (2.3d) são requisitos do escalonamento tais quais os conjuntos de restrições (2.1b), (2.1c) e (2.1d) da formulação da Seção 2.3.1. As restrições (2.3e) e (2.3f) definem os tempos de espera do servidor. As restrições (2.3f) determinam o n-ésimo tempo de espera do servidor, enquanto que as restrições (2.3e) o fazem para as demais posições da sequência. As restrições (2.3g) e (2.3h) são requisitos de integralidade e as restrições (2.3i) são de não negatividade.

2.3.3 Formulação F3 de Gan et al. (2012)

Na formulação F3, as variáveis E_b e v_{bj} são equivalentes àquelas da formulação F1. S_b e P_b indicam, respectivamente, os tempos de *setup* e de processamento da b-ésima tarefa atribuída. As demais variáveis de decisão são dadas por:

$$\delta_b = \begin{cases} 1, & \text{se a } b\text{-ésima tarefa alocada é processada na máquina 2,} \\ 0, & \text{caso contrário.} \end{cases}$$

$$z_{db} = \begin{cases} 1, & \text{se as tarefas na } d \text{ e } b\text{-ésima posições não são alocadas na mesma máquina,} \\ 0, & \text{caso contrário.} \end{cases}$$

$$z_{db}^{+} = \begin{cases} 1, & \text{se as tarefas na } d \text{ e } b\text{-ésima posições são alocadas, respectivamente, nas} \\ & \text{máquinas 2 e 1,} \\ 0, & \text{caso contrário.} \end{cases}$$

$$z_{db}^{-} = \begin{cases} 1, & \text{se as tarefas na } d \text{ e } b\text{-ésima posições são alocadas, respectivamente, nas} \\ & \text{máquinas 1 e 2,} \\ 0, & \text{caso contrário.} \end{cases}$$

Assim, a formulação segue como:

$$\text{Minimizar} \quad C_{max} \tag{2.4a}$$

Sujeito a:

$$C_{max} \geq E_b + S_b + P_b \qquad b = 1, \ldots, n, \tag{2.4b}$$

$$\sum_{j \in J} v_{bj} = 1 \qquad b = 1, \ldots, n, \tag{2.4c}$$

$$\sum_{b=1}^{n} v_{bj} = 1 \qquad j \in J, \tag{2.4d}$$

$$S_b = \sum_{j \in J} v_{bj} s_j \qquad b = 1, \ldots, n, \tag{2.4e}$$

$$P_b = \sum_{j \in J} v_{bj} p_j \qquad b = 1, \ldots, n, \tag{2.4f}$$

$$E_b \geq E_{b-1} + S_{b-1} \qquad b = 2, \ldots, n, \tag{2.4g}$$

$$E_d \geq E_b + S_b + P_b - BMz_{db} \qquad \forall b, d = 1, \ldots, n; b < d, \tag{2.4h}$$

$$z_{db}^{+} - z_{db}^{-} = \delta_d - \delta_b \qquad d, b = 1, \ldots n, \tag{2.4i}$$

$$z_{db} = z_{db}^{+} + z_{db}^{-} \qquad d, b = 1, \ldots n, \tag{2.4j}$$

$$v_{bj} \in \{0, 1\} \qquad j \in J; b = 1, \ldots n, \tag{2.4k}$$

$$\delta_b \in \{0, 1\} \qquad b = 1, \ldots n, \tag{2.4l}$$

$$z_{db}, z_{db}^{+}, z_{db}^{-} \in \{0, 1\} \qquad d, b = 1, \ldots n, \tag{2.4m}$$

$$E_b, S_b, P_b \geq 0 \qquad b = 1, \ldots, n, \tag{2.4n}$$

O conjunto de restrições (2.4b) definem o *makespan*. As restrições (2.4c) e (2.4d) implicam que apenas uma tarefa pode ser alocada na b-ésima posição da sequência e que cada tarefa só pode ser atribuída a apenas uma posição. As restrições (2.4e) e (2.4f) determinam, respectivamente, os tempos de *setup* e processamento da b-ésima tarefa alocada.

Conforme as restrições (2.4g), o *setup* da b-ésima tarefa só pode ser iniciado após o término do *setup* da $(b-1)$-ésima tarefa alocada. No caso de tarefas alocadas na mesma máquina, uma tarefa só pode iniciar após o fim do processamento daquela imediatamente anterior a ela, isso é garantido pelas restrições (2.4h).

As restrições (2.4i) e (2.4j) definem que uma tarefa só pode ser processada em uma das máquinas. As restrições (2.4k), (2.4l) e (2.4m) são requisitos de integralidade e o conjunto (2.4n) são restrições de não negatividade.

2.3.4 Formulação F4 de Hasani et al. (2014a)

O modelo baseia-se na ideia de que qualquer sequenciamento pode ser decomposto em um conjunto de blocos $B_1, \ldots, B_d$, com $d \leq n$. Assim, cada bloco B é composto por uma tarefa de primeiro nível j^1 enquanto o segundo é composto por um conjunto de tarefas de segundo nível $\lambda = \{j_1^2, \ldots, j_d^2\}$, que satisfazem $p^1 \geq s_1^2 + \cdots + s_d^2 + p_1^2 + \cdots + p_d^2$, conforme mostrado na Figura 2.3.

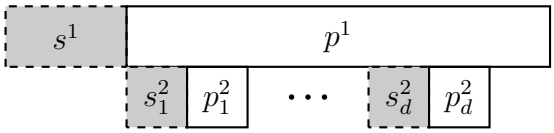

Figura 2.3: Representação de um bloco.
Fonte: Adaptado de Hasani et al. (2014a).

O bloco B_d pode ainda ser considerado como uma tarefa modificada A_d, cujos tempos de *setup* e processamento são obtidos pelas Equações (2.5) e (2.6), respectivamente. Desse modo, as tarefas $j \in J$ podem ser divididas em um conjunto de tarefas A_d, $d = 1, \ldots, n$, e serem escalonadas sequencialmente alternando-se as máquinas, ou seja, A_1 na máquina 1, A_2 na máquina 2, A_3 na máquina 1 e assim por diante.

$$S_d = s^1 \tag{2.5}$$

$$P_d = p^1 - \sum_{i \in \lambda_d} (s_i^2 + p_i^2) \tag{2.6}$$

Aplicando os conceitos no exemplo dado na Figura 2.4, pode-se definir quatro blocos B_1, B_2, B_3 e B_4. O bloco B_1 é composto pela tarefa de primeiro nível 1 e a tarefa de segundo nível $\lambda_1 = \{2\}$; B_2 é definido pela tarefa de primeiro nível 3 e a tarefa de segundo nível $\lambda_2 = \{4\}$; B_3 é composto pela tarefa de primeiro nível 5 e não possui tarefas de segundo nível ($\lambda_3 = \{\emptyset\}$); por fim B_4 é constituído pela tarefa de primeiro nível 6 e pelas tarefas de segundo nível $\lambda_4 = \{7, 8\}$.

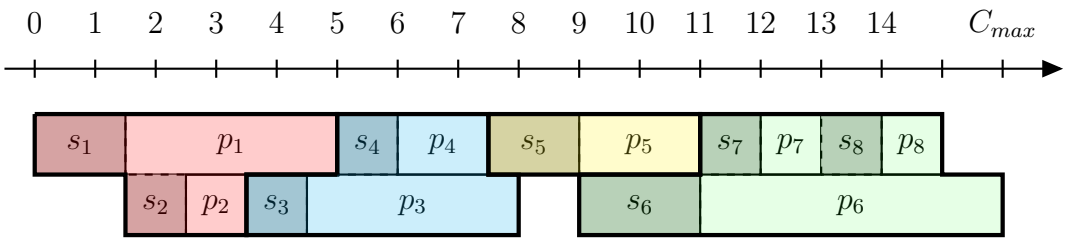

Figura 2.4: Escalonamento com quatro blocos.
Fonte: Adaptado de Hasani et al. (2014a).

Das Equações (2.5) e (2.6), obtém-se o escalonamento das tarefas modificadas A_1, A_2, A_3 e A_4, como o mostrado na Figura 2.5, cujo *makespan* modificado é dado por C_A. Portanto, C_{max} pode ser obtido pela soma de C_A com os tempos de *setup* e processamento de todas as tarefas de segundo nível, conforme a Equação (2.7), onde $\Lambda = \bigcup_{d=1}^{n} \lambda_d$.

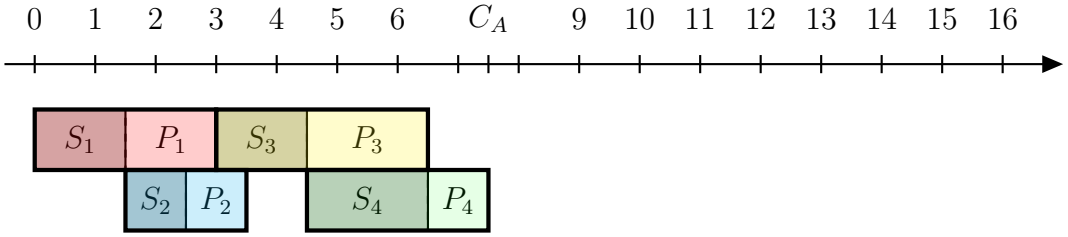

Figura 2.5: Escalonamento das tarefas modificadas.

$$C_{max} = C_A + \sum_{i \in \Lambda} (s_i^2 + p_i^2) \tag{2.7}$$

Na formulação F4, a variável B_{dj}^f é usada para representar um bloco. O índice $d = 1, \ldots, n$ representa a posição do bloco na sequência, $f \in \{1, 2\}$ indica o nível do bloco e $j \in J$, a tarefa. E_d indica tempo de início do *setup* do d-ésimo bloco ou tarefa modificada. Assim, tem-se que:

$$B_{dj}^f = \begin{cases} 1, & \text{se a tarefa } j \text{ é alocada no nível } f \text{ do } d\text{-ésimo bloco,} \\ 0, & \text{caso contrário.} \end{cases}$$

E a formulação segue:

$$\text{Minimizar} \quad C_A + \sum_{d=1}^{n} \sum_{j \in J} (s_j + p_j) B_{dj}^2 \tag{2.8a}$$

Sujeito a:

$$\sum_{d=1}^{n} \sum_{f=1}^{2} B_{dj}^f = 1 \qquad j \in J, \tag{2.8b}$$

$$\sum_{j \in J} B^1_{dj} \leq 1 \qquad d = 1, \ldots, n, \tag{2.8c}$$

$$S_d \geq \sum_{j \in J} s_j B^1_{dj} \qquad d = 1, \ldots, n, \tag{2.8d}$$

$$P_d \geq \sum_{j \in J} p_j B^1_{dj} - \sum_{j \in J} (s_j + p_j) B^2_{dj} \qquad d = 1, \ldots, n, \tag{2.8e}$$

$$E_d + S_d \leq E_{d+1} \qquad d = 1, \ldots, n-1, \tag{2.8f}$$

$$E_d + S_d + P_d \leq E_{d+2} \qquad d = 1, \ldots, n-2, \tag{2.8g}$$

$$C_A \geq E_n + S_n + P_n \tag{2.8h}$$

$$C_A \geq E_{n-1} + S_{n-1} + P_{n-1} \tag{2.8i}$$

$$\sum_{j \in J} B^2_{dj} \leq \sum_{j \in J} H(j) B^1_{dj} \qquad d = 1, \ldots n, \tag{2.8j}$$

$$B^f_{dj} \in \{0, 1\} \qquad d = 1, \ldots n; j \in J; f \in \{1, 2\}, \tag{2.8k}$$

$$E_d, S_d, P_d \geq 0 \qquad d = 1, \ldots, n, \tag{2.8l}$$

$$C_A \geq 0 \tag{2.8m}$$

A função objetivo (2.8a) define C_{max} tal qual o explicado para a Equação (2.7). As restrições (2.8b) garantem que cada tarefa pertença a um bloco. Conforme as restrições (2.8c) apenas uma tarefa pertence ao primeiro nível de cada bloco. Os conjuntos de restrições (2.8d) e (2.8e) equivalem, respectivamente, as Equações (2.5) e (2.6).

De acordo com as restrições (2.8f) e (2.8g), uma tarefa modificada só pode iniciar seu *setup* após o término do *setup* da sua antecessora na sequência e após o término do processamento da sua antecessora na mesma máquina. As restrições (2.8h) e (2.8i) definem o *makespan* do escalonamento das tarefas modificadas. No conjunto de restrições (2.8j), se j for a tarefa de primeiro nível para algum bloco B_i, então $H(j)$ indica o número máximo de tarefas de segundo nível para o mesmo bloco. As restrições (2.8k) são os requisitos de integralidade e (2.8l) e (2.8m) de não negatividade.

2.3.5 Formulação F5 de Hamzadayi e Yildiz (2017)

Na formulação, E_i indica o tempo de início do processamento da tarefa i e as demais variáveis de decisão são:

$$w^l_{ij} = \begin{cases} 1, & \text{se a tarefa } i \text{ é processada antes da tarefa } j \text{ na máquina } l, \\ 0, & \text{caso contrário.} \end{cases}$$

$$u_{ij} = \begin{cases} 1, & \text{se o } \textit{setup} \text{ da tarefa } i \text{ é realizado antes do } \textit{setup} \text{ da tarefa } j, \\ 0, & \text{caso contrário.} \end{cases}$$

$$\varphi_j^l = \begin{cases} 1, & \text{se a tarefa } j \text{ é processada na máquina } l, \\ 0, & \text{caso contrário.} \end{cases}$$

A formulação segue:

$$\text{Minimizar} \quad C_{max} \tag{2.9a}$$

Sujeito a:

$$C_{max} \geq E_i + p_i \qquad \forall i \in J, \tag{2.9b}$$

$$\sum_{l \in M} \varphi_i^l = 1 \qquad \forall i \in J, \tag{2.9c}$$

$$\sum_{j \in J} w_{ij}^l \leq \varphi_i^l \qquad \forall i \in J; l \in M, \tag{2.9d}$$

$$\sum_{i \in J} w_{ij}^l \leq \varphi_j^l \qquad \forall j \in J; l \in M, \tag{2.9e}$$

$$w_{jj}^l = 0 \qquad \forall j \in J; l \in M, \tag{2.9f}$$

$$1 + \sum_{i \in J} \sum_{j \in J} w_{ij}^l = \sum_{i=1}^{n} \varphi_i^l \qquad \forall l \in M, \tag{2.9g}$$

$$w_{ij}^l \leq u_{ij} \qquad \forall i, j \in J; i \neq j; l \in M, \tag{2.9h}$$

$$w_{ik}^l - BM(6 - \varphi_i^l - \varphi_j^l - \varphi_k^l - u_{ij} - u_{jk} - u_{ik}) \leq 0 \qquad \forall i, j, k \in J; i \neq j; j \neq k; i \neq k; l \in M \tag{2.9i}$$

$$E_i + BM(4 - \varphi_i^l - \varphi_j^l - u_{ji} - w_{ji}^l) \geq E_j + p_j + s_{ji} \qquad \forall i, j \in J; i \neq j; l \in M, \tag{2.9j}$$

$$E_j + BM(3 - \varphi_i^h - \varphi_j^l - u_{ij}) \geq E_i + w_{kj}^l s_{kj} \qquad \forall i, j, k \in J; i \neq j; j \neq k; i \neq k; h, l \in M, \tag{2.9k}$$

$$E_i + BM(2 - \varphi_i^h - \varphi_j^l - u_{ij}) \geq E_j + w_{ki}^h s_{ki} \qquad \forall i, j, k \in J; i \neq j; j \neq k; i \neq k; h, l \in M, \tag{2.9l}$$

$$w_{ij}^l \in \{0, 1\} \qquad \forall i, j \in J; l \in M, \tag{2.9m}$$

$$u_{ij} \in \{0, 1\} \qquad \forall i, j \in J, \tag{2.9n}$$

$$\varphi_i^l \in \{0, 1\} \qquad \forall i \in J; l \in M, \tag{2.9o}$$

$$E_i \geq 0 \qquad \forall i \in J. \tag{2.9p}$$

O conjunto de restrições (2.9b) definem o *makespan*. As restrições (2.9c) garantem que cada tarefa deve ser alocada em apenas uma máquina. Os conjuntos de restrições (2.9d) e (2.9e) impedem que as tarefas i e j sejam processados sequencialmente na máquina l se ambas não tiverem sido atribuídas à máquina l. E o conjunto (2.9f) determina que uma tarefa não pode ser alocada sequencialmente com ela própria.

O conjunto de restrições (2.9g) definem as tarefas adjacentes a serem processadas na mesma máquina. Esse é exemplificado na Figura 2.6, que ilustra as tarefas atribuídas a uma mesma máquina l, a tarefa j é a última da sequência. É importante salientar que a formulação não considera operações de *setup* para a primeira tarefa alocada em uma máquina.

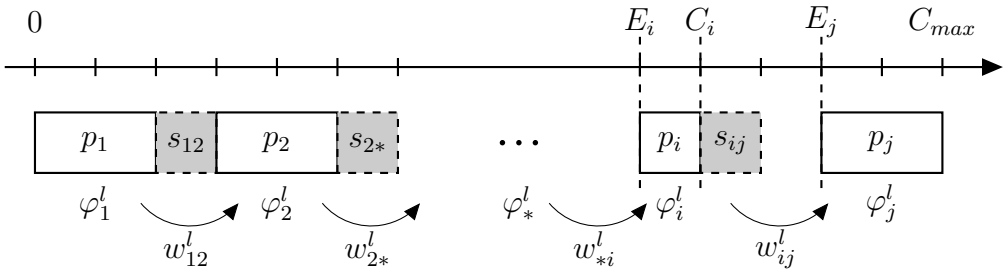

Figura 2.6: Sequência de tarefas na mesma máquina.

As restrições (2.9h), definem a interdependência das operações de *setup* e a sequência de tarefas processadas em qualquer máquina. De acordo com as restrições (2.9i), se as tarefas i, j e k forem sucessivamente atribuídas na mesma máquina l, logo $w_{ik}^l = 0$.

As restrições (2.9j) são usadas para definir a sequência de tarefas processadas pela mesma máquina, desse modo, uma tarefa só pode iniciar seu processamento após o término da sua antecessora. Vale notar que nessa formulação, o tempo de *setup* de uma tarefa é adicionado ao tempo de término da sua predecessora, como o mostrado na Figura 2.6, onde tem-se $w_{ij}^l = 1$ e o *setup*, s_{ij}, é realizado imediatamente após o término de i, em C_i, e antes do início de j, em E_j.

Os conjuntos de restrições (2.9k) e (2.9l) determinam a sequência das operações de *setup* pelo servidor e asseguram que essa operação, para qualquer tarefa, seja iniciada após a conclusão do *setup* da tarefa anterior. Os conjuntos (2.9m), (2.9n) e (2.9o) são requisitos de integralidade e o conjunto (2.9p) são restrições de não negatividade.

