



UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA A
MINIMIZAÇÃO DO ATRASO PONDERADO NO ESCALONAMENTO DE
TAREFAS EM MÁQUINAS PARALELAS

MARCOS THOMAZ DA SILVA

Dezembro de 2018

Manaus - AM

MARCOS THOMAZ DA SILVA

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA A
MINIMIZAÇÃO DO ATRASO PONDERADO NO ESCALONAMENTO DE
TAREFAS EM MÁQUINAS PARALELAS

Dissertação apresentada ao Programa de Pós-graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas (PPGI/IComp, UFAM) como parte dos requisitos necessários à obtenção do título de Mestre em Informática .

Orientadora: Rosiane de Freitas Rodrigues, Dra.

Dezembro de 2018

Manaus - AM

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

S586h Silva, Marcos Thomaz da
Hibridização de Métodos Exatos e Heurísticos Para a
Minimização do Atraso Ponderado no Escalonamento de Tarefas
em Máquinas Paralelas / Marcos Thomaz da Silva. 2018
112 f.: il. color; 31 cm.

Orientadora: Rosiane deFreitas
Dissertação (Ciência da Computação) - Universidade Federal do
Amazonas.

1. algoritmos genéticos. 2. busca local. 3. heurísticas. 4.
programação linear inteira. 5. escalonamento de tarefas. I.
deFreitas, Rosiane II. Universidade Federal do Amazonas III. Título



PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA



FOLHA DE APROVAÇÃO

"Hibridização de métodos exatos e heurísticos para a minimização do atraso ponderado no escalonamento de tarefas em máquinas paralelas"

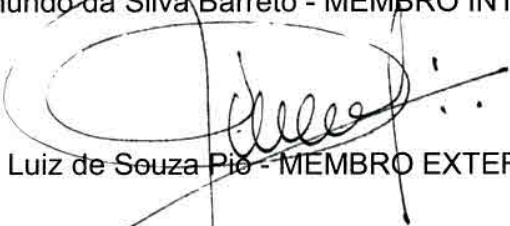
MARCOS THOMAZ DA SILVA

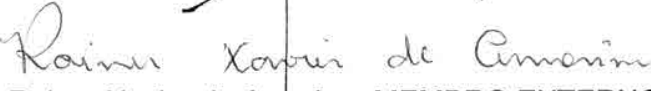
Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:


Profa. Rosiane de Freitas Rodrigues - PRESIDENTE


Prof. José Reginaldo Hughes Carvalho - MEMBRO INTERNO


Prof. Raimundo da Silva Barreto - MEMBRO INTERNO


Prof. José Luiz de Souza Pio - MEMBRO EXTERNO


Prof. Rainer Xavier de Amorim - MEMBRO EXTERNO

Manaus, 17 de Dezembro de 2018

*A Deus,
a minha esposa e filhos,
a minha família,
aos professores,
aos amigos e colegas.*

Agradecimentos

Agradeço a Deus que me permitiu atingir este objetivo, me dando forças e me guiando.

A minha esposa Kelly e meus filhos Marcos Guilherme e Maria Eduarda pela paciência, carinho, alegria e incentivo a todos os momentos.

A meus pais por acreditarem em mim, incentivando e sempre me direcionando a buscar cada vez mais.

A minha orientadora, professora Rosiane, que tanto incentivou e apoiou, não como uma professora, mas como uma amiga.

A toda a equipe do IComp / PPGI, pela atenção e cortesia no atendimento.

Aos professores e técnico-administrativos da Ufam e a Fapeam, que fizeram com que tudo isso fosse possível.

Resumo da Dissertação apresentada ao PPGI/IComp/UFAM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.

HIBRIDIZAÇÃO DE MÉTODOS EXATOS E HEURÍSTICOS PARA A
MINIMIZAÇÃO DO ATRASO PONDERADO NO ESCALONAMENTO DE
TAREFAS EM MÁQUINAS PARALELAS

MARCOS THOMAZ DA SILVA

Dezembro/2018

Orientadora: Profa. Rosiane de Freitas Rodrigues, Dra.

Nesta dissertação estão sendo apresentados os resultados da investigação realizada sobre problemas de escalonamento em máquinas paralelas, com foco na minimização do atraso ponderado total das tarefas, com tempos de processamento e prazos estimados de término arbitrários. Este é um problema clássico muito encontrado em indústrias, ambientes de produção e cenários onde o atraso na realização de tarefas ou produção pode gerar multas ou penalidades. A estratégia de resolução aplicada faz uso de um método híbrido exato-heurístico, onde a execução de um Algoritmo Genético fortemente baseado em Busca Local, denominado GLS, fornece um conjunto de soluções de ótimos locais para ser incorporado em uma formulação de programação linear inteira tri-indexada, otimizando o processo de resolução enumerativo implícito. Sendo a parametrização um problema inerente aos algoritmos genéticos e meta-heurísticas em geral, foi utilizada a ferramenta iRace para a otimização e definição de parâmetros. O solver IBM ILog CPLEX e a biblioteca dinâmica UFFLP foram utilizados para avaliar o conjunto de soluções obtido através das heurísticas utilizando a formulação de programação inteira. Os experimentos computacionais foram realizados em instâncias criadas com base no *benchmark* disponível na OR-Library com 40, 50, 100, 150, 200, 300 e 500 tarefas, em ambientes com 2, 4, 10 e 20 máquinas paralelas, obtendo resultados competitivos frente às melhores estratégias disponibilizadas na literatura, e apresentando a robustez do método em instâncias maiores.

Palavras-chave: algoritmos genéticos, atraso ponderado, busca local, escalonamento de tarefas, heurísticas, máquinas paralelas, programação linear inteira.

Abstract of Dissertation presented to PPGI/IComp/UFAM as a partial fulfillment of the requirements for the degree of Master in Informatics.

HYBRIDIZATION EXACT AND HEURISTIC METHODS FOR MINIMIZING WEIGHTED TARDINESS JOB SCHEDULING IN PARALLEL MACHINES

MARCOS THOMAZ DA SILVA

December/2018

Advisor: Prof. Rosiane de Freitas Rodrigues, D.Sc.

This dissertation presents the results of the investigation carried out on parallel machine scheduling problems, with focus on minimizing the total weighted tardiness of the jobs, with arbitrary processing times and deadlines. This is a classic scheduling problem that is often found in industries, production environments, and scenarios where delayed completion of jobs can lead to penalties. The applied resolution strategy makes use of an exact-heuristic hybrid method, where the execution of a strongly Local Search-based Genetic Algorithm, called GLS, provides a set of optimal location solutions to be incorporated in a tri-indexed integer linear programming formulation, optimizing the implicit enumerative resolution process. How the parameterization is an inherent problem in genetic algorithms and meta-heuristics in general, the iRace tool was used for optimization and parameter definition. The IBM ILog CPLEX solver and UFFLP dynamic library was used to evaluate the solution set obtained through heuristics using the integer linear programming formulation. The computational experiments were performed for instances created similar of the OR-Library with 40, 50, 100, 150, 200, 300 and 500 tasks in 2, 4, 10 and 20 parallel machines, obtaining competitive results against the best strategies available in the literature, and presenting the robustness of the method in larger instances.

Keywords: genetic algorithms, weighted tardiness, local search, scheduling jobs, heuristics, parallel machines, linear integer programming.

Sumário

Lista de Figuras	j
Lista de Tabelas	l
Lista de Algoritmos	n
Lista de Siglas	o
1 Introdução	1
1.1 Contexto e descrição do problema	2
1.2 Motivação	5
1.3 Hipótese	7
1.4 Objetivos	7
1.5 Organização do trabalho	8
2 Fundamentação Teórica	9
2.1 Otimização combinatória	9
2.2 Problemas de escalonamento	11
2.3 Métodos de resolução	16
2.3.1 Programação linear inteira	17
2.3.2 Heurísticas e meta-heurísticas	24
2.3.3 Métodos híbridos	34
2.4 Otimização de parâmetros	35
3 Escalonamento em Máquinas Paralelas com Minimização de Atraso Total	
Ponderado	36
3.1 Trabalhos relacionados	38

3.2	Formulações matemáticas	41
3.2.1	Formulação de programação linear inteira mista para o escalonamento em máquinas paralelas com tempo ocioso	41
3.2.2	Formulação de programação linear inteira mista para o escalonamento em uma máquina sem tempo ocioso	43
3.2.3	Formulação baseada no modelo de fluxo em redes e roteamento de veículos para o o escalonamento em máquinas paralelas - <i>Arc-time indexed formulation</i>	44
4	Método Híbrido Heurístico-Exato Proposto	46
4.1	Escolha da formulação e método de aplicação	48
4.2	Genetic + local search - GLS	49
4.2.1	Geração da população inicial	50
4.2.2	Função de aptidão ou <i>fitness</i>	53
4.2.3	Operador genético de cruzamento	53
4.2.4	Busca local iterada - GPI	55
4.2.5	Operador genético de mutação	57
4.2.6	Diversidade populacional	59
4.2.7	Elitismo e novas gerações	59
4.3	Armazenamento de valores para método exato	60
4.4	Aplicação do método exato	61
4.5	Pacote <i>iRace</i> e Parametrização da execução	62
4.6	Considerações sobre o método	65
5	Resultados	68
5.1	Ambiente computacional	69
5.2	Instâncias de teste	69
5.3	Observações da execução	71
5.4	Análise dos resultados	75
5.4.1	Comparação dos resultados com métodos da literatura	78
5.4.2	Resumo de resultados obtidos	80
6	Conclusões	84
6.1	Considerações finais	85

<i>SUMÁRIO</i>	i
6.2 Trabalhos futuros	86
Referências	87
Apêndices	92
A Tabela Completa de Resultados	93

Lista de Figuras

1.1	Exemplo de aplicação do problema envolvendo máquinas paralelas.	3
1.2	Exemplo de aplicação do problema envolvendo máquinas paralelas.	3
1.3	Exemplo de operação em uma pequena indústria.	5
1.4	Instância com 6 tarefas para 2 máquinas.	6
2.1	Exemplo de árvore de <i>branch-and-bound</i>	18
2.2	Exemplo de solução gráfica.	19
2.3	Resultado do <i>branch</i>	19
2.4	Conjunto de soluções antes e depois do corte.	21
2.5	Exemplo de árvore - <i>branch-and-cut</i>	24
2.6	Funcionamento de heurística de busca local em um determinado espaço de busca.	26
2.7	Exemplo de seleção por roleta.	28
2.8	Exemplo de seleção por torneio.	29
2.9	Exemplo de cruzamento/recombinação usando método uniforme.	31
2.10	Cruzamento/recombinação usando método de um ponto.	31
2.11	Cruzamento/recombinação usando método de dois pontos.	32
2.12	Mutação aplicada a um determinado indivíduo.	32
2.13	Inversão aplicada a um determinado indivíduo.	33
3.1	Penalidade aplicada ao atraso.	37
4.1	Arquitetura geral do método híbrido heurístico-exato proposto.	47
4.2	Esquema de execução e obtenção de solução pelo método híbrido.	49
4.3	População inicial gerada com seleção do melhor indivíduo (EDD).	52
4.4	Dados de uma instância modelo.	53

4.5	Distribuição dos genes em máquinas e cálculo do atraso total ponderado.	53
4.6	Criação de novos indivíduos através do cruzamento uniforme.	54
4.7	Movimento <i>swap</i> realizado pela busca local.	56
4.8	Movimento <i>move</i> realizado pela busca local.	56
4.9	Operador genético de mutação com $q = 2$	57
4.10	Representação do escalonamento no modelo de fluxo em redes.	60
4.11	Arquivo de parâmetros do <i>iRace</i>	63
5.1	Primeiras 11 linhas de uma instância com 250 tarefas	70
5.2	Composição dos nomes de arquivos de instâncias.	70
5.3	Dispersão de tarefas de cada uma das instâncias analisadas para o problema $P \sum w_j T_j$ para 300 tarefas e 20 máquinas	71
5.4	Rastreio das soluções encontradas para o problema $P \sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 1	72
5.5	Rastreio das soluções encontradas para o problema $P \sum w_j T_j$ para 300 tarefas e 2 máquinas - instância 1	72
5.6	Rastreio das soluções encontradas para o problema $P \sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 116	73
5.7	Rastreio das soluções para o problema $P \sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 1 - sem população inicial	73
5.8	Melhores Resultados para Instância de 100 tarefas e 4 máquinas.	80
5.9	Efetividade dos 4 métodos para 200 instâncias.	83
5.10	Efetividade do método para 325 instâncias.	83

Lista de Tabelas

2.1	Símbolos usados em problemas de escalonamento.	12
2.2	Notação de três campos - problemas de escalonamento.	13
3.1	Classificação dos problemas de escalonamento com antecipação e atraso de tarefas.	40
3.2	Resumo de formulações.	45
4.1	Parâmetros utilizados na execução do híbrido.	64
5.1	Resumo de Resultados Obtidos.	76
5.2	Comparativo entre métodos para instâncias com 100 tarefas e 2 máquinas.	78
5.3	Comparativo entre métodos para instâncias com 100 tarefas e 4 máquinas.	79
5.4	Comparativo entre métodos para instâncias com 100 tarefas e 10 máquinas.	81
5.5	Comparativo entre métodos para instâncias com 300 tarefas e 2, 4, 10 e 20 máquinas	82
A.1	Resultados para $P \sum w_j T_j$ com 40 jobs em 2 máquinas.	94
A.2	Resultados para $P \sum w_j T_j$ com 40 jobs em 4 máquinas.	95
A.3	Resultados para $P \sum w_j T_j$ com 40 jobs em 10 máquinas.	96
A.4	Resultados para $P \sum w_j T_j$ com 50 jobs em 2 máquinas.	97
A.5	Resultados para $P \sum w_j T_j$ com 50 jobs em 4 máquinas.	98
A.6	Resultados para $P \sum w_j T_j$ com 50 jobs em 10 máquinas.	99
A.7	Resultados para $P \sum w_j T_j$ com 100 jobs em 2 máquinas.	100
A.8	Resultados para $P \sum w_j T_j$ com 100 jobs em 4 máquinas.	101
A.9	Resultados para $P \sum w_j T_j$ com 100 jobs em 10 máquinas.	102
A.10	Resultados para $P \sum w_j T_j$ com 150 jobs em 2 máquinas.	103
A.11	Resultados para $P \sum w_j T_j$ com 150 jobs em 4 máquinas.	104

A.12 Resultados para $P \sum w_j T_j$ com 150 <i>jobs</i> em 10 máquinas.	105
A.13 Resultados para $P \sum w_j T_j$ com 200 <i>jobs</i> em 2 máquinas.	106
A.14 Resultados para $P \sum w_j T_j$ com 200 <i>jobs</i> em 4 máquinas.	107
A.15 Resultados para $P \sum w_j T_j$ com 200 <i>jobs</i> em 10 máquinas.	108
A.16 Resultados para $P \sum w_j T_j$ com 300 <i>jobs</i> em 2 máquinas.	109
A.17 Resultados para $P \sum w_j T_j$ com 300 <i>jobs</i> em 4 máquinas.	110
A.18 Resultados para $P \sum w_j T_j$ com 300 <i>jobs</i> em 10 máquinas.	111
A.19 Resultados para $P \sum w_j T_j$ com 300 <i>jobs</i> em 20 máquinas.	112
A.20 Resultados para $P \sum w_j T_j$ com 500 <i>jobs</i> em 2 máquinas.	113
A.21 Resultados para $P \sum w_j T_j$ com 500 <i>jobs</i> em 4 máquinas.	114
A.22 Resultados para $P \sum w_j T_j$ com 500 <i>jobs</i> em 10 máquinas.	115
A.23 Resultados para $P \sum w_j T_j$ com 500 <i>jobs</i> em 20 máquinas.	116
A.24 Resultados para $P \sum w_j T_j$ com 500 <i>jobs</i> em 30 máquinas.	117

Lista de Algoritmos

1	Algoritmo <i>branch-and-bound</i>	20
2	Algoritmo de Plano de Cortes.	21
3	Algoritmo <i>branch-and-cut</i>	23
4	Algoritmo Genético.	27
5	Método de Seleção por Roleta.	29
6	Método de Seleção por Torneio.	30
7	Algoritmo de construção da população inicial baseado no algoritmo de Fisher-Yates.	50
8	Algoritmo de construção da população inicial.	52
9	Cruzamento Uniforme com Busca Local.	55
10	Algoritmo de busca local de de Freitas et al. (2008)	57
11	Algoritmo do operador genético de mutação.	58

Lista de Siglas

AG - Algoritmo Genético.

CDN - *Content Delivery Network* / Rede de Fornecimento de Conteúdo.

CP - *Constraint Programming* / Programação por Restrições

CPLEX - IBM Ilog Cplex, *solver* utilizado na resolução de problemas de otimização combinatória.

DNA - *Deoxyribonucleic acid* ou ácido desoxirribonucleico, um composto orgânico cujas moléculas contêm as instruções genéticas que coordenam o desenvolvimento e funcionamento de todos os seres vivos e alguns vírus.

EDD - *Earliest Due Date*, método que ordena as tarefas em ordem crescente de tempo estimado de término.

FO - Função objetivo também chamada de função meta.

GLS - Algoritmo Genético + Busca Local.

GPI - Generalized Pairwise Interchanges / Troca Generalizada de Pares.

HILL - *Hill Climbing* (Busca Local).

ILS - *Iterated Local Search* / Busca Local Iterada.

IP - *Integer Programming* / Programação Inteira.

iRace - Ferramenta de apoio a parametrização de algoritmos.

Job - Tarefa, atividade a ser realizada.

LTP - *Longest Processing Time* / Tempo de Maior de Processamento.

LS - *Local Search* / Busca Local.

OR-Library - Biblioteca de instâncias com diversos problemas de otimização.

PD - Programação Dinâmica.

PLI - Programação Linear Inteira.

PLIM - Programação Linear Inteira Mista.

PMTWTSP - *Parallel Machines Total Weighted Tardiness Scheduling Problem* ou Pro-

blema da Soma dos Atrasos Ponderados em Máquinas Paralelas.

PO - Pesquisa Operacional.

PVS - PVS-Studio, ferramenta de análise de código e busca de erros.

SA - *Simulated Annealing*.

SPT - *Shortest Processing Time* / Tempo de Processamento mais Curto.

Solver - Ferramenta de apoio utilizada na resolução de problemas, capaz de aplicar algoritmos dados certos modelos matemáticos.

UFFLP - Biblioteca de apoio a problemas de otimização combinatória.

WEDD - *Weighted Earliest Due Date*, método que ordena as tarefas em ordem crescente de tempo estimado de término.

WLTP - *Weighted Longest Processing Time* / Maior Tempo de Processamento Ponderado.

WSPT - *Weighted Shortest Processing Time* / Tempo de Processamento Ponderado mais Curto.

Capítulo 1

Introdução

Escalonamento pode ser dito como um processo que auxilia a tomada de decisões, objetivando a alocação de recursos para a execução de tarefas, em um ou mais períodos de tempo, e tendo como meta a obtenção de um ou mais objetivos (mono-objetivo ou multiobjetivo respectivamente) (Pinedo, 2016). Basicamente trata da atribuição de recursos para serem aplicados na execução de tarefas, por um ou mais períodos de tempo, buscando a eficiência dessa alocação, normalmente através do sequenciamento de tarefas e distribuição eficientes dos recursos dentro do tempo. (Brucker, 2007)

Ao tratar do escalonamento em ambientes de produção industriais podemos citar a execução de tarefas (onde cada tarefa possui características específicas) em uma ou mais máquinas, ou seja, onde existe um ambiente diversificado de processamento destas tarefas. Esses ambientes de processamento podem ter a disposição uma única máquina ou diversas máquinas, e neste caso, as máquinas podem divergir quanto a capacidade e/ou função. São inúmeros os tipos de problema que podem ser encontrados nesse tipo de ambiente (Brucker, 2007), o que por si só já apresenta um desafio, mas a resolução dos mesmos em ambiente computacional possui um desafio ímpar por se tratar de um conjunto especialmente difícil. A realização deste trabalho surge da busca por solucionar uma parte deste conjunto de problemas, oferecendo uma estratégia de solução eficiente e competitiva com outras propostas na literatura.

No restante deste capítulo é apresentado o problema abordado neste trabalho, juntamente com sua descrição e contexto (Seção 1.1), a motivação que inspirou seu desenvolvimento (Seção 1.2), os objetivos (Seção 1.4) e a organização da dissertação (Seção 1.5).

1.1 Contexto e descrição do problema

Problemas de escalonamento tem sido estudados há muito tempo, com especial atenção desde que os computadores puderam ser utilizados como ferramentas de apoio na modelagem e busca de solução (Brucker, 2007). Ferramentas computacionais são aplicadas diretamente através da implementação de algoritmos (e nesse ponto podemos citar heurísticas e meta-heurísticas) ou através de formulações matemáticas (executadas por softwares específicos, chamados *solvers*). Tais problemas são frequentemente vistos em ambientes de produção industriais ou em cenários onde é necessário realizar a alocação de recursos, normalmente escassos, para a execução de um dado conjunto de tarefas. Ao citar recursos, é preciso ter em mente que esta é apenas uma forma usada para nomear qualquer item cuja disponibilidade seja limitada, e que realize alguma ação dentro de um dado período de tempo.

A Figura 1.1 representa um exemplo de indústria que aborda problemas de escalonamento envolvendo máquinas paralelas, em especial, envolvendo situações com penalização pelo atraso (abordado nesta dissertação), onde existe uma determinada fábrica (Fábrica 3) responsável pela produção de impressoras e que, para a entrega de seu produto final necessita de produtos oriundos de outras duas fábricas, sendo uma delas fornecedora de bens duráveis (parte eletrônica das impressoras, dada pela Fábrica 1), e outra fábrica responsável por bens não duráveis (Fábrica 2 responsável pela fabricação de tonners e cartuchos utilizados pelas impressoras). Existindo a necessidade de ambos os tipos de produtos para a construção de seu produto final, a mesma usa de contratos para garantir as entregas, sendo geradas multas em caso de atraso (ou seja, uma penalidade que varia conforme o produto que sofreu o atraso na entrega). Escalonar corretamente os pedidos para as Fábricas 1 e 2 afeta diretamente nos lucros, uma vez que se houver um atraso no fornecimento de um dos produtos pode acarretar em grande prejuízo (tecnologia defasada, vendas perdidas, rompimento em contratos de fornecimento de produtos, entre outros).