Capítulo 3

Aspectos Metodológicos

Este capítulo descreve os métodos propostos para a resolução dos problemas CS abordados nesse trabalho. Tendo em conta a pouca diversidade dos métodos exatos propostos na literatura, são apresentados duas formulações matemáticas com variáveis binárias com índices de tempo, sendo uma baseada na formulação indexada no tempo (TIF, *Time-Indexed Formulation*) e a outra no modelo de arcos indexados no tempo (ATIF, *Arc-Time-Indexed Formulation*). O TIF é capaz de prover soluções para o problema $P, S_1|s_j|C_{max}$ enquanto que o ATIF lida com o problema $P, S_1|s_{ij}|C_{max}$. Assume-se que todos as tarefas estão disponíveis no instante de tempo 0, porém, ao contrário do ATIF, o TIF considera a realização de *setup* da primeira tarefa alocada em cada máquina.

Devido o problema pertencer à classe $\mathcal{NP}$-difícil, e por isso os métodos exatos exigem um tempo computacional de ordem exponencial em relação ao tamanho do problema, são apresentados dois algoritmos heurísticos fundamentados na meta-heurística *Iterated Local Search* (ILS) (LOURENÇO et al., 2003). São denominados de HILS-CSvr e ILS_{CS}. Ambos utilizam a abordagem multi-start e recebem como parâmetros I_R e I_{ILS}, que são respectivamente o número máximo de: (i) *restarts*, e (ii) perturbações consecutivas sem melhorias na solução do procedimento ILS.

3.1 *Time-indexed formulation*

A formulação indexada no tempo (TIF) foi originalmente desenvolvida por Dyer e Wolsey (1990) para problemas de *scheduling* usando variáveis binárias y_j^t para indicar que uma tarefa j inicia seu processamento no tempo t. A formulação assume um horizonte de tempo de 0 a T, onde T é um limite superior do tempo máximo de conclusão de uma tarefa em alguma solução ótima.

A diferença entre o clássico e o TIF proposto para o problema CS, é que esse último utiliza a variável y_j^t para indicar que a tarefa j inicia seu tempo de *setup* no instante t, e consequentemente, o seu processamento se iniciará em $t + s_j$. Define-se ainda o tempo de duração da tarefa j como $\tau_j = s_j + p_j$. Desse modo, a formulação consiste em:

$$\text{Minimizar} \quad C_{max} \tag{3.1a}$$

Sujeito a:

$$\sum_{t=0}^{T-\tau_j} y_j^t = 1 \qquad \forall j \in J, \tag{3.1b}$$

$$\sum_{j \in J} \sum_{s=max\{0,t-\tau_j+1\}}^{t} y_j^s \leq m \qquad t = 0, \ldots, T-1, \tag{3.1c}$$

$$C_{max} \geq \sum_{t=0}^{T-\tau_j} (t + \tau_j) y_j^t \qquad \forall j \in J, \tag{3.1d}$$

$$\sum_{(j,t') \in D_t} y_j^{t'} \leq 1 \qquad t = 0, \ldots, T-\tau_j, \tag{3.1e}$$

$$y_j^t \in \{0,1\} \qquad \forall j \in J; t = 0, \ldots, T-\tau_j. \tag{3.1f}$$

A função objetivo (3.1a) minimiza o *makespan*. As restrições (3.1b) garantem que a execução da tarefa j seja iniciada apenas uma vez no período de tempo t. O conjunto (3.1c) são restrições impondo que cada tarefa j deve ser processada durante τ_j períodos consecutivos e que a quantidade de tarefas em execução, ao mesmo tempo, é no máximo igual ao número de máquinas. As restrições (3.1d) determinam o *makespan*, ou seja, o tempo de término da última tarefa ainda em processamento.

Tendo em vista que não pode haver sobreposição de *setups*, as restrições (3.1e) são definidas. Estas significam que dado D_t o conjunto de variáveis $y_j^{t'}$, que quando ativas implicarão na ocorrência de *setup* no tempo t, no máximo uma tarefa poderá realizar *setup* naquele instante. As restrições (3.1f) definem o domínio das variáveis de decisão.

3.2 *Arc-time-indexed formulation*

A formulação de arcos indexados no tempo foi originalmente desenvolvido por Pessoa et al. (2010) para o problema $1||\sum w_j T_j$. O modelo ATIF proposto para o problema CS

baseia-se naquele de Silva et al. (2018) para uma variante do problema de aceitação de tarefas e escalonamento com tempos de *setup* dependentes da sequência.

A formulação também assume um horizonte de tempo de 0 a T. Cada variável é associada a um arco em uma rede onde os nós possuem índices de tarefa e tempo. Essa rede é definida pelo grafo acíclico $G = (V, A)$ e os arcos $((i, t-p_i-s_{ij})(j, t))$ são denotados por (i, j, t). O conjunto de nós é definido por $V = N \cup D$, onde o conjunto $N = \{(j, t) : j \in J, t = 0, \ldots, T - p_j\}$ contém os vértices associados as tarefas e $D = \{(0, t) : t = 0, \ldots, T\}$ é o conjunto de vértices associado com a tarefa *dummy* 0. Os tempos ociosos ocorrem antes da tarefa ser processada. A tarefa j poderá começar ser processada quando chegar ao vértice (j, t), mas não necessariamente no instante t devido a existência de ociosidade. A tarefa 0 é a primeira e a última tarefa a ser processada em cada máquina, com $p_0 = 0$ e $s_{0j} = s_{j0} = 0, \forall j \in J$. A sequência em cada máquina é representada por um caminho, que começa no nó $(0, 0)$ e termina em um nó $(0, t), t \leq T$.

O conjunto de arcos é dado por $A = A^1 \cup A^2 \cup A^3 \cup A^4$, onde o conjunto $A^1 = \{(i, j, t) = ((i, t - p_i - s_{ij})(j, t)) : (i, t - p_i - s_{ij}) \in N, (j, t) \in N, j \in J \setminus \{i\}\}$ contém os arcos que conectam os vértices de N; $A^2 = \{(0, j, 0) = ((0, 0)(j, 0)) : (j, 0) \in N\}$ contém todos os arcos que conectam os vértices de D para N; O conjunto $A^3 = \{(j, 0, t) = ((j, t - p_j - s_{j0})(0, t)) : (j, t - p_j - s_{j0}) \in N, (0, t) \in D, j \in J\}$ contém todos os arcos que conectam os vértices de N para D; O conjunto $A^4 = \{(j, j, t) = ((j, t - 1)(j, t)) : (j, t - 1) \in N, (j, t) \in N\}$ contém os arcos associados aos tempos ociosos. Os arcos (j, j, t) armazenam as informações referentes a tarefa j, enquanto esta permanece ociosa, evitando o conflito de *setup* e permitindo que o tempo de *setup* da tarefa j para a sua sucessora seja computado corretamente (SILVA et al., 2018).

Seja $c_a = c(i, j, t)$ o custo de um arco $a = (i, j, t) \in A^1 \cup A^2$, que é o custo se a tarefa j inicie seu processamento no instante t, $c(i, j, t) = f_j(t + p_j)$. Para todos os arcos $a \in A^3 \cup A^4$ teremos que $c_a = 0$. O conjunto $N^j = \{(i, t) \in N : i = j\}$ contém os arcos associados com a tarefa j. Para cada vértice $v \subset V$, temos $\delta^-(v)$ e $\delta^+(v)$ os conjuntos representando os arcos que entram e saem de v, respectivamente. A formulação pode ser escrita da seguinte maneira:

$$\min \quad C_{max} \tag{3.2a}$$

Subject to:

$$\sum_{a \in \delta^-(N^j) \setminus A^4} x_a = 1, \qquad \forall j \in J \tag{3.2b}$$

$$\sum_{a \in A^2} x_a = m, \tag{3.2c}$$

$$\sum_{a \in \delta^-(\{v\})} x_a - \sum_{a \in \delta^+(\{v\})} x_a = 0, \qquad \forall v \in V \setminus \{(0,0),(0,t)\} \tag{3.2d}$$

$$C_{max} \geq \sum_{a \in \delta^-(N^j)} c_a x_a, \qquad \forall j \in J \tag{3.2e}$$

$$\sum_{a' \in D_t} x_{a'} \leq 1, \qquad t = 0, \ldots, T-1 \tag{3.2f}$$

$$x_a \in \{0,1\} \qquad \forall a \in A \tag{3.2g}$$

A função objetivo (3.2a) minimiza o *makespan*. O conjunto de restrições (3.2b) indicam que uma tarefa deve ser executada apenas uma vez. A restrição (3.2c) garante que m máquinas serão usadas para o processamento das tarefas. As restrições (3.2d) asseguram a preservação de fluxo no grafo previamente definido. As restrições (3.2e) determinam o *makespan*. No conjunto de restrições (3.2f), o arco a' é definido por (i, j, t') e D_t é o conjunto de variáveis $x_{a'}$, que quando ativas implicarão na ocorrência de *setup* no tempo t, e no máximo uma tarefa poderá realizar *setup* naquele instante. As restrições (3.2g) definem o domínio das variáveis de decisão.

A Figura 3.1 mostra o diagrama de Gantt para a solução ótima de uma instância exemplo com 9 tarefas e 3 máquinas. As informações das tarefas são encontradas na Tabela 3.1. Nesse caso o valor do *makespan* é de 12 unidades de tempo e a solução é composta pelas variáveis não-nulas x_a associadas aos seguintes arcos: $(0,1,0)$, $(0,6,0)$, $(0,7,0)$, $(1,1,1)$, $(1,4,6)$, $(2,0,12)$, $(3,5,7)$, $(4,2,10)$, $(5,0,11)$, $(6,3,4)$, $(7,8,5)$, $(8,9,8)$ e $(9,0,11)$. O grafo associado a essa solução é representado na Figura 3.2. O arco $(1,1,1)$ indica que a máquina M1 permanece ociosa entre os tempos 0 e 1, atrasando assim o ínicio do processamento da tarefa 1. Isso acontece para evitar o conflito entre as operações de *setup* da tarefa 8, na máquina M3, e a tarefa 4 na máquina M1

3.3 Algoritmo HILS-CSvr

O HILS-CSvr é um algoritmo híbrido que utiliza os princípios da meta-heurística ILS para resolver uma versão relaxada do problema onde não se consideram as restrições de

Tabela 3.1: Instância exemplo com 9 tarefas

j	p_j	s_{ij}								
		1	2	3	4	5	6	7	8	9
1	4	0	2	1	1	2	2	2	1	3
2	2	2	0	1	2	1	2	2	1	2
3	2	3	1	0	2	1	2	2	2	3
4	3	3	1	2	0	1	1	3	2	2
5	4	2	2	2	2	0	1	1	2	4
6	3	3	2	1	2	1	0	3	1	2
7	4	2	2	2	2	1	3	0	1	2
8	2	3	1	1	2	1	2	2	0	1
9	3	2	1	2	1	1	2	2	4	0

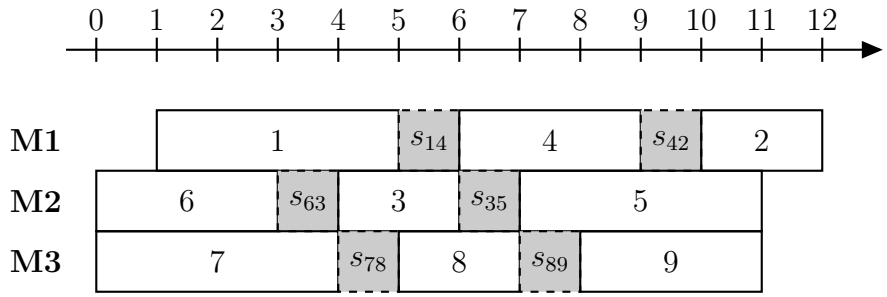

Figura 3.1: Diagrama de Gantt de uma instância exemplo com 9 tarefas e 3 máquinas.

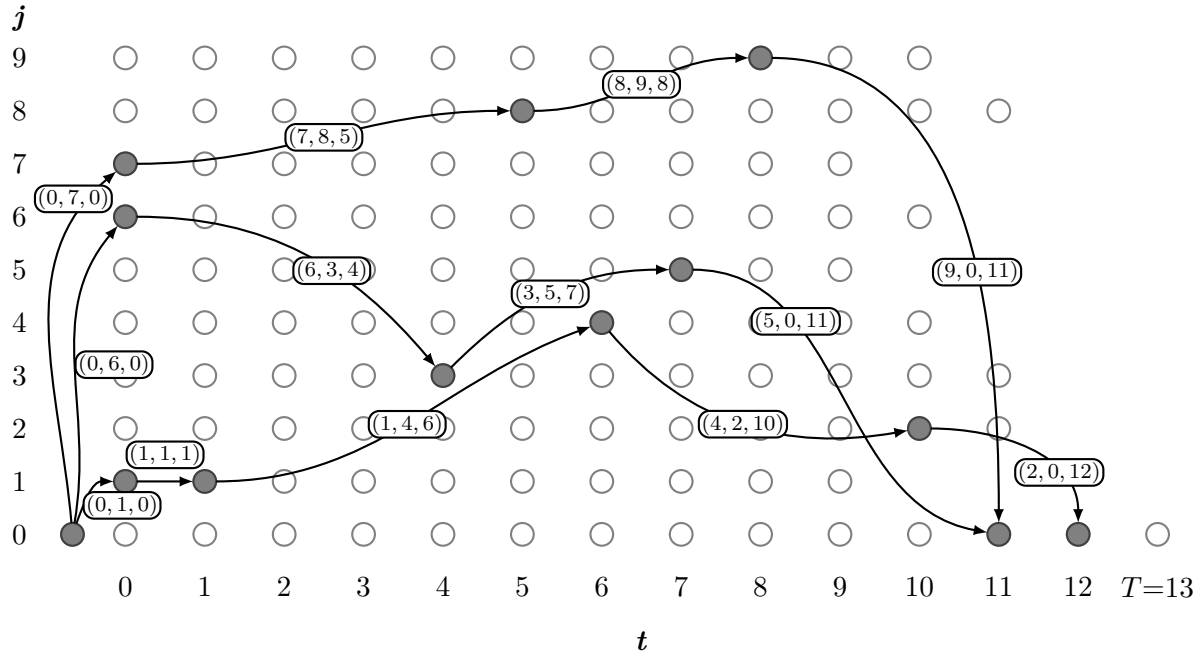

Figura 3.2: Representação dos caminhos em um grafo associado a solução ATIF.

conflitos de *setup*. Uma versão restrita do modelo descrito na Seção 3.2 ou um procedimento heurístico baseado na fase construtiva da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) (FEO; RESENDE, 1995) é, então, empregado para gerar soluções viáveis para o problema original a partir da solução relaxada.

O HILS-CSvr é apresentado no Algoritmo 1. O método recebe, além de I_R e I_{ILS}, os parâmetros α e I_{GSV}, que são utilizados no procedimento heurístico para geração da solução viável dada uma solução relaxada. O parâmetro α, controla o nível de gulosidade e aleatoriedade durante a geração da solução viável. I_{GSV} é semelhante ao I_R.

Algoritmo 1: HILS-CSvr

```
1  Procedimento HILS-CSvr(I_R, I_ILS, α, I_GSV)
2  f* ← ∞
3  P = ∅;   // P é o conjunto de soluções já visitadas
4  para i ← 1,...,I_R faça
5      s ← GeraSolucaoInicial(α)
6      s' ← s
7      Iter_ILS ← 0
8      enquanto Iter_ILS < I_ILS faça
9          s ← BuscaLocal(s)
10         se f(s) ≤ f(s') e f(s) < f* então
11             se f(s) < f(s') então
12                 Iter_ILS ← 0
13             fim
14             s' ← s
15             se s' ∉ P então
16                 P ← P ∪ s'
17                 se instância é de médio/grande porte então
18                     f'' ← GeraSolucaoViavel(s', α, I_GSV)
19                 senão
20                     se f* = ∞ então
21                         T ← f(s') + n   // T é o horizonte de tempo
22                     senão
23                         T ← f*
24                     fim
25                     f'' ← ATIF(s', T)
26                 fim
27                 se f'' < f* então
28                     s* ← s'   f* ← f''
29                 fim
30             fim
31         fim
32         s ← Perturbacao(s)
33         Iter_ILS ← Iter_ILS + 1
34     fim
35 fim
36 retorna s*
37 fim HILS-CSvr
```

A solução inicial (linha 5) é melhorada por meio de um procedimento de busca local (linha 9) combinado com o mecanismo de perturbação (linha 32). Sempre que a busca

local retornar uma solução s com o valor objetivo $f(s) \leq f(s')$, a melhor solução corrente (s') é atualizada (linhas 10–29). Uma vez que isso ocorre, um método para gerar soluções viáveis é utilizado caso a solução já não tenha sido visitada (linhas 15–29). A solução viável é obtida de duas maneiras: ATIF (linha 25) ou o procedimento `GeraSolucaoViavel` (linha 18). A chamada desses métodos depende do porte da instância (linha 17), nesse trabalho considerou-se instâncias de médio/grande porte aquelas com $n \geq 49$ ou $m \geq 10$. Por fim, a heurística retorna a melhor solução viável s^* entre todas as iterações.

3.3.1 Geração da solução inicial

O procedimento para geração da solução inicial funciona da seguinte forma. Inicialmente m tarefas pertencentes a J são escolhidas, aleatoriamente, e atribuídas uma a cada máquina. Em seguida, procedimentos de inserção são aplicados até que todas as outras tarefas tenham sido adicionadas a solução.

A cada chamada da função para construir uma solução inicial, o método decide de forma aleatória a estratégia (sequencial ou paralela) e critério de inserção (mais barata ou mais próxima) a serem adotados. A estratégia de inserção sequencial consiste em considerar uma máquina diferente a cada iteração do procedimento construtivo, ao passo que a estratégia paralela leva em conta todas as máquinas simultaneamente ao avaliar a inserção mais barata ou próxima.

Ambas as estratégias de inserção desconsideram a máquina com o maior tempo de processamento durante o cálculo do custo de inserção. Seja J' o conjunto de tarefas escalonadas e J'' o conjunto das tarefas não escalonadas. A inserção mais barata de uma tarefa $h \in J''$ entre duas tarefas $i \in J'$ e $j \in J'$ é dada por $\min\{s_{ih} + s_{hj} - s_{ij} + p_h \mid i, j \in J,' h \in J''\}$. Já a inserção mais próxima é dada por $\min\{s_{ih} + p_h \mid i \in J', h \in J''\}$.

3.3.2 Busca local

A fase de busca local é baseada no procedimento *Variable Neighborhood Descent* (VND) (MLADENOVIC; HANSEN, 1997) com ordem aleatória na escolha das vizinhanças. Tal procedimento seleciona aleatoriamente uma vizinhança inexplorada para continuar a busca sempre que outra não obtiver sucesso em encontrar uma solução melhorada. Se ocorrer uma melhoria, todas as vizinhanças serão reconsideradas para serem selecionadas.

Foram considerados dois grupos de estruturas de vizinhança: Vizinhança entre Máqui-

nas (VE) e intra Máquinas (VI). Na primeira, os movimentos de busca são realizados entre duas maquinas, enquanto que na segunda, os movimentos são realizados entre as tarefas da própria máquina. Nesse caso, seleciona-se uma estrutura de vizinhança VE aleatoriamente. Caso a solução seja melhorada, todas as vizinhanças VE serão reconsideradas para uma nova seleção, caso contrário, a estrutura é eliminada de VE.

A cada melhora da solução retornada pelas estruturas VE, é também realizada uma busca local utilizando as estruturas VI, que ocorre do mesmo modo que o descrito para VE. Uma vez que VE esteja vazia, é realizada uma última busca intra máquinas.