É possível notar portanto que existe uma capacidade limitada de produção (no exemplo apresentado na figura anterior o recurso limitado pode ser visto como sendo a produção das fábricas 1 e 2), e que esta deve ser capaz de atender as demandas solicitadas. Nesse ponto a fabricação tanto dos bens duráveis como dos bens não duráveis pode ser visto como uma tarefa (ou *job*) a ser realizada, tendo cada uma delas características espe-

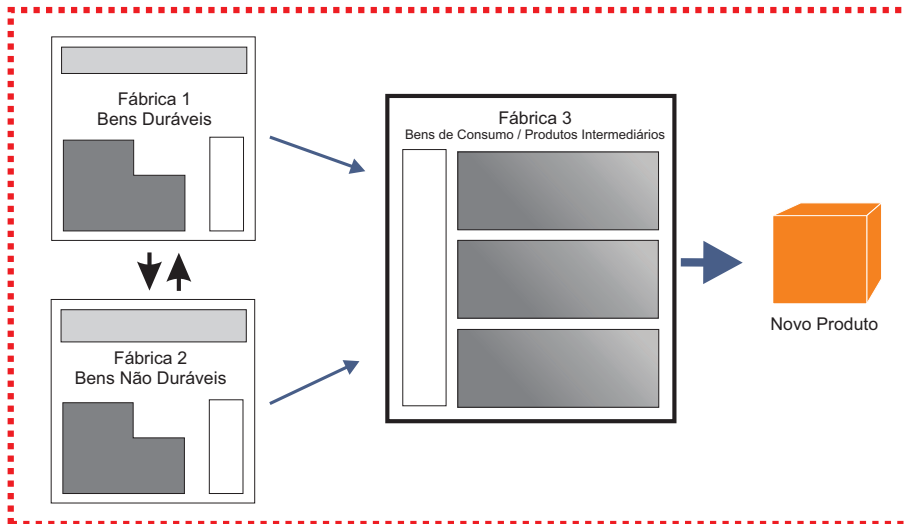


Figura 1.1: Exemplo de aplicação do problema envolvendo máquinas paralelas.

cíficas como tempo de produção (uma vez que produtos eletrônicos diferentes tem tempos diferentes de fabricação, assim como diferentes tipos de tonner e cartuchos). Outro ponto comum a todas as tarefas é que as mesmas possuem uma data estimada de entrega que varia conforme a tarefa (a solicitação de certos tipos de eletrônicos, devido a possíveis problemas de montagem devem ser solicitados em tempo mais curto), bem como, cada um destes materiais possui em seu contrato uma multa diferente (pois determinados tipos de material podem impactar mais ou menos negativamente na produção das impressoras). Dessa forma, resumidamente, a situação que temos é, dado um conjunto de tarefas, para serem executadas em um conjunto de máquinas paralelas, buscar reduzir a penalidade resultante de atrasos na execução. Nesse ponto, temos três características básicas comuns a todas as tarefas: um tempo de produção, uma data esperada de entrega e uma penalidade em caso de atraso na produção. Uma forma simplificada desse modelo é representado pela Figura 1.2.

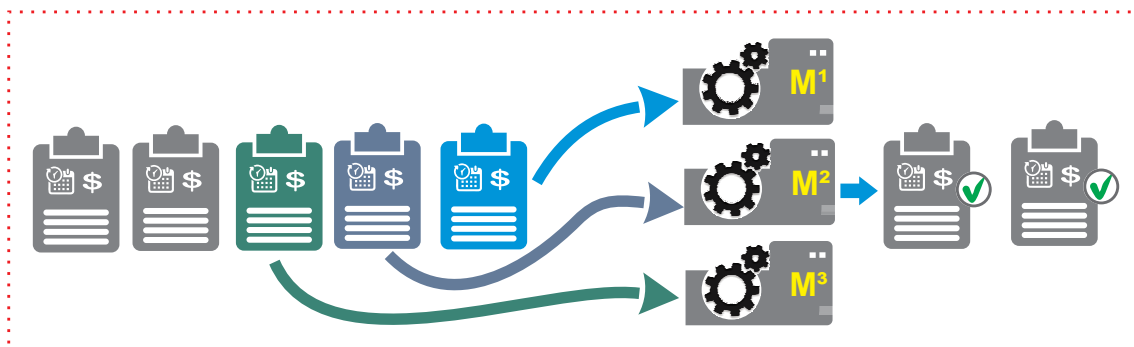


Figura 1.2: Exemplo de aplicação do problema envolvendo máquinas paralelas.

Em problemas de escalonamento as tarefas podem possuir outras características tais como ordem de precedência, data/hora mínima para início, data/hora máxima de término, tempo de preparação, critérios de urgência, além de características específicas que gerem dependência das máquinas, onde uma tarefa deve ser executada por uma máquina específica, ou passar por um grupo de máquinas em uma determinada ordem. Em outras palavras, são inúmeras situações que podem ser tratadas e representadas.

Este trabalho aborda a minimização de atraso total ponderado (*weighted tardiness*) (Lee, 2013), mais especificamente, a minimização do atraso ponderado em máquinas paralelas idênticas (Brucker, 2007). Este tipo de escalonamento busca indicar a ordem de alocação de tarefas para processamento em máquinas paralelas, onde o tempo de processamento de cada tarefa, bem como sua penalidade e data esperada de término são arbitrárias (Sen et al., 2003). O objetivo é reduzir essa penalidade total, indicando a distribuição de tarefas para cada máquina, bem como a ordem de execução das mesmas, sendo este um quadro comum em ambientes industriais.

Além da grande relevância teórica, a importância do tema pode ser notada ao avaliar o contexto de uma indústria, que gera um determinado conjunto de produtos a partir de um certo volume de insumos. As máquinas utilizadas na produção da indústria são um recurso escasso e nem sempre uniforme (um ambiente industrial pode possuir equipamentos com diferentes configurações, velocidades, capacidade e desempenho), cuja utilização, se feita de forma adequada, pode acarretar na minimização de perdas e/ou maximização de lucros, uma vez que pode evitar tanto a sobrecarga quanto a subutilização de equipamentos. Do mesmo modo o problema permanece relevante após a fase de fabricação, uma vez que os produtos gerados precisam ser distribuídos de forma eficiente, reduzindo custos com armazenamento, transportes e frete, uma vez que os caminhões utilizados na distribuição podem ter características distintas (capacidade de carga, velocidade, autorização de acesso a determinados locais) e gerar custos distintos (autonomia, pedágios e outros). Avaliando diretamente, os problemas que envolvem o escalonamento estariam presentes em diversos locais de um mesmo setor. Uma amostra deste modelo pode ser visto na Figura 1.3.

Tratando mais precisamente da etapa de produção, se o uso das máquinas no setor produtivo for feito de forma inadequada, acarretará em prejuízos ou desperdício de recursos: encargo com funcionários, horas adicionais de produção, aumento no consumo de energia elétrica (Liu et al., 2014)(Fang and Lin, 2013), desgaste de máquinas e atrasos

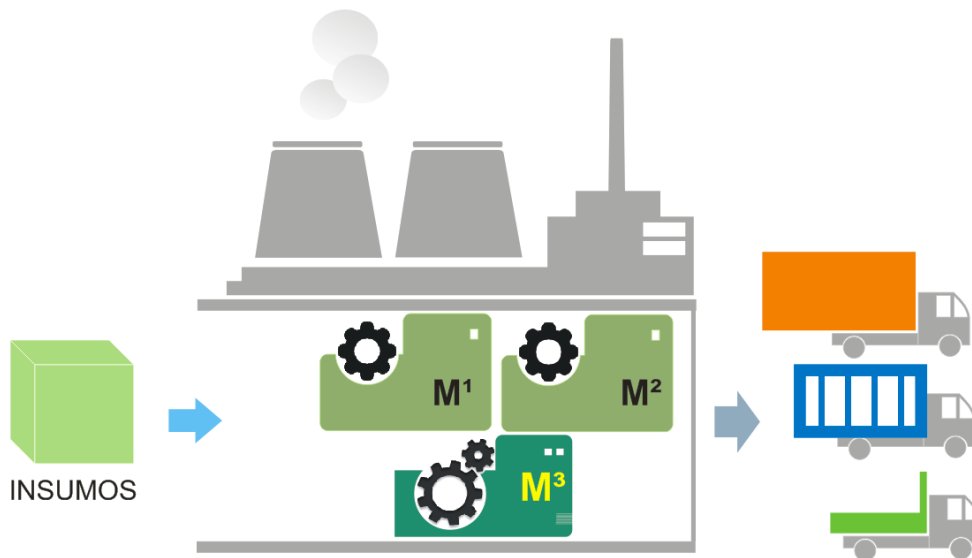


Figura 1.3: Exemplo de operação em uma pequena indústria.

na produção, caso este que pode gerar prejuízo financeiro maior, uma vez que determinadas indústrias possuem contratos com determinação e multa por descumprimento de prazos (Brucker, 2007).

1.2 Motivação

Abordar problemas com alta relevância teórica e aplicação prática, por si só já representam motivação para o trabalho. Problemas de escalonamento podem ser categorizados como tal, uma vez que realizar a alocação de recursos é uma tarefa bem abstrata, cuja aplicação pode ser vista em diferentes áreas (ambientes de produção da indústria, execução de projetos na seleção de profissionais para atribuição de atividades, seleção de computadores ou processadores para execução em ambientes multitarefa ou ambientes distribuídos), resumindo, pode ser aplicado para quaisquer ambientes onde exista a necessidade de executar tarefas na busca de um determinado objetivo (maximizar lucros, minimizar perdas, eliminar ociosidade em máquinas entre outros).

Outra grande motivação está na oportunidade de abordar um tema amplo e desafiador, dada sua dificuldade computacional de resolução (por fazer parte de uma generalização do problema $1||\sum T_j$ (notação melhor explicada no Capítulo 2, Seção 2.2), provado ser um problema NP-Difícil por Lenstra et al. (1977), Lawler (1977) e por Du and Leung (1990)).

De forma a melhor elucidar o problema, a Figura 1.4 apresenta uma instância contendo seis tarefas para serem executadas. Cada tarefa j possui um tempo de processamento (p_j), data de término sugerida (d_j) e penalidade caso ocorra atraso (w_j). Neste exemplo não são citadas regras de precedência das tarefas (sendo apenas um modelo simplificado), portanto, o problema principal a ser resolvido está centrado mais precisamente em identificar a sequência de execução das tarefas, de modo que não se tenha atraso no processamento, ou que este seja minimizado, reduzindo portanto a penalidade.

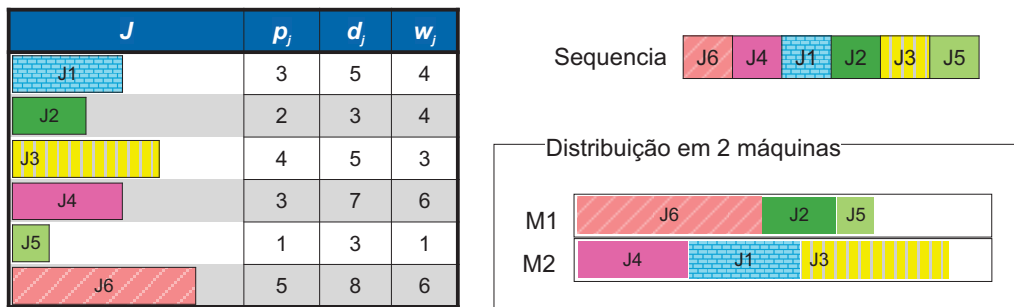


Figura 1.4: Instância com 6 tarefas para 2 máquinas.

Para o processamento das tarefas apresentadas na Figura 1.4, existem diferentes abordagens e resoluções, que podem variar de acordo com a quantidade de máquinas disponíveis e configurações destas máquinas. O problema poderia ser resolvido utilizando uma única máquina onde, nesse cenário, a quantidade de possíveis soluções para serem verificadas seria dada por uma combinatória, resultando em um total de $n!$ (n fatorial) soluções (onde n seria a quantidade de tarefas), ou seja, aproximadamente 720 soluções possíveis, que deveriam ser analisadas, e então selecionada a melhor dentre elas (a que gera menor atraso ou penalidade).

O cenário pode ser um pouco mais complexo ao tratar de mais de uma máquina. Neste caso, o problema vai depender da configuração das máquinas, que podem ser idênticas (mesma configuração), uniformes (velocidades de processamento diferentes entre si, mas mesma velocidade independente da tarefa) ou não relacionadas (onde o tempo de processamento é diferente para cada máquina e tarefa). Em casos de execução em mais de uma máquina, a quantidade de soluções possíveis aumenta consideravelmente, passando para $(n!)^i$ soluções possíveis (sendo i a quantidade de máquinas). Para o nosso exemplo apresentado na Figura 1.4, um total de $(6!)^2 = 518400$ soluções (considerando apenas duas máquinas distintas).

Dado o grande número de soluções possíveis, o uso de meta-heurísticas se torna mais

atrativo e dependendo do cenário a única solução plausível, por avaliar pequenos conjuntos de soluções, selecionando algumas delas, chamadas de **ótimos locais**. Dentre estas soluções, é obtida a melhor delas, porém, justamente por não avaliar todas as soluções possíveis, pode não retornar a solução ótima desejada para a solução do problema. Este acaba sendo outro motivador, uma vez que visa utilizar métodos diversos, tentando obter uma melhor solução.

1.3 Hipótese

É possível resolver de forma eficiente o problema da minimização do atraso total ponderado, e obter bons resultados utilizando uma estratégia híbrida exato-heurística, com um método aproximado fornecendo um espaço de busca para uma formulação de programação linear inteira.

Tal hipótese pode ser comprovada ao avaliar a qualidade dos resultados obtidos, atingindo as soluções ótimas ou obtendo limites superiores melhores, além de possibilitar a obtenção de resultados ao resolver o problema para instâncias bem maiores em um menor tempo de execução.

1.4 Objetivos

O objetivo desta dissertação é o de elaborar uma estratégia algorítmica híbrida exata-heurística que seja competitiva frente a outras estratégias existentes na literatura, trazendo bons resultados para o problema de escalonamento em máquina paralelas idênticas, com minimização do atraso ponderado, de tal forma a se conseguir resolver grandes instâncias em um menor tempo computacional e obter melhores limites superiores, quando não as soluções ótimas. Para que este objetivo geral seja alcançado, foram definidos os seguintes objetivos específicos:

- a. Avaliar, propor e aplicar melhorias em método híbrido exato-heurístico construído.
- b. Avaliar os métodos heurísticos mais recentes e de melhor desempenho para problemas similares, de modo a extrair características que possam refinar o método híbrido em estudo.

- c. Avaliar as formulações matemáticas e métodos exatos existentes para o problema, de modo a também incorporar no método híbrido.
- d. Definir uma estratégia para análise empírica e de desempenho dos métodos elaborados, de tal forma a tornar mais rica a apresentação e discussão dos resultados obtidos.

1.5 Organização do trabalho

A presente dissertação está dividida em cinco capítulos. No Capítulo 1 é mostrada uma contextualização do problema, indicando a importância e seus desafios. Também são indicados os objetivos, a motivação e a organização desta dissertação. No Capítulo 2 estão dispostos os conceitos e definições necessários a compreensão do documento. No Capítulo 3 o assunto do trabalho é abordado de forma mais direta, incluindo trecho específico por indicar trabalhos relacionados ao tema. No Capítulo 4 é apresentado o método aplicado na solução do problema. No Capítulo 5 são apresentados os resultados obtidos quando da aplicação do método aplicado, além de retratar dados relevantes acerca dos resultados e aplicação dos procedimentos. No Capítulo 6 são expostas as considerações finais do trabalho, bem como recomendações sobre pesquisas futuras.

Capítulo 2

Fundamentação Teórica

De forma a elucidar a área, tema e problema abordados nesta dissertação, é necessário que seja realizada uma breve introdução aos principais termos, conceitos e definições. As seções seguintes tratam sobre Otimização Combinatória, Métodos Exatos, Métodos Aproximados e Híbridos, além de uma introdução acerca de Problemas de Escalonamento.

2.1 Otimização combinatória

Otimização combinatória é a área de Ciência da Computação que objetiva resolver problemas onde é necessário selecionar um conjunto discreto e finito de dados, a partir de um conjunto de soluções possíveis, satisfazendo um conjunto de critérios.

Tais problemas podem ser formulados utilizando de programação inteira (nesse caso, resolvidos através de Algoritmos de Programação Inteira) ou através de heurísticas e meta-heurísticas (Goldberg et al., 2016). Estes problemas estão relacionados com a busca de uma solução ou mais soluções, que permitam minimizar ou maximizar o valor de uma determinada função objetivo, respeitando um determinado conjunto de restrições, e tendo como base uma quantidade limitada de recursos (máquinas, pessoas, automóveis, e outros) (de Freitas Rodrigues, 2009) (Martí and Reinelt, 2011).

Problemas de otimização combinatória podem ser resolvidos de diferentes formas, variando conforme sua complexidade, recursos, tempo, espaço de soluções, restrições entre outros fatores. Dentre as formas de resolução, podemos citar o teste mais básico que avaliaria todo o espaço de soluções disponíveis (também chamado de força-bruta), formulações matemáticas, algoritmos de busca local, programação dinâmica, algoritmos

genéticos e meméticos, evolução diferencial dentre outras.

Problemas de otimização podem ser vistos e abordados em diferentes ambientes e setores (Müller-Merbach, 1981). Na tecnologia ao efetuar a seleção de profissionais para atendimento e resolução de problemas específicos, na distribuição de dados dentro de uma rede de computadores, na replicação de conteúdo através de uma *CDN* e no escalonamento de processadores em computadores. Na logística ao tratar de roteamento de veículos, sequenciamento distribuição de tarefas em máquinas, empacotamento de caixas e objetos em contêineres. Na produção em operações de corte de placas ou madeira, seleção de embalagens, armazenamento em galpões. Também podem ser vistos na educação e saúde, na criação de quadros de horário, distribuição de professores e sequenciamento de genes de DNA.

A complexidade destes problemas faz com que seja necessária a seleção correta de algoritmos eficientes e escaláveis. Uma forma de avaliar a eficiência de um algoritmo pode ser realizada ao avaliarmos a complexidade do mesmo. A complexidade de algoritmos é a análise que leva em consideração o tempo de execução de um dado algoritmo, baseado no tamanho da instância de entrada. Ao avaliar a ordem de crescimento no tempo de execução de um algoritmo baseado no tamanho da entrada, estamos estudando a eficiência assintótica do algoritmo (Cormen et al., 2002). Para representar essa eficiência assintótica existem notações - Notação Θ , Notação O (*big "O"*), Notação o (*little "o"*) e Notação Ω -, cada uma representando um limite diferente de crescimento. Um algoritmo cujo tempo de execução não varia conforme o tamanho da entrada, pode ser indicado com complexidade $O(1)$, enquanto outros, cujo tempo de execução varia conforme a instância de entrada, podem ser indicados com complexidade $O(n^k)$, onde n indicaria o tamanho da instância de entrada e k uma constante.

Se existe um algoritmo de tempo polinomial para resolver um dado problema (indicando que é um problema tratável), podemos dizer que este algoritmo está na classe P . Do mesmo modo, quando for possível verificar uma solução proposta para um dado problema em tempo polinomial no tamanho da entrada, é possível dizer que o problema está na classe NP e a solução proposta é chamada de **certificado**. Para que o problema esteja em NP o tempo de verificação do certificado precisa ser polinomial no tamanho da entrada e do certificado. Assim sendo, todo problema em P está automaticamente em NP . São chamados de NP -Completo aqueles problemas que estão em NP e que são conhecidamente

difíceis (Cormen, 2017).

Em questões que envolvam a solução de problema de natureza combinatória, a complexidade é um fator basilar, devido a enormidade de possibilidades a serem averiguadas na busca de uma solução ótima. Problemas de escalonamento fazem parte do conjunto de problemas de natureza combinatória, e muitas vezes, exigem maior dedicação em sua solução devido a sua complexidade e características. As próximas seções do capítulo descrevem tais problemas e algumas das abordagens utilizadas na solução dos mesmos.

2.2 Problemas de escalonamento

Os problemas de escalonamento (também chamados de programação de produção) são aqueles cujo objetivo é o planejamento e o controle da execução de tarefas de produção e serviços. Consistem em alocar a execução de tarefas de forma a realizar seu processamento de forma mais eficiente, atendendo a um dado grupo de critérios. Essa alocação otimizar uma dada **função objetivo**, que pode ser atingido através do sequenciamento de um conjunto de operações por um conjunto de recursos, executados ao longo do tempo. Além do objetivo comum, podem existir regras específicas a serem atendidas (**restrições**), que avaliam se uma solução é válida ou não, além de permitir avaliar penalizações ou perdas em casos específicos. Deve levar em consideração também o **ambiente** onde serão executados, tendo em vista a influência direta na qualidade da solução. As funções objetivo são funções matemáticas que determinam a qualidade da solução, em função de um conjunto de variáveis de decisão, e que almejam a minimização ou maximização deste do valor obtido pela função matemática. Em casos da busca pela minimização, dentro de diversos outros problemas, é possível àqueles relacionados a data de entrega de uma data produção, onde podem ser geradas multas por atraso na produção, ou aluguel adicional de espaços em caso de antecipação. Já ao buscar objetivos como de maximização, podemos estar tratando de aumento de lucratividade baseado em custos de produção, vendas, ou períodos de oferta (Brucker, 2007). Algumas das características dos problemas de escalonamento são apresentadas na Tabela 2.1:

Tabela 2.1: Símbolos usados em problemas de escalonamento.

Termo	Símbolo
<i>Jobs</i> (Tarefas)	J_j
<i>Machines</i> (Máquinas)	i
<i>Processing Time</i> (Tempo de Processamento)	p_j
<i>Processing Time</i> (Tempo de Processamento dependente da máquina)	p_{ij}
<i>Release Date</i> (Data de disponibilidade)	r_j
<i>Due Date</i> (Data Prevista de Término)	d_j
<i>Weight</i> (Peso, Prioridade ou Importância)	w_j
Velocidade de Processamento da máquina i	s_i
Velocidade de Processamento da tarefa j variando conforme a máquina i	s_{ij}
Tempo de completude de uma tarefa j	C_j
Maior tempo de completude das tarefas (<i>Makespan</i>)	C_{max}
Latência de uma tarefa j (dado por $C_j - d_j$)	L_j
Maior tempo de latência (dado por $\max(L_1, L_2, \dots, L_n)$)	L_{max}
Indicador de atraso para uma tarefa j (valor 1 caso $C_j > d_j$, ou 0 caso contrário)	U_j
Atraso de uma dada tarefa j (<i>tardiness</i> , dado por $\max(0, L_j)$)	T_j
Antecipação de uma dada tarefa j (<i>earliness</i> , dado por $\max(d_j - C_j, 0)$)	E_j

Dada a grande variedade de problemas de escalonamento, foi criada uma notação especial para auxiliar na classificação desses problemas. Esta representação chamada **Notação de Três Campos**, proposta por [Graham et al. \(1979\)](#), foi criada para organizar e categorizar os diferentes problemas de escalonamento. A Notação de Três Campos permite definir um problema, classificando-o por seu ambiente de máquinas, características e restrições e a função objetivo. Tem como símbolos os itens apresentados na Tabela 2.2:

Tabela 2.2: Notação de três campos - problemas de escalonamento.

Símbolo	Descrição
α	Descreve o ambiente de máquinas
β	Descreve as características especiais do processamento
γ	Descreve a função objetivo (a ser maximizado ou minimizado)

Ambiente de máquinas (α)

São tratados problemas envolvendo ambientes com uma só máquina (chamado de ambiente mono processado ou *single machine*) ou ambientes com mais máquinas (máquinas paralelas ou *parallel machines*). Ambientes com máquinas paralelas, por poderem possuir características distintas, tratam três modelos diferentes:

- a) **Máquinas paralelas idênticas** (P_m), onde cada máquina possui a mesma velocidade (dada como 1), sendo assim, cada tarefa j é executada dentro de seu tempo de processamento p_j indiferente da máquina onde é processada.
- b) **Máquinas paralelas uniformes** (Q_m), onde existem máquinas distintas, com velocidades de execução distintas (diferentes s_i), porém, não levam em consideração a tarefa que está sendo executada, ou seja, não variam de acordo com a tarefa (processamento de uma tarefa j em uma máquina i , é dado por $p_{ij} = p_j/s_i$).
- c) **Máquinas paralelas não relacionadas** (R_m), onde cada máquina i executa cada uma das tarefas j com velocidades que podem ser diferentes, ou seja, a velocidade de processamento de uma tarefa pode variar conforme a máquina onde está sendo processada, assim como, cada máquina i pode processar cada tarefa j com velocidades distintas. Neste caso, uma tarefa j , executando em uma dada máquina i , possui uma velocidade de processamento s_{ij} , portanto, seu tempo de processamento é dado por $p_{ij} = p_j/s_{ij}$.

Flowshop (F_m)

Em um ambiente *Flowshop*, temos m máquinas em sequência. Todas as tarefas seguem a mesma sequência de máquinas. Ao concluir sua execução em uma dada máquina,

a tarefa entra em uma fila, aguardando a execução na máquina seguinte, até que tenha passado por todas as máquinas.

***Jobshop* (J_m)**

No *Jobshop* existem m máquinas, porém, diferentemente do *Flowshop*, a sequência é diferente para cada uma das tarefas, e pode ser iniciada em paralelo (desde que não haja outra tarefa sendo executada na mesma máquina).

***Openshop* (O_m)**

No ambiente *Openshop* todas as tarefas, devem ser executadas em todas as máquinas, porém, não existe uma sequência definida para ser executada.

Características de processamento (β)

As características de processamento poderiam ser compreendidas como restrições a serem aplicadas. Cada uma leva em consideração um determinado ambiente ou motivações, e fazem com que a forma de resolução de um determinado problema seja modificado. Estas características podem ser aplicadas em conjunto, sendo comum em ambientes de produção esse aspecto. Podem ser vistas como características:

- a) **Data de disponibilidade (*release date*)** - indica que uma dada tarefa j não pode iniciar antes do tempo r_j . Quando esta característica não é indicada, as tarefas podem ser processadas a partir do tempo zero.
- b) **Sequência dependente de tempos de preparação (*Sequence dependent setup time*)** - indica que existe um tempo de preparação para a tarefa k , que depende da tarefa anterior j , sendo esse tempo de preparação dado por s_{jk} . Esse tempo de *setup* ou tempo de preparação, pode variar conforme a máquina (normalmente em ambientes de máquinas paralelas não relacionadas). Nestes casos existe a presença dessa variação, dada por s .
- c) **Preempção (*preemption* - *prmp*)** - indica que as tarefas podem ser interrompidas a qualquer momento e substituída por outra na máquina, podendo a execução da tarefa ser retomada. Ao ser retomada a execução, o tempo de processamento a ser considerado é apenas o restante que faltou ser executado. Caso não seja apresentada essa característica na formulação, é subentendido que não é aceita a preempção, ou seja, a tarefa deve iniciar e terminar sem interrupções.

- d) **Restrições de precedência** (*precedence constraints - prec*) - Indica que existem relações de precedência entre as tarefas, isto é, para uma dada tarefa poder ser executada, a mesma não deve ter nenhuma dependência, ou seja, as tarefas que deveriam ser executadas antes da tarefa corrente já devem ter sido concluídas. Normalmente é dado por um Grafo Acíclico Direcionado. Caso não seja indicado no problema, considera-se que não existe esta restrição.
- e) **Restrições de elegibilidade da máquina** (*machine eligibility restrictions - M_j*) - é aplicada em casos onde determinadas tarefas precisam ser executadas por um determinado grupo de máquinas. Normalmente esta restrição é aplicada em casos onde o tipo da tarefa necessita ser executada em máquinas com características específicas. Caso M_j (um subconjunto de máquinas) não seja informado, todas as tarefas podem ser executadas por qualquer uma das máquinas disponíveis.
- f) **Permutação** (*permutation - $prmu$*) - indica que a ordem das tarefas no sistema deve seguir em conformidade com uma fila, isto é, padrão FIFO (*First-In-First-Out*). Caso não seja informado, qualquer ordem pode ser aplicada.
- g) **Bloqueio** (*blocking - block*) - indica que existe uma relação entre determinadas tarefas, isto é, uma dada tarefa só pode ter a execução em uma máquina concluída, se a máquina seguinte por onde a mesma deverá ser processada, já estiver liberada. Aplicado em casos onde existe a necessidade de processar as tarefas em mais de uma máquina.
- h) **Sem espera** (*no-wait - nwt*) - Indica que não pode haver tempo ocioso para uma tarefa entre sua execução em mais de uma máquina. Aplicado também em casos onde existe a necessidade de execução das tarefas em mais de uma máquina.

Funções objetivo (γ)

São as metas a serem atingidas em problemas de escalonamento. A partir de um dado ambiente, e um conjunto de regras, as funções objetivo indicam que meta deseja ser alcançada, sendo esta meta a minimização ou maximização de um determinado atributo das tarefas, máquinas e o processamento. Cabe ressaltar que as funções objetivo, para que sejam cumpridas, por vezes é necessário realizar a ordenação e distribuição das tarefas

pelas máquinas (a distribuição em caso de máquinas paralelas). Algumas das funções objetivo podem ser vistos abaixo:

- a) Minimizar o tempo total (*Makespan* - C_{max}).
- b) Minimizar a latência máxima (*Lateness* - L_{max}).
- c) Minimizar o tempo total de processamento ($\sum C_j$).
- d) Minimizar o número de tarefas tardias ($\sum U_j$).
- e) Minimizar o atraso total ($\sum T_j$).
- f) Minimizar o tempo total de processamento ponderado ($\sum C_j w_j$).
- g) **Minimizar o atraso total ponderado** ($\sum T_j w_j$).
- h) Minimizar o número de tarefas tardias ponderadas ($\sum U_j w_j$).
- i) Minimizar a antecipação total das tarefas ($\sum E_j$).
- j) Minimizar a antecipação ponderada de tarefas ($\sum E_j w_j$).
- k) Minimizar a antecipação e atraso totais ($\sum E_j + \sum T_j$).
- l) Minimizar a antecipação e atraso ponderados ($\sum E_j \alpha_j + \sum T_j \beta_j$, com α e β sendo, respectivamente, o peso para a antecipação e do atraso de uma dada tarefa j).

2.3 Métodos de resolução

Nesta seção são abordados alguns dos métodos aplicados na solução de problemas de escalonamento, tratando àqueles capazes de obter a melhor solução possível ao custo de uma maior porção de tempo (programação linear inteira ou métodos exatos), mas também métodos capazes de retornar boas soluções em uma pequena fração de tempo (métodos aproximados).

2.3.1 Programação linear inteira

Métodos exatos são aqueles capazes de retornar uma solução garantindo se a mesma é a solução ótima global ou não. Uma solução é dita como ótima global se dado um contexto e critério de otimização, não existir nenhuma outra solução com resultado melhor do que ela. Na abordagem exata, pode ser utilizado um modelo de Programação Linear (Prado, 2010), uma técnica de otimização utilizada para encontrar o ótimo global, seja ele máximo ou mínimo, em situações nas quais temos diversas alternativas de escolha sujeitas a algum tipo de restrição ou regulamentação. É possível indicar também a utilização de modelos matemáticos que podem ser executados através do uso de *solvers*, além de métodos de enumeração e algoritmos específicos.

A implementação mais comum de um método exato consiste na enumeração explícita, também chamada de "força bruta", onde é avaliado e enumerado todo o conjunto de soluções possíveis, retornando a melhor solução contida neste conjunto, porém, existem outras abordagens, como os métodos de enumeração implícita, que conseguem retornar a solução ótima sem a necessidade de avaliar todo o conjunto de soluções. Os métodos de enumeração, amplamente utilizados na resolução de problemas, são aqueles que realizam uma busca no conjunto de prováveis soluções do problema, avaliando todo o conjunto de soluções disponível e retornando uma solução ótima. Dentre os métodos de enumeração podemos citar o *Branch-and-Bound*.

2.3.1.1 Método *Branch-and-Bound*

Algoritmos *Branch-and-Bound* fazem uso de uma estratégia de enumeração implícita. Baseiam-se na ideia de construção de uma enumeração "inteligente" que elege pontos candidatos a solução ótima de um problema. O termo *branch* refere-se ao fato do método particionar o espaço de soluções, enquanto que o termo *bound* indica que a prova de otimalidade é dada através da utilização de limites, obtidos ao longo da enumeração (Goldbarg and Luna, 2005).

Como forma de detalhar o método, assume-se que P representa um problema de minimização para ser resolvido, e que S' representa um subconjunto, parte do conjunto S de soluções factíveis do problema P . Então, é definido um limite inferior para o problema em sua forma relaxada, isto é, ignorando as restrições de integralidade, e resolvendo o

problema utilizando algum método de programação linear contínua (Simplex, Dual Simplex). Caso a solução obtida seja inteira, isto é, todas as variáveis de decisão obtidas pelo método de programação linear contínua sejam inteiras, o problema é tido como resolvido, caso contrário, tem-se início o processo de **branching** ou **ramificação** (Brucker, 2007).

No processo de *branching*, S é substituído por problemas menores S_i ($i = 1, 2, \dots, r$), sendo $\bigcup_{i=1}^r S_i = S$, e cada S_i . Sendo representado por uma estrutura em árvore, chamada árvore de recursões (*branching tree*), S atua como raiz desta árvore, sendo cada um dos problemas menores S_i um ramo originado a partir da raiz. De forma semelhante, cada um dos problemas S_i pode gerar novos problemas (ou ramos) na árvore. Para cada novo ramo adicionado a árvore, são incorporadas restrições, sobre uma variável de decisão. Problemas de otimização discreta criados no processo de *branching* são chamados de subproblemas (Brucker, 2007). Na Figura 2.1 é apresentado um exemplo de árvore de *branch-and-bound*, mostrando as podas e ramificações.

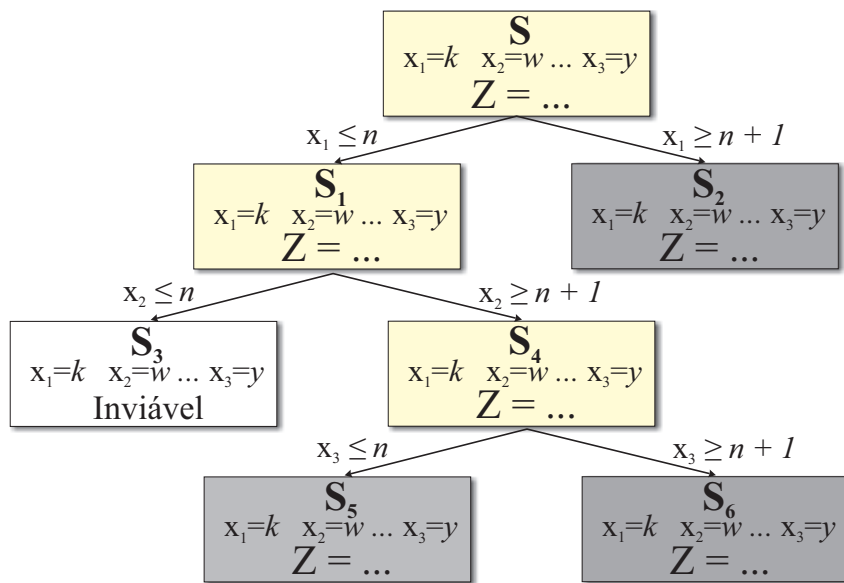


Figura 2.1: Exemplo de árvore de *branch-and-bound*.

Ao passo que o processo de *branching* vai sendo executado, também é realizado o processo de **bounding** (ou definição de limites), sendo este o passo responsável por dar continuidade ou interromper a execução do método. Considere U o limite superior de P . Neste ponto é possível que para um subproblema criado, a restrição adicionada o torna inviável, dessa forma, não existem novas ramificações a partir deste nó e o mesmo é podado. Em outros casos, o limite inferior (*lower bound*) de um subproblema é maior ou igual a U , então o mesmo não sofre novas ramificações por não poder retornar soluções

melhores para P . Caso ainda exista alguma variável de decisão contínua, o processo de *branching* tem continuidade, criando novos subproblemas, com novas restrições. Quando a solução obtida para o nó é inteira, não há novas ramificações a partir deste nó, sendo a solução deste nó, a melhor solução inteira no momento, sendo seu valor atribuída para U . O processo é concluído quando todos os subproblemas tiverem sido resolvidos, isto é, quando não for possível gerar novos ramos. Neste ponto, o valor de U é retornado, sendo ele a solução ótima para o problema. A Figura 2.2 apresenta um exemplo de resolução do método *branch-and-bound* através do modo gráfico, enquanto que a Figura 2.3 apresenta a mesma árvore após o processo de "*branch*".

O cálculo do valor de U portanto atua de forma importante no processo, uma vez que evita que sejam avaliadas soluções incapazes de melhorar a solução vigente do problema. Em alguns casos, o uso de heurísticas é realizado para a geração do valor inicial de U .

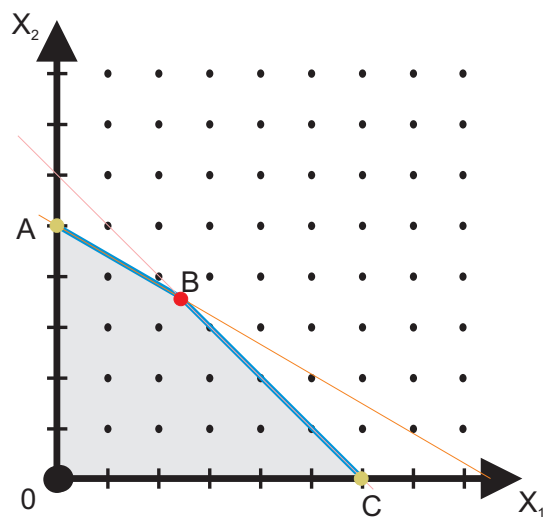


Figura 2.2: Exemplo de solução gráfica.

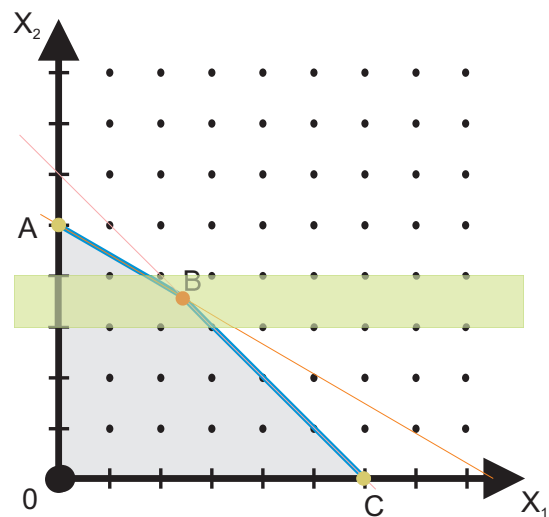


Figura 2.3: Resultado do *branch*.

A técnica *branch-and-bound* possui ampla aplicabilidade, podendo passar por modificações e adaptações, além de permitir ser ajustada para atuar com diferentes estratégias de implementação. As modificações podem ser realizadas nas técnicas de desenvolvimento da árvore de enumeração (busca em profundidade, busca em largura, variantes híbridas), técnicas de formação da árvore (variante de [Dakin \(1965\)](#), variante de [Land and Doig \(1960\)](#), Método das penalidades de [Beale and Small \(1965\)](#) entre outras), e ainda ter variações nas técnicas complementares para obtenção de limites (através de relaxação Lagrangeana, cortes e algoritmos heurísticos através do uso de meta-heurísticas). Uma versão resumida do algoritmo *branch-and-bound* é apresentado no Algoritmo 1.

Algoritmo 1 Algoritmo *branch-and-bound*.

```
 $L \leftarrow S;$ 
 $U \leftarrow$  Valor da melhor solução corrente (upper bound)
 $melhorAtual \leftarrow$  Melhor solução corrente
enquanto  $L \neq \emptyset$  faça
  Seleciona um nó de ramificação  $k$  de  $L$ 
  Remove  $k$  de  $L$ 
  Gerar os filhos  $C_i$  (para  $i = 1, 2, \dots, n_k$ )
  Calcular os limites inferiores correspondentes  $LB_i$ 
  para  $i \leftarrow 1$  até  $n_k$  faça
    se  $LB_i < U$  então
      se  $C_i$  é uma solução inteira então
        se  $LB_i$  é uma solução melhor que  $U$  então
           $U \leftarrow LB_i$ 
           $melhorAtual \leftarrow$  Solução correspondente de  $C_i$ 
        fim se
      senão
        Adiciona  $C_i$  para  $L$ 
      fim se
    fim se
  fim para
fim enquanto
```

Além do método *branch-and-bound*, é possível fazer uso de outros métodos exatos tais como: programação dinâmica, métodos de corte, geração de colunas, *branch-and-cut* (método que combina o *branch-and-bound* e planos de corte), *branch-and-price*, *branch-cut-and-price*, entre outros (de Freitas Rodrigues, 2009).

2.3.1.2 Planos de corte

Os planos de corte de Gomory (1960) atuam eliminando pontos contínuos extremos, sem remover soluções inteiras factíveis de um determinado problema. Iniciam seu processamento através de uma relaxação linear do problema e vão adicionando novas desigualdades de forma a remover soluções inválidas para o problema. De forma geral, pode ser definido como um método que atua adicionando uma desigualdade, e esta é satisfeita por todo o conjunto de soluções viáveis, e dessa forma, acabam por cortar soluções inviáveis (valores contínuos).

O processo de adição de cortes de forma a não podar soluções viáveis do problema pode ser visto na Figura 2.4, onde uma ponto B indica a solução ótima relaxada para um dado problema P . Pela adição de uma desigualdade c , é gerado um corte, removendo a

solução contínua do problema e obtendo uma nova solução.

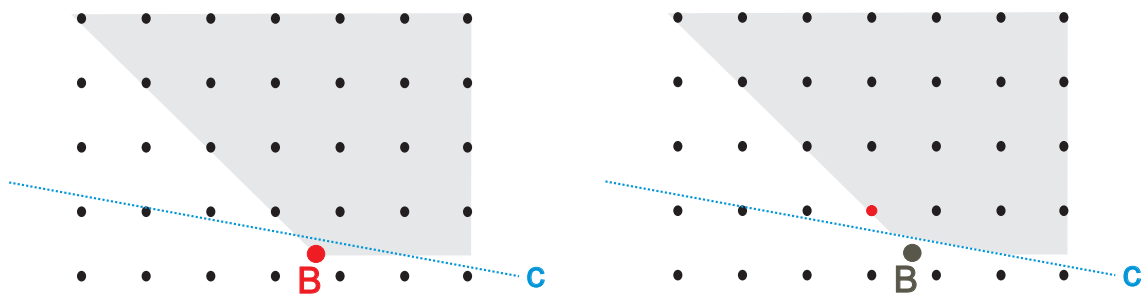


Figura 2.4: Conjunto de soluções antes e depois do corte.

A aplicação destes cortes repetidas vezes garante obter a solução ótima do problema, uma vez que remove todas as soluções inviáveis do problema e vai buscando a melhor solução após cada corte. Uma versão resumida do plano de cortes pode ser vista no Algoritmo 2.

Algoritmo 2 Algoritmo de Plano de Cortes.

$P_0 \leftarrow$ Formulação inicial do problema (relaxada)
 $t \leftarrow 0$
repita
 $z^t \leftarrow$ Solução da formulação P_t
se z^t não é uma solução inteira **então**
 $d \leftarrow$ desigualdade para eliminar z^t do problema
 $P_{t+1} \leftarrow P_t$ acrescida da desigualdade d
 $t \leftarrow t + 1$
fim se
até z^t seja uma solução inteira

2.3.1.3 Método *Branch-and-Cut*

O método *branch-and-cut* atua combinando um algoritmo *branch-and-bound* com a técnica de planos de corte de Gomory. Proposto por Padberg and Rinaldi (1987), o *branch-and-cut* teve sua aplicação inicialmente realizada para solucionar o problema do Caixeiro Viajante. Com a combinação destes dois métodos, o tempo necessário para resolver instâncias maiores é reduzido. Nesse algoritmo, é inserida em uma versão inicial relaxada do problema, um conjunto de restrições válidas, isto é, restrições que eliminam apenas as soluções contínuas, mantendo todo o conjunto de soluções válidas do problema.