3.3.2.1 Estruturas VE

As seguintes estruturas VE foram implementadas.

- *Cross* — A sequência de duas máquinas são divididas em dois segmentos, em seguida, o segmento inicial de cada uma é unido no segmento final da outra, no mesmo ponto de corte, como mostrado na Figura 3.3(a).
- *Reallocation* (h) — Um número h de tarefas adjacentes é retirado de uma máquina e realocado em outra, com $1 \leq h \leq 3$. Na Figura 3.3(b) é exemplificado o movimento para $h = 1$.
- *Swap* (h_1, h_2) — Um número h_1 de tarefas adjacentes é trocado de máquina com um número h_2 de tarefas adjacentes de outra máquina, com $1 \leq h_1 \leq 3$ e $1 \leq h_2 \leq h_1$. A Figura 3.3(c) exemplifica o funcionamento dessa vizinhança para $h_1 = 3$ e $h_2 = 2$.

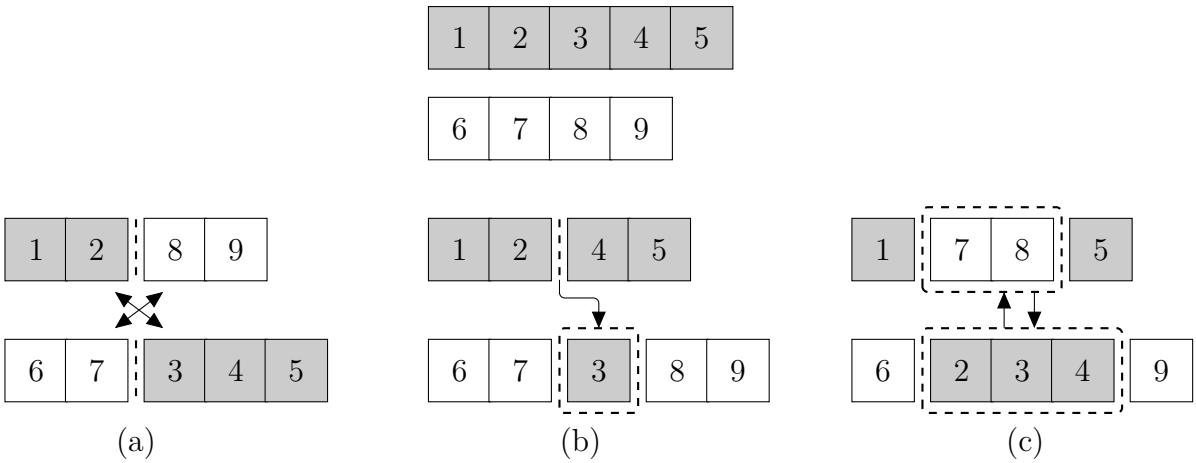

Figura 3.3: Vizinhanças VE: *Cross* (a), *Reallocation*(1) (b) e *Swap*$(3, 2)$ (c).

3.3.2.2 Estruturas VI

As seguintes estruturas VI foram consideradas.

- *Reinsertion* (h) — h tarefas adjacentes são reinseridas em outra posição na máquina, com $1 \leq h \leq 4$. Na figura Figura 3.4(a) é mostrado um exemplo para essa vizinhança com $h = 2$.
- *Block Reverse* — A ordem de processamento de um bloco de tarefas é invertida, conforme ilustrado (para um bloco de tamanho 3) na Figura 3.4(b).
- *Job Exchange* — Duas tarefas são trocadas de posição na sequência, assim como o demonstrado na Figura 3.4(c).

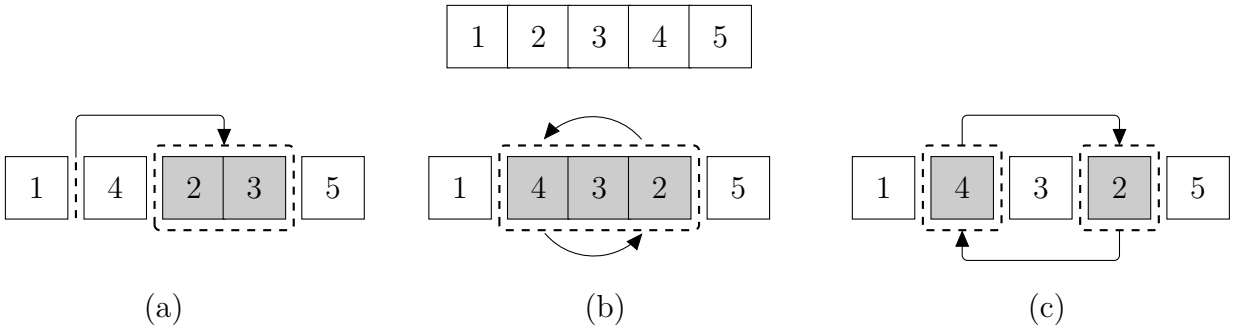

Figura 3.4: Vizinhanças VI: *Reinsertion*(2) (a), *Block Reverse* (b), *Job Exchange* (c).

3.3.3 Solução viável

A solução viável é obtida de duas maneiras: ATIF ou o procedimento `GeraSolucaoViavel`. Ambos recebem a sequências pré-definidas, bastando realizar os ajustes para evitar os conflitos de *setup*. Para o ATIF, descrito na Seção 3.2, uma vez que a sequência está pré-definida, é declarado no modelo apenas as variáveis que preservam a sequência, resultando em uma diminuição considerável da quantidade de variáveis e no tempo de execução do modelo. Ainda para o ATIF, em máquinas com até de 4 tarefas alocadas é permitida a permutação na sequência.

O procedimento `GeraSolucaoViavel`, descrito pelo Algoritmo 2, é empregado para gerar a solução viável das instâncias de médio/grande porte, pois nessas instâncias o modelo ATIF requer um tempo CPU relativamente elevado. Os termos usados nesse algoritmo estão descritos na Tabela 3.2. O algoritmo recebe como entrada a solução relaxada s' e os parâmetros α e I_{GSV} previamente descritos. As tarefas são inseridas na

solução uma a uma, de acordo com a sequência pré-definida s'_k, em s', para cada máquina k (linhas 7–31).

Algoritmo 2: GeraSolucaoViavel

```
1  Procedimento GeraSolucaoViavel(s', α, I_GSV)
2  f'' ← ∞
3  enquanto iter_GSV < I_GSV faça
4      TP_t ← 0,  t = 1, ..., T_max   // T_max é um valor suficientemente grande
5      CT_k ← 0,  ∀k ∈ M
6      M_a ← k,  ∀k ∈ M
7      enquanto M_a ≠ ∅ faça
8          para k ∈ M_a faça
9              se CT_k = 0 então
10                 g(k) ← p^k_{j'}
11             senão
12                 repita
13                     ST ← CT_k
14                     para t ← ST, ..., ST + d^k_{jj'} faça
15                         se TP_t ≠ 0 então
16                             CT_k ← t
17                         fim
18                     fim
19                 até CT_k = ST;
20                 g(k) ← ST + d^k_{jj'} + p^k_{j'}
21             fim
22         fim
23         Insira em LCR os α × |M_a| menores valores g(k), k ∈ M_a
24         Selecione, aleatoriamente, um elemento k ∈ LCR
25         CT_k ← g(k)
26         Limpe LCR
27         Atualize TP
28         s'_k ← s'_k − {j'}
29         se s'_k = ∅ então
30             M_a ← M_a − {k}
31         fim
32     fim
33     C_max ← max{CT_k | k ∈ M}
34     se C_max < f'' então
35         f'' ← C_max
36         iter_GSV ← 0
37     fim
38     iter_GSV ← iter_GSV + 1
39 fim
40 retorna f''
41 fim GeraSolucaoViavel
```

A cada iteração, é calculado o tempo de conclusão da próxima tarefa da sequência

Tabela 3.2: Termos do algoritmo `GeraSolucaoViavel`

Termos	Definição
TP_t	Se $TP_t \neq 0$ indica que no instante t está sendo realizado *setup*, ($t = 1, \ldots, T_{max}$).
CT_k	O tempo de conclusão na máquina k, $\forall k \in M$
LCR	Lista de candidatos restrita
M_a	O conjunto de máquinas com tarefas não alocadas
ST	Armazena temporariamente o instante de tempo que uma tarefa poderá começar sem ocasionar conflitos de *setup*
$p^k_{j'}$	O tempo de processamento da atual tarefa na sequência da máquina k
$d^k_{jj'}$	O tempo de *setup* da atual tarefa j' ao suceder a última tarefa j, alocadas na máquina k
$g(k)$	Armazena o tempo de término temporário da máquina k

de cada máquina (linhas 8–22). Se a tarefa j' é a primeira da sequência na máquina k, seu tempo de conclusão será igual ao seu tempo de processamento $p^k_{j'}$ (linha 10), caso contrário, são analisadas as possibilidades de conflitos na sua inclusão, caso haja, seu tempo de início é ajustado de modo a evitá-los (linhas 11–21) e então o tempo de conclusão é computado (linha 20). De modo semelhante a fase construtiva do GRASP, uma máquina dentre aquelas com o menor tempo de término é aleatoriamente escolhida (linhas 23–24), e então o tempo em CT_k é atualizado (linha 25). Uma vez que todas as tarefas já tenham sido computadas na solução viável, o *makespan* será igual ao maior tempo de conclusão dentre todas as máquinas (linha 33). Por fim, é retornado o valor objetivo f'' da solução viável.

3.3.4 Mecanismos de perturbação

De acordo com Lourenço et al. (2003), o movimento de perturbação é responsável por transformar uma solução de boa qualidade em um bom ponto de partida para se realizar a busca local, evitando assim ótimos locais. Em oposição aos movimentos realizados na busca local, os movimentos de perturbação não possuem como objetivo a melhoria da solução. As perturbações adotadas pelo algoritmo são as seguintes.

- *Machine reallocation* — Duas tarefas, em máquinas distintas, são selecionadas aleatoriamente e reinseridas em posições arbitrárias da outra máquina. O movimento é realizado múltiplas vezes.

- *Machine swap* — Duas tarefas, em máquinas distintas, são selecionadas aleatoriamente e permutadas. O movimento é realizado múltiplas vezes.

3.4 Algoritmo ILS_{CS}

O algoritmo ILS_{CS} é mais simples que o HILS-CSvr e também alterna entre procedimentos de intensificação e pertubação com o objetivo de escapar de ótimos locais. Por conveniência primeiro é apresentada a representação da solução e o método de avaliação da mesma, e logo após é mostrado o algoritmo ILS_{CS} proposto.

3.4.1 Representação da solução

A solução é representada pela sequência de prioridade $(j_1, j_2, j_3, \ldots, j_k, \ldots, j_{|J|})$, onde a tarefa na k-ésima posição da sequência será a k-ésima tarefa a ser processada. As tarefas são atribuídas uma a uma, seguindo a ordem de prioridade, na máquina onde possuírem menor tempo de conclusão. Para avaliar o custo da solução é aplicado o procedimento chamado de *decoding algorithm*. O *decoding algorithm* proposto para a avaliação da solução é baseado naquele desenvolvido por Hamzadayi e Yildiz (2017). Por conveniência, os denotaremos por MDA (*Modified Decoding Algorithm*) e ODA (*Original Decoding Algorithm*), respectivamente. Eles são mostrados nos Algoritmos 3 e 4.

O Algoritmo 3 funciona da seguinte maneira: a estrutura do diagrama de Gantt é inicializada (linhas 2–4); identifica a máquina na qual a tarefa j possui o menor tempo de conclusão (linhas 8–27); e atualiza o diagrama de Gantt (linhas 28–38). Depois que todas as tarefas são atribuídas (linhas 5–39) é retornado o *makespan* (linha 40).

Vale ressaltar que no pseudocódigo do *decoding algorithm* apresentado em Hamzadayi e Yildiz (2017) existem alguns erros que comprometem a avaliação da solução. Nas linhas 15 e 31 do ODA, originalmente teríamos $CS = 0$, o que permitiria a sobreposição de tarefas na mesma máquina. Pois CS armazena o último instante de tempo na qual foi realizado *setup*, e o tempo de conclusão da última tarefa atribuída na mesma máquina pode ser maior que o valor atual de CS. Para exemplificar, usando os dados da Tabela 3.1, o escalonamento mostrado na Figura 3.5 é gerado pela sequência de prioridade $(4, 7, 9, 6, 8, 2, 5, 1, 3)$. Veja que a tarefa 5 é a sétima a ser processada. Antes de ser alocada temos que $CS = 6$, e o custo seria computado com o *setup* da tarefa 5 iniciando nesse instante de tempo, o que provocaria a sobreposição com a tarefa 6, e assim, o erro na avaliação do custo. Por isso, é necessário a substituição $CS = 0$ por $CS \leq E_{j,m}$, o que permite evitar essa situação.

O Algoritmo 4 opera do seguinte modo: a estrutura do diagrama de Gantt é inicializada (linhas 2–5); identifica a máquina na qual a tarefa j possui o menor tempo de

Algoritmo 3: Original Decoding Algorithm

```
1  Procedimento OriDecAlgorithm (s(.))
2  S_{j,m} ← 0  ∀j ∈ J, ∀m ∈ M   // Tempo que a máquina m inicia a processar a tarefa j
3  E_{j,m} ← 0  ∀j ∈ J, ∀m ∈ M   // Tempo que a máquina m conclui a tarefa j
4  CS ← 0                          // Tempo em que o servidor fica disponível
5  para k ← 1 to |J| faça
6      j ← s(k)                    // Tarefa na k-ésima posição da sequência
7      f' ← ∞
8      para m ← 1 to |M| faça
9          se E_{j,m} = 0 então
10             se f' ≥ p_j então
11                 f' ← p_j
12                 m' ← m
13             fim
14         senão
15             se CS ≤ E_{j,m} então
16                 se f' ≥ E_{j,m} + s_ij + p_j então
17                     f' ← E_{j,m} + s_ij + p_j  // i é a última tarefa alocada na máquina m
18                     m' ← m
19                 fim
20             senão
21                 se f' ≥ CS + s_ij + p_j então
22                     f' ← CS + s_ij + p_j
23                     m' ← m
24                 fim
25             fim
26         fim
27     fim
28     se E_{j,m'} = 0 então
29         S_{j,m'} ← 0 E_{j,m'} ← p_j
30     senão
31         se CS ≤ E_{j,m'} então
32             S_{j,m'} ← E_{j,m'} + s_ij
33         senão
34             S_{j,m'} ← CS + s_ij
35         fim
36         E_{j,m'} ← S_{j,m'} + p_j
37         CS ← S_{j,m'}
38     fim
39 fim
40 retorna max{E_{j,m'} | j ∈ J, m ∈ M}
```

conclusão (linhas 9–14); atribui a tarefa j a máquina onde será processado (MDA:linha 19) e atualiza o diagrama de Gantt (linhas 20–22). Após atibuir todas a tarefas (linhas 6–23) é retornado o *makespan* (linha 24). Além da sequência da prioridade, o MDA recebe como parâmetro o *makespan* da melhor solução conhecida na fase atual do procedimento de busca local do algoritmo meta-heurístico proposto. Isto permite que avaliação da solu-

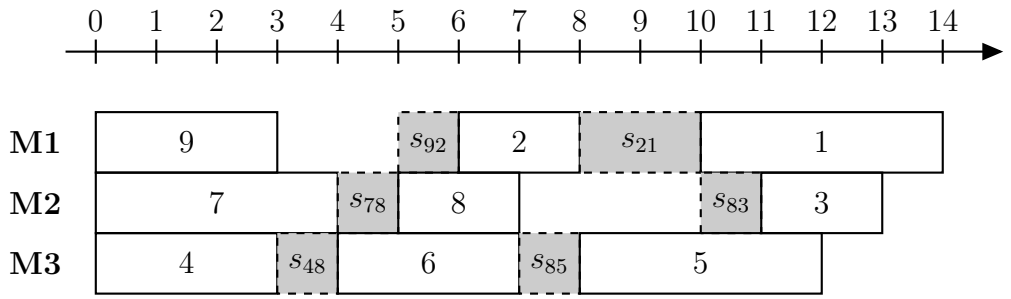

Figura 3.5: Diagrama de Gantt para uma solução viável gerada pelo ODA.

ção atual seja interrompida caso seu custo em algum momento antes de fim da avaliação ultrapasse o melhor custo conhecido (linhas 15–18), proporcionando um ganho em relação ao tempo de execução. Ambos os algoritmos possuem complexidade $\mathcal{O}(nm)$. Porém o MDA consegue ser mais rápido por possuir um número reduzido de operações aritméticas e seleção e também uma estratégia de interrupção da avaliação.

Algoritmo 4: Modified Decoding Algorithm

```
1  Procedimento ModDecAlgorithm (s(.), f(s'))
2  A_j ← ∅  ∀j ∈ J ;                          // Máquina onde a tarefa j será processada
3  S_j ← 0  ∀j ∈ J ;                  // Tempo de início do processamento da tarefa j
4  E_m ← 0  ∀m ∈ M ;                          // Tempo de conclusão da máquina m
5  CS ← 0
6  para k ← 1 to |J| faça
7      j ← s(k)
8      f' ← ∞
       /* Se a máquina m não tiver tarefas alocadas:  s_ij = 0                  */
9      para m ← 1 to |M| faça
10         se f' > max(E_m, CS) + s_ij + p_j então
11             f' ← max(E_m, CS) + s_ij + p_j
12             m' ← m
13         fim
14     fim
15     se f' > f(s') então
16         E_m' ← f'
17         Vá para linha 24
18     fim
19     A_j ← m'
20     S_j ← max(E_m', CS) + s_ij
21     E_m' ← f'
22     CS ← S_j
23 fim
24 retorna max{E_m|m ∈ M}
```

3.4.2 ILS_{CS}

O fluxograma do algoritmo ILS_{CS} é apresentado na Figura 3.6.

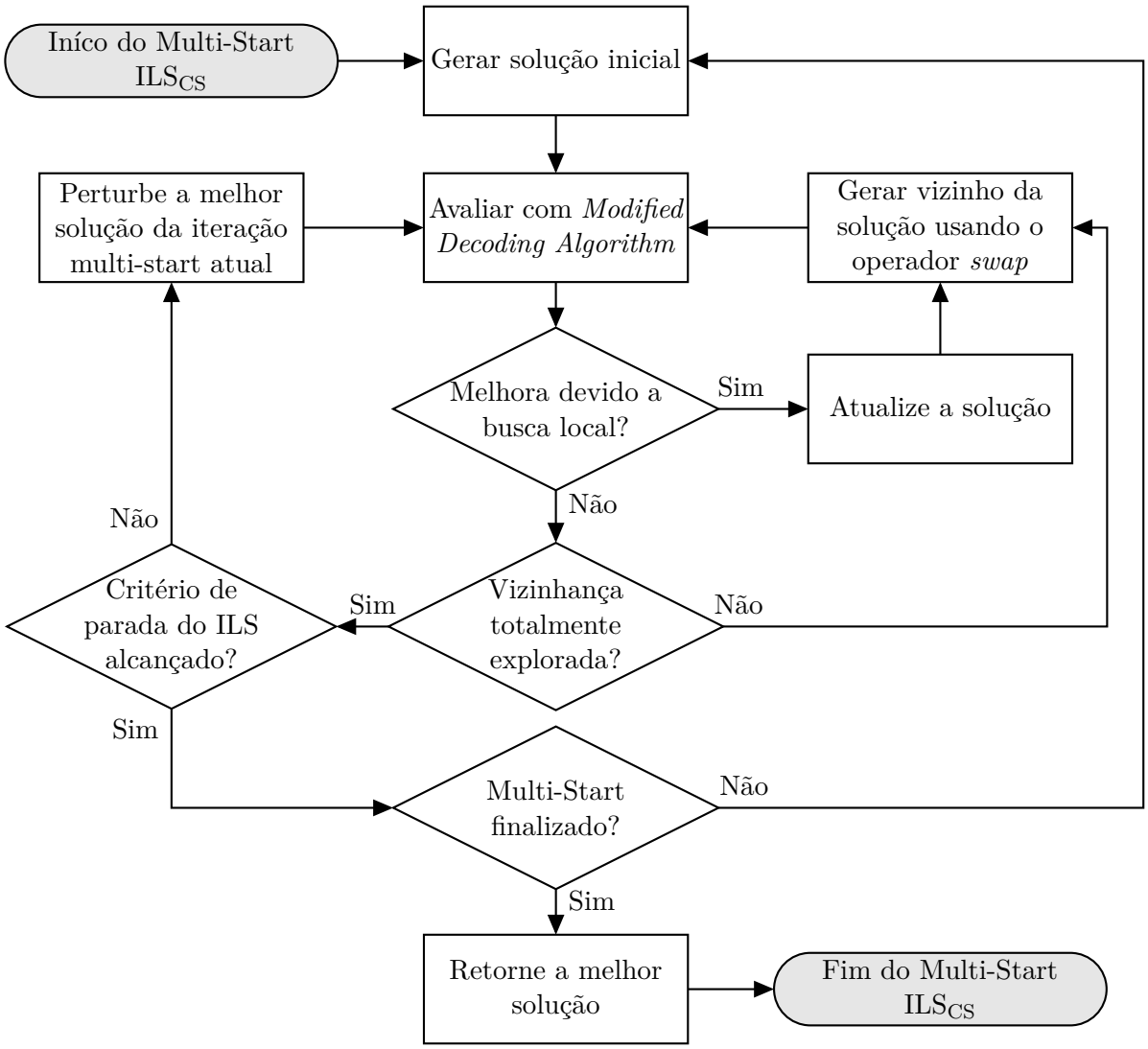

Figura 3.6: Fluxograma do algoritmo ILS_{CS}

A heurística multi-start inicia gerando uma solução inicial usando um procedimento de inserção completamente aleatório. Em seguida, a fase de busca local é realizada usando o operador de troca (*swap*), conforme descrito em Algoritmo 5. Observe que as tarefas das m primeiras posições só são trocadas por aquelas após a m-ésima posição, de modo a evitar gerar soluções simétricas ao aplicar o algoritmo de decodificação que, por sua vez, atribui as m primeiras tarefas da sequência para cada máquina. Optou-se por adotar a estratégia de primeira melhora (*first improvement*), pois ela ajudou a acelerar o procedimento de busca local sem comprometer a qualidade da solução quando comparada à estratégia de melhor melhora (*best improvement*). A complexidade geral de enumerar ($\mathcal{O}(n^2)$) e avaliar ($\mathcal{O}(nm)$) todos os movimentos de troca é da ordem de $\mathcal{O}(n^3m)$. Além disso, sempre que

uma solução é modificada, seja por um movimento de busca local ou por um movimento de perturbação, a busca é reiniciada.

Algoritmo 5: Busca Local

```
1  Procedimento BuscaLocal(s)
2  para i ← 1 até |J| faça
3  |   k ← max(|M|, i) + 1
4  |   para j ← k até |J| faça
5  |   |   s' ← Swap(s(i), s(j)) ;        // Tarefas nas posições i,j-ésimas posições são
   |   |    trocadas
6  |   |   f ← ModDecAlgorithm(s', f(s))
7  |   |   se f < f(s) então
8  |   |   |   s ← s'
9  |   |   |   Vá para linha 2
10 |   |   fim
11 |   fim
12 fim
13 retorna s
```

Vale ressaltar que outras estruturas de vizinhança, como a reinserção de tarefas, foram implementadas, mas experimentos preliminares revelaram que eles não contribuem para melhorar a qualidade da solução. Portanto, optou-se por usar apenas a vizinhança de troca. Para exemplificar o movimento de troca considere a sequência de prioridade $(4, 7, 9, 6, 8, 2, 5, 1, 3)$, que gera a solução mostrada previamente na Figura 3.5. Ao trocarmos de posições as tarefas 1 e 9, a sequência resultante gera a solução mostrada na Figura 3.7, e essa é uma solução ótima para o problema da instância da Tabela 3.1.

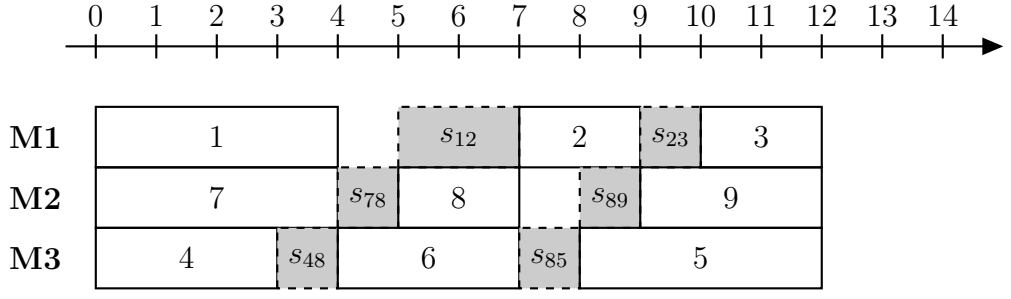

Figura 3.7: Diagrama de Gantt para uma solução ótima gerada pelo MDA.

Além disso, se uma solução local ótima for encontrada e o número máximo de perturbações consecutivas sem melhorias for menor que I_{ILS} (isto é, o critério de parada ILS não for atendido), o algoritmo modifica a solução da atual iteração multi-start por meio de um mecanismo de perturbação e a busca continua a partir dessa solução perturbada. O mecanismo de perturbação consiste em múltiplos *swaps*. Mais especificamente, 1, 2 ou 3 pares de tarefas distintas são escolhidas aleatoriamente e depois trocadas. Se o critério de

parada do ILS for atendido, o algoritmo será reiniciado desde o início. Se o número máximo de reinicializações for atingido, o ILS_{CS} será encerrado e retornará a melhor solução encontrada.

Uma observação importante é que não é permitido que o MDA seja interrompido quando calcula-se o custo de uma solução inicial ou perturbada, pois nesses casos deve-se determinar o custo de qualquer maneira. Portanto, em tais situações, simplesmente é fornecido um número suficientemente grande como custo de entrada para o procedimento. Isso naturalmente impedirá que o MDA seja prematuramente interrompido.

Capítulo 4

Resultados Computacionais

Este capítulo apresenta os resultados obtidos pela aplicação do modelo ATIF e do algoritmo ILS_{CS} e faz uma comparação com os resultados de outros métodos reportados na literatura. Mediante a indisponibilidade das instâncias na literatura, novas instâncias foram geradas conforme Hamzadayi e Yildiz (2017) e Kim e Lee (2012) para o problema com tempos de *setup* dependentes e independentes da sequência, respectivamente. A formulação F5, o SA e o AG de Hamzadayi e Yildiz (2017) e as formulações de Kim e Lee (2012), Gan et al. (2012), Hasani et al. (2014a) foram reimplementadas.

Os algoritmos heurísticos foram implementados em linguagem de programação C++ (g++ 5.4.0) e os experimentos foram executados em um Intel Core i7, com 3,4 GHz, 16 GB de memória RAM e sistema operacional Linux Ubuntu 16.04. Os métodos exatos, sob o mesmo ambiente computacional, foram executados no resolvedor CPLEX 12.7 considerando apenas uma *thread*. Nos testes reportados posteriormente, todos os métodos heurísticos foram executados 10 vezes para cada instância.

Primeiro (Seção 4.1) são descritas as características das instâncias, depois são apresentados os experimentos realizados para a calibração dos parâmetros das heurísticas na Seção 4.2. Na Seção 4.3 são apresentados e comparados os resultados dos métodos exatos e heurísticos. Por fim (Seção 4.4), têm-se os resultados do impacto na performance do MDA dobre o ODA.

4.1 Instâncias

Para o problema com tempos de *setup* dependentes da sequência, as instâncias foram geradas como em Balakrishnan et al. (1999), Hamzadayi e Yildiz (2017). O número de

máquinas varia de 2 a 10, ou seja, $m = 2$, 3, 4, 5, 7 e 10, enquanto que o número de tarefas foi escolhido em função do número de máquinas, ou seja, $n = 3m$, $4m$, $5m$, $7m$ e $10m$, levando a um total de 30 grupos de instâncias. Cinco problemas de teste foram gerados para cada grupo, resultando em um total de 150 instâncias. Os tempos de processamento e *setup* foram gerados uniformemente a partir do intervalo $[10, 100]$ e $[5, 50]$, respectivamente. Os valores dos tempos de *setup* foram ajustados para respeitar a desigualdade triangular: $s_{i,j} \leq s_{i,k} + s_{k,j}, \forall i \neq j \neq k \in J$.

Com tempos de *setup* independentes da sequência, conforme Kim e Lee (2012), geradas aleatoriamente usando a combinação de quatro parâmetros $(n, m, \varepsilon, \rho)$: ε é o fator de diversidade para variação dos tempos de *setup* e processamento e ρ é o fator de severidade do tempo de *setup*, dado por $\rho = m \times \mu(s)/\mu(p)$, onde $\mu(s)$ e $\mu(p)$ indicam a média do tempos de *setup* e processamento. Assim, os tempos de *setup* e processamento serão gerados aleatoriamente como valores inteiros nos intervalos $s_j = [\mu(s) - \varepsilon \times \mu(s); \mu(s) + \varepsilon \times \mu(s)]$ e $p_j = [\mu(p) - \varepsilon \times \mu(p); \mu(p) + \varepsilon \times \mu(p)]$, para toda tarefa $j \in J$.

O número de máquinas e tarefas seguem o esquema descrito no primeiro parágrafo dessa seção, porém para cada combinação de máquinas utilizou-se apenas 3 combinações de tarefas sendo aquelas de maior número e com no máximo 50 tarefas. Para as instâcias com 2 máquinas, adotou-se $n \in \{10, 20, 30, 40, 50\}$. Do mesmo modo que Kim e Lee (2012) tomou-se $\mu(p) = 25$ para cada combinação dos parâmetros $\varepsilon = \{0,1; 0,3\}$ e $\rho = 0,7$. Também foram gerados cinco problemas de teste para cada combinação, resultando num total de 200 instâncias.

4.2 Parâmetros das meta-heurísticas propostas

Para a HILS-CSvr, dada a similaridade na representação da solução, para os parâmetros I_{ILS} e I_R, adotou-se os mesmos valores que Subramanian et al. (2014), com $I_{ILS} = 4 \times n$ e $I_R = 20$. De acordo com os referidos autores, estes valores parecem fornecer uma boa relação entre qualidade da solução e o tempo de CPU. Em relação aos parâmetros do procedimento `GeraSolucaoViavel`, tomou-se $\alpha = 0,5$ e $I_{GSV} = n \times m$, pois, considerando a rapidez a cada chamada de tal procedimento pelo HILS-CSvr, esses valores forneceram melhores resultados.

No ILS_{CS}, para calibrar tais parâmetros, foram selecionadas um conjunto de 15 instâncias desafiadoras. Primeiro, foram conduzidos experimentos de calibração para escolher um valor para I_{ILS}. Foram selecionados diferentes valores para este parâmetro, a saber,

50, 100 e 150, mantendo $I_R = 1$. As Figuras 4.1 e 4.2 apresentam as médias dos *gaps* e tempos de CPU, em segundos, respectivamente, para cada configuração. Os *gaps* médios foram calculados entre a média da solução e a melhor solução encontrada

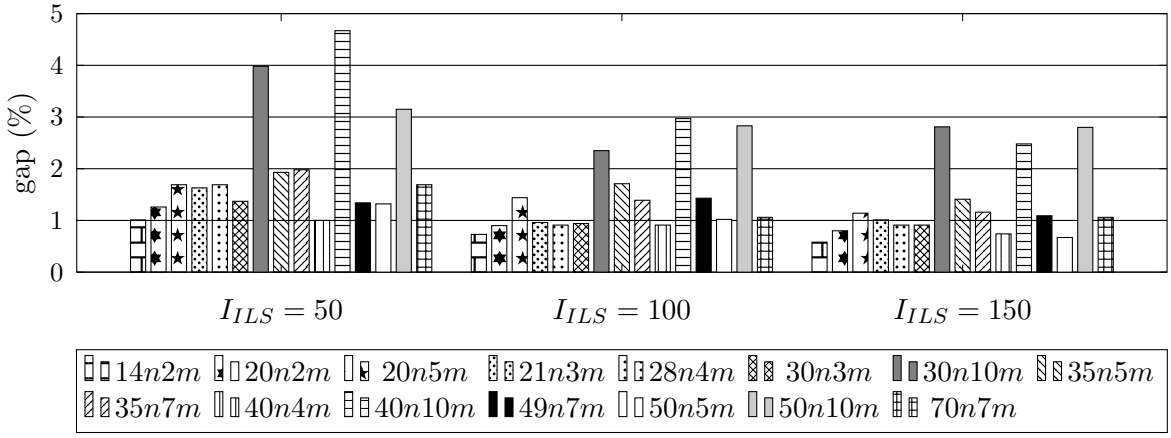

Figura 4.1: Média dos *gaps* obtidos na calibração do parâmetro I_{ILS}

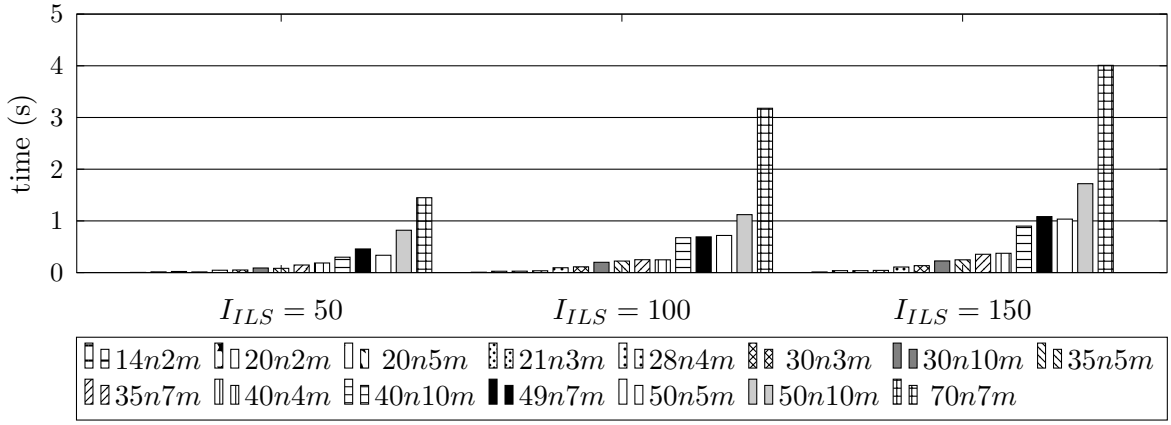

Figura 4.2: Média dos tempos de CPU (s) obtidos na calibração do parâmetro I_{ILS}

A partir dos resultados obtidos, adotou-se $I_{ILS} = 100$, uma vez que oferece um *trade-off* razoável entre a qualidade da solução e o tempo de CPU.

O próximo passo foi realizar experimentos de ajuste para escolher um valor para I_R. Foram utilizados três valores diferentes: 5, 10 e 15, e os resultados em termos de *gaps* médios e tempo de CPU, em segundos, são reportados nas Figuras 4.3 e 4.4, respectivamente. Por acreditar que ofereça um bom compromisso entre a qualidade da solução e o tempo da CPU decidiu-se selecionar $I_R = 10$.

4.3 Resultados

No começo são relatados os resultados agregados médios obtidos pelas abordagens exatas em cada grupo de instâncias e depois são comparados com aqueles encontrados

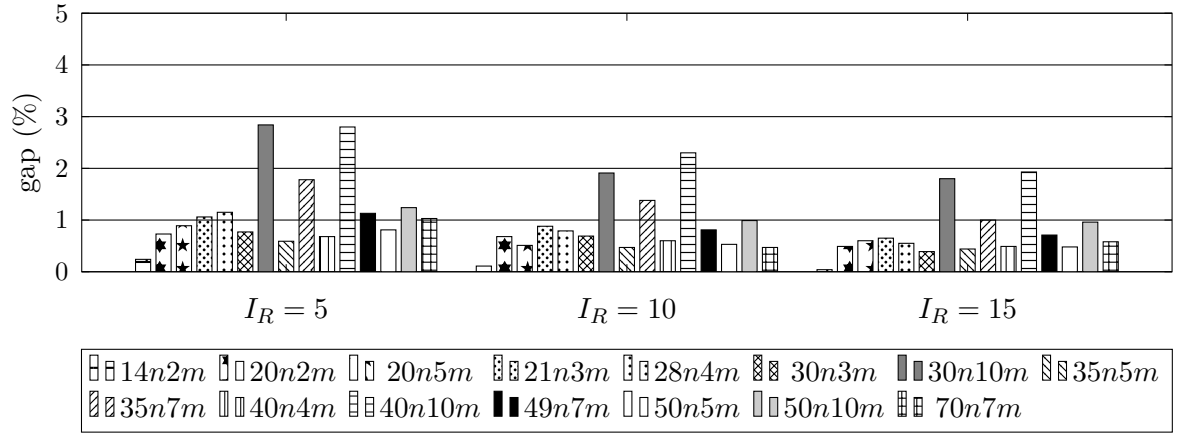

Figura 4.3: Média dos *gaps* obtidos na calibração do parâmetro I_R

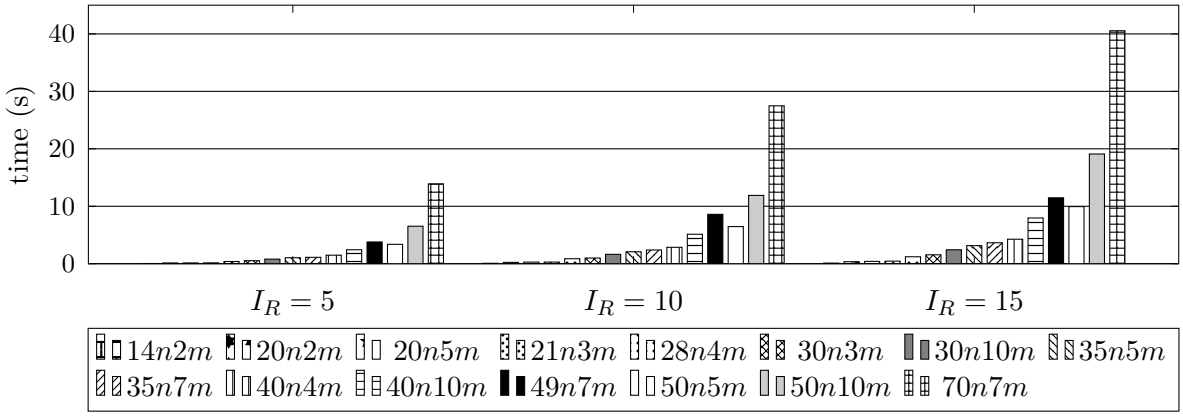

Figura 4.4: Média dos tempos de CPU (s) obtidos na calibração do parâmetro I_R

pelas meta-heurísticas. Resultados detalhados são fornecidos no Apêndice A.