Para cada subproblema da árvore de ramificação o *branch-and-cut* aplica um algoritmo de plano de corte para gerar um limite inferior, ou seja, para cada subproblema

gerado na árvore, são introduzidas restrições à formulação, reduzindo o número de ramificações gerado. Aliado a este fato, podem ser adicionados cortes controlados por algum critério. Este critério pode ser definido de acordo com o problema. Com a aplicação destes critérios adicionais, juntamente com o fato de serem considerados cortes dos estágios de enumeração, permite que o *branch-and-cut* encontre a solução ótima de um problema sem que seja necessário adicionar todas as restrições do problema original (Rodrigues, 1996).

Podem ser incorporadas também técnicas de pré-processamento, onde o objetivo é melhorar a relaxação linear através de considerações algébricas, onde as variáveis e restrições são analisadas evitando redundância e variáveis tem valores fixados, sem com isso afetar o resultado obtido. Heurísticas e meta-heurísticas podem ser aplicadas para auxiliar no processo de definição dos limites, reduzindo o tempo de processamento e número de ramificações. Na Figura 2.5 é possível avaliar a criação da árvore pelo *branch-and-cut*, onde temos um problema inicial S , onde são aplicados os planos de corte (adição de restrições) gerando S' . Na operação de *branch* realizada sobre S' são gerados dois filhos $S1$ e $S2$. $S1$ é avaliado, e sobre ele novamente é aplicado o método de planos de corte, dando origem a $S'1$. Na operação de *branch* realizada em $S'1$ são gerados 2 subproblemas $S3$ e $S4$, e $S3$ é podado por não fornecer uma solução viável. Novamente a técnica de planos de corte é aplicada, desta vez sobre $S4$ dando origem a $S'4$, que em sua operação de *branch* dá origem a dois novos subproblemas, $S5$ e $S6$, ambos encerrados obtendo solução inteira válida. Continuando a análise da árvore é avaliado o nó $S2$, este por sua vez, mesmo não possuindo uma solução inteira não passa pelo processo de *branch*, pois possui um valor de solução maior do que o limite superior obtido ao realizar as operações em subproblemas já avaliados (considerando um problema de minimização), sendo encerrado por limite.

É possível avaliar que a operação de *branch* não foi realizada em $S2$ pois o subproblema em questão já apresentava solução pior do que o limite superior do problema obtido em processamento anterior. O IBM Ilog Cplex, usado como *solver* na implementação do híbrido, utiliza o algoritmo *branch-and-cut* para solucionar os problemas de Programação Inteira. Uma versão simplificada do algoritmo *branch-and-cut* pode ser avaliado no Algoritmo 3.

Algoritmo 3 Algoritmo *branch-and-cut*.

$L \leftarrow S$;
 $U \leftarrow$ Valor da melhor solução corrente (*upper bound*)
 $melhorAtual \leftarrow$ Melhor solução corrente
enquanto $L \neq \emptyset$ **faça**
 Seleciona um nó de ramificação k de L
 Remove k de L
 Aplica **plano de corte** sobre k
 Gerar os filhos C_i (para $i = 1, 2, \dots, n_k$)
 Calcular os limites inferiores correspondentes LB_i
 para $i \leftarrow 1$ até n_k **faça**
 se $LB_i < U$ **então**
 se C_i é uma solução inteira **então**
 se LB_i é uma solução melhor que U **então**
 $U \leftarrow LB_i$
 $melhorAtual \leftarrow$ Solução correspondente de C_i
 fim se
 senão
 se LB_i é uma solução melhor que U **então**
 Adiciona C_i para L
 fim se
 fim se
 fim se
 fim para
fim enquanto

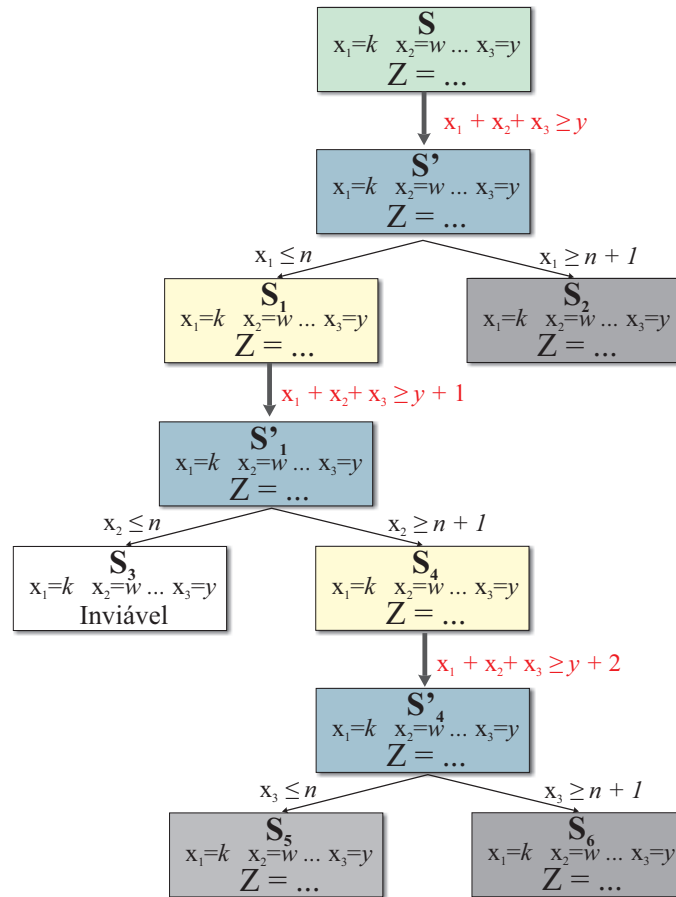


Figura 2.5: Exemplo de árvore - *branch-and-cut*.

2.3.2 Heurísticas e meta-heurísticas

O conjunto de heurísticas e meta-heurísticas, também chamados de métodos aproximados, são amplamente utilizados em aplicações reais que envolvem cenários complexos. Tal situação ocorre devido ao fato destes métodos retornarem boas soluções (em alguns casos obtendo a solução ótima), dentro de um espaço de tempo menor que o demandado normalmente por métodos exatos, e exigindo um esforço computacional menor.

O nome heurística vem de uma palavra grega, cujo significado é descobrir. Trata-se de uma técnica computacional que alcança uma solução viável, baseado na experiência do método. Por gerar resultados aproximados (ou resultados ótimos locais), pode não retornar a solução ótima para o problema (Goldberg et al., 2016). Os métodos heurísticos podem ser divididos em sete (Rodrigues, 1996):

- a) Heurísticas de construção.
- b) Heurísticas de melhorias.

- c) Heurísticas de decomposição.
- d) Heurísticas de particionamento.
- e) Heurísticas de relaxação.
- f) Heurísticas baseadas em formulações matemáticas.
- g) Heurísticas de restrição do espaço de soluções.

Já uma meta-heurística pode ser definida como um conjunto de conceitos e métodos, que podem ser aplicados na resolução de problemas com uma grande quantidade de soluções possíveis. Uma meta-heurística pode também ser definida como uma estratégia de busca, que tenta explorar eficientemente um dado conjunto de possíveis soluções, ou ainda, pode ser vista como um conjunto de métodos e modelos algorítmicos gerais, que sofrem pequenas modificações em busca de torná-las adaptáveis para resolução de problemas específicos (Goldberg et al., 2016). São muito aplicadas na solução de problemas de otimização combinatória. Origens das meta-heurísticas são encontradas em comunidades de inteligência artificial e pesquisa operacional (Blum, 2012).

Existem diversas meta-heurísticas, algumas inspiradas em comportamentos, outras na natureza. Dentre elas cabe citar: Otimização de Colônia de Formigas (*ACO*, do inglês, *Ant Colony Optimization*), Algoritmos Evolucionários ou Genéticos (*AG*), Algoritmos Meméticos (*AM*), Busca Local Iterada ou *Iterated Local Search (ILS)*, *Simulated Annealing (SA)*, Busca Tabu (*TS*, do inglês *Tabu Search*), algoritmo do Morcego (*BA*, do inglês *Bat Algorithm*), Busca harmônica (*HS*, do inglês *Harmony Search*) e outros (Yang, 2008).

Uma classificação das meta-heurísticas foi indicada por Melián et al. (2003): Meta-heurísticas construtivas, Meta-heurísticas evolutivas, meta-heurísticas de busca por entornos e técnicas de relaxação.

2.3.2.1 Heurísticas de busca local

Busca Local (*Local Search*) pode ser definida como uma função, método ou algoritmo que seleciona a partir de um ponto qualquer de um conjunto S (conjunto finito) uma solução inicial, fazendo uma avaliação dos valores próximos a posição atual (chamados também de vizinhos ou vizinhança), elegendo uma solução possível (chamada de ótimo local). Após a escolha, é feito um salto entre os valores disponíveis, e novamente os

vizinhos são avaliados. Essa operação é realizada repetidamente buscando gerar diversas soluções ótimas locais (Brucker, 2007). Ou seja, dado um determinado conjunto S a função de busca local tenta inicialmente retornar uma solução s^* (com $s^* \in S$), e a partir dela, avaliar os elementos próximos (chamados de vizinhos), dados por $N(s)$ (onde $N(s)$ é um subconjunto de S). A partir de $N(s)$, uma nova solução é obtida e a operação é refeita (Brucker, 2007).

Nos problemas de minimização de atraso, o algoritmo define, para cada uma das soluções encontradas, uma vizinhança composta por um conjunto de soluções com características semelhantes. Então, a partir da solução encontrada, é percorrida toda a vizinhança em busca de uma solução com um valor menor e, se for encontrada, o processo retoma a avaliação da vizinhança da nova solução. O processo se repete enquanto forem sendo encontradas novas soluções. Caso não seja encontrada uma solução melhor que a solução buscada, o algoritmo é encerrado. A Figura 2.6 apresenta um gráfico de soluções na aplicação de uma heurística de busca local.

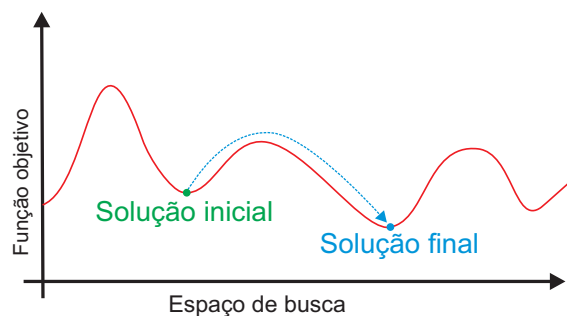


Figura 2.6: Funcionamento de heurística de busca local em um determinado espaço de busca.

2.3.2.2 Algoritmos genéticos

Propostos inicialmente por Holland (1992), os algoritmos genéticos (AG) se baseiam no processo na Teoria da Evolução de Darwin. São de busca heurística de otimização global, sendo parte de uma classe específica de algoritmos evolutivos, cuja obtenção de novas soluções é obtida por técnicas semelhantes as encontradas na biologia evolutiva tais como seleção natural, recombinação/cruzamento e mutação. Os AGs possuem uma vasta aplicação em diversas áreas científicas, tendo seu uso muito comum na solução de problemas de otimização, aprendizado de máquina, ambientes de simulação, análise de modelos econômicos dentre outros. Em seu funcionamento, um algoritmo genético parte de uma

população inicial e, após a evolução desta população (e aplicação de alguns métodos algorítmicos), é obtida a solução.

Uma população inicial normalmente é constituída a partir de indivíduos gerados de forma aleatória, ou através de algumas estratégias que buscam ter alguns indivíduos com melhor valor de *fitness*. Cada indivíduo (também chamado de cromossomo) é constituído de genes, que representam as características a serem avaliadas pela função de *fitness* (ou função de aptidão). Após a geração da população inicial, é executada a avaliação das soluções obtidas, ou seja, o valor de *fitness* de cada um dos indivíduos. De maneira análoga a Teoria da Evolução, onde os indivíduos com maior capacidade de adaptação sobrevivem, os algoritmos genéticos tratam da evolução de indivíduos em busca de um bom conjunto de soluções para um dado problema. A evolução ocorre ao realizarmos mudanças nos cromossomos disponíveis, conseqüentemente gerando novos indivíduos, e avaliando o resultado gerado. Essas mudanças são dadas por intermédio de técnicas denominadas **operadores genéticos**. Os operadores genéticos são responsáveis por produzir novas soluções seja através da recombinação de genes de alguns indivíduos, quanto pela mudança na sequencia de genes de um cromossomo. Basicamente temos os seguintes operadores genéticos: **Cruzamento ou recombinação, Mutação e Inversão**.

Após a geração e avaliação da população inicial, os AGs atuam de forma cíclica, aplicando os operadores genéticos responsáveis pela criação de novos indivíduos (que realiza a evolução da população), seguido da avaliação das soluções encontradas e por fim selecionando os melhores indivíduos para fazer parte da próxima geração (**elitismo**). O processo se repete enquanto o critério de parada não é atingido, o que pode ser representado por um número máximo de gerações, tempo máximo de execução ou ter alcançado uma determinada solução limite. O Algoritmo 4 indica um AG e seu esquema de funcionamento.

Algoritmo 4 Algoritmo Genético.

```
Gera população inicial
Avalia soluções
enquanto Critério de parada não for atingido faça
    Evolui população
    Avalia soluções
    Seleciona indivíduos da próxima geração
fim enquanto
retorne Melhor solução obtida
```

2.3.2.2.1 Função de *fitness* ou avaliação de aptidão

Este é o componente responsável por calcular o valor de aptidão de cada indivíduo, sendo portanto o componente mais importante de um AG. É o método responsável por analisar as soluções produzidas, indicando a qualidade das mesmas. Seu desempenho afeta diretamente não só a qualidade dos resultados de um AG, como também seu desempenho (uma vez que é executada diversas vezes durante o processo completo de um AG). Em problemas de otimização combinatoria esta função é dada pela função objetivo.

2.3.2.2.2 Método de seleção

Em algumas das operações realizadas por um AG, em diversas etapas é necessário realizar a seleção de um ou mais indivíduos. Basicamente todo operador genético precisa atuar sobre um ou mais indivíduos da população. Tal processo se dá por meio de um método de seleção. Existem diversos métodos de seleção, porém, é possível destacar os métodos de seleção por Roleta (Figura 2.7) e por Torneio.

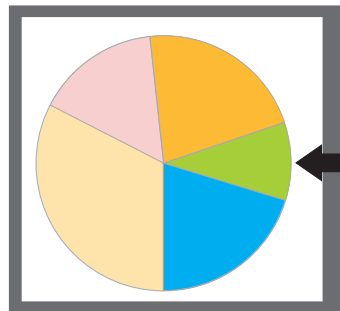


Figura 2.7: Exemplo de seleção por roleta.

No método de seleção por Roleta os indivíduos são selecionados de acordo com sua função de aptidão. Quanto melhor for o valor de aptidão de um indivíduo, maiores são suas chances de ser selecionado. A atuação deste algoritmo se assemelha a uma roleta, onde os indivíduos com melhores valores de *fitness* possuem maior porção da roleta. O algoritmo responsável pela seleção por Roleta, tem seu funcionamento relativamente simples de ser implementado: é feita a soma do valor de *fitness* de todos os cromossomos da população e armazenado em uma variável s ; em seguida é gerado um número aleatório r que está entre 0 e s ; a população é então percorrida e novamente seus valores vão sendo somados em uma variável t . Quando o valor de t estiver maior ou igual ao valor de r , o indivíduo que foi adicionado por último é retornado. Um exemplo de implementação

pode ser visto no Algoritmo 5.

Algoritmo 5 Método de Seleção por Roleta.

```
 $s \leftarrow 0$   
para todo indivíduo da População faça  
     $s \leftarrow s + \textit{fitness}$  do indivíduo  
fim para  
 $r \leftarrow$  Valor aleatório entre 0 e  $s$   
 $t \leftarrow 0$   
para todo indivíduo da População faça  
     $t \leftarrow t + \textit{fitness}$  do indivíduo  
    se  $t \geq r$  então  
        retorne Indivíduo  
    fim se  
fim para
```

Um dos pontos desfavoráveis deste tipo de implementação se dá devido a necessidade de passar por todos os indivíduos duas vezes para a seleção de um único cromossomo, o que pode ser um complicador por impactar diretamente no desempenho do AG, se considerar uma população muito grande.

Além do método de Seleção por Roleta, existe também o método de Seleção por Torneio (Figura 2.8), onde um número n de indivíduos da população é selecionado de forma aleatória, formando um subgrupo. Neste subgrupo o indivíduo será selecionado baseado em uma probabilidade p (definida previamente). Essa operação é repetida para cada indivíduo que deve ser selecionado.

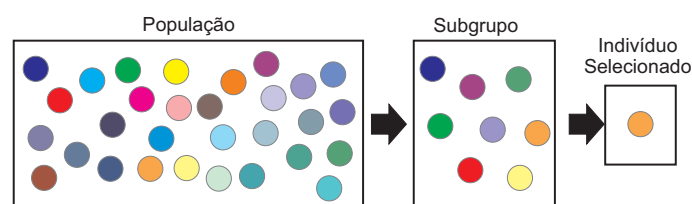


Figura 2.8: Exemplo de seleção por torneio.

O método de Seleção por Torneio tem um desempenho melhor do que o método de Seleção por Roleta, porém, sua eficiência varia conforme alguns parâmetros, tais como a quantidade de indivíduos a ser selecionada para o subgrupo e o valor da probabilidade p . Uma implementação deste método de seleção pode ser visto no Algoritmo 6.

Algoritmo 6 Método de Seleção por Torneio.

```
 $p \leftarrow 0.8$   
Seleciona  $n$  indivíduos da população  
 $k \leftarrow$  valor aleatório entre 0 e 1  
se  $r < k$  então  
    retorne Melhor indivíduo  
senão  
    retorne Pior indivíduo  
fim se
```

2.3.2.2.3 Cruzamento ou recombinação

Considerado o operador genético predominante, o cruzamento ou recombinação é responsável por produzir novos indivíduos a partir da combinação de características de cromossomos já existentes. Esse operador genético atua inicialmente selecionando dois indivíduos da população e recombinaando seus genes a partir de algum tipo de estratégia. O novo indivíduo gerado é então adicionado a população, e caso apresente melhor valor de *fitness* (nesse contexto, indicando o resultado da função objetivo), ele pode ser integrado ao conjunto de soluções, ou substituir uma das soluções existentes, passando para a geração seguinte. Existem diversas estratégias de recombinação genética, dentre as quais vale destacar o método uniforme, o método de um ponto, e o método de dois pontos.

No **Método Uniforme** (Figura 2.9) inicialmente são selecionados dois indivíduos da população, denominados *Parent1* e *Parent2*. Para a escolha destes indivíduos é utilizado um dos métodos indicados na subseção anterior (Métodos de Seleção). Após a escolha dos pais, é gerada uma máscara de cruzamento, que nada mais é que uma sequência composta de valores 0 ou 1, de mesmo tamanho que os cromossomos. Também são gerados dois novos indivíduos (*Offspring1* e *Offspring2*). A máscara de cruzamento é então percorrida e, para cada valor igual a 1 existente na máscara de cruzamento, o *Offspring1* recebe a característica do *Parent1*, enquanto que o *Offspring2* recebe o gene do *Parent2*. Após o preenchimento inicial, as lacunas existentes nos dois novos cromossomos são preenchidas, desta vez, o *Offspring1* complementa seu cromossomo com genes do *Parent2*, enquanto que *Offspring2* complementa seu cromossomo com genes do *Parent1*. Um fator que pode ser controlado nesse tipo de cruzamento é o percentual mínimo de genes a serem obtidos inicialmente. Esse valor pode implicar na obtenção de soluções melhores, porém, pode variar conforme o caso devendo portanto ser obtido através de experimentação ou através do uso de ferramentas específicas para essa finalidade.

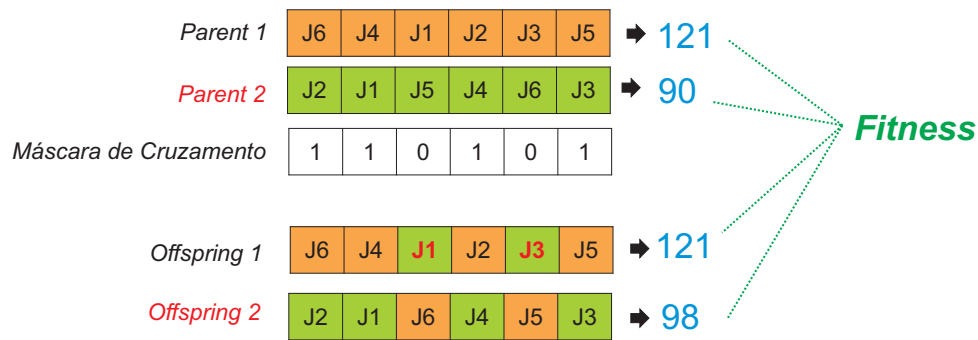


Figura 2.9: Exemplo de cruzamento/recombinação usando método uniforme.

No **Método de Cruzamento em um Ponto** (Figura 2.10), de forma semelhante ao cruzamento uniforme, são selecionados dois indivíduos da população. Também é gerado um valor aleatório p entre 1 e $N - 1$ (onde N é dado pelo tamanho dos cromossomos). Esse valor será responsável por indicar o ponto de cruzamento. Dessa forma, toda sequência de genes de *Parent1* que estiver antes do ponto de cruzamento é copiado para *Offspring1*, e toda a sequência de genes de *Parent2* que estiver antes do ponto de cruzamento é copiado para *Offspring2*. O restante da sequência genética de *Offspring1* é complementada com genes de *Parent2*, e de maneira semelhante, a sequência genética de *Offspring2* é complementada com genes de *Parent1*.

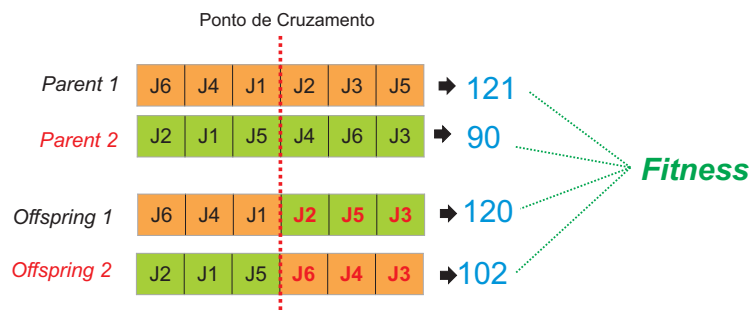


Figura 2.10: Cruzamento/recombinação usando método de um ponto.