4.3.1 Abordagens exatas para o $P, S_1|s_j|C_{max}$

Esta seção apresenta a comparação entre as médias dos resultados dos métodos exatos, para cada grupo de instâncias do problema $P, S_1|s_j|C_{max}$. São reportados a média (Avg.), mínimo (Min.) e máximo (Max.) *gaps* (%) entre as instancias de cada grupo retornados pelo CPLEX. A coluna g indica o grupo, enquanto a coluna #Opt indica o número de soluções ótimas encontradas. Também são mostrados as médias de tempo, em segundos, usados por cada método. O símbolo " - " indica que o modelo correspondente não conseguiu resolver a relaxação linear de pelo menos uma instância do grupo.

O tempo limite de 1 hora foi imposto para cada método em cada instância. Apesar do ILS_{CS} ter sido originalmente proposto para o problema com tempos de *setup* dependentes da sequência, este também pode ser aplicado ao problema com tempos independentes, bastando substituir o termo s_{ij} por s_j e permitindo a realização de *setup* na primeira

tarefa atribuída a uma máquina no Algoritmo 4. Desse modo, testes com ILS_{CS} foram realizados e o custo encontrado fornecido como um limite superior para o *solver* em cada modelo.

Nas Tabelas 4.1 e 4.2 são mostrados os resultados médios agregados dos *gaps* e tempos de execução, respectivamente, para o problema de escalonamento com duas máquinas. Das 50 instâncias testadas, o TIF conseguiu resolver 44 (88%), seguido de F4 com 35 (70%), F2 com 29 (58%), F1 com 22 e F3 (44%) com 20 (40%) das instâncias resolvidas. Das instâncias que o TIF não foi capaz de resolver, a maioria delas possui 50 tarefas com maiores tempos de *setup* devido a combinação de parâmetros (ε, ρ), nessas o modelo F4 obteve melhor desempenho pois mesmo não resolvendo na otimalidade consegue retornar *gaps* melhores.

Em relação ao tempo de execução, nos grupos de 1 a 5 o modelo F4 possui melhor desempenho, por outro lado nos grupos 8 a 10 o tempo máximo é alcançado sem retornar a solução ótima. O TIF possui o aumento no tempo de execução mais estável independente das características das instâncias dadas pela combinação de parâmetros (ε, ρ).

Os resultados obtidos para o problema $P, S_1|s_j|C_{max}$ são mostrados na Tabela 4.3, para esse problema apenas o TIF e as formulações F1 e F2 são aplicáveis para resolução. O TIF domina em todas as instâncias, tanto em relação aos *gaps* quanto ao tempo de resolução. Das 150 instâncias os modelos F1 e F2 conseguiram resolver apenas 35 (23%) e 28 (19%) instâncias, respectivamente. Os tempos de execução são mostrados Tabela 4.4

Tabela 4.1: Resultados médios agregados dos métodos exatos – $P_2, S_1|s_j|C_{max}$

ε	g	n	Gap (%)																			
			TIF				F1				F2				F3				F4			
			#opt	min	avg	max	#opt	min	avg	max	#opt	min	avg	max	#opt	min	avg	max	#opt	min	avg	max
	1	10	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	2	20	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
0,1	3	30	5	0,00	0,00	0,00	2	0,00	2,11	-	5	0,00	0,00	0,00	0	0,38	0,38	-	5	0,00	0,00	0,00
	4	40	4	0,00	7,01	35,07	0	-	-	-	4	0,00	0,00	-	0	-	-	-	5	0,00	0,00	0,00
	5	50	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-	0	-	-	-	5	0,00	0,00	0,00
	6	10	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	7	20	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
0,3	8	30	5	0,00	0,00	0,00	0	19,12	21,00	22,50	0	1,75	7,12	12,79	0	0,20	0,20	-	0	0,11	0,40	0,61
	9	40	5	0,00	0,00	0,00	0	36,02	37,09	-	0	26,62	30,17	33,03	0	-	-	-	0	0,67	1,17	2,19
	10	50	0	30,89	33,96	-	0	-	-	-	0	36,96	40,65	43,14	0	-	-	-	0	1,06	3,74	5,39

Tabela 4.2: Tempos médios agregados dos métodos exatos – $P_2, S_1|s_j|C_{max}$

ε	g	n	CPU (s)				
			TIF	F1	F2	F3	F4
	1	10	1,51	0,08	0,10	0,40	0,02
	2	20	32,83	22,43	4,47	56,10	0,02
0,1	3	30	449,95	3330,73	56,78	3600	0,04
	4	40	2183,98	3600	2709,38	3600	0,06
	5	50	2303,09	3600	3600	3600	0,15
	6	10	2,18	0,13	0,12	0,46	4,28
	7	20	48,84	62,10	21,94	354,60	79,70
0,3	8	30	590,79	3600	3600	3600	3600
	9	40	1447,03	3600	3600	3600	3600
	10	50	3600	3600	3600	3600	3600

Tabela 4.3: Resultados médios agregados dos métodos exatos – $P, S_1|s_j|C_{max}$

ε	g	m	n	Gap (%)											
				TIF				F1				F2			
				#opt	min	avg	max	#opt	min	avg	max	#opt	min	avg	max
0,1	1	3	15	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	2		21	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	3		30	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	4	4	20	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	5		28	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	6		40	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	7	5	25	5	0,00	0,00	0,00	0	1,32	1,32	-	0	4,58	4,58	-
	8		35	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	9		50	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	10	7	21	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	11		28	5	0,00	0,00	0,00	0	-	-	-	0	17,07	17,07	-
	12		35	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	13	10	30	5	0,00	0,00	0,00	2	0,00	0,82	2,04	0	4,17	5,39	-
	14		40	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	15		50	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
0,3	16	3	15	5	0,00	0,00	0,00	5	0,00	0,00	0,00	5	0,00	0,00	0,00
	17		21	5	0,00	0,00	0,00	4	0,00	0,64	3,21	2	0,00	6,78	14,20
	18		30	5	0,00	0,00	0,00	0	-	-	-	0	35,42	35,42	-
	19	4	20	5	0,00	0,00	0,00	3	0,00	3,78	12,37	1	0,00	7,40	-
	20		28	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	21		40	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	22	5	25	5	0,00	0,00	0,00	0	17,33	17,33	-	0	-	-	-
	23		35	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	24		50	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	25	7	21	5	0,00	0,00	0,00	1	0,00	9,28	15,05	0	11,36	14,54	17,03
	26		28	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	27		35	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	28	10	30	5	0,00	0,00	0,00	0	23,81	23,81	-	0	-	-	-
	29		40	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-
	30		50	5	0,00	0,00	0,00	0	-	-	-	0	-	-	-

4.3.2 Abordagens exatas para o $P, S_1|s_{ij}|C_{max}$

A Tabela 4.5 apresenta a comparação entre as médias dos resultados dos métodos exatos, para cada grupo de instâncias. O tempo limite de 1 hora foi imposto para cada método em cada instância. A melhor solução encontrada pelo ILS_{CS} foi fornecida como um limite inicial primal para o resolvedor. São reportados a média (Avg.), mínimo (Min.) e máximo (Max.) *gaps* (%) entre as instancias de cada grupo retornados pelo CPLEX. A coluna g indica o grupo, enquanto a coluna #Opt indica o número de soluções ótimas encontradas. Também são mostrados as médias de tempo, em segundos, usados por cada

Tabela 4.4: Tempos médios agregados dos métodos exatos – $P, S_1|s_j|C_{max}$

$\varepsilon = 0,1$						$\varepsilon = 0,3$					
g	m	n	CPU (s)			g	m	n	CPU (s)		
			TIF	F1	F2				TIF	F1	F2
1	3	15	3,01	4,61	5,08	16	3	15	6,67	17,58	32,25
2		21	19,68	407,06	186,60	17		21	53,75	2559,82	3073,60
3		30	53,84	3600	3600	18		30	548,41	3600	3600
4	4	20	5,37	188,95	299,56	19	4	20	10,14	2843,96	3191,03
5		28	62,73	3600	3600	20		28	159,86	3600	3600
6		40	244,27	3600	3600	21		40	1160,63	3600	3600
7	5	25	18,82	3600	3600	22	5	25	32,57	3600	3600
8		35	36,42	3600	3600	23		35	165,56	3600	3600
9		50	582,88	3600	3600	24		50	1686,35	3600	3600
10	7	21	6,02	132,82	162,85	25	7	21	3,44	3122,03	3600
11		28	10,08	3600	3600	26		28	18,50	3600	3600
12		35	60,39	3600	3600	27		35	63,27	3600	3600
13	10	30	3,42	3108,26	3600	28	10	30	5,84	3600	3600
14		40	22,88	3600	3600	29		40	35,76	3600	3600
15		50	82,69	3600	3600	30		50	90,44	3600	3600

método. O símbolo " - " indica que o modelo correspondente não conseguiu resolver a relaxação linear de pelo menos uma instância do grupo.

Os resultados obtidos sugerem que a ATIF supera, em média, a formulação F5 em quase todas as instâncias. Os menores *gaps* médios ilustram que o ATIF conseguiu encontrar melhores limites inferiores para os casos em que a solução ótima não pôde ser determinada. Além disso, em 14 grupos (1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17 e 21), o ATIF foi capaz de encontrar a solução ótima de pelo menos uma instância e um total de 65 soluções ótimas. O modelo de Hamzadayi e Yildiz (2017), por outro lado, só conseguiu resolver 15 instâncias com até 9 tarefas. O ATIF também domina a formulação F5 ao considerar o tempo de CPU necessário para resolver as instâncias em que a solução ótima foi encontrada.

Apesar de resolver instâncias maiores quando comparado ao melhor modelo existente, o ATIF não conseguiu provar a otimalidade de instâncias com mais de 21 tarefas, o que ainda é um tamanho limitado. Portanto, o uso de algoritmos heurísticos ainda parece ser mais adequado para fornecer boas soluções viáveis para problemas práticos envolvendo mais tarefas.

Tabela 4.5: Resultados médios agregados do métodos exatos – $P, S_1|s_{ij}|C_{max}$

g	m	n	Gap (%)								CPU (s)	
			ATIF				F5				ATIF	F5
			#Opt	Min	Avg	Max	#Opt	Min	Avg	Max		
1	2	6	5	0,00	0,00	0,00	5	0,00	0,00	0,00	0,1	0,9
2		8	5	0,00	0,00	0,00	5	0,00	0,00	0,00	0,6	157,6
3		10	5	0,00	0,00	0,00	0	46,67	51,20	54,13	3,4	3600
4		14	5	0,00	0,00	0,00	0	75,00	76,22	77,69	198,7	3600
5		20	0	46,68	46,95	47,49	0	82,24	83,99	85,56	3600	3600
6	3	9	5	0,00	0,00	0,00	5	0,00	0,00	0,00	0,3	397,2
7		12	5	0,00	0,00	0,00	0	52,40	56,54	61,33	4,3	3600
8		15	5	0,00	0,00	0,00	0	66,32	69,29	73,24	45,1	3600
9		21	1	0,00	26,28	33,17	0	75,97	78,94	80,86	3488,1	3600
10		30	0	47,59	47,89	48,42	0	82,85	83,60	84,78	3600	3600
11	4	12	5	0,00	0,00	0,00	0	43,66	46,93	53,71	1,1	3600
12		16	5	0,00	0,00	0,00	0	58,26	62,63	65,79	76,5	3600
13		20	4	0,00	5,31	26,54	0	70,30	71,55	73,39	1989,9	3600
14		28	0	47,06	47,59	47,88	0	76,26	76,81	77,29	3600	3600
15		40	0	–	–	–	0	82,36	84,36	85,33	3600	3600
16	5	15	5	0,00	0,00	0,00	0	42,60	46,71	49,73	7,0	3600
17		20	5	0,00	0,00	0,00	0	54,88	60,78	65,64	626,1	3600
18		25	0	27,20	30,89	32,28	0	58,95	68,41	72,27	3600	3600
19		35	0	47,82	48,12	48,43	0	75,49	76,96	79,60	3600	3600
20		50	0	–	–	–	0	83,50	84,64	85,21	3600	3600
21	7	21	5	0,00	0,00	0,00	0	39,24	45,82	54,85	420,0	3600
22		28	0	22,82	26,81	29,82	0	55,81	61,32	68,62	3600	3600
23		35	0	33,40	41,50	47,91	0	64,00	68,82	71,79	3600	3600
24		49	0	48,64	48,64	–	0	76,91	78,06	79,63	3600	3600
25		70	0	–	–	–	0	–	–	–	3600	3600
26	10	30	0	14,84	17,58	20,56	0	43,86	46,58	49,20	3600	3600
27		40	0	31,72	32,13	32,78	0	58,26	61,14	63,50	3600	3600
28		50	0	48,40	48,85	49,38	0	–	–	–	3600	3600
29		70	0	–	–	–	0	–	–	–	3600	3600
30		100	0	–	–	–	0	–	–	–	3600	3600

4.3.3 Abordagens meta-heurísticas propostas

Nessa seção são apresentados os resultados entre o métodos ILS_{CS} e HILS-CSvr. Na subseção seguinte são mostrados resultados entre as meta-heurísticas da literatura. Também é realizada a análise e comparação desses com os resultados do melhor dentre os dois métodos aqui propostos. O resultados agregados médios das abordagens ILS_{CS} e HILS-CSvr são mostrados na Tabela 4.6.

Tabela 4.6: Resultados médios agregados encontrados por ILS_{CS} e HILS-CSvr

g	m	n	Gap (%)						CPU (s)	
			ILS_{CS}			HILS-CSvr			ILS_{CS}	HILS-CSvr
			Min	Avg	Max	Min	Avg	Max		
1	2	6	0,00	0,00	0,00	0,00	0,00	0,00	0,006	0,114
2		8	0,00	0,00	0,00	0,64	0,71	0,74	0,013	0,134
3		10	0,00	0,06	0,45	0,27	0,31	0,70	0,025	0,232
4		14	0,00	0,19	0,44	0,16	0,38	0,77	0,068	0,922
5		20	0,11	0,71	1,20	0,09	0,40	0,90	0,220	3,722
6	3	9	0,00	0,08	0,13	0,28	0,36	0,37	0,018	0,613
7		12	0,00	0,05	0,23	0,60	1,24	1,91	0,041	0,682
8		15	0,00	0,56	1,11	0,46	1,28	1,93	0,085	1,245
9		21	0,28	0,84	1,38	0,24	0,88	1,36	0,253	3,523
10		30	0,67	1,06	1,44	0,00	0,49	1,00	0,900	10,594
11	4	12	0,00	0,03	0,09	0,19	0,74	1,07	0,042	2,036
12		16	0,00	0,55	1,12	0,85	2,29	3,45	0,113	1,808
13		20	0,06	0,71	1,19	0,30	1,38	2,43	0,233	2,893
14		28	0,09	0,58	1,19	0,24	0,91	1,52	0,705	7,809
15		40	0,38	0,91	1,29	0,00	0,55	0,96	2,624	25,268
16	5	15	0,00	0,15	0,56	0,11	1,35	2,10	0,088	5,087
17		20	0,00	0,60	1,07	0,87	2,14	3,40	0,247	4,227
18		25	0,00	0,54	1,00	1,43	2,56	4,02	0,534	6,231
19		35	0,00	0,65	1,22	0,54	1,32	2,30	1,815	19,997
20		50	1,26	1,80	2,24	0,00	0,67	1,29	6,135	5,301
21	7	21	0,00	0,51	1,06	1,56	3,56	5,37	0,317	23,793
22		28	0,00	0,73	1,49	2,93	4,41	5,94	0,922	14,703
23		35	0,00	0,82	1,60	2,06	3,70	5,28	2,128	26,354
24		49	0,27	0,97	1,43	0,13	1,86	3,26	6,792	6,159
25		70	0,00	0,46	1,00	0,68	1,95	3,19	24,790	17,667
26	10	30	0,00	1,13	1,85	5,82	11,62	19,06	1,466	3,429
27		40	0,00	1,29	2,46	6,15	9,62	13,18	4,535	8,214
28		50	0,00	0,94	1,80	4,82	8,79	12,26	10,470	14,139
29		70	0,00	0,85	1,55	7,38	9,61	12,52	39,072	33,201
30		100	0,00	0,88	1,71	6,46	8,71	10,48	147,168	93,378

De acordo com os resultados, o ILS_{CS} obteve um desempenho superior em termos de *gaps* mínimos, médios e máximos para cada grupo, quando comparado ao HILS-CSvr. Das 150 instâncias, ILS_{CS} encontrou os melhores resultados em 126 (84%), enquanto o HILS-CSvr encontrou 62 ($41,3\%$). Este último obteve melhores resultados em termos de *gaps* mínimos em 24 instâncias distribuídas em 8 grupos, porém em 4 e 3 desses grupos o ILS_{CS} obteve melhores resultados em termos de *gaps* médios e máximos, respectivamente. Também pode-se notar que a diferença no desempenho em relação à qualidade da solução tende a aumentar com o tamanho da instância. Isso mostra que o valor dos *gaps* é mais consistente para o ILS_{CS}, do que para o HILS-CSvr.

Quanto a escalabilidade dos algoritmos propostos, em média, ambos os algoritmos apresentaram uma escalabilidade razoável, especialmente para instâncias com até 50 tarefas. O ILS_{CS} obteve os maiores tempos em apenas 4 grupos, caracterizados principalmente por possuírem mais de 50 tarefas e/ou 10 máquinas. O que pode ser justificado pelo fato que nessas instâncias o HILS-CSvr usa o procedimento `GeraSolucaoViavel`, que é chamado apenas caso haja melhora na solução relaxada.

Outro aspecto que deve-se ter em mente é que, para o ILS_{CS}, a avaliação de movimento é computacionalmente cara ($\mathcal{O}(nm)$) e o tempo de CPU provavelmente aumentará drasticamente com o tamanho da instância. No entanto, o tempo médio de CPU ficou abaixo de 150 segundos para as instâncias de 100 tarefas, o que pode ser considerado aceitável, dada a natureza altamente desafiadora do problema.

4.3.4 Abordagens meta-heurísticas da literatura

Para o SA e AG reimplementados, considerou-se os parâmetros propostos por Hamzadayi e Yildiz (2017). Porém, tendo em vista que esses algoritmos possuem um tempo de CPU muito pequeno, decidiu-se manter o tempo de execução igual a média dos tempos de execução do ILS_{CS} para cada grupo de instâncias. Permitindo assim, um número maior de gerações para o AG, enquanto que no SA, o parâmetro de temperatura sofre um *restart* quando atinge a temperatura final, mantendo seu decréscimo até o tempo estabelecido ser atingido.