O **Método de Cruzamento em Dois Pontos** (Figura 2.11) se assemelha ao método citado anteriormente, exceto que ao invés de efetuar as trocas em apenas um ponto, são selecionados dois pontos de intersecção.

Existem estratégias onde mais de dois pontos são inseridos, além de outros procedimentos, como cruzamento baseado em média, ou ainda, avaliando o peso/disponibilidade de cada característica (como a aplicada ao utilizar Evolução Diferencial). Cada estratégia de cruzamento pode ter melhores ou piores resultados dependendo da natureza do problema, bem como tamanhos da instância.



Figura 2.11: Cruzamento/recombinação usando método de dois pontos.

2.3.2.2.4 Mutação

A mutação é a operação genética responsável por modificar, aleatoriamente, alguns genes de um indivíduo. A mutação é uma operação muito importante nos AGs, uma vez que é capaz de produzir novas características que não foram repassadas ao indivíduo na sua criação. Ao modificar um ou mais genes por vezes produz bons resultados sendo capaz de encontrar boas soluções, que em alguns casos, acabam por ser melhores que as soluções produzidas através de outros operadores, uma vez que modificam apenas um pequeno conjunto de genes. Com isso mantém boa parte das características existentes, já obtidas através de outras operações. Outro fator importante no uso da mutação ao se trabalhar com AGs em problemas de natureza combinatória se dá ao fato dos AGs convergirem muito rapidamente, e por vezes, ficarem presos em ótimos locais. A modificação de pequenas partes da solução causa uma pequena perturbação, que pode ser suficiente para fugir de soluções intermediárias. A Figura 2.12 a mutação sendo aplicada a um determinado indivíduo, permutando dois genes.

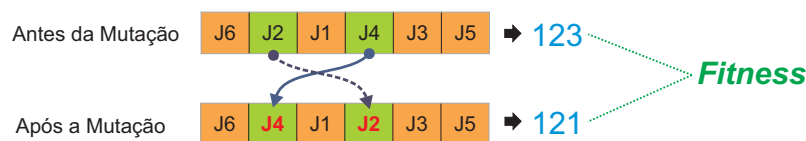


Figura 2.12: Mutação aplicada a um determinado indivíduo.

A aplicação da mutação não é realizada sobre toda a população a todo o tempo, ocorrendo apenas quando atingir um dado valor, ou em alguns casos, existir a necessidade de aumento na diversidade da população. A taxa de mutação, bem como a diversidade população mínima são parâmetros que podem influenciar na qualidade das soluções obtidas e portanto, exigem cuidado e testes em sua escolha.

2.3.2.2.5 Inversão

A inversão é um dos operadores genéticos citados por Holland (1992), porém, pouco utilizado. Isso porque sua aplicação pode causar modificação em um grande conjunto de características de um dado cromossomo, além de ter um custo computacional maior do que o operador de mutação. Esse operador seleciona aleatoriamente dois pontos i e f (com i e f maior do que 1 e menores do que o tamanho do cromossomo). Feito isso, inverte a sequência genética que está entre i e f . Seu comportamento pode ser visto na Figura 2.13.

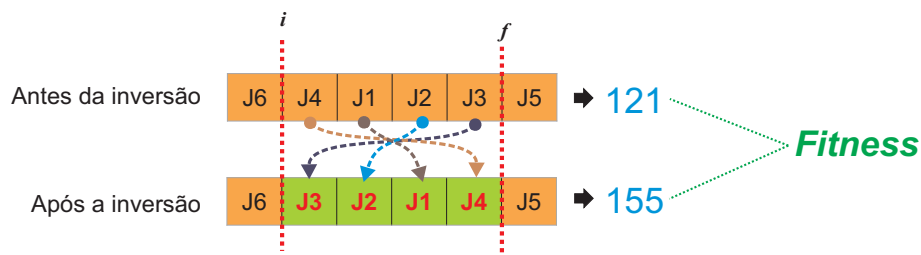


Figura 2.13: Inversão aplicada a um determinado indivíduo.

2.3.2.2.6 Parâmetros genéticos

O funcionamento e eficiência dos algoritmos genéticos está ligada diretamente aos parâmetros aplicados ao mesmo. Estes parâmetros traçam comportamentos, determinam operações além de determinar situações mais extremas como condições de parada e retorno da solução. Em se tratando de AGs, são diversos os parâmetros possíveis:

- Tamanho da população.
- Tipo de seleção e de cruzamento.
- Taxa de cruzamento, de mutação e de inversão.
- Probabilidade de mutação.
- Probabilidade de inversão.
- Intervalo de geração.
- Número de gerações.
- Tempo máximo de execução.

i) Percentual mínimo de diversidade populacional.

Cada um dos parâmetros citados acima implica em diferenças comportamentais por parte do algoritmo genético, acarretando em um desempenho melhor ou pior, além de gerar variação nas soluções obtidas. Para um bom desempenho de um AG (considerando eficiência e eficácia) faz-se necessária a realização de um grande número de testes, avaliando a combinação de valores para os diferentes parâmetros aplicados no AG, ou então, a utilização de ferramentas específicas para este fim, como é o caso da ferramenta *iRace*. Uma abordagem mais detalhada sobre otimização de parâmetros é apresentada na Seção 2.4.

2.3.3 Métodos híbridos

Métodos híbridos são aqueles que se utilizam de uma ou mais técnicas para obtenção das soluções. Quando citamos a utilização de diversas técnicas, combinadas, está sendo feita referência ao fato de que é possível serem trabalhadas combinações de uma ou mais heurísticas (ou meta-heurísticas) alimentando um método exato, ou então uma heurística alimentando uma meta-heurística. Combinar diferentes estratégias e métodos pode permitir aliar pontos fortes pertencentes a cada uma delas, obtendo assim melhores resultados. O uso de métodos combinados também é viável por permitir minimizar problemas originados em uma dada técnica, como por exemplo, reduzir o espaço de soluções viáveis que serão avaliadas por uma formulação matemática ou algoritmo guloso. Uma vantagem destes métodos está na possibilidade de aproximar o resultado obtido da solução ótima (usando por exemplo um método exato), ou ainda, comprovar a veracidade e qualidade de uma solução obtida através da aplicação de um conjunto de heurísticas e meta-heurísticas (Pinedo, 2010). Abordagens híbridas não são recentes (Srinivasan, 1971), contudo, a capacidade computacional bem como os algoritmos e heurísticas vêm evoluindo, tornando os métodos e algoritmos híbridos uma opção interessante. Dada sua capacidade de combinar não apenas características de heurísticas e meta-heurísticas, mas também por agregar recursos de Programação Inteira (IP), e Programação por Restrições (CP) (Melián et al., 2003), faz com que sejam vistos como boa aplicação para diversos problemas de natureza combinatória.

2.4 Otimização de parâmetros

Determinar valor de parâmetros a serem aplicados em algoritmos usados na solução de problemas com natureza combinatória, por si só, já é comprovadamente de natureza combinatória, tornando seus ajustes e calibração demasiadamente complexos (López-Ibáñez et al., 2011). Problemas de otimização possuem uma função objetivo, utilizada para verificar a qualidade de uma solução candidata. Esta função é normalmente repassada como um procedimento Caixa-Preta, não sendo possível relacionar ou analisar os impactos gerados por parâmetros usados nos algoritmos que criam a solução (Rudolf, 2016).

Existem estudos e pesquisas sendo realizadas afim de tratar deste problema, como a Otimização Sequencial de Parâmetros (SPO) e Estimativa de Relevância de Parâmetro e Calibração de Valor (REVAC) (Nannen and Eiben, 2006) (Nannen and Eiben, 2007), além de diversas técnicas heurísticas e da possibilidade de ajustes manuais nos parâmetros baseado em experimentação.

Também existem ferramentas específicas para este fim como **iRace**¹, ParamILS², SMAC³ e SPOT⁴.

A correta escolha dos parâmetros pode fazer com que um determinado algoritmo fuja de ótimos locais, e ainda, prevenir a convergência prematura de algoritmos, sendo portanto, de alta relevância ao lidar com problemas de otimização usando heurísticas e meta-heurísticas.

¹<http://iridia.ulb.ac.be/irace/>

²<http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/>

³<http://ml.informatik.uni-freiburg.de/papers/11-LION5-SMAC.pdf>

⁴<https://cran.r-project.org/web/packages/SPOT/index.html>

Capítulo 3

Escalonamento em Máquinas Paralelas com Minimização de Atraso Total

Ponderado

Um cenário comum encontrado na indústria é a busca pelo atendimento correto de prazos. Em alguns casos, cuja produção possui datas limites de entrega, incluindo ainda, multas e quebras contratuais em casos de atrasos. Seja na fabricação de produtos intermediários (que servem para a fabricação de outros), seja para a entrega e venda direta, a existência de atrasos no cumprimento de prazos acarreta em perdas e prejuízos. Esse cenário pode ser visto em períodos específicos da indústria e comércio como datas especiais e comemorativas (Natal, Dia das Crianças, entre outros). Atrasos na produção podem gerar queda nas vendas. Sabendo que o comércio nesses períodos é alto, porém limitado a datas específicas, a produção insuficiente ou fora do período poderiam ser desastrosos.

Empresas de software contratadas para desenvolvimento de programas de computador voltados ao controle de chão-de-fábrica, sistemas financeiros, empresas de investimento. A ocorrência de atrasos nestes setores pode impactar de forma negativa, gerando perdas imensuráveis. Este é o foco deste trabalho, situações que podem ser visualizadas como uma sequência de tarefas a ser realizada em um ambiente limitado de máquinas, cujo atraso na realização de cada uma das tarefas acarreta em problemas ou algum tipo de prejuízo.

O assunto deste trabalho é a Minimização do Atraso Total Ponderado em Máquinas Paralelas, que poderia ser assim descrito: dado um conjunto de tarefas $J=\{1, 2, 3, \dots, j\}$,

para serem executadas em um grupo de máquinas $M=\{1, 2, \dots, m\}$. Cada máquina pode processar apenas uma tarefa por vez, e uma vez tendo iniciado sua execução, o procedimento não pode ser interrompido (ou seja, sem preempção). Cada tarefa j possui um tempo de processamento p_j , um peso w_j (com $w_j \geq 0$) e um instante previsto para término d_j . O atraso da tarefa j , dado por $T_j = \max(0, C_j - d_j)$, onde C_j indica o instante real de término da tarefa j . O objetivo é encontrar a sequencia correta de distribuição das tarefas para as máquinas, de forma que minimize a soma das penalidades originadas pelos atrasos, tendo portanto como função objetivo $\sum w_j T_j$.

Encontrar a melhor distribuição destas tarefas é uma tarefa complexa, demandando tempo computacional dificilmente disponível, uma vez que a quantidade de soluções candidatas é enorme (visto que o problema é de natureza combinatória). Uma expressão matemática capaz de representar o número de soluções possíveis pode ser dada por $(N!)^m$, sendo N a quantidade de tarefas a ser executada e m a quantidade de máquinas disponíveis.

Conforme [Xiao and Li \(2002\)](#), controlar a data de término de tarefas é um passo importante e necessário no planejamento de operações em uma organização. Mesmo assim, decidir de forma errônea o planejamento de entrega, como citado, pode acarretar em prejuízos, uma vez que o ambiente produtivo pode não acompanhar o planejamento que fora realizado. Quando o planejamento não condiz com o ambiente de produção, gera o atraso na produção, o que em alguns casos pode ocasionar em prejuízos, seja devido a cláusulas contratuais junto ao cliente, seja devido a perda na oportunidade da venda. A Figura 3.1 ilustra a definição de atraso (T_j), onde a penalidade aumenta em conformidade com o atraso.

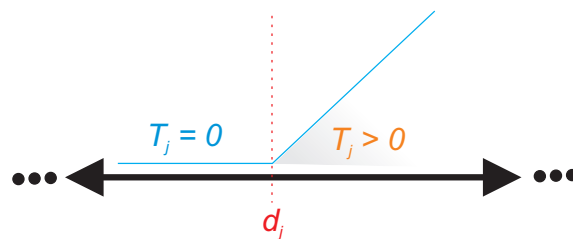


Figura 3.1: Penalidade aplicada ao atraso.

Na literatura existem diversas abordagens buscando propor uma melhor solução ao problema. Algumas utilizando unicamente métodos exatos, através de formulações matemáticas, outras utilizando de heurísticas e meta-heurísticas.

Para instâncias maiores, os métodos exatos, mesmo usando enumeração implícita e

conseguindo retornar a melhor solução possível (ótimo global), acabam gerando um esforço enorme (bem como um consumo muito grande de recursos), inviabilizando sua utilização. Na busca por resolver instâncias maiores, heurísticas e meta-heurísticas foram criadas e aplicadas; Também foram realizados testes e experimentos combinando técnicas e estratégias heurísticas. Um conjunto de trabalhos relacionados é apresentado na seção seguinte onde é apresentada uma breve descrição acerca dos trabalhos.

3.1 Trabalhos relacionados

O tema abordado nesta dissertação trata de um conteúdo rico e com vasto histórico. Uma destas pesquisas é a produzida por Lee et al. (1997), abordando o problema caracterizado por $1|s_{ij}|\sum w_j T_j$, onde é construída uma heurística para tratar do problema, aplicando dentre outras técnicas, a regra *WSPT*, uma estratégia que sequencia as tarefas de forma crescente, baseado no resultado da penalidade de atraso dividida pelo tempo de processamento da tarefa (w_j/p_j). Este trabalho já indica a possibilidade da heurística construída atuar em conjunto com outras existentes (*simulated annealing* e busca tabu).

Montalvão and Pessoa (2016) apresentam um algoritmo que combinava uma busca local iterada (*ILS*) e Programação Dinâmica tratando de uma versão robusta do problema de minimização de atraso ponderado em uma máquina. A busca local utilizada na abordagem (*GPI*) havia sido inicialmente proposta por de Freitas et al. (2008) para tratar do problema de minimização de atraso ponderado em ambientes de máquinas paralelas idênticas.

No trabalho de Feili et al. (2012), o problema foi abordado fazendo uso e comparando três métodos distintos: algoritmos genéticos (*AG*), *simulated annealing* (*SA*) e *hill climbing* (*HILL*), onde em seu quadro comparativo inicial obteve melhores resultados (tanto em relação a tempo quanto valor da FO) obtidos pelo SA. Nos resultados apresentados, aqueles obtidos pelo SA são expressivamente melhores que os outros (*AG* e *HILL*). Em comparação posterior, são apresentados resultados obtidos através de abordagens híbridas (*SA+AG* e *SA+HILL*), com os resultados obtidos apenas na aplicação do SA. As abordagens híbridas apresentaram resultados melhores, cabendo destaque ao método *SA+HILL*, que obteve os melhores resultados dentre os apresentados.

Nos resultados apresentados por Tanaka and Fujikuma (2011), onde foi tratado o problema da minimização do atraso ponderado em uma máquina, com tempos de liberação

arbitrários, cabe destaque a resolução apresentada, onde foi usada Programação Dinâmica e obtidos bons resultados. Cabe destaque nesse trabalho que ressalta o fato de que, com a metodologia utilizada, o mesmo é capaz de resolver instâncias com 300 (trezentas) tarefas para o problema $1||\sum w_j T_j$.

No trabalho de [Sivapragasam and Suppiah \(2017\)](#) foi utilizado um algoritmo genético como estratégia de resolução, que utilizava uma variação de cruzamento uniforme. Apesar de tratar pequenas instâncias, o destaque do mesmo é dado devido a utilização de diversas heurísticas de despacho utilizadas: EDD, WEDD, SPT, WSPT, LPT e WLPT.

No artigo de [de Freitas et al. \(2008\)](#) é apresentada uma heurística de busca local com movimentos GPI. No trabalho são resolvidas instâncias com 40 e 50 tarefas, distribuídas em 2 e 4 máquinas. A importância deste trabalho pode ser vista no fato do mesmo servir para diversos outros trabalhos.

[Pessoa et al. \(2010\)](#) nesse trabalho apresentam um algoritmo exato baseado na formulação *arc-time-indexed*, aplicado a máquinas paralelas idênticas. O algoritmo foi executado para instâncias com 40, 50 e 100 tarefas, e 2 e 4 máquinas. Um diferencial desta formulação é que a mesma não avalia variáveis que não gerem impacto na função objetivo (valores com penalidade igual a zero), permitindo assim que instâncias maiores possam ser verificadas.

[Della Croce et al. \(2012\)](#) em seu trabalho apresenta uma heurística de busca local iterada, voltado ao problema $P||\sum w_j T_j$. Aqui novamente está presente o algoritmo de busca local GPI, bem como o GPI com modificações. Como resultados foram apresentadas soluções para diferentes proporções de instâncias (40, 50, 100 e 300 tarefas e 2, 4 e 10 máquinas). Em destaque, a resolução do problema para grandes instâncias com 300 tarefas e 20 máquinas.

Nos trabalhos de [Mensendiek et al. \(2015\)](#) e [Liaw \(2016\)](#) a solução também é dada através de uma abordagem exata, através do método *branch-and-bound*. Ambos tratam da minimização do atraso em máquinas paralelas idênticas. As abordagens utilizadas em ambos os trabalhos foram semelhantes, exceto pelo fato do segundo ter avaliado um cenário que permite que a execução das tarefas seja interrompida, ou seja, permite preempção.

Tabela 3.1: Classificação dos problemas de escalonamento com antecipação e atraso de tarefas.

Problemas	Estratégias de resolução	Referências
$1 \sum T_j$	Busca local baseada em movimentos GPI e Programação Dinâmica	Montalvão and Pessoa (2016)
$1 s_{ij} \sum w_j T_j$	<i>Simulated Annealing</i> , algoritmo genético, <i>hill climbing</i> e abordagens híbridas (SA+AG e SA+HILL)	Feili et al. (2012)
$1 r_j \sum w_j T_j$	Programação dinâmica	Tanaka and Fujikuma (2011)
$P \sum T_j$	Algoritmo genético	Demirel et al. (2011)
$P \sum w_j T_j$	Algoritmo Genético	Sivapragasam and Suppiah (2017)
$P \sum w_j T_j$	Lista sequencial de busca local baseada em movimentos GPI, com critério de desempate	de Freitas et al. (2008)
$P \sum w_j T_j$	ILS com técnicas de busca em vizinhança de grande porte (VLNS)	Della Croce et al. (2012)
$P \sum w_j T_j$	Algoritmo exato <i>Branch-Cut-and-Price</i>	Pessoa et al. (2010)
$P \sum T_j$	<i>Branch-and-bound</i>	Mensendiek et al. (2015)
$P prmp \sum T_j$	<i>Branch-and-bound</i>	Liaw (2016)
$Q p_j = p, prmp \sum T_j$	Algoritmo de tempo polinomial	Kravchenko and Werner (2009)
$Q p_j = 1 \sum w_j T_j$	Algoritmo de tempo polinomial	Dourado et al. (2010)
$Q prmp \sum T_j$	Algoritmo de tempo polinomial - Q2TT	Lushchakova (2012)
$R_m r_j \sum_{j=1}^n w_j T_j = \sum_{j=1}^n w_j (C_j - d_j)^+$, onde MP é múltiplas máquinas	Heurística baseada em decomposição e busca em vizinhança variável (VNS)	Bilyk and Mönch (2010)
$R \sum w_j T_j$	Híbrido utilizando Algoritmo genético e <i>Simulated Annealing</i>	Shafiei Nikabadi and Naderi (2016)

Continua na próxima página

Tabela 3.1 – continuação da página anterior

Problemas	Estratégias de resolução	Referências
$R \sum w_j T_j$	Híbrido - GRASP- <i>Greedy Randomized Adaptive Search Procedure</i> e ILS	João et al. (2014)
$R s_{ij} \sum w_j T_j$	Busca Tabu	Lee et al. (2013)

3.2 Formulações matemáticas

Uma forma de realizar o escalonamento objeto deste trabalho é através de uma formulação de Programação Linear Inteira ou Inteira Mista. Existem na literatura alguns exemplos de formulação modeladas com o intuito de resolver o problema. Nesta seção serão apresentadas formulações estudadas que contribuíram no trabalho.

3.2.1 Formulação de programação linear inteira mista para o escalonamento em máquinas paralelas com tempo ocioso

Dentre as formulações avaliadas, está a apresentada por Arenales et al. (2007). A formulação leva em consideração que as tarefas devem estar disponíveis no instante zero ($r_j = 0$) e que cada tarefa possui sua própria data de término sugerida (d_j) e que qualquer máquina pode processar qualquer uma das tarefas, sendo que cada tarefa deve ser processada apenas uma vez. Ao iniciar a execução de uma tarefa, não é permitida a interrupção (ou seja, sem preempção) e cada máquina pode executar uma (e apenas uma) tarefa por vez. Os valores de n , m e M são, respectivamente, o total de tarefas, o total de máquinas e uma constante suficientemente grande (conhecida pelo termo *big M*). Dadas estas características, a formulação pode ser dada pelo que segue:

$$\text{Min. } \sum_{j=1}^n w_j T_j \quad (1)$$

$$\text{St: } \sum_{k=1}^m \sum_{i=0}^n x_{ijk} = 1 \quad j = 1, \dots, n. \quad (2)$$

$$\sum_{j=1}^n x_{0jk} \leq 1 \quad k = 1, \dots, m. \quad (3)$$

$$\sum_{i=0, i \neq h}^n x_{ihk} - \sum_{j=0, j \neq h}^n x_{hjk} = 0 \quad h = 1, \dots, n \text{ e } k = 1, \dots, m. \quad (4)$$

$$C_{0k} = 0 \quad k = 1, \dots, m. \quad (5)$$

$$C_{jk} \geq C_{ik} - M + (p_{jk} + s_{ijk} + M)x_{ijk} \quad i = 0, \dots, n; j = 1, \dots, n \text{ e } k = 1, \dots, m. \quad (6)$$

$$T_i \geq C_{ik} - d_i \quad i = 1, \dots, n \text{ e } k = 1, \dots, m. \quad (7)$$

$$T_i \geq 0 \quad i = 1, \dots, n. \quad (8)$$

$$x_{ijk} \in \{0, 1\} \quad i, j = 0, \dots, n \text{ e } k = 1, \dots, m. \quad (9)$$

Na formulação acima a função objetivo (1) minimiza o atraso total ponderado das tarefas, sendo n a quantidade de tarefas. Todas as tarefas devem estar disponíveis no instante zero e possuir as seguintes características:

- Data de término sugerida (d_j).
- Tempo de processamento (p_{jk} que indica o tempo de processamento da tarefa j na máquina k para ambientes de máquinas paralelas com diferentes velocidades, mas que pode ser alterado para p_j em ambientes de máquinas paralelas idênticas).
- Penalidade em caso de atraso (w_j).
- Tempo de preparação (s_{ijk} que indica o tempo de preparação da máquina k , para processar a tarefa j , após ter processado a tarefa i). Este valor é zero caso esta restrição não seja necessária.

A variável de decisão desta formulação é binária, dada por x_{ijk} que recebe 1 se a tarefa i precede a tarefa j na máquina k e zero caso contrário. Outra variável que pode receber valor é C_{ik} que indica o instante de término da tarefa i na máquina k ; e T_j que representa o atraso da tarefa j . As restrições (2) e (3) garantem uma sequencia única

de tarefas para cada uma das máquinas, com a restrição (2) garantindo que cada tarefa j alocada na máquina k seja precedida por apenas uma tarefa, enquanto que a restrição (3) garante que cada máquina k tenha uma sequência exclusiva de tarefas se for utilizada. A conjunto de restrições (4) define que cada tarefa j preceda a uma única tarefa, excetuando-se a tarefa zero, responsável por indicar início e fim de uma sequência de tarefas em uma dada máquina k . A restrição (5) indica que o tempo de completude da tarefa 0, em qualquer máquina k seja igual a 0 (indicando o início de uma sequência). No conjunto de restrições (6) são verificados os instantes de conclusão das tarefas nas máquinas onde são executadas. Na restrição (7) a restrição é aplicada para garantir que não existam valores negativos sendo considerados como atraso nas tarefas. As restrições (8) e (9) indicam o domínio das variáveis utilizadas. Cabe ressaltar que as restrições acima não impedem a ocorrência de tempo ocioso entre as tarefas.

3.2.2 Formulação de programação linear inteira mista para o escalonamento em uma máquina sem tempo ocioso

Outra formulação avaliada é a proposta no trabalho de [Feili et al. \(2012\)](#) e apresentada abaixo.

$$\text{Min } \sum_{i=1}^n (w_i T_i) \quad (1)$$

$$\text{St: } \sum_j x_{ij} = 1 \quad (2)$$

$$\sum_i x_{ij} = 1 \quad (3)$$

$$C_0 = 0 \quad (4)$$

$$C_j \geq C_{j-1} + \sum_{i=1}^n x_{ij} P_i + \sum_i \sum_{h \neq i} x_{h,j-1} x_{ij} S_{h,i} \quad (5)$$

$$T_i \geq \sum_{j=1}^n c_j x_{ij} - d_i \forall i \quad (6)$$

$$T_i \geq 0 \forall i \quad (7)$$

$$x_{ij} = 0, 1 \quad (8)$$

A formulação acima indica que é realizado o escalonamento de n tarefas, tendo como função objetivo (1) o mesmo que a formulação anterior. Como o problema abordado por Feili et al. (2012) envolve tarefas com tempos de configuração dependentes da sequencia, esta situação deve ser tratada, e isso pode ser visto nas restrições (2) e (3), onde x_{ij} possui valor 1 se a tarefa i é executada em prioridade j , e 0 caso contrário. $S_{h,i}$ apresentado na restrição (5) se refere ao tempo de configuração requerido quando a tarefa i é executada após a tarefa h . C_j indica o tempo de completude da tarefa com prioridade j . A restrição (2) indica que apenas uma tarefa pode ser alocada em cada prioridade, enquanto que a restrição (3) garante que cada tarefa esteja em apenas uma prioridade. Equações (4) e (5) mostram como é calculado o tempo de completude em cada prioridade. As restrições (6) e (7) tratam do cálculo do atraso para cada tarefa. Por fim, a restrição (8) indica uma limitação binária.

3.2.3 Formulação baseada no modelo de fluxo em redes e roteamento de veículos para o o escalonamento em máquinas paralelas - *Arc-time indexed formulation*

Esta formulação proposta por Pessoa et al. (2010) para a minimização do atraso ponderado em ambiente mono e multiprocessado, é baseado no modelo de fluxo em redes e roteamento de veículos (*Arc-time indexed formulation*). Nessa formulação o escalonamento é realizado no intervalo de tempo entre 0 e T , onde as máquinas estão ociosas no instante 0 e devem estar novamente ociosas no instante T . Destaca-se que o conjunto de tarefas $J = \{1, \dots, n\}$ (sendo n a quantidade de tarefas) é processado por um conjunto de máquinas $M = \{1, \dots, m\}$ (sendo m a quantidade de máquinas).

As variáveis binárias x_{ij}^t (com $i \neq j$) indicam que a tarefa i termina sua execução e a tarefa j inicia a sua execução em um dado instante t , em uma das máquinas disponíveis. Quando $i = 0$, é indicado que a tarefa j iniciou seu processamento em um instante t em alguma máquina que estava ociosa no intervalo de $t - 1$ e t , sendo que, as variáveis x_{0j}^0 indicam que a tarefa j iniciou seu processamento em alguma máquina ociosa. As variáveis x_{i0}^t indicam que a tarefa i finaliza seu processamento em um instante t , e que a máquina em que foi processada ficará ociosa entre os instantes t e $t + 1$. Se o valor de t for igual a T , indica que a tarefa i será concluída no instante final do intervalo de tempo.

$$\text{Min. } \sum_{i \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t=p_i}^{T-p_j} f_j(t+p_j)x_{ij}^t \quad (1)$$

$$\text{St: } \sum_{i \in J_+ \setminus \{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^t = 1 \quad (\forall j \in J) \quad (2)$$

$$\sum_{j \in J_+ \setminus \{i\}, t-p_j \geq 0} x_{ji}^t - \sum_{j \in J_+ \setminus \{i\}, t+p_i+p_j \leq T} x_{ij}^{t+p_i} = 0 \quad (\forall i \in J; t = 0, \dots, T-p_i) \quad (3)$$

$$\sum_{j \in J_+, t-p_j \geq 0} x_{j0}^t - \sum_{j \in J_+, t+p_j+1 \leq T} x_{0j}^{t+1} = 0 \quad (t = 0, \dots, T-1) \quad (4)$$

$$\sum_{j \in J_+} x_{0j}^0 = m \quad (5)$$

$$x_{ij}^t \in \mathbb{Z}_+ \quad (\forall i \in J_+; \forall j \in J_+ \setminus \{i\}; t = p_i, \dots, T-p_j) \quad (6)$$

$$x_{00}^t \in \mathbb{Z}_+ \quad (t = 0, \dots, T-1) \quad (7)$$

As restrições (3), (4) e (5) definem o fluxo em redes de m máquinas, representadas sobre um grafo acíclico dado por $G = (V, A)$, sendo que neste fluxo onde existem apenas uma origem e um destino, qualquer solução pode ser decomposta em um conjunto de m caminhos, cada um deles correspondendo a um escalonamento (sequência de tarefas e tempo ocioso) de cada uma das máquinas. A restrição (2) determina que cada tarefa deve pertencer a exatamente um caminho, e portanto, processada em somente uma máquina. A formulação acima considera a função objetivo dada por f_j , que representa a função de custo de uma dada tarefa j , sendo executada sobre o tempo de completude C_j .

No decorrer do capítulo foram apresentadas 3 formulações, dispostas na Tabela 3.2 abaixo:

Tabela 3.2: Resumo de formulações.

Número de Restrições	Número de Variáveis	Tipos de Variáveis	Ambiente de Processamento	Tipo de Formulação	Referência
$O(n^2m)$	$O(n^2m)$	mono, bi e tri-indexadas	mono e multiprocessado	PIM	Arenales et al. (2007)
$O(n^2)$	$O(n^2)$	bi-indexadas	monoprocessado	PI	Feili et al. (2012)
$O(nT)$	$O(n^2T)$	tri-indexadas	mono e multiprocessado	PI	Pessoa et al. (2010)

Dentre as formulações estudadas, foi utilizada na solução apresentada neste trabalho, uma versão relaxada da *Arc-Time* ([Pessoa et al., 2010](#)), implementada na linguagem C/C++ em conjunto com a biblioteca dinâmica UFFLP e o *solver* CPLEX, por se mostrar mais adequada e adaptável ao modelo construído.

Capítulo 4

Método Híbrido Heurístico-Exato

Proposto

Como exposto no decorrer do trabalho, a minimização das penalidades geradas pelo atraso na execução de tarefas é um assunto importante nos mais diversos ambientes, além de expressar uma rica pesquisa, uma vez que trata de um problema conhecidamente difícil. Durante a construção deste trabalho, a aplicação de uma metodologia utilizando heurísticas e meta-heurísticas se mostrou mais apropriada, uma vez que as instâncias que seria analisadas seriam grandes e de custo computacional alto, porém, como ponto crítico desta abordagem, está a comprovação dos resultados, uma vez que parte das instâncias testadas ainda não dispõem de resultados comparativos disponíveis na literatura. Com essa mentalidade, e visando resolver o problema de minimização do atraso total ponderado em máquinas paralelas, garantindo não apenas a eficácia do código, como também sua eficiência e efetividade, foi realizada a construção de um método híbrido exato-heurístico, aliando as qualidades existentes em ambos os métodos, aproximados e exatos. O uso de abordagens híbridas não são recentes e podem ser vistas no trabalho de [Srinivasan \(1971\)](#), que combinou uma técnica apresentada por [Emmons \(1969\)](#) juntamente com um modelo envolvendo Programação Dinâmica - PD.

O trabalho proposto por [Feili et al. \(2012\)](#), utilizando de algoritmos genéticos, acabou por influenciar a análise deste modelo. Definir bem os parâmetros genéticos, seria imprescindível para a implementação bem sucedida de um AG, além de impactar diretamente nos resultados. Isso oferece uma possibilidade ímpar, uma vez que a calibragem deste algoritmo é parte importante do trabalho e, se bem realizada, oferece boas soluções.

A seleção dos parâmetros, além de comprovadamente complexa, torna-se crucial para o bom desempenho de uma heurística e, em se tratando de AGs, o cenário é ainda mais complicado devido a quantidade de parâmetros a serem verificados. Para contornar este problema, foi avaliada, testada e utilizada uma ferramenta específica voltada para a otimização e definição de parâmetros (citados no Capítulo 2.4), corroborando com a possibilidade de obtenção de boas soluções com implementação realizada.

A ideia de construir uma solução híbrida, mesclando uma abordagem aproximada de um algoritmo exato tornou-se factível e eficaz. Uma heurística responsável por selecionar o conjunto de soluções a serem avaliadas por um método exato, reduzindo o tempo de busca. O método exato, nesse contexto, seria utilizado para obtenção de uma solução melhor do que aquela obtida pela heurística. Ainda que o método exato não consiga apresentar uma melhor solução, o mesmo desempenha o papel de comprovar a qualidade da solução obtida, garantindo assim a robustez do método.

Para tanto, foi utilizada uma formulação matemática já conceituada, cuja aplicação se daria através da utilização de um *solver*. A Figura 4.1 apresenta uma ideia geral da arquitetura proposta para o método híbrido heurístico-exato construído.

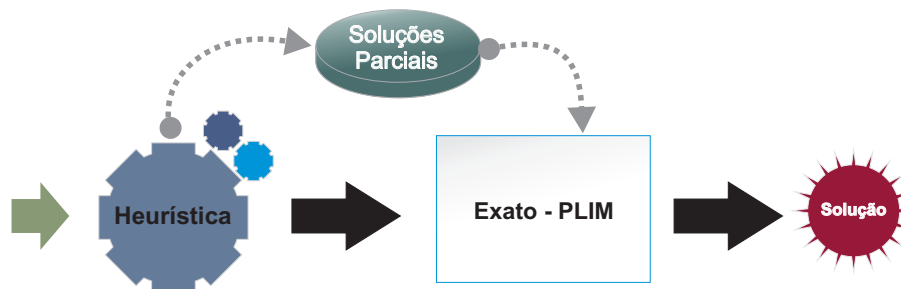


Figura 4.1: Arquitetura geral do método híbrido heurístico-exato proposto.

Como forma de construir o método citado, ficou decidido como passo inicial, a seleção da formulação matemática a ser usada, juntamente com os recursos técnicos (*solver* e ferramentas adicionais) a serem usados em sua aplicação. Isso porque a decisão entre o método aproximado a ser utilizado já havia sido tomada durante a pesquisa, quando foi definida a utilização de um AG. Como passos seguintes, ficou definida a construção do modelo híbrido utilizando as combinações de técnicas estabelecidas (AG + Exato), além da parametrização do método de forma mais trabalhada, e posterior realização de experimentos computacionais mais intensos.

4.1 Escolha da formulação e método de aplicação

A escolha da formulação foi realizada tendo como principais motivadores: *i.* a ideia inicial da formulação, *ii.* sua aplicação a instancias maiores e *iii.* a adequação da mesma, quando relaxada, ao método proposto. Dentre formulações analisadas, a apresentada por [Pessoa et al. \(2010\)](#) teve destaque especial, sendo definida para uso no método híbrido.

A escolha da formulação aos avaliarmos os requisitos envolvidos. A formulação *Arc-Time* atendia a todos os requisitos do problema, além de que, por tratar de mesmo tema trabalhado nesta dissertação, foi marcado como item relevante a ser implementado. Aliado a estes fatos, a formulação também apresentava resultados excelentes quando aplicado em instâncias mais maiores e complexas, seguindo portanto os requisitos pensados inicialmente como parâmetros para definir a formulação. Por mostrar um bom desempenho ao tratar de grandes instâncias, a formulação tornou-se ainda mais relevante no método. Mesmo que o conjunto de soluções analisadas no método híbrido fosse muito menor, ainda assim a formulação teria que avaliar um grande conjunto de informações, visto a complexidade das instâncias avaliadas, que poderiam exigir um alto processamento computacional.

Uma vez selecionada a formulação (que pode ser avaliada melhor na Subseção 3.2.3, localizada na página 44), passamos a avaliar o *solver* a ser aplicado. Surgiram diversas opções muito interessantes, divididos entre opções pagas e opções gratuitas. Dentre as opções destacamos: IBM ILog Cplex ([IBM, 2016](#)) e Gurobi (que apesar de serem soluções proprietárias, possuem versões de avaliação e versões gratuitas para uso acadêmico), Coin CBC e CLPK (código aberto). Devido a quantidade de documentação, bem como ferramental adicional (ambiente gráfico com diversas configurações, análise gráfica e possibilidades de estruturação) o *solver* selecionado foi o IBM ILog Cplex.

De forma a possibilitar modificações futuras, bem como testes com outros *solvers*, foi adotada a ferramenta UFFLp, criada por [Pessoa and Uchoa \(2011\)](#). Ela abstrai a necessidade de conhecer a implementação dentro do *solver*, sendo possível realizar a substituição deste sem a necessidade de modificação do código. Uma vantagem em se trabalhar utilizando essa ferramenta também se dá em situações onde a aplicação do método não exige poder computacional, o que permite que seja utilizado um *solver* de menor capacidade como o Microsoft Excel.

Uma vez decididas as tecnologias e técnicas a serem aplicadas no método híbrido, um esboço mais detalhado pode ser construído, sendo representado pela Figura 4.2.

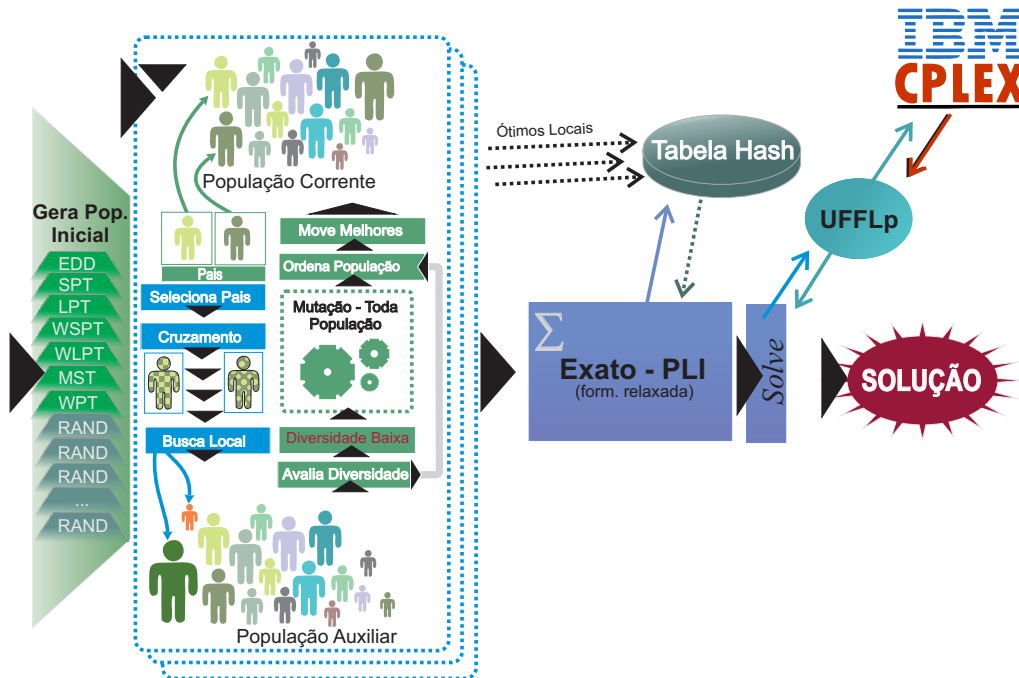


Figura 4.2: Esquema de execução e obtenção de solução pelo método híbrido.

4.2 Genetic + local search - GLS

O GLS é uma abordagem aproximada que envolve a utilização de um Algoritmo Genético (AG), além de ser fortemente baseado em busca local. Essencialmente, logo após uma nova solução ser obtida através dos operadores genéticos (*offspring*), a mesma é passada como parâmetro para um algoritmo de busca local. O algoritmo de busca local utilizado neste trabalho (GPI) foi primeiramente apresentado por de Freitas et al. (2008), que também serviu de inspiração para a **representação** da solução no algoritmo genético, onde cada gene representa uma tarefa a ser executada. Logo, um indivíduo (ou cromossomo) é dado por uma sequência de tarefas. Assim como os demais algoritmos genéticos, o GLS possui algumas operações a serem realizadas, sendo estas descritas nas subseções seguintes.

4.2.1 Geração da população inicial

Algumas etapas são necessárias para a obtenção da solução por parte de um AG, sendo a primeira destas etapas, a geração da população inicial. Nesse ponto as soluções são construídas de forma aleatória, para serem trabalhadas posteriormente por operadores genéticos. Visando gerar boa diversidade de indivíduos, porém garantindo que alguns tenhamos boas soluções iniciais, algumas estratégias foram utilizadas na construção da população.

A população é construída inicialmente utilizando o algoritmo de embaralhamento Fisher-Yates (Fisher, Ronald A.; Yates, 1963), mostrado a seguir. Este algoritmo representa uma forma simples de gerar a população inicial, garantindo que as mesmas serão aleatórias.

Algoritmo 7 Algoritmo de construção da população inicial baseado no algoritmo de Fisher-Yates.

```
population ← GeraPopulation(popSize)
para  $k \leftarrow 1$  até popSize faça
  para  $i \leftarrow 1$  até n faça
    population[k][i] ← i
  fim para
fim para
se  $k < \text{popSize}$  então
  para  $i \leftarrow n$  até 1 faça
     $j \leftarrow$  Valor aleatório entre 0 e  $i$ 
    se  $i \neq j$  então
      SwapValues(population,  $i$ ,  $j$ )
    fim se
  fim para
fim se
```

Uma vez gerada a população inicial, é realizada a avaliação da população. Nessa avaliação, toda a população tem seus valores de aptidão calculados, e ordenados de forma crescente (menor valor de *fitness* primeiro). Logo após essa ordenação, a população sofre uma pequena modificação, onde os 7 piores indivíduos da população são substituídos por heurísticas específicas relacionadas abaixo.

- a) *Earliest due-date* (EDD, onde as tarefas são ordenadas de forma crescente pelo *due-date*, ou tempo estimado de conclusão da tarefa).
- b) *Shortest processing time* (SPT, onde as tarefas são ordenadas de forma crescente

pelos tempos de processamento).

- c) *Longest processing time* (LPT, onde as tarefas são ordenadas de forma decrescente pelos tempos de processamento).
- d) *Weighted shortest processing time* (WSPT, onde as tarefas são ordenadas de forma crescente pela proporção entre a penalidade aplicada ao atraso da tarefa e seu tempo de processamento, ou seja, w_j/p_j).
- e) *Weighted longest processing time* (WLPT, onde as tarefas são ordenadas de forma decrescente pela proporção entre a penalidade aplicada ao atraso da tarefa e seu tempo de processamento).
- f) *Minimum slack time* (MST, onde as tarefas são ordenadas de forma crescente pela diferença entre o tempo estimado de conclusão da tarefa e seu tempo de processamento, ou seja, $d_j - p_j$).
- g) *Weighted processing time* (WPT, onde as tarefas são ordenadas de forma crescente pelo produto do penalidade aplicada ao atraso da tarefa e seu tempo de processamento, ou seja, $w_j * p_j$).

Após a substituição das piores soluções pelas heurísticas citadas, os novos indivíduos tem seus valores de aptidão calculados e novamente é realizada a ordenação das soluções, porém agora, também é registrado um arco (para uso no método exato) que representa a melhor solução obtida. Detalhe importante é que a quantidade de indivíduos de uma população é um dos parâmetros que pode ser ajustado na intenção de obter melhores resultados. O algoritmo de construção da população inicial, portanto, pode ser representado pelo apresentado abaixo.

Após a geração e avaliação da população inicial, o indivíduo com o menor *fitness*, isto é, melhor função objetivo (FO) é selecionado como melhor solução. A Figura 4.3 representa a geração de uma população inicial, construída a partir da estratégia descrita, onde um dos indivíduos com menor *fitness* é selecionado como melhor solução. Na imagem, o item em destaque (*EDD*) representa o indivíduo que obteve a melhor solução dentro da população existente. Mesmo o EDD sendo uma regra capaz de obter a solução ótima para o problema que avalia unicamente o total de atraso ($\sum T_j$), o mesmo não ocorre no problema tratado nesta proposta, que deve considerar além do atraso, os pesos a serem

Algoritmo 8 Algoritmo de construção da população inicial.