A Tabela 4.7 apresenta os resultados agregados médios encontrados pelas abordagens meta-heurísticas. Os *gaps* médios relatados são em relação à melhor solução encontrada dentre todos os métodos. Também são relatados os tempos de CPU, em segundos.

Os resultados mostram que o ILS_{CS} obteve claramente um desempenho superior em

Tabela 4.7: Resultados médios agregados dos métodos meta-heurísticos – $P, S_1|s_{ij}|C_{max}$

g	m	n	Gap (%)									CPU (s)
			ILS_{CS}			SA			GA			
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	
1	2	6	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,006
2		8	0,00	0,00	0,00	0,00	0,10	0,48	0,10	1,10	2,35	0,013
3		10	0,00	0,06	0,45	0,43	1,62	2,83	1,07	3,18	4,55	0,025
4		14	0,00	0,19	0,44	0,36	1,49	2,37	2,24	3,90	5,33	0,068
5		20	0,00	0,60	1,09	1,02	1,81	2,53	3,81	5,20	6,49	0,220
6	3	9	0,00	0,08	0,13	0,00	0,56	1,24	0,69	2,03	4,00	0,018
7		12	0,00	0,05	0,23	0,90	2,09	3,36	2,90	5,02	6,69	0,041
8		15	0,00	0,56	1,11	0,94	2,09	3,16	3,25	5,12	6,60	0,085
9		21	0,00	0,56	1,10	0,78	1,60	2,42	3,65	5,23	6,83	0,253
10		30	0,00	0,38	0,77	1,10	1,79	2,19	3,81	4,93	5,97	0,900
11	4	12	0,00	0,03	0,09	0,28	1,47	3,36	3,22	4,91	6,53	0,042
12		16	0,00	0,55	1,12	1,20	2,01	3,02	4,31	6,52	8,08	0,113
13		20	0,00	0,65	1,13	1,34	2,24	3,16	4,16	6,30	7,89	0,233
14		28	0,00	0,49	1,10	1,58	2,32	3,14	4,25	5,96	7,76	0,705
15		40	0,00	0,52	0,90	1,80	2,49	3,18	4,35	5,38	6,65	2,624
16	5	15	0,00	0,15	0,56	0,90	1,99	3,02	2,96	5,72	7,78	0,088
17		20	0,00	0,60	1,07	1,53	2,48	3,60	4,30	6,23	8,29	0,247
18		25	0,00	0,54	1,00	1,08	2,35	3,37	5,23	7,19	9,10	0,534
19		35	0,00	0,65	1,22	2,30	3,05	3,77	5,13	6,82	8,27	1,815
20		50	0,00	0,53	0,96	2,71	3,20	3,67	4,36	5,54	6,88	6,135
21	7	21	0,00	0,51	1,06	1,80	3,14	4,86	6,39	9,87	13,69	0,317
22		28	0,00	0,73	1,49	2,02	3,10	4,20	6,15	8,40	11,12	0,922
23		35	0,00	0,82	1,60	2,91	3,74	4,61	5,81	8,86	11,39	2,128
24		49	0,00	0,70	1,16	3,45	4,28	4,87	5,91	8,03	9,73	6,792
25		70	0,00	0,46	1,00	4,60	5,09	5,54	5,67	7,68	9,36	24,790
26	10	30	0,00	1,13	1,85	3,27	5,10	6,66	8,76	13,53	18,59	1,466
27		40	0,00	1,29	2,46	4,75	6,23	7,40	11,01	14,36	19,36	4,535
28		50	0,00	0,94	1,80	5,25	6,45	7,43	10,14	13,85	17,22	10,470
29		70	0,00	0,85	1,55	7,33	8,53	9,43	10,05	13,14	15,65	39,072
30		100	0,00	0,88	1,71	9,07	9,93	10,63	9,81	12,01	14,53	147,168

termos de *gaps* mínimos, médios e máximos para cada grupo, quando comparado ao SA e ao GA. Observe que o GA e o SA não conseguiram encontrar as soluções ótimas conhecidas em instâncias muito pequenas envolvendo menos de 10 tarefas. No total, das 65 soluções ótimas conhecidas, o ILS_{CS} encontrou 51 (78%), enquanto o SA e o GA encontraram apenas 28 (43%) e 12 (18%), respectivamente. Além disso, a diferença no desempenho em relação à qualidade da solução tende a aumentar com o tamanho da instância. Outra constatação é que o ILS_{CS} tende a ser mais robusto, pois o valor dos *gaps* é mais consistente do que aqueles encontrados por SA e GA, independentemente do tamanho.

4.4 Impacto do MDA sobre o ODA

A Figura 4.5 apresenta a performance de tempo do ILS_{CS} para as variações de implementação do procedimento *decoding algorithm*. O gráfico mostra a variação percentual média de tempo para cada grupo de instância. O cálculo da variação foi realizado em relação ao menor tempo médio dentre todos os métodos para cada grupo. O método ODA-WB corresponde ao método ODA com a estratégia de interrupção de avaliação pelo melhor custo conhecido, enquanto que o método MDA-NB, corresponde ao MDA sem essa estratégia.

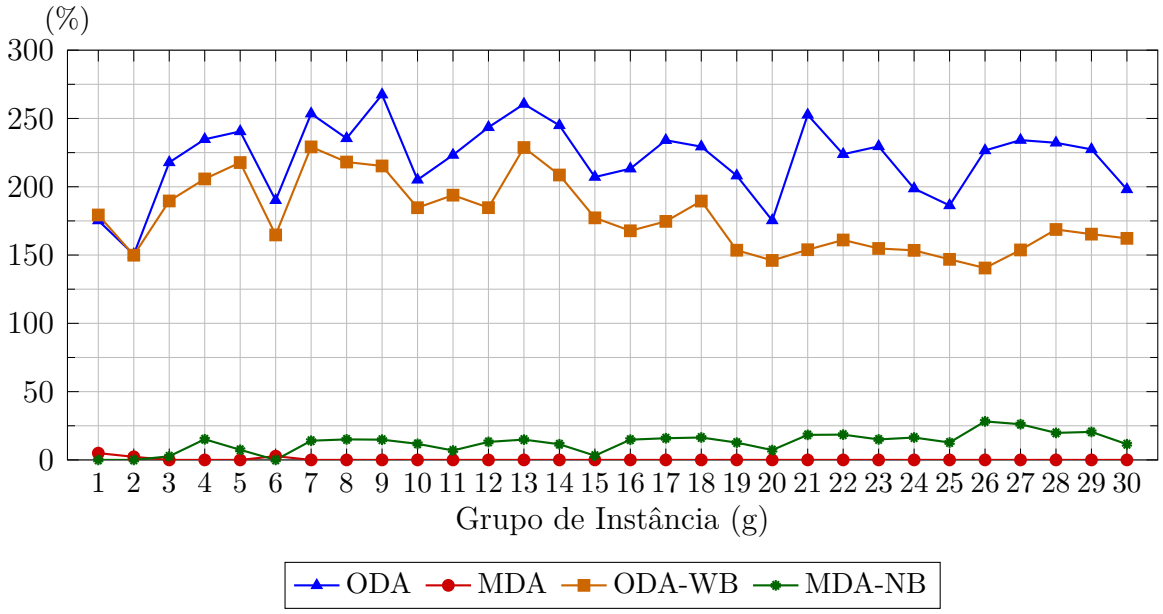

Figura 4.5: Variação média do tempo entre ODA e MDA para cada grupo de instâncias

Pela análise dos resultados pode-se verificar que: (i) a estratégia de interrupção de avaliação provoca uma redução de tempo de 0 a 100% entre os métodos ODA e ODA-WB; e para os métodos MDA e MDA-NB entre 0 e 25% de redução; (ii) para instâncias muito pequenas e com uma relação $n/m \approx m$ é perceptível que a estratégia de interrupção

de avaliação na tenha o efeito esperado, podendo até mesmo possuir tempo de execução levemente maior; (iii) O MDA possui uma redução de tempo com variação de 175 a 275% a depender do grupo de instância em relação ao ODA. Esses resultados sugerem que as modificações realizadas no ODA ajudaram o ILS_{CS} a obter acelerações significativas, aumentando assim drasticamente seu desempenho em tempo de execução.

Capítulo 5

Considerações Finais

Neste trabalho, foram propostas abordagens exatas e meta-heurísticas para o problema de escalonamento de máquinas paralelas idênticas com servidor único e tempos de *setup* dependentes e independentes da sequência. Para ambos o objetivo é minimizar o *makespan*. Os métodos exatos consistem em uma formulação indexada no tempo (TIF) e outra de arcos indexados no tempo (ATIF), e são utilizados para a resolução do problema com os tempos de *setup* independentes e dependentes da sequência, respectivamente. Para o segundo caso foram apresentados, também, dois algoritmos meta-heurísticos *multi-start* baseado em busca local iterada e denotados por HILS-CSvr e ILS_{CS}. O primeiro é um algoritmo híbrido que utiliza os princípios do ILS para resolver uma versão relaxada do problema e então emprega uma versão restrita do modelo ATIF ou um procedimento heurístico baseado na fase construtiva da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para gerar soluções viáveis. O segundo itera sobre uma sequência de prioridade de tarefas executando movimentos de troca e usando um algoritmo de decodificação modificado (MDA), que é uma versão melhorada do original desenvolvido por Hamzadayi e Yildiz (2017), para avaliar a solução.

Experimentos computacionais foram realizados em 200 instâncias divididas em 25 grupos para o problema com tempos de *setup* independentes da sequência, dos quais 10 grupos para o problema com apenas duas máquinas e 15 com um número arbitrário de máquinas. Os resultados obtidos pelo TIF foram comparados com os modelos de Kim e Lee (2012), Gan et al. (2012) e Hasani et al. (2014a). O TIF foi capaz de resolver 97% das instâncias na otimalidade, demostrando-se superior aos demais modelos comparados.

Para o problema com tempos de *setup* dependentes da sequência, os experimentos computacionais foram realizados em 150 instâncias divididas em 30 grupos, cada um com um cenário distinto em relação ao número de tarefas e máquinas. Os resultados

obtidos pelo ATIF foram comparados com a formulação de programação linear inteira mista (MILP) de Hamzadayi e Yildiz (2017). Os resultados do HILS-CSvr e ILS_{CS} foram comparados entre si e o melhor de ambos comparados com o *simulated annealing* (SA) e o algoritmo genético (GA), também desenvolvidos por Hamzadayi e Yildiz (2017).

O ATIF superou claramente o modelo de Hamzadayi e Yildiz (2017) e foi capaz de resolver instâncias com até 21 tarefas para otimalidade, enquanto o segundo só pôde provar a otimalidade de instâncias envolvendo até 9 tarefas. Além disso, o ATIF também forneceu, em média, melhores limites inferiores para as instâncias não resolvidas. O ILS_{CS} obteve um desempenho superior em termos de *gaps* mínimos, médios e máximos para cada grupo, quando comparado ao HILS-CSvr. Das 150 instâncias, ILS_{CS} encontrou os melhores resultados em 126 (84%), enquanto o HILS-CSvr encontrou 62 ($41,3\%$). Da mesma forma, o ILS_{CS} também superou as outras duas abordagens, encontrando 78% das soluções ótimas conhecidas, contra 43% e 18% do SA e GA, respectivamente. Além disso, o ILS_{CS} sistematicamente produziu, em média, os melhores limites superiores para aquelas instâncias onde as soluções ótimas não são conhecidas.

Também foram conduzidos experimentos para ilustrar os benefícios do uso do MDA em vez do algoritmo de decodificação original (ODA). Os resultados mostram os significativos aumentos de velocidade alcançados pelo ILS_{CS} ao implementar o primeiro algoritmo, que em alguns casos foi 275% mais rápido do que ao usar o último procedimento. Portanto, o MDA desempenhou um papel crítico no desempenho em tempo de execução da meta-heurística.

Referências

ABDEKHODAEE, A. H.; WIRTH, A. Scheduling parallel machines with a single server: Some solvable cases and heuristics. *Computers and Operations Research*, v. 29, n. 3, p. 295–315, 2002.

ABDEKHODAEE, A. H.; WIRTH, A.; GAN, H. S. Equal processing and equal setup time cases of scheduling parallel machines with a single server. *Computers & Operations Research*, v. 31, n. 11, p. 1867–1889, sep 2004.

ABDEKHODAEE, A. H.; WIRTH, A.; GAN, H.-S. Scheduling two parallel machines with a single server: the general case. *Computers & Operations Research*, v. 33, n. 4, p. 994–1009, apr 2006.

ALLAHVERDI, A. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, v. 246, n. 2, p. 345 – 378, 2015. ISSN 0377-2217.

BALAKRISHNAN, N.; KANET, J. J.; SRIDHARAN, V. Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research*, v. 26, n. 2, p. 127 – 141, 1999. ISSN 0305-0548.

BRUCKER, P. et al. Complexity results for parallel machine problems with a single server. *Journal of Scheduling*, v. 5, n. 6, p. 429–457, nov 2002.

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. 2-3, p. 255–270, 1990.

FEO, T. A.; RESENDE, M. G. C. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, v. 6, n. 2, p. 109–133, mar 1995. ISSN 1573-2916.

FUCHIGAMI, H. Y.; RANGEL, S. A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*, Elsevier B.V., v. 25, p. 425–436, 2018.

GAN, H.-S.; WIRTH, A.; ABDEKHODAEE, A. A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. *Computers & Operations Research*, v. 39, n. 9, p. 2242–2247, sep 2012.

GLASS, C. A.; SHAFRANSK, Y. M.; STRUSEVICH, V. A. Scheduling for Parallel Dedicated Machines with a Single Server. In: *Naval Research Logistics, 47,*: IEEE, 2000. v. 47, p. 304–328.

GRAHAM, R. L. et al. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In: *Discrete Optimization II*: Elsevier, 1979, (Annals of Discrete Mathematics, v. 5). p. 287–326.

GUIRCHOUN, S.; SOUHKAL, A.; MARTINEAU, P. Complexity results for parallel machine scheduling problems with a server in computer systems. *In Proceedings of the 2nd multidisciplinary international conference on scheduling: Theory and applications*, p. 232–236, 2005.

HALL, N. G.; POTTS, C. N.; SRISKANDARAJAH, C. Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, v. 102, n. 3, p. 223–243, jun 2000.

HAMZADAYI, A.; YILDIZ, G. Hybrid strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Simulation Modelling Practice and Theory*, v. 63, p. 104–132, apr 2016.

HAMZADAYI, A.; YILDIZ, G. Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. *Computers & Industrial Engineering*, v. 106, p. 287–298, apr 2017.

HASANI, K.; KRAVCHENKO, S. A.; WERNER, F. Block models for scheduling jobs on two parallel machines with a single server. *Computers & Operations Research*, v. 41, n. 1, p. 94–97, 2014.

HASANI, K.; KRAVCHENKO, S. A.; WERNER, F. Minimizing total weighted completion time approximately for the parallel machine problem with a single server. *Information Processing Letters*, v. 114, n. 9, p. 500–503, 2014.

HASANI, K.; KRAVCHENKO, S. A.; WERNER, F. Simulated annealing and genetic algorithms for the two-machine scheduling problem with a single server. *International Journal of Production Research*, v. 52, n. 13, p. 3778–3792, 2014.

HASANI, K.; KRAVCHENKO, S. A.; WERNER, F. Minimizing the makespan for the two-machine scheduling problem with a single server: Two algorithms for very large instances. *Engineering Optimization*, v. 48, n. 1, p. 173–183, jan 2015.

HUANG, S.; CAI, L.; ZHANG, X. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering*, v. 58, n. 1, p. 165–174, feb 2010.

JIANG, Y.; DONG, J.; JI, M. Preemptive scheduling on two parallel machines with a single server. *Computers & Industrial Engineering*, v. 66, n. 2, p. 514–518, 2013.

KIM, M.-Y.; LEE, Y. H. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Computers & Operations Research*, v. 39, n. 11, p. 2457–2468, nov 2012.

KOULAMAS, C. P. Scheduling two parallel semiautomatic machines to minimize machine interference. *Computers & Operations Research*, v. 23, n. 10, p. 945–956, oct 1996.

KRAVCHENKO, S. A.; WERNER, F. Parallel machine scheduling problems with a single server. *Mathematical and Computer Modelling*, v. 26, n. 12, 1997.

KRAVCHENKO, S. A.; WERNER, F. *Scheduling on Parallel Machines with Single and Multiple Servers*. Brazil: 15th Int. Conf. on CAD/CAM, Robotics and Factories of Future, 1999. 3747–3769 p.

KRAVCHENKO, S. A.; WERNER, F. A heuristic algorithm for minimizing mean flow time with unit setups. *Information Processing Letters*, v. 79, n. 6, p. 291–296, sep 2001.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated Local Search. In: _____. *Handbook of Metaheuristics*. Boston, MA: Springer US, 2003. p. 320–353. Disponível em: <https://doi.org/10.1007/0-306-48056-5{_}11>.

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097–1100, 1997.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research*, v. 63, n. 5, p. 511–623, 1996. Disponível em: <http://link.springer.com/10.1007/BF02125421>.

OU, J.; QI, X.; LEE, C.-Y. Parallel machine scheduling with multiple unloading servers. *Journal of Scheduling*, v. 13, n. 3, p. 213–226, jun 2010.

PESSOA, A. et al. 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.

PINEDO, M. L. *Scheduling: Theory, algorithms and systems*. 3. ed.: Springer, 2008.

PINEDO, M. L. *Planning and scheduling in manufacturing and services*. 2. ed.: Springer-Verlag New York, 2009.

POTTS, C. N.; STRUSEVICH, V. A. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, v. 60, n. S1, 2009.

SILVA, Y. L. T. V.; SUBRAMANIAN, A.; PESSOA, A. A. Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. *Computers & Operations Research*, v. 90, p. 142–160, 2018.

SOUZA, M. J. F. *Inteligência computacional para otimização*. Departamento de Computação, Universidade Federal de Ouro Preto, 2011. Notas de aula.

SU, C. Online LPT algorithms for parallel machines scheduling with a single server. *Journal of Combinatorial Optimization*, v. 26, n. 3, p. 480–488, oct 2013.

SUBRAMANIAN, A.; BATTARRA, M.; POTTS, C. N. An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, v. 52, n. 9, p. 2729–2742, 2014.

WANG, G.; CHENG, T. C. E. An approximation algorithm for parallel machine scheduling with a common server. *Journal of the Operational Research Society*, v. 52, n. 2, p. 234–237, feb 2001.

WERNER, F.; KRAVCHENKO, S. A. Scheduling with multiple servers. *Automation and Remote Control*, v. 71, n. 10, p. 2109–2121, oct 2010.

ZHANG, L.; WIRTH, A. On-line scheduling of two parallel machines with a single server. *Computers & Operations Research*, v. 36, n. 5, p. 1529–1553, 2009.

APÊNDICE A – Resultados detalhados

A.1 Resultados detalhados para o $P, S_1|s_j|C_{max}$

Nas tabelas a seguir são mostrados os limites superiores (UB) e inferiores (LB) e o tempo de execução para cada método exato. Se LB = UB indica que a solução ótima foi encontrada. Formulação F1 e F2 (KIM; LEE, 2012), F3 (GAN et al., 2012) e F4 (HASANI et al., 2014a).