Gera população usando método Fisher-Yates.

para $k \leftarrow 1$ **até** $popSize$ **faça**

 Calcula valor de *fitness* para $population[k]$

fim para

 Ordena população.

 Constroi cromossomos usando EDD, SPT, LPT, WSPT, WLPT

para $k \leftarrow popSize - 6$ **até** $popSize$ **faça**

 Calcula valor de *fitness* para $population[k]$

fim para

 Ordena população.

$melhorSolucao \leftarrow population[1]$

 Registra arcos de toda população *population*

aplicados, que variam conforme a tarefa. Desta forma, o destaque dado na imagem a regra EDD é apenas a ilustração da criação da população com a seleção do melhor indivíduo, que pode variar conforme a instância que esta sendo resolvida.

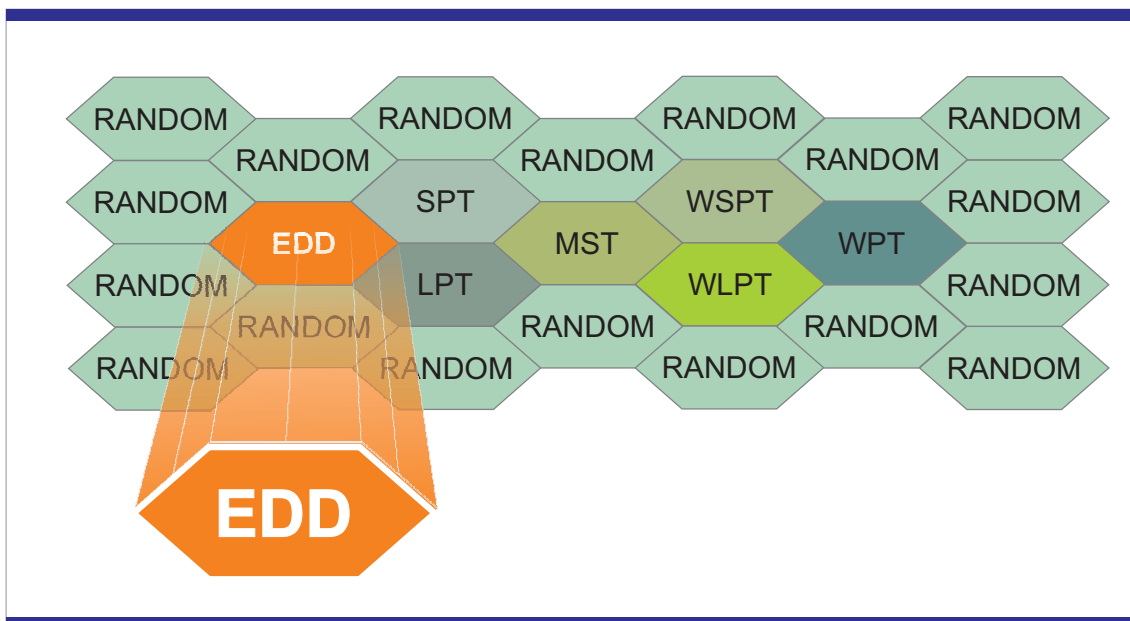


Figura 4.3: População inicial gerada com seleção do melhor indivíduo (EDD).

Após avaliar toda a população e selecionar a melhor solução o procedimento entra em um laço, onde novas soluções são geradas a partir de operadores genéticos de cruzamento e mutação, etapas estas citadas em subseções posteriores.

4.2.2 Função de aptidão ou *fitness*

Como forma de elucidar o cálculo do valor de *fitness* de uma solução, cabe observar os dados da instância de exemplo apresentada na Figura 4.4.

J_i	p_i	d_i	w_i
J1	3	3	5
J2	2	6	5
J3	3	6	8
J4	4	4	10
J5	6	6	4
J6	7	10	3
J7	3	11	2
J8	3	8	8

Figura 4.4: Dados de uma instância modelo.

Para o cálculo da função de *fitness* do AG, cada uma das tarefas é atribuída a primeira máquina disponível, e assim, avaliado o cenário que pode ou não acarretar em uma penalidade relacionada ao atraso de uma tarefa. Caso sejam gerados atrasos, os mesmos são somados e o valor é retornado. O valor desta distribuição pode ser visualizado na Figura 4.5, onde é apresentada a forma de alocação de uma tarefa em uma dada máquina a partir do cromossomo utilizado.

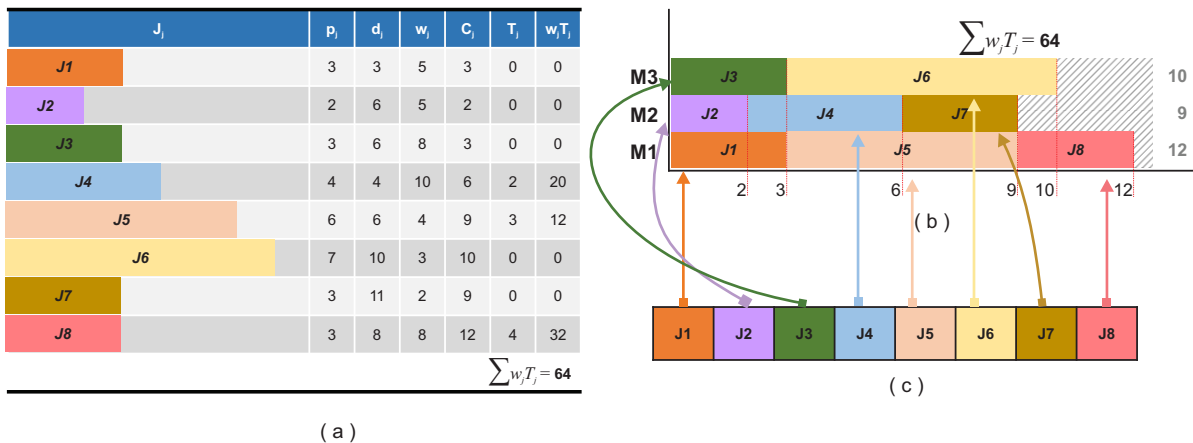


Figura 4.5: Distribuição dos genes em máquinas e cálculo do atraso total ponderado.

4.2.3 Operador genético de cruzamento

Após a construção da população inicial e avaliação das soluções, é aplicado o operador genético de cruzamento em busca da obtenção de indivíduos melhores (condição em que

o valor da FO é menor do que os valores disponíveis). Os novos indivíduos (*offspring*) possuem portanto característica de dois indivíduos pertencentes a geração corrente. A escolha destes dois indivíduos é dada a partir da seleção por torneio. O método de seleção utilizado foi escolhido devido ao mesmo possuir um melhor desempenho em relação ao método de seleção por roleta. O Algoritmo 6 (página 30) apresenta um pseudo-código deste tipo de seleção. O código é executado duas vezes, uma para cada "pai" que será utilizado pelo operador de cruzamento. Após o término dos *rounds* do torneio, os dois melhores indivíduos (com menor *fitness*) são escolhidos.

Selecionados os pais, é indicado qual dos dois será o principal. Para isso, os pais selecionados são comparados entre si, e aquele com menor *fitness* (π_i) é indicado como principal (sendo o outro indivíduo indicado por π_j). Tal separação é apenas visando elucidar o método, uma vez que são gerados 2 novos indivíduos a partir do cruzamento, cada um deles utilizando um pai como principal. Cabe ressaltar que foi utilizado o cruzamento uniforme como método de reprodução. Tal opção foi selecionada após experimentos, obtendo melhor resultado que o cruzamento em um ponto, e o cruzamento e dois pontos.

Selecionados os pais, é gerada (aleatoriamente) uma sequência binária que representa as características predominantes de cada pai a permanecerem em cada um dos novos indivíduos. Mantidas as características predominantes, o cromossomo é complementado a partir das características de π_j (para o *offspring 1*), aplicando apenas os genes ainda não inseridos, e na ordem em que aparecem (para o *offspring 2* as características complementares são obtidas de π_i). Um exemplo de cruzamento pode ser visto na Figura 4.6.

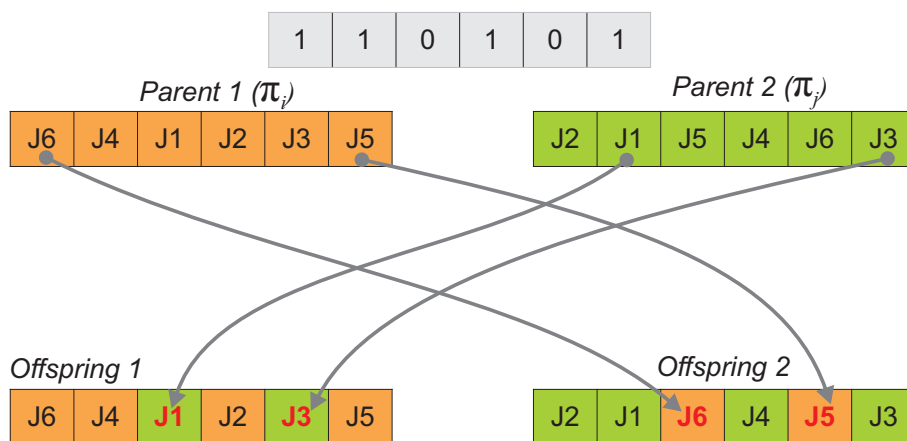


Figura 4.6: Criação de novos indivíduos através do cruzamento uniforme.

Após a geração dos novos indivíduos, cada um deles é passado como parâmetro para um algoritmo de busca local visando obter melhoria na solução encontrada. Após a con-

clusão da busca local, as soluções são classificadas, onde é identificada a **melhor** e a **pior** solução chamadas respectivamente de s_m e s_p (a comparação é baseado no valor de *fitness* de cada uma delas). Se o indivíduo s_m for mais apto que o melhor cromossomo encontrado até o momento, s_m assume a posição de melhor indivíduo da população e o mesmo é registrado como arco para ser avaliado pelo método exato. Caso contrário, é calculada a taxa de mutação para este cromossomo e, caso necessário, a mutação é aplicada para s_m . Também é calculado o valor da taxa de mutação para a solução s_p e também aplicada a mutação se o valor obtido for maior que a taxa de mutação indicada como parâmetro. Após a execução destas rotinas, os novos indivíduos são adicionados na população. O pseudocódigo de reprodução pode ser visto no Algoritmo 9.

Algoritmo 9 Cruzamento Uniforme com Busca Local.

```

Seleciona  $\pi_i$  e  $\pi_j$  pelo Método de Seleção Torneio
Cria dois novos cromossomos vazios  $o_i$  e  $o_j$ 
 $m \leftarrow$  Mascara aleatória de valores binários
Copia genes de  $\pi_i$  para  $o_i$  conforme  $m$ 
Copia genes de  $\pi_j$  para  $o_j$  conforme  $m$ 
Complementa cromossomo  $o_i$  com valores restantes vindos de  $\pi_j$ 
Complementa cromossomo  $o_j$  com valores restantes vindos de  $\pi_i$ 
Realiza busca local em  $o_i$  e  $o_j$ )
 $s_m \leftarrow$  Melhor solução entre  $o_i$  e  $o_j$ 
 $s_p \leftarrow$  Melhor solução entre  $o_i$  e  $o_j$ 
se  $fitness(s_m) < fitness(melhorSolucao)$  então
     $s_m$  substitui a melhor solução corrente
    Armazena a nova melhor solução para ser utilizada no método exato
senão
     $t \leftarrow$  Valor aleatorio entre 0 e 1
    se  $t > taxaMutacao$  então
        Aplica Mutacao na Solução( $s_m$ )
    fim se
fim se
 $t \leftarrow$  Valor aleatorio entre 0 e 1
se  $t > taxaMutacao$  então
    Aplica Mutacao na Solução( $s_p$ )
fim se

```

4.2.4 Busca local iterada - GPI

A busca local iterada aplicada neste trabalho foi proposta por de Freitas et al. (2008), onde é realizada uma busca local na vizinhança por movimentos troca generalizada de pares (GPI). Basicamente movimentos GPI são dados por dois tipos de operações, sendo

a primeira deles o movimento de troca (*swap*) e a segunda dada pela remoção e reinserção de uma tarefa (*move*). Na operação de *swap* é realizada a troca de duas tarefas i e j , não necessariamente adjacentes (Figura 4.7)

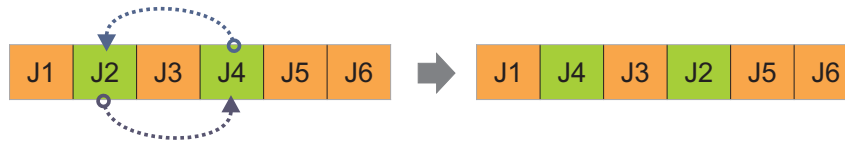


Figura 4.7: Movimento *swap* realizado pela busca local.

No operação *move* é selecionada uma tarefa da posição i e inserida na posição j , movendo assim outras tarefas que estão situadas entre as posições i e j , como apresentado pela Figura 4.8.



Figura 4.8: Movimento *move* realizado pela busca local.

As operações são realizadas até que não seja mais obtida melhora da solução corrente, sendo que, uma busca completa na vizinhança tem o custo de $O(n^2)$ movimentos GPI (além do custo de conversão de uma lista sequencial em escalonamento para máquinas paralelas que é de $O(n \log_m)$).

Seja um escalonamento π , representado por uma lista sequencial de, uma solução inicial do problema $P || \sum w_j T_j$, onde todas as máquinas estão disponíveis no instante 0. A busca da vizinhança quando aplicada a π termina ao ser encontrada uma dada solução π' melhor que a atual, situação esta em que a busca é reiniciada, onde π recebe o valor de π' . A busca completa é dada como concluída se após uma quantidade N de iterações não houver melhoria da solução. Ocorre que em uma busca local, diversas soluções semelhantes podem ser encontradas em uma vizinhança. Uma forma de evitar que o código fique preso a ótimos locais é através da aplicação de uma fórmula, dada de forma a realizar uma comparação em casos onde o custo de duas soluções é o mesmo. A fórmula é dada por $b(\pi) = \sum_{j=1}^n d_{\pi_j} x(n - j + 1)$, sendo d_{π_j} a data de término sugerida (*due date*) das tarefas, n o número de tarefas e x representando o número de iterações. O processo de busca local é repetido por x iterações (sendo $x = knm$, com m indicando a quantidade de máquinas disponíveis e k uma certa constante).

Quando o número da iteração corrente é múltiplo de uma dada constante r , a busca local é reiniciada a partir de uma nova sequência de passos. Esta ação é tomada na tentativa de evitar que busca seja capaz de fugir de ótimos locais, explorando assim outras regiões do espaço de busca. O Algoritmo 10 apresenta de forma resumida os passos aplicados na busca local.

Algoritmo 10 Algoritmo de busca local de de Freitas et al. (2008).

```

 $i \leftarrow 1$ 
enquanto  $i < N$  faça
  se  $i$  é múltiplo de  $r$  então
    Reinicia permutações
  fim se
  Aplica movimentos GPI até não ser possível melhorar a solução
  se valor de  $\pi'$  for melhor que o valor de  $\pi$  então
     $\pi'$  substitui solução corrente
  fim se
  Aplica  $k$  movimentos de troca escolhidos aleatoriamente
   $i \leftarrow i + 1$ 
fim enquanto
retorne  $\pi$ 

```

4.2.5 Operador genético de mutação

O operador genético de mutação exerce uma função importante dentro de um algoritmo genético. Além de produzir novos indivíduos, continuando a evolução da população, ele atua com o objetivo de fugir de ótimos locais. Na estratégia proposta a mutação é aplicada selecionando-se aleatoriamente duas posições da sequência de tarefas que compõem o cromossomo, i e j . Então é realizada a permutação destes genes, com o gene localizado na posição i sendo movido para a posição j , ao tempo que o gene da posição j vai para a posição i . Essa operação de permuta de genes é realizada uma quantidade q de vezes. A Figura 4.9 demonstra a execução do operador de mutação sendo aplicado com um valor de $q = 2$, ou seja, duas repetições de permutação.

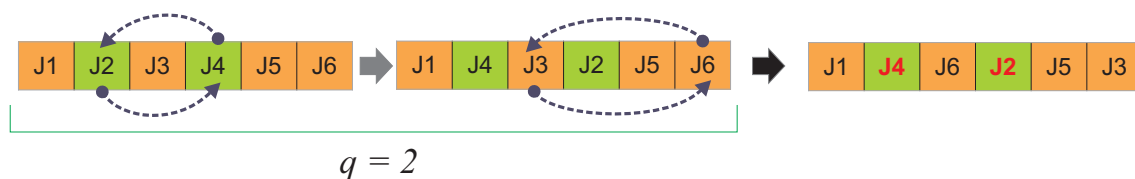


Figura 4.9: Operador genético de mutação com $q = 2$.

Após a realização das permutações, é aplicada uma busca local (GPI) sobre o indivíduo que sofreu a mutação e, caso o mesmo obtenha um valor de *fitness* melhor do que a solução vigente, este assume a posição de melhor solução e é armazenada em forma de arco para ser utilizado no método exato.

A mutação é dada em apenas dois momentos: logo após a aplicação do operador genético de cruzamento, se a taxa de mutação for atingida, e quando a diversidade populacional estiver baixa. Em ambos os casos, a ocorrência da mutação é realizada sob situações específicas. Para a ocorrência após a operação de reprodução, é necessário que o valor da probabilidade de mutação de um dado indivíduo seja superior a taxa de mutação. A **taxa de mutação** verificada para a aplicação da mutação é um parâmetro que exige cuidado ao ser aplicado: se for muito alto pode não ocorrer a mutação e o algoritmo ficar preso em ótimos locais; se for muito baixo a ocorrência de mutação muito frequente pode impactar negativamente por fazer com que um indivíduo perca características importantes adquiridas, sendo portanto, necessária a realização de uma avaliação cuidadosa acerca deste percentual. Tal cuidado também é necessário ao se definir a quantidade de repetições da operação de permuta, uma vez que, se for muito alta, também impacta negativamente gerando perda de características. O Algoritmo 11 apresenta um modelo sintetizado do operador de mutação.

Algoritmo 11 Algoritmo do operador genético de mutação.

```
Dada um indivíduo  $\pi$ 
 $q \leftarrow$  Número de repetições
 $i \leftarrow 1$ 
enquanto  $i < q$  faça
    Atribui dois valores aleatórios distintos para  $i$  e  $j$ 
    Troca o gene da posição  $i$  com o gene da posição  $j$  do indivíduo  $\pi$ 
     $i \leftarrow i + 1$ 
fim enquanto
Calcula valor de fitness de  $\pi$ 
Executa busca local na solução  $\pi$ 
se  $fitness(s_m) < fitness(melhorSolucao)$  então
     $\pi$  substitui a melhor solução corrente
    Armazena a nova melhor solução para ser utilizada no método exato
fim se
retorne  $\pi$ 
```

4.2.6 Diversidade populacional

Um problema conhecido dos algoritmos genéticos é a convergência prematura, fazendo com que o método fique preso a ótimos locais, ou não seja capaz de gerar novos indivíduos que levem a explorar novos espaços de busca. Após repetidas operações de cruzamento e mutação, por vezes a diversidade da população cai, reduzindo a capacidade de produção de novos indivíduos com características distintas, e assim, a dificultando a obtenção de melhores soluções. A diversidade populacional é o indicador que avalia a repetição dos genes em posições semelhantes entre um conjunto de membros.

Sempre que um novo indivíduo é introduzido na população, a mesma passa por uma análise que compara a semelhança entre todos os membros, avaliando para isso, cada um dos genes. Como resultado dessa operação temos um percentual d_p indicando o grau de diversidade da população atual. Como forma de controle, é considerado um valor de diversidade mínima d_m , de forma que, quando a diversidade da população está abaixo da diversidade mínima (isto é $d_p < d_m$), é aplicada a mutação a alguns indivíduos da população, fazendo com que o grau de diversidade volte a valores aceitáveis.

Essa garantia de diversidade permite que o processo siga com condições de produzir novas soluções, explorando o espaço de busca.

4.2.7 Elitismo e novas gerações

Se por um lado a diversidade baixa pode acarretar em problemas, a perda de dos melhores cromossomos de uma população também, uma vez que sem a transferência das melhores soluções de uma geração para outra, um AG poderia ser comparado a uma busca aleatória. No método construído, logo após a realização das operações de cruzamento e mutação, toda a população é analisada, isto é, tem seus valores de *fitness* calculados. Logo após, os cromossomos são ordenados, ficando o indivíduo mais apto na primeira posição da população. A população nesse ponto tem duas vezes o tamanho normal, pois as soluções de uma geração não são completamente substituídas pelos novos cromossomos gerados.

Metade da população é passada para a geração seguinte. Mais precisamente, os melhores indivíduos da população. A cada geração, o processo de elitismo é repetido buscando assim a evolução a cada nova geração. O número de gerações varia conforme a instância, sendo este um parâmetro também indicado no início da execução do método. Ao término

da última geração, todas as soluções contidas na população são armazenadas em formato de arcos para posterior execução do método exato.

4.3 Armazenamento de valores para método exato

A cada novo indivíduo obtido, cujo valor de sua FO seja a melhor encontrada, o escalonamento que esta solução representa é armazenada em forma de arcos para ser utilizada pelo método exato. Um arco dentro deste contexto, indica a alocação de uma dada tarefa a uma máquina dentro do tempo, onde o valor desse arco (variável tri-indexado) é representado pela tarefa atual, a tarefa futura e o tempo. Como forma de armazenar os arcos, é utilizada uma tabela de dispersão (ou tabela *hash*) com endereçamento aberto como tratamento de colisões. A Figura 4.10 apresenta um conjunto de arcos obtidos a partir de um conjunto de três soluções usando a formulação citada.