Tabela A.1: Resultados - $P_2, S_1|s_j|C_{max}$

ε	Instância	g	TIF		F1		F2		F3		F4		CPU (s)				
			UB	LB	UB	LB	UB	LB	UB	LB	UB	LB	TIF	F1	F2	F3	F4
0,1	10n2m1a2p1	1	182	182	182	182	182	182	182	182	182	182	0,83	0,09	0,08	0,40	0,01
	10n2m1a2p2		185	185	185	185	185	185	185	185	185	185	1,40	0,09	0,16	0,19	0,01
	10n2m1a2p3		175	175	175	175	175	175	175	175	175	175	2,24	0,08	0,06	0,98	0,06
	10n2m1a2p4		188	188	188	188	188	188	188	188	188	188	1,16	0,07	0,08	0,16	0,01
	10n2m1a2p5		181	181	181	181	181	181	181	181	181	181	1,94	0,09	0,10	0,28	0,01
	20n2m1a2p1	2	350	350	350	350	350	350	350	350	350	350	33,79	21,50	4,03	60,66	0,03
	20n2m1a2p2		354	354	354	354	354	354	354	354	354	354	38,10	24,46	4,86	40,76	0,01
	20n2m1a2p3		350	350	350	350	350	350	350	350	350	350	31,09	19,31	6,20	51,43	0,01
	20n2m1a2p4		359	359	359	359	359	359	359	359	359	359	29,42	23,21	2,14	71,39	0,03
	20n2m1a2p5		350	350	350	350	350	350	350	350	350	350	31,75	23,65	5,12	56,29	0,02
	30n2m1a2p1	3	523	523	523	523	523	523	-	-	523	523	242,68	2771,65	55,58	3600	0,05
	30n2m1a2p2		527	527	527	508,5	527	527	-	-	527	527	239,61	3600	48,74	3600	0,07
	30n2m1a2p3		522	522	-	-	522	522	-	-	522	522	793,29	3600	65,46	3600	0,04
	30n2m1a2p4		520	520	520	520	520	520	520	518	520	520	212,70	3081,99	65,10	3600	0,04
	30n2m1a2p5		527	527	527	501	527	527	527	525	527	527	761,45	3600	49,05	3600	0,03
	40n2m1a2p1	4	698	453,2	-	-	698	698	-	-	698	698	3600	3600	1615,71	3600	0,10
	40n2m1a2p2		699	699	-	-	699	699	-	-	699	699	3097,44	3600	1907,46	3600	0,05
	40n2m1a2p3		696	696	-	-	696	696	-	-	696	696	699,60	3600	2901,22	3600	0,06
	40n2m1a2p4		690	690	-	-	-	-	-	-	690	690	1029,72	3600	3600	3600	0,04
	40n2m1a2p5		697	697	-	-	697	697	-	-	697	697	2493,11	3600	3522,51	3600	0,06
	50n2m1a2p1	5	860	860	-	-	-	-	-	-	860	860	1574,06	3600	3600	3600	0,18
	50n2m1a2p2		865	865	-	-	-	-	-	-	865	865	1785,59	3600	3600	3600	0,18
	50n2m1a2p3		870	870	-	-	-	-	-	-	870	870	3096,75	3600	3600	3600	0,08
	50n2m1a2p4		872	872	-	-	-	-	-	-	872	872	2769,99	3600	3600	3600	0,14
	50n2m1a2p5		865	865	-	-	-	-	-	-	865	865	2289,07	3600	3600	3600	0,17
0,3	10n2m2a2p1	6	183	183	183	183	183	183	183	183	183	183	2,56	0,14	0,15	0,18	0,69
	10n2m2a2p2		170	170	170	170	170	170	170	170	170	170	2,22	0,13	0,16	0,50	1,08
	10n2m2a2p3		159	159	159	159	159	159	159	159	159	159	1,87	0,09	0,06	0,35	0,19
	10n2m2a2p4		180	180	180	180	180	180	180	180	180	180	1,84	0,12	0,09	0,57	0,38
	10n2m2a2p5		178	178	178	178	178	178	178	178	178	178	2,40	0,18	0,16	0,71	19,03

Continua na próxima página

Tabela A.1: (*Continuação*)

ε	Instância	g	TIF		F1		F2		F3		F4		CPU (s)				
			UB	LB	UB	LB	UB	LB	UB	LB	UB	LB	TIF	F1	F2	F3	F4
	20n2m2a2p1	7	345	345	345	345	345	345	345	345	345	345	42,08	70,43	34,79	867,72	317,13
	20n2m2a2p2		356	356	356	356	356	356	356	356	356	356	42,42	50,32	19,60	574,19	62,04
	20n2m2a2p3		361	361	361	361	361	361	361	361	361	361	69,24	88,85	21,82	115,42	10,84
	20n2m2a2p4		364	364	364	364	364	364	364	364	364	364	49,48	62,27	12,47	30,65	7,38
	20n2m2a2p5		325	325	325	325	325	325	325	325	325	325	40,96	38,61	21,01	185,03	1,09
	30n2m2a2p1	8	513	513	513	406	513	495	-	-	513	509,98	218,97	3600	3600	3600	3600
	30n2m2a2p2		500	500	500	393	500	465,71	-	-	500	499,36	468,75	3600	3600	3600	3600
	30n2m2a2p3		476	476	476	385	476	425,2	-	-	476	475,5	411,40	3600	3600	3600	3600
	30n2m2a2p4		492	492	492	388	492	429,07	492	491	492	489	144,61	3600	3600	3600	3600
	30n2m2a2p5		514	514	514	398,37	514	505	-	-	514	511	1710,23	3600	3600	3600	3600
	40n2m2a2p1	9	695	695	696	429	695	469,96	-	-	707	691,5	539,37	3600	3600	3600	3600
	40n2m2a2p2		692	692	-	-	692	507,8	-	-	696	688,58	1242,66	3600	3600	3600	3600
	40n2m2a2p3		659	659	659	415,8	659	441,32	-	-	664	656,48	2457,95	3600	3600	3600	3600
	40n2m2a2p4		671	671	-	-	672	466,88	-	-	673	668,5	1826,50	3600	3600	3600	3600
	40n2m2a2p5		694	694	694	444	694	497,47	-	-	696	690,5	1168,68	3600	3600	3600	3600
	50n2m2a2p1	10	851	548,88	-	-	850	516,63	-	-	888	847	3600	3600	3600	3600	3600
	50n2m2a2p2		846	584,71	-	-	845	482,35	-	-	851	842	3600	3600	3600	3600	3600
	50n2m2a2p3		-	-	-	-	873	496,38	-	-	918	868,5	3600	3600	3600	3600	3600
	50n2m2a2p4		-	-	-	-	833	491,21	-	-	874	830	3600	3600	3600	3600	3600
	50n2m2a2p5		866	558,72	-	-	865	545,34	-	-	885	862	3600	3600	3600	3600	3600

Tabela A.2: Resultados - $P, S_1|s_j|C_{max}$ e $\varepsilon = 0,1$

Instância	g	TIF		F1		F2		CPU (s)		
		UB	LB	UB	LB	UB	LB	TIF	F1	F2
15n3m1a2p1	1	163	163	163	163	163	163	3,01	3,74	4,87
15n3m1a2p2		164	164	164	164	164	164	2,9	6,32	6,37
15n3m1a2p3		159	159	159	159	159	159	2,89	3,99	5,09
15n3m1a2p4		164	164	164	164	164	164	3,88	4,21	5,63
15n3m1a2p5		159	159	159	159	159	159	2,37	4,77	3,43
21n3m1a2p1	2	226	226	226	226	226	226	16,17	174,82	200,28
21n3m1a2p2		226	226	226	226	226	226	42,56	295,53	116,25
21n3m1a2p3		223	223	223	223	223	223	13,47	552,94	169,05
21n3m1a2p4		227	227	227	227	227	227	12,87	323,69	279,39
21n3m1a2p5		228	228	228	228	228	228	13,32	688,29	168,04
30n3m1a2p1	3	317	317	-	-	-	-	60,9	3600	3600
30n3m1a2p2		324	324	-	-	-	-	55,93	3600	3600
30n3m1a2p3		311	311	-	-	-	-	48,96	3600	3600
30n3m1a2p4		322	322	-	-	-	-	60,07	3600	3600
30n3m1a2p5		321	321	-	-	-	-	43,31	3600	3600
20n4m1a2p1	4	161	161	161	161	161	161	5,17	252,45	204,8
20n4m1a2p2		155	155	155	155	155	155	4,32	62,88	169,83
20n4m1a2p3		155	155	155	155	155	155	5,07	364,32	65,46
20n4m1a2p4		158	158	158	158	158	158	6,07	189,02	879,39
20n4m1a2p5		159	159	159	159	159	159	6,24	76,06	178,31
28n4m1a2p1	5	222	222	-	-	-	-	80,84	3600	3600
28n4m1a2p2		218	218	-	-	-	-	24,91	3600	3600
28n4m1a2p3		214	214	-	-	-	-	22,93	3600	3600
28n4m1a2p4		219	219	-	-	-	-	158,11	3600	3600
28n4m1a2p5		215	215	-	-	-	-	26,85	3600	3600
40n4m1a2p1	6	306	306	-	-	-	-	399,87	3600	3600
40n4m1a2p2		309	309	-	-	-	-	438,86	3600	3600

Continua na próxima página

Tabela A.2: (*Continuação*)

Instância	g	TIF		F1		F2		CPU (s)		
		UB	LB	UB	LB	UB	LB	TIF	F1	F2
40n4m1a2p3		311	311	-	-	-	-	174,71	3600	3600
40n4m1a2p4		304	304	-	-	-	-	108,39	3600	3600
40n4m1a2p5		317	317	-	-	-	-	99,53	3600	3600
25n5m1a2p1	7	156	156	-	-	-	-	39,2	3600	3600
25n5m1a2p2		154	154	-	-	-	-	7,02	3600	3600
25n5m1a2p3		153	153	-	-	-	-	10,86	3600	3600
25n5m1a2p4		152	152	152	150	153	146	25,19	3600	3600
25n5m1a2p5		157	157	-	-	-	-	11,81	3600	3600
35n5m1a2p1	8	213	213	-	-	-	-	39,19	3600	3600
35n5m1a2p2		213	213	-	-	-	-	37,05	3600	3600
35n5m1a2p3		210	210	-	-	-	-	32,13	3600	3600
35n5m1a2p4		210	210	-	-	-	-	39,14	3600	3600
35n5m1a2p5		211	211	-	-	-	-	34,61	3600	3600
50n5m1a2p1	9	297	297	-	-	-	-	794,63	3600	3600
50n5m1a2p2		301	301	-	-	-	-	1287,36	3600	3600
50n5m1a2p3		303	303	-	-	-	-	198,12	3600	3600
50n5m1a2p4		300	300	-	-	-	-	164,59	3600	3600
50n5m1a2p5		299	299	-	-	-	-	469,7	3600	3600
21n7m1a2p1	10	94	94	94	94	94	94	1,26	195,44	245,92
21n7m1a2p2		93	93	93	93	93	93	3,71	106,19	106,93
21n7m1a2p3		92	92	92	92	92	92	2,43	102,08	162,48
21n7m1a2p4		96	96	96	96	96	96	19,68	75,21	47,29
21n7m1a2p5		94	94	94	94	94	94	3,00	185,19	251,65
28n7m1a2p1	11	121	121	-	-	123	102	17,52	3600	3600
28n7m1a2p2		120	120	-	-	-	-	9,68	3600	3600
28n7m1a2p3		122	122	-	-	-	-	7,34	3600	3600
28n7m1a2p4		120	120	-	-	-	-	5,94	3600	3600
28n7m1a2p5		121	121	-	-	-	-	9,95	3600	3600
35n7m1a2p1	12	148	148	-	-	-	-	66,95	3600	3600
35n7m1a2p2		147	147	-	-	-	-	44,55	3600	3600
35n7m1a2p3		151	151	-	-	-	-	145,68	3600	3600
35n7m1a2p4		149	149	-	-	-	-	20,68	3600	3600
35n7m1a2p5		149	149	-	-	-	-	24,08	3600	3600
30n10m1a2p1	13	97	97	97	96	97	92	3,56	3600	3600
30n10m1a2p2		97	97	97	97	-	-	3,27	3569,3	3600
30n10m1a2p3		98	98	98	96	98	92	2,27	3600	3600
30n10m1a2p4		97	97	97	96	98	92	4,44	3600	3600
30n10m1a2p5		96	96	96	96	96	92	3,58	1171,99	3600
40n10m1a2p1	14	125	125	-	-	-	-	20,65	3600	3600
40n10m1a2p2		126	126	-	-	-	-	20,06	3600	3600
40n10m1a2p3		127	127	-	-	-	-	21,00	3600	3600
40n10m1a2p4		125	125	-	-	-	-	34,41	3600	3600
40n10m1a2p5		125	125	-	-	-	-	18,28	3600	3600
50n10m1a2p1	15	152	152	-	-	-	-	114,23	3600	3600
50n10m1a2p2		152	152	-	-	-	-	75,36	3600	3600
50n10m1a2p3		153	153	-	-	-	-	30,61	3600	3600
50n10m1a2p4		152	152	-	-	-	-	54,26	3600	3600
50n10m1a2p5		150	150	-	-	-	-	138,98	3600	3600

Tabela A.3: Resultados - $P, S_1|s_j|C_{max}$ e $\varepsilon = 0,3$

Instância	g	TIF		F1		F2		CPU (s)		
		UB	LB	UB	LB	UB	LB	TIF	F1	F2
15n3m2a2p1	16	164	164	164	164	164	164	6,91	17,25	32,73
15n3m2a2p2		153	153	153	153	153	153	5,08	9,67	21,06
15n3m2a2p3		164	164	164	164	164	164	4,28	27,75	61,05
15n3m2a2p4		171	171	171	171	171	171	7,29	7,08	12,55
15n3m2a2p5		166	166	166	166	166	166	9,81	26,15	33,86
21n3m2a2p1	17	223	223	223	223	223	203,67	41,28	2870,1	3600
21n3m2a2p2		224	224	224	224	224	224	39,86	3160,93	3090,7
21n3m2a2p3		220	220	220	220	220	220	56,55	608,79	1477,28
21n3m2a2p4		237	237	237	237	237	210,81	86,36	2559,3	3600
21n3m2a2p5		234	234	234	226,5	234	200,77	44,69	3600	3600
30n3m2a2p1	18	315	315	-	-	-	-	519,07	3600	3600
30n3m2a2p2		311	311	-	-	-	-	876,11	3600	3600
30n3m2a2p3		336	336	-	-	-	-	262,12	3600	3600
30n3m2a2p4		314	314	-	-	-	-	641,35	3600	3600
30n3m2a2p5		319	319	-	-	319	206	443,41	3600	3600
20n4m2a2p1	19	156	156	156	156	156	156	10,15	1446,64	1555,17
20n4m2a2p2		159	159	159	139,33	159	135,88	12,57	3600	3600
20n4m2a2p3		160	160	160	160	161	151	6,66	3118,6	3600
20n4m2a2p4		157	157	158	147,67	158	144	13,5	3600	3600
20n4m2a2p5		151	151	151	151	-	-	7,79	2454,55	3600
28n4m2a2p1	20	203	203	-	-	-	-	211,32	3600	3600
28n4m2a2p2		229	229	-	-	-	-	190,65	3600	3600
28n4m2a2p3		203	203	-	-	-	-	109,31	3600	3600
28n4m2a2p4		206	206	-	-	-	-	168,63	3600	3600
28n4m2a2p5		204	204	-	-	-	-	119,4	3600	3600
40n4m2a2p1	21	310	310	-	-	-	-	1404,83	3600	3600
40n4m2a2p2		318	318	-	-	-	-	1270,06	3600	3600
40n4m2a2p3		326	326	-	-	-	-	1317,17	3600	3600
40n4m2a2p4		307	307	-	-	-	-	641,13	3600	3600
40n4m2a2p5		311	311	-	-	-	-	1169,95	3600	3600
25n5m2a2p1	22	142	142	-	-	-	-	21,33	3600	3600
25n5m2a2p2		150	150	-	-	-	-	17,89	3600	3600
25n5m2a2p3		146	146	-	-	-	-	78,19	3600	3600
25n5m2a2p4		153	153	-	-	-	-	18,79	3600	3600
25n5m2a2p5		148	148	150	124	-	-	26,66	3600	3600
35n5m2a2p1	23	206	206	-	-	-	-	375,94	3600	3600
35n5m2a2p2		196	196	-	-	-	-	141,25	3600	3600
35n5m2a2p3		211	211	-	-	-	-	141,52	3600	3600
35n5m2a2p4		218	218	-	-	-	-	114,73	3600	3600
35n5m2a2p5		210	210	-	-	-	-	54,37	3600	3600
50n5m2a2p1	24	283	283	-	-	-	-	2209,63	3600	3600
50n5m2a2p2		287	287	-	-	-	-	1134,31	3600	3600
50n5m2a2p3		293	293	-	-	-	-	1921,06	3600	3600
50n5m2a2p4		288	288	-	-	-	-	1363,82	3600	3600
50n5m2a2p5		290	290	-	-	-	-	1802,92	3600	3600
21n7m2a2p1	25	91	91	92	80,56	92	76,33	3,51	3600	3600
21n7m2a2p2		96	96	96	88,75	96	85	6,23	3600	3600
21n7m2a2p3		91	91	93	79	92	77	3,37	3600	3600
21n7m2a2p4		87	87	87	87	88	78	1,43	1204,91	3600
21n7m2a2p5		90	90	91	80,67	91	75,94	2,66	3600	3600
28n7m2a2p1	26	122	122	-	-	-	-	16,48	3600	3600
28n7m2a2p2		122	122	-	-	-	-	12,94	3600	3600
28n7m2a2p3		122	122	-	-	-	-	41,33	3600	3600
28n7m2a2p4		116	116	-	-	-	-	11,09	3600	3600
28n7m2a2p5		115	115	-	-	-	-	10,65	3600	3600

Continua na próxima página

Tabela A.3: (*Continuação*)

Instância	g	TIF		F1		F2		CPU (s)		
		UB	LB	UB	LB	UB	LB	TIF	F1	F2
35n7m2a2p1	27	141	141	-	-	-	-	20,39	3600	3600
35n7m2a2p2		144	144	-	-	-	-	178,52	3600	3600
35n7m2a2p3		146	146	-	-	-	-	40,34	3600	3600
35n7m2a2p4		148	148	-	-	-	-	60,97	3600	3600
35n7m2a2p5		132	132	-	-	-	-	16,12	3600	3600
30n10m2a2p1	28	86	86	-	-	-	-	8,96	3600	3600
30n10m2a2p2		82	82	84	64	-	-	2,45	3600	3600
30n10m2a2p3		92	92	-	-	-	-	3,03	3600	3600
30n10m2a2p4		86	86	-	-	-	-	7,3	3600	3600
30n10m2a2p5		90	90	-	-	-	-	7,48	3600	3600
40n10m2a2p1	29	114	114	-	-	-	-	37,83	3600	3600
40n10m2a2p2		115	115	-	-	-	-	54,47	3600	3600
40n10m2a2p3		113	113	-	-	-	-	16,23	3600	3600
40n10m2a2p4		109	109	-	-	-	-	31,63	3600	3600
40n10m2a2p5		111	111	-	-	-	-	38,65	3600	3600
50n10m2a2p1	30	140	140	-	-	-	-	36,59	3600	3600
50n10m2a2p2		145	145	-	-	-	-	130,1	3600	3600
50n10m2a2p3		140	140	-	-	-	-	93,16	3600	3600
50n10m2a2p4		142	142	-	-	-	-	50,66	3600	3600
50n10m2a2p5		139	139	-	-	-	-	141,71	3600	3600

A.2 Resultados detalhados para o $P, S_1|s_{ij}|C_{max}$

Para os métodos meta-heurísticos são reportados a melhor solução (Best) encontrada por cada um e o valor médio das soluções (Avg) nas 10 execuções em cada instância. A coluna Avg CPU(s) indica a média dos tempos obtidos nas 10 execuções. Os tempos de execução dos métodos SA e GA (HAMZADAYI; YILDIZ, 2017) são os mesmos do ILS_{CS}. Em relação ao ATIF, são mostrados os limites superiores (UB) e inferiores (LB) e o tempo de execução.

Tabela A.4: Resultados - $P, S_1|s_{ij}|C_{max}$

Instância	ATIF			ILS_{CS}		HILS-CSvr		SA		GA		Avg CPU(s)	
	UB	LB	CPU(s)	Best	Avg	Best	Avg	Best	Avg	Best	Avg	ILS_{CS}	HILS-CSvr
6n2m1	188	188	0,16	188	188,00	188	188,00	188	188,00	188	188,00	<0,01	0,04
6n2m2	201	201	0,13	201	201,00	201	201,00	201	201,00	201	201,00	<0,01	0,24
6n2m3	179	179	0,11	179	179,00	179	179,00	179	179,00	179	179,00	<0,01	0,08
6n2m4	151	151	0,09	151	151,00	151	151,00	151	151,00	151	151,00	<0,01	0,05
6n2m5	188	188	0,11	188	188,00	188	188,00	188	188,00	188	188,00	<0,01	0,17
8n2m1	244	244	0,75	244	244,00	244	244,00	244	244,00	244	245,50	<0,01	0,22
8n2m2	170	170	0,41	170	170,00	170	170,00	170	170,30	170	172,30	<0,02	0,13
8n2m3	245	245	0,74	245	245,00	245	245,00	245	245,00	245	246,50	<0,01	0,13
8n2m4	209	209	0,56	209	209,00	210	210,00	209	209,00	210	211,00	<0,01	0,14
8n2m5	219	219	0,67	219	219,00	225	225,70	219	219,70	219	223,30	<0,01	0,06
10n2m1	264	264	2,52	264	264,30	264	264,50	264	270,40	265	273,80	0,03	0,20
10n2m2	286	286	3,31	286	286,00	286	286,00	286	289,20	286	293,00	0,02	0,20
10n2m3	270	270	2,48	270	270,50	270	270,00	275	277,00	277	282,60	0,03	0,21

Continua na próxima página

Tabela A.4: (*Continuação*)

Instância	ATIF			ILS_{CS}		HILS-CSvr		SA		GA		Avg CPU(s)	
	UB	LB	CPU(s)	Best	Avg	Best	Avg	Best	Avg	Best	Avg	ILS_{CS}	HILS-CSvr
10n2m4	386	386	4,89	386	386,00	389	389,10	386	390,10	393	396,30	0,03	0,32
10n2m5	344	344	3,82	344	344,00	346	346,00	345	347,10	346	352,20	0,03	0,22
14n2m1*	467	467	326,52	467	467,50	467	468,50	467	471,70	473	479,60	0,06	1,11
14n2m2	388	388	32,47	388	389,80	388	388,40	388	396,00	397	406,40	0,07	0,75
14n2m3	402	402	465,48	402	402,80	402	403,10	402	405,80	414	416,90	0,07	0,94
14n2m4	503	503	50,86	503	503,00	507	509,10	512	514,40	516	524,70	0,07	0,97
14n2m5	444	444	118,29	444	444,80	444	444,00	444	449,10	453	462,00	0,07	0,83
20n2m1*	633	336,49	3600	633	637,30	634	637,10	638	644,30	655	667,50	0,23	3,81
20n2m2	658	350,86	3600	658	661,20	660	662,80	666	668,30	682	689,70	0,22	4,03
20n2m3	518	272	3600	518	520,90	516	517,00	523	527,90	543	549,70	0,22	3,15
20n2m4	594	315,08	3600	594	597,10	594	595,00	600	604,00	614	622,20	0,22	3,78
20n2m5	617	328,26	3600	617	621,50	616	617,50	624	630,00	640	647,10	0,22	3,84
9n3m1	207	207	0,39	207	207,00	209	209,00	207	208,30	209	212,20	0,02	0,75
9n3m2	163	163	0,23	163	163,00	163	163,00	163	165,40	166	167,40	0,02	0,50
9n3m3	151	151	0,26	151	151,60	151	151,00	151	151,80	152	153,60	0,02	0,39
9n3m4	151	151	0,32	151	151,00	151	151,00	151	151,20	151	153,90	0,02	0,47
9n3m5	227	227	0,54	227	227,00	228	228,90	227	227,10	227	229,90	0,02	0,97
12n3m1	256	256	2,66	256	256,60	259	264,10	258	261,00	267	270,50	0,05	0,63
12n3m2	269	269	2,61	269	269,00	269	269,50	278	278,90	278	287,00	0,04	0,51
12n3m3	206	206	10,49	207	207,00	207	207,70	207	209,70	209	214,20	0,04	0,60
12n3m4	271	271	2,44	271	271,10	276	276,40	272	276,90	280	286,30	0,04	0,61
12n3m5	271	271	3,11	271	271,00	271	272,40	271	274,60	278	280,80	0,04	1,07
15n3m1	285	285	63,16	285	286,30	288	290,10	285	290,40	293	299,20	0,09	1,27
15n3m2	370	370	14,92	370	371,00	370	371,60	374	376,40	378	385,20	0,09	1,29
15n3m3	329	329	120,72	329	330,40	332	333,30	331	335,10	342	345,60	0,09	1,24
15n3m4	297	297	13,92	297	299,30	297	300,60	302	304,90	307	314,20	0,08	1,11
15n3m5	303	303	12,55	303	305,70	304	308,10	307	310,00	315	320,20	0,08	1,31
21n3m1	470	314,92	3600	470	472,10	466	471,60	474	477,60	488	494,80	0,26	3,33
21n3m2	483	324,86	3600	483	487,40	485	489,00	488	491,40	504	511,00	0,26	3,69
21n3m3	502	335,49	3600	502	503,50	503	506,10	504	508,00	518	527,30	0,26	3,75
21n3m4*	381	381	3040,32	387	388,60	385	387,00	391	393,80	400	405,80	0,25	3,04
21n3m5	512	345,71	3600	512	515,80	515	515,20	515	520,50	530	538,50	0,24	3,80
30n3m1	657	343,10	3600	657	658,80	653	655,90	662	666,90	677	688,30	0,88	10,62
30n3m2*	624	325,75	3600	624	627,00	621	624,10	628	634,30	653	658,30	0,94	10,71
30n3m3	574	296,10	3600	574	576,10	568	569,30	579	583,90	602	605,00	0,86	9,82
30n3m4	592	308,75	3600	592	593,80	587	590,80	603	604,00	612	621,70	0,90	10,87
30n3m5	583	305,55	3600	583	585,90	581	584,60	591	594,90	601	606,10	0,92	10,96
12n4m1	215	215	1,12	215	215,30	215	216,20	216	216,90	217	219,90	0,04	2,59
12n4m2	229	229	1,44	229	229,00	229	229,00	229	231,60	236	238,80	0,05	2,79
12n4m3	167	167	0,89	167	167,00	167	168,80	167	169,90	176	179,30	0,04	1,15
12n4m4	115	115	0,47	115	115,00	115	115,40	115	117,60	120	123,10	0,04	1,51
12n4m5	213	213	1,61	213	213,00	215	216,60	215	215,80	218	220,60	0,04	2,15
16n4m1	243	243	100,31	245	245,90	246	249,80	248	250,40	256	259,70	0,11	2,16
16n4m2	247	247	112,34	247	248,30	250	253,20	250	252,40	257	261,90	0,11	1,74
16n4m3	224	224	79,77	224	224,40	226	231,20	226	227,30	233	238,80	0,11	1,70
16n4m4	228	228	41,16	228	230,70	232	234,10	232	232,90	238	245,10	0,11	1,55
16n4m5	218	218	48,96	218	219,10	218	220,40	220	222,50	228	232,10	0,12	1,89
20n4m1	309	227	3600	309	310,90	311	315,10	314	317,10	326	333,00	0,23	2,66
20n4m2	318	318	2677,52	321	322,70	321	323,80	327	328,60	328	337,30	0,23	2,80
20n4m3	338	338	506,89	340	343,40	341	344,60	344	348,90	356	361,60	0,24	3,01
20n4m4	357	357	842,06	357	360,00	359	362,50	362	364,10	374	383,20	0,25	3,08
20n4m5	327	327	2323,17	330	330,90	329	332,90	332	335,20	342	346,30	0,22	2,93
28n4m1	402	209,51	3600	402	404,00	403	404,70	408	410,10	422	425,10	0,71	7,53
28n4m2	429	224	3600	429	430,80	427	431,20	435	438,90	446	456,60	0,71	7,40
28n4m3	416	220,24	3600	416	418,70	418	419,40	426	428,50	432	441,80	0,68	7,85
28n4m4*	417	217,68	3600	417	419,40	419	421,80	425	427,80	438	444,20	0,75	7,95
28n4m5	436	229,21	3600	436	437,30	436	440,10	439	443,40	451	457,40	0,68	8,32
40n4m1*	633	0	3600	633	638,30	632	634,80	646	651,70	663	669,60	2,68	25,88
40n4m2	627	0	3600	627	629,00	622	626,90	638	641,30	654	659,50	2,54	25,02
40n4m3	651	0	3600	651	654,30	650	652,30	663	667,20	684	689,20	2,50	23,94
40n4m4	672	0	3600	672	674,60	669	673,40	680	685,30	694	702,40	2,67	28,52
40n4m5	550	0	3600	550	553,00	548	550,70	562	565,00	574	580,60	2,73	22,98
15n5m1	187	187	9,38	187	187,20	187	190,10	188	190,90	195	199,70	0,09	4,49

Continua na próxima página

Tabela A.4: (*Continuação*)

Instância	ATIF			ILS_{CS}		HILS-CSvr		SA		GA		Avg CPU(s)	
	UB	LB	CPU(s)	Best	Avg	Best	Avg	Best	Avg	Best	Avg	ILS_{CS}	HILS-CSvr
15n5m2	187	187	18,11	187	187,00	188	188,00	187	188,90	194	196,20	0,09	7,38
15n5m3	169	169	3,20	169	169,30	169	170,20	172	173,60	174	179,50	0,09	4,65
15n5m4	168	168	2,03	168	168,60	168	170,40	169	170,90	169	176,80	0,08	3,03
15n5m5	187	187	2,22	187	187,20	187	191,50	190	191,50	193	197,20	0,09	5,89
20n5m1	244	244	349,81	245	246,10	248	251,10	248	250,40	256	259,10	0,27	5,15
20n5m2*	233	233	1766,41	236	236,90	238	240,00	238	241,40	243	250,00	0,24	5,05
20n5m3	248	248	427,93	250	252,40	254	256,30	255	256,90	262	267,10	0,24	3,58
20n5m4	289	289	273,15	291	292,40	293	296,60	296	297,40	303	306,90	0,24	3,86
20n5m5	214	214	313,34	215	216,60	215	219,50	219	221,30	226	230,40	0,24	3,49
25n5m1	342	234,78	3600	342	344,70	345	348,60	347	350,60	361	366,40	0,53	5,16
25n5m2	229	166,72	3600	229	230,20	234	237,30	232	237,20	244	251,70	0,54	8,18
25n5m3	357	243,71	3600	357	359,10	362	365,00	361	364,40	373	378,30	0,52	5,89
25n5m4	357	241,77	3600	357	358,20	362	367,00	359	362,50	375	379,30	0,57	6,22
25n5m5	310	211,13	3600	310	311,40	314	316,80	313	316,30	324	330,80	0,52	5,70
35n5m1	405	211,34	3600	405	408,70	408	410,30	415	419,00	431	435,30	1,84	17,49
35n5m2*	404	208,34	3600	404	406,50	405	410,40	413	416,00	424	431,80	1,84	17,55
35n5m3	412	212,81	3600	412	414,00	414	417,40	420	423,10	428	435,10	1,74	25,66
35n5m4	421	218,65	3600	421	425,30	425	428,00	432	434,50	446	451,60	1,88	20,43
35n5m5	402	209,23	3600	402	402,90	403	404,90	411	413,70	420	429,50	1,76	18,85
50n5m1	600	0	3600	600	603,00	591	595,50	618	621,30	632	637,90	6,69	5,44
50n5m2*	666	0	3600	666	668,70	656	660,80	679	684,80	690	698,10	6,22	5,47
50n5m3	656	0	3600	656	660,90	648	653,30	676	677,30	688	693,80	5,58	5,38
50n5m4	650	0	3600	650	653,60	644	647,00	668	671,20	677	684,90	6,41	5,19
50n5m5	647	0	3600	647	649,90	640	643,50	665	667,10	672	682,20	5,77	5,03
21n7m1	179	179	416,48	180	180,10	180	180,20	181	183,60	190	194,20	0,29	16,32
21n7m2	158	158	75,64	158	159,70	162	165,50	163	165,00	169	173,10	0,31	24,81
21n7m3	180	180	1418,52	182	182,20	183	189,70	183	185,40	189	199,00	0,33	22,41
21n7m4	205	205	132,73	206	206,70	209	211,80	209	212,00	218	225,40	0,33	33,50
21n7m5	154	154	56,70	154	155,50	159	163,10	159	160,90	169	174,20	0,33	21,91
28n7m1	229	175,14	3600	229	231,10	238	240,10	234	238,10	243	250,10	0,89	18,51
28n7m2	267	191,48	3600	267	268,60	274	278,90	272	275,50	279	286,20	0,91	15,49
28n7m3	290	204,08	3600	290	292,60	300	303,20	298	299,00	307	314,00	0,93	9,41
28n7m4	267	187,37	3600	267	268,30	272	276,50	272	273,50	284	287,50	0,93	12,36
28n7m5	215	165,94	3600	215	216,60	221	225,00	218	221,10	232	235,70	0,96	17,75
35n7m1	311	162	3600	311	313,40	318	321,90	319	322,30	329	338,90	2,05	21,35
35n7m2	275	183,15	3600	275	277,40	280	285,00	283	285,90	289	299,10	2,16	30,51
35n7m3	316	202,30	3600	316	320,30	321	327,00	328	330,00	343	346,90	2,16	26,65
35n7m4*	351	202,43	3600	351	353,20	362	367,70	361	363,10	371	380,50	2,17	31,57
35n7m5	332	173	3600	332	333,50	337	342,40	340	342,80	345	359,80	2,11	21,69
49n7m1	435	0	3600	435	439,20	434	444,80	448	455,40	468	473,30	6,65	6,35
49n7m2	447	0	3600	447	448,30	449	453,00	463	464,60	466	478,50	6,88	6,01
49n7m3	481	0	3600	481	483,60	480	486,90	497	499,00	502	516,50	6,98	6,32
49n7m4	433	222,40	3600	433	436,40	429	441,10	449	452,80	462	468,90	6,63	6,25
49n7m5*	466	0	3600	466	470,20	467	471,60	483	486,80	497	506,30	6,82	5,86
70n7m1	624	0	3600	624	626,40	632	637,90	652	655,60	652	669,90	26,04	16,96
70n7m2	632	0	3600	632	634,10	632	640,70	659	662,40	669	681,50	24,59	17,11
70n7m3	624	0	3600	624	628,40	632	638,00	653	655,50	664	671,00	25,25	20,14
70n7m4*	620	0	3600	620	623,00	624	631,30	648	651,90	657	666,60	24,48	18,01
70n7m5	570	0	3600	570	572,20	571	581,80	599	600,70	602	616,50	23,60	16,11
30n10m1	180	143	3600	180	180,80	186	195,70	183	186,10	191	199,10	1,33	2,86
30n10m2	187	150	3600	187	188,50	198	208,60	193	196,70	205	215,10	1,56	3,96
30n10m3	182	155	3600	182	185,20	192	202,10	189	192,20	204	209,40	1,48	3,85
30n10m4	171	144	3600	171	172,90	179	193,10	177	181,30	186	195,20	1,41	3,01
30n10m5*	195	162	3600	195	198,00	214	222,00	203	205,30	209	220,00	1,56	3,47
40n10m1	267	180,94	3600	267	270,20	285	290,60	279	284,00	299	306,80	4,70	7,82
40n10m2	274	186,34	3600	274	275,90	283	292,20	283	287,80	302	306,50	4,52	8,68
40n10m3	258	176,16	3600	258	262,30	268	281,20	270	273,90	288	295,00	4,45	8,76
40n10m4*	244	166,13	3600	244	248,10	265	268,40	260	261,90	266	282,00	4,88	8,75
40n10m5	230	154,61	3600	230	232,80	249	261,30	241	244,40	258	264,80	4,14	7,06
50n10m1	316	161,96	3600	316	319,90	330	347,00	334	338,50	347	360,30	10,37	15,47
50n10m2	332	171,30	3600	332	336,10	346	359,70	350	353,00	367	373,90	10,24	13,01
50n10m3	326	166,23	3600	326	327,60	342	349,40	342	344,20	352	366,30	9,73	14,64
50n10m4	330	167,03	3600	330	331,80	348	360,90	349	351,20	368	375,00	11,20	14,29

Continua na próxima página

Tabela A.4: (*Continuação*)

Instância	ATIF			ILS_{CS}		HILS-CSvr		SA		GA		Avg CPU(s)	
	UB	LB	CPU(s)	Best	Avg	Best	Avg	Best	Avg	Best	Avg	ILS_{CS}	HILS-CSvr
50n10m5*	314	161	3600	314	317,80	330	343,10	328	335,30	348	366,10	10,81	13,29
70n10m1	435	0	3600	435	439,80	472	481,80	472	476,30	476	498,00	42,16	30,47
70n10m2	449	0	3600	449	451,60	469	481,50	480	482,70	486	499,30	38,59	39,62
70n10m3	455	0	3600	455	459,00	480	494,80	489	492,60	509	514,00	37,93	31,34
70n10m4	445	0	3600	445	447,70	481	487,50	473	480,70	490	506,70	37,42	35,39
70n10m5	415	0	3600	415	419,50	458	463,80	446	453,80	459	469,70	39,27	29,19
100n10m1	665	0	3600	665	672,90	704	721,60	731	732,20	730	742,60	147,21	77,45
100n10m2	648	0	3600	648	653,30	694	703,00	701	710,80	713	720,10	151,22	79,16
100n10m3	645	0	3600	645	648,80	687	706,30	704	710,60	713	729,80	158,42	89,72
100n10m4	653	0	3600	653	657,70	699	710,10	711	716,40	717	729,60	141,21	110,50
100n10m5	672	0	3600	672	679,30	711	728,00	734	739,10	732	755,10	137,78	110,06

*Instâncias utilizadas na calibração dos parâmetros do ILS_{CS}