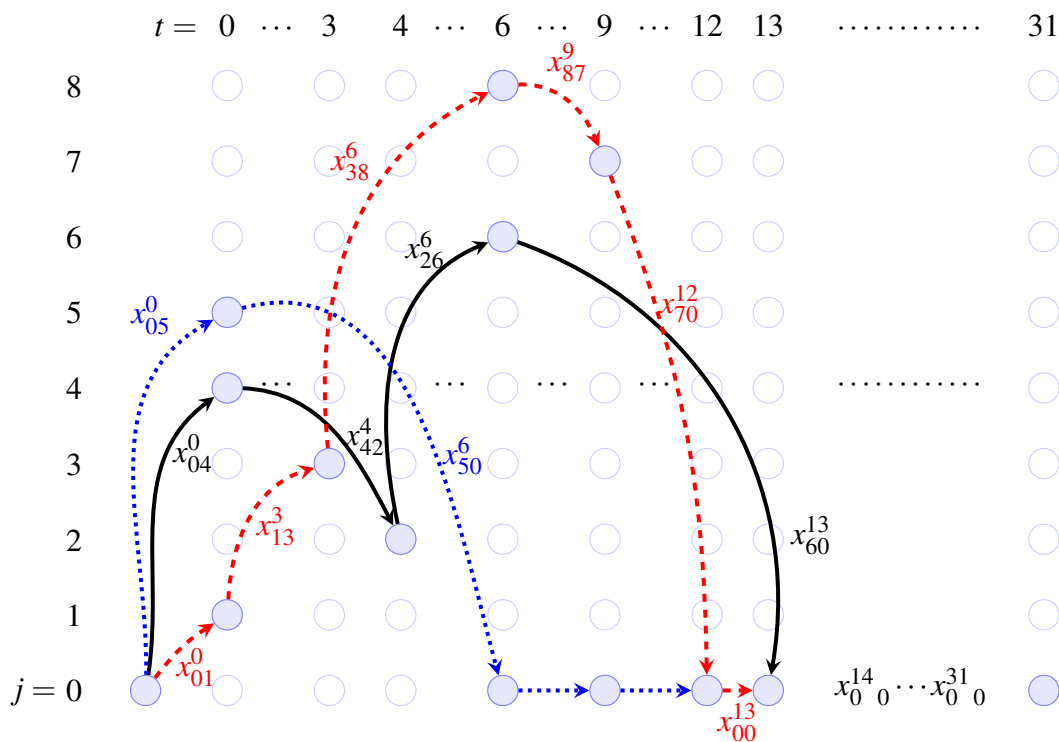


Figura 4.10: Representação do escalonamento no modelo de fluxo em redes.

Fonte: Amorim (2013).

A tabela *hash* criada para conter os arcos é construída com um tamanho máximo predeterminado, isso para evitar que o método exato seja sobrecarregado com um alta quantidade de soluções, que poderia acarretar na ineficiência do método. A função responsável por adicionar os arcos avalia a disponibilidade de espaço dentro da tabela *hash*

e, havendo espaço disponível, a solução é adicionada, caso contrário, é feita uma avaliação das soluções já armazenadas e, existindo um arco com valor da FO pior que a nova solução, o mesmo é removido, dando espaço para a nova solução encontrada, garantindo que apenas as melhores soluções farão parte do espaço de busca do método exato.

Durante o período de experimentação, foi avaliado que o tempo utilizado pelo método exato estava baixo, sendo que um dos fatores que acarretavam tal situação era a baixa quantidade de arcos acumulados. Como medida, todos os arcos de cada geração (incluindo os arcos da população inicial) são também adicionados. Dessa forma são armazenados em forma de arcos todos os indivíduos da população inicial e de cada uma das gerações (após a aplicação do método responsável pelo elitismo), além das melhores soluções que foram sendo encontradas durante a execução do método heurístico. Com isso, o espaço de busca do método exato passou a ser mais eficiente sem sobrecarga do método.

Após o encerramento do procedimento heurístico, é dado início a execução do método exato, citado na seção seguinte.

4.4 Aplicação do método exato

As soluções armazenadas em forma de arcos são avaliadas usando através do algoritmo *branch-and-cut*, utilizado pelo *solver* Cplex IBM (2016), tendo como apoio na sua utilização e manipulação a ferramenta UFFLP, criada por Pessoa and Uchoa (2011).

Para a aplicação do método exato, foi utilizada uma formulação relaxada do problema utilizando um modelo de programação linear inteira. A formulação utilizada (*Arc-time indexed formulation*) foi apresentada por Pessoa et al. (2010), apresentada na subseção 3.2.3 (página 44) da presente dissertação. Nesta formulação, são aplicadas restrições de forma que atendam a um determinado conjunto de arcos que foram registrados durante o processo executado pelo *GLS*.

Conforme os arcos vão sendo utilizados na construção da formulação, os mesmos são removidos da tabela *hash*, evitando assim que sejam reavaliadas soluções semelhantes. A solução obtida pela formulação (e avaliada pelo *solver*) pode obter como retorno 2 valores distintos:

1. Ineficaz: quando a solução gerada pela formulação não pode ter um resultado

válido.

2. Calculado: quando é obtida uma solução a partir da formulação.

Em casos onde a solução é dada como calculada, o valor da FO obtida é comparada ao melhor resultado obtido anteriormente (pelo método heurístico ou outra execução do método exato). Nesta situação, podem ocorrer casos onde a formulação exata confirma a solução encontrada, de forma que, além de validar o método heurístico aplicado, ainda o denota como um bom resultado, ou ainda, consegue encontrar uma solução melhor a partir dos arcos registrados.

Se existirem arcos disponíveis para realizar nova análise após o processo descrito acima, o procedimento de geração da formulação e análise por parte do *solver* é realizado novamente, e se repete até que todos os arcos registrados tenham sido avaliados.

4.5 Pacote *iRace* e Parametrização da execução

O pacote *iRace* implementa o procedimento chamado *iterated racing*, uma extensão do *Iterated F-race (I/F-Race)* proposta por (Balaprakash et al., 2007). Implementado como um pacote da linguagem R¹, tem como principal objetivo fornecer os parâmetros mais adequados para algoritmos voltados a problemas de otimização. Seu funcionamento consiste em, indicados um conjunto de instâncias e parâmetros, o pacote executa o algoritmo com parâmetros iniciais e posteriormente testa os mesmos parâmetros em outras instâncias, na sequência, variando novamente os parâmetros. A cada teste os dados relacionados a execução do algoritmo são armazenados para comparação futura. Ao final da execução, o *iRace* determina os parâmetros baseado em suas análises (distribuição de probabilidades). Como análise dos parâmetros e seus resultados, o *iRace* faz uso de técnicas específicas de análise como por exemplo Teste-T e Friedman.

Para a execução do *iRace* é necessária a configuração de alguns arquivos relacionados ao problema e instâncias, sendo divididos, principalmente, em um arquivo relacionado ao cenário, responsável por indicar o local das instâncias de teste e de treino (se forem diferentes), tempo máximo de execução, quantidade de iterações, tipo de teste estatístico a ser aplicado, quantidade de repetições e semente a ser aplicada (para permitir a reprodução de ambiente). O segundo arquivo principal é responsável por descrever quais os parâmetros

¹<https://www.r-project.org/>

são aceitos pelo algoritmo a ser avaliado, indicando a forma de aplicação, forma de variação e limites a serem testados para cada um dos parâmetros. Na Figura 4.11 apresenta um exemplo de arquivo de parâmetros utilizado pelo *iRace*.

```

19 ## For i,r: a pair of unquoted numbers representing minimum and
20 ## maximum values.
21
22 ## 5: A conditional parameter can be defined according to the values of
23 ## one or several other parameters. This is done by adding a
24 ## character '|' followed by an R expression involving the names of
25 ## other parameters. This expression must return TRUE if the
26 ## condition is satisfied, FALSE otherwise.
27 ## timeMIP      ""      o (1800, 2000, 2500, 3600, 4800, 6400,
28 ## para timeGLS coloquei para validar de 30' a 10h, pegando de meia em meia hor
29
30 # 1:           2:           3: 4:           5:
31 timeGLS      ""           o (800, 2400)
32 timeMIP      ""           o (800, 1200)
33 repeat       ""           i (1, 5)
34 diversity    ""           r (0.5, 0.95)
35 popSize      ""           c (10, 15, 20, 40)
36 perc         ""           c (1)

```

Figura 4.11: Arquivo de parâmetros do *iRace*.

Visando obter os melhores resultados, bem como a eficiência do método, diversos parâmetros foram aplicados na execução dos métodos, principalmente no que diz respeito ao método aproximado. A escolha dos parâmetros, por si só, já é conhecidamente complexa, o que pode dificultar a pesquisa, inviabilizando-a inclusive, se não for realizada adequadamente. Desta forma foi utilizada a ferramenta de otimização de parâmetros, *iRace* (apresentada na seção 2.4). Foram avaliados pela ferramenta os seguintes parâmetros:

- a) Tempo máximo de execução do método heurístico (dado em segundos).
- b) Tempo máximo de execução do método exato (dado em segundos).
- c) Quantidade de repetições a ser realizada.
- d) Similaridade máxima permitida (aplicada para análise da diversidade adaptativa).
- e) Tamanho da população.
- f) Número máximo de gerações.
- g) Probabilidade de mutação.

Para cada um destes valores foram aplicadas variações de forma a permitir calibrar o método, além de possibilitar que o método seja melhor utilizado em cenários que apresentem mudanças de *hardware* (aumento ou diminuição de recursos), uma vez que, existindo a possibilidade de testes em computadores com maior capacidade de processamento e

memória, soluções mais robustas poderiam ser exploradas, aumentando a quantidade de arcos a serem armazenados, assim como o tamanho da população e número de gerações.

Com relação aos itens *a* e *b* indicados acima (tempo de execução), a ferramenta não estimou valores, uma vez que aumentando o tempo, havia melhora nas soluções. Estes parâmetros portanto, foram fixados, e posteriormente removidos da análise do *iRace*. O mesmo ocorreu com o parâmetro responsável por indicar a quantidade de repetições, onde foi fixado um parâmetro limitando a 10 repetições.

Após a análise da ferramenta, os parâmetros ficaram assim definidos:

Tabela 4.1: Parâmetros utilizados na execução do híbrido.

Parâmetro	n	Valor
Tempo máximo de execução do método heurístico	≤ 200	1 hora
	> 200	2 horas
Tempo máximo de execução do método exato	≤ 200	1 hora
	> 200	2 horas
Quantidade de repetições por instância	*	10
Siimilaridade máxima permitida na população (heurística)	*	73%
Tamanho da População	*	87
Número máximo de gerações	*	100

Durante a fase de experimentação outros parâmetros foram testados devido ao refinamento realizado no método, principalmente, refinamentos realizados no método heurístico. Alguns destes parâmetros tem relação com a estratégia de resolução (tipo de seleção de cromossomos para cruzamento e tipo de cruzamento), outros relacionados a parametrização específica para cada estratégia (probabilidade de mutação, quantidade máxima de gerações sem modificação - usado na aplicação da mutação). Para estes parâmetros, a seleção foi realizada de forma empírica, comparando resultados obtidos com cada implementação.

O método de seleção de cromossomos para cruzamento que foi utilizado (Seleção por Torneio) ocorreu devido ao desempenho. A necessidade de explorar um maior conjunto de soluções em uma mesma porção de tempo fez necessário um maior desempenho. Mesmo ignorando tal ponto, ao avaliar empiricamente os resultados obtidos comparando ambas implementações dos métodos de seleção, a Seleção por Roleta apresentava resultados piores do que aqueles obtidos quando utilizado o método de Seleção por Torneio.

De forma semelhante ocorreu com a escolha no tipo de cruzamento, onde foram avaliados os métodos de Corte em Um Ponto, Corte em Dois Pontos e Cruzamento Uniforme. Ao avaliar os três métodos, foram comparados os resultados obtidos em cada um deles, onde optamos pelo uso do método de cruzamento uniforme por propiciar melhores resultados, além de prover mudanças mais significativas no cromossomo, logo, permitindo explorar um maior espaço de soluções e assim obter melhores resultados.

4.6 Considerações sobre o método

Uma forma resumida de apresentar a execução do método híbrido pode ser dado pelos seguintes passos:

1. É iniciado o método usando a abordagem aproximada (método heurístico).
 - 1.1. O AG constrói a população inicial usando heurísticas específicas (elencadas na Subseção 4.2.1), e a população é preenchida com indivíduos aleatórios diferentes.
 - 1.2. A FO de todas as soluções é avaliada e selecionada a melhor solução dentre elas, definindo o limite superior.
 - 1.3. Todas as soluções são armazenadas em forma de arcos.
 - 1.4. É dado início a parte de construção de gerações.
 - 1.5. É construída uma população auxiliar com o dobro do tamanho da população atual.
 - 1.6. Toda a população atual é copiada para uma população auxiliar.
 - 1.7. São aplicadas diversas operações de cruzamento.
 - 1.7.1. São selecionados dois indivíduos através de Torneio.
 - 1.7.2. É realizada a operação de Cruzamento Uniforme, produzindo 2 novos indivíduos.
 - 1.7.3. Os indivíduos passam por uma busca local (movimentos GPI) e adicionados na população auxiliar.
 - 1.7.4. Se um indivíduo possui melhor valor de *fitness*, assume como melhor solução (limite superior), e é armazenado em forma de arco (*job-job-time*).

- 1.8. É verificado o valor da similaridade dos membros da população a cada operação de cruzamento.
 - 1.8.1. É calculado o valor da similaridade entre todos os indivíduos da população.
 - 1.8.2. Se a diversidade está baixa, todas as soluções, exceto a melhor da população, passam por mutação.
 - 1.8.3. Sempre que um item passa por mutação, é novamente aplicada a busca local.
 - 1.8.4. Se algum dos indivíduos que passaram por mutação obteve um valor de *fitness* melhor que a solução corrente, este é armazenado em forma de arco e um novo limite superior é atribuído.
- 1.9. O processo de cruzamento e mutação (Itens 1.7. e 1.8., respectivamente) ocorrem até que toda a população auxiliar seja preenchida.
- 1.10. Ao final de uma geração, ocorre uma seleção de indivíduos na população auxiliar, onde os melhores são copiados para a população corrente, enquanto que os demais são descartados (melhor baseado no valor de *fitness*).
- 1.11. Todos os indivíduos da população são armazenados em forma de arcos.
- 1.12. O ciclo recomeça a partir do item 1.5., encerrando ao atingir um limite máximo de gerações ou tenha esgotado o tempo máximo para processamento.
2. É concluída a parte do método aproximado dando início ao método exato.
 - 2.1. É solicitado um conjunto de arcos, e estes são inseridos na formulação.
 - 2.2. Modelo é carregado pela biblioteca UFFLp no IBM ILog CPLEX.
 - 2.3. Se o resultado obtido é comparado com o valor da solução corrente.
 - 2.4. Se existem mais arcos a serem explorados, procedimento reinicia a partir do Item 2.1., caso contrário, encerra o subprocesso.
3. O algoritmo reporta a solução obtida (armazena em arquivo) e encerra sua execução.

Além do ferramental tradicional, e análise de parâmetros feitas utilizando a ferramenta, diversos testes realizados puderam prover mudanças nos algoritmos empregados

no método. Algumas das mudanças realizadas no código foram indicadas por uma ferramenta de Análise de Código e Cobertura de Erros chamada *PVS-Studio*. Outras mudanças foram realizadas para permitir o uso da ferramenta *iRace*. Um outro ajuste importante é a possibilidade de realizar o rastreamento das soluções obtidas, onde ao final do método híbrido, cada um dos valores obtidos pela função de *fitness* é armazenado em um arquivo (indiferente de serem melhores ou não), permitindo avaliar o comportamento do método durante sua execução (variação de soluções entre outras informações).

Outras abordagens foram realizadas com a finalidade de obter melhores soluções e assim ampliar a capacidade do exato de analisar novos espaços de soluções tais como: mudança da formulação, aumento na coleta de arcos durante a execução do método aproximado, armazenamento de arcos aleatórios, além de mudanças de implementação do modelo exato onde foi implementado um algoritmo de *branch-and-bound* para avaliação os arcos. Todos os testes realizados, bem como as modificações seguidas foram realizadas objetivando a obtenção de melhores soluções, dentro de um menor espaço de tempo. O híbrido mostrou-se relevante e competitivo, além de dar suporte e base para melhorias que podem prover ganhos mais relevantes para instâncias maiores (instâncias com 500 tarefas).

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados obtidos através de experimentos realizados na aplicação do método proposto, objetivando tratar do problema de minimização da soma de penalidades aplicadas na ocorrência de atrasos no processamento de tarefas em um ambiente de máquinas paralelas. Os resultados apresentados nas Tabelas A.1, A.2, A.4, A.5, A.7 e A.8 foram obtidos executando o híbrido e comparando com os resultados gerados pelos métodos UILS (*unified heuristic, based on local search* proposto por [Kramer and Subramanian \(2015\)](#)) e BCP (*branch-cut-and-price* [Pessoa et al. \(2010\)](#)) e também com os resultados reportados por [Della Croce et al. \(2012\)](#). Neste comparativo ficou demonstrado que o método proposto é competitivo com as principais abordagens disponíveis na literatura voltadas ao problema em questão, indicando inclusive, resultados muito bons para instâncias maiores (acima de 100 *jobs*). Nas Tabelas A.3, A.6 e A.9 a comparação pode ser realizada com os resultados apresentados por [Della Croce et al. \(2012\)](#), onde novamente obteve bons resultados, sendo que nas instâncias com 100 *jobs* distribuídos em 10 máquinas, foram obtidos resultados melhores do que os relatados em 8 das instâncias analisadas, tendo resultado menor apenas em uma instância (obtendo resultado equivalente ao relatado na literatura nas outras 16 instâncias). Na execução do método para instâncias maiores, com 300 *jobs* para duas (Tabela A.16), quatro (Tabela A.17), dez (Tabela A.18) e vinte (Tabela A.19) máquinas, os resultados obtidos, quando comparados com resultados existentes na literatura [Della Croce et al. \(2012\)](#), também apresentaram um bom resultado: para 300 *jobs* e 2 máquinas ficou melhor em 10 instâncias, para 300 *jobs* e 4 máquinas ficou melhor em 10 instâncias, para 300 *jobs*, 10 e 20 máquinas, o método híbrido ficou melhor em 11. Também são relatados resultados obtidos em instâncias maiores com 150

e 200 *jobs* para 2, 4 e 10 máquinas (Tabelas A.10, A.11, A.12, A.13, A.14 e A.15), e com 500 *jobs* para 2, 4, 10 e 20 máquinas (Tabelas A.20, A.21, A.22 e A.23). Dentre os resultados apresentados, alguns não puderam ser comparados devido a indisponibilidade de resultados para tal. Do mesmo modo são apresentados no presente trabalho de modo a corroborar com a eficiência do método proposto e contribuir para pesquisas futuras em mesma área.

5.1 Ambiente computacional

Os experimentos e resultados computacionais relatadas no presente trabalho foram executados em um computador com processador Intel(R) Core(TM) i7-6700HQ de 2.6 GHz, 4 núcleos, 8 processadores lógicos, 16 GB de memória RAM, disco rígido de 1TB, sistema operacional Windows 10 Home 64 *bits*. Todos os algoritmos utilizados foram construídos na linguagem C/C++. Foi utilizada a ferramenta/biblioteca UFFLp juntamente com o *solver* IBM Ilog CPLEX na resolução de problemas de programação inteira. Como apoio aos códigos construídos foi utilizada a ferramenta de análise de erros PVS-Stúdio. Na seleção de parâmetros foi utilizada a ferramenta *iRace*, que possuía como dependência a linguagem R. Para geração dos gráficos foi utilizada a ferramenta GNUPlot¹.

5.2 Instâncias de teste

As instâncias utilizadas nos experimentos computacionais realizados foram construídas a partir daquelas disponibilizadas pela *OR-Library* (Congram and Potts, 1998). Dentre as instâncias utilizadas, as de 40, 50 e 100 tarefas, foram obtidas diretamente do site da *OR-Library*. As demais instâncias, de 150, 200, 300 e 500 tarefas, foram construídas seguindo a mesma metodologia empregada pela *OR-Library*. Nessa metodologia, as instâncias foram criadas aleatoriamente, com tarefas disponibilizadas em sequência $(1, 2, \dots, n)$, sendo n a quantidade de tarefas). Cada uma das tarefas possuindo um tempo de processamento (p_j) , gerado a partir de uma distribuição uniforme entre 1 e 100; uma penalidade (w_j) , gerada a partir de uma distribuição uniforme entre 1 e 10; e uma data estimada de término (d_j) . Buscando obter instâncias com diferen-

¹<http://www.gnuplot.info/>

tes níveis de complexidade, foram utilizadas diferentes regras de distribuição uniforme para a geração da data estimada de término das instâncias. Para um dado intervalo de datas de término sugeridas, $RDD(RDD = 0.2, 0.4, 0.6, 0.8, 1.0)$ e uma determinada média correspondente ao fator de atraso $TF(TF = 0.2, 0.4, 0.6, 0.8, 1.0)$, uma data de término sugerida é gerada aleatoriamente através da distribuição uniforme no intervalo $[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$, onde $P = \sum_{i=1}^n p_j$.

Buscando oferecer alternativas relacionadas a cada nível de dificuldade, foram geradas 5 instâncias para cada 25 pares de valores de RDD e TF , dessa forma, gerando 125 instâncias para cada valor de n . Um exemplo que pode ser visto na Figura 5.1, onde são apresentadas as primeiras 11 linhas do arquivo, sendo as linhas de 2 a 11 as características das tarefas 1, 2, ..., 10, respectivamente.

1	250	← n (número de tarefas)		
2	77	5334	2] dados das tarefas
3	5	4789	4	
4	92	4512	1	
5	81	5153	7	
6	12	4813	6	
7	74	4756	3	
8	89	4414	6	
9	3	5349	2	
10	6	5278	3	
11	79	4948	4	

\uparrow p_j \uparrow d_j \uparrow w_j

Figura 5.1: Primeiras 11 linhas de uma instância com 250 tarefas

Cada instância tem seus dados em arquivos separados. Como foram inicialmente criadas para ambientes monoprocessados, as instâncias não possuem dentro de seus arquivos uma referência a quantidade de máquinas. A indicação da identificação da instância, juntamente com a quantidade de máquinas é feita no nome do arquivo, como pode ser visto na Figura 5.2.

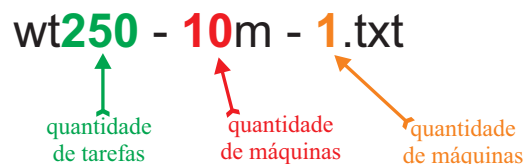


Figura 5.2: Composição dos nomes de arquivos de instâncias.

Seguindo a metodologia, as instâncias possuem diferentes níveis de complexidade, dados principalmente devido a data de término estimado de cada uma das tarefas. Na

Figura 5.3 é mostrada a dispersão das tarefas para cada uma das 25 instâncias do problema com 300 tarefas e 20 máquinas.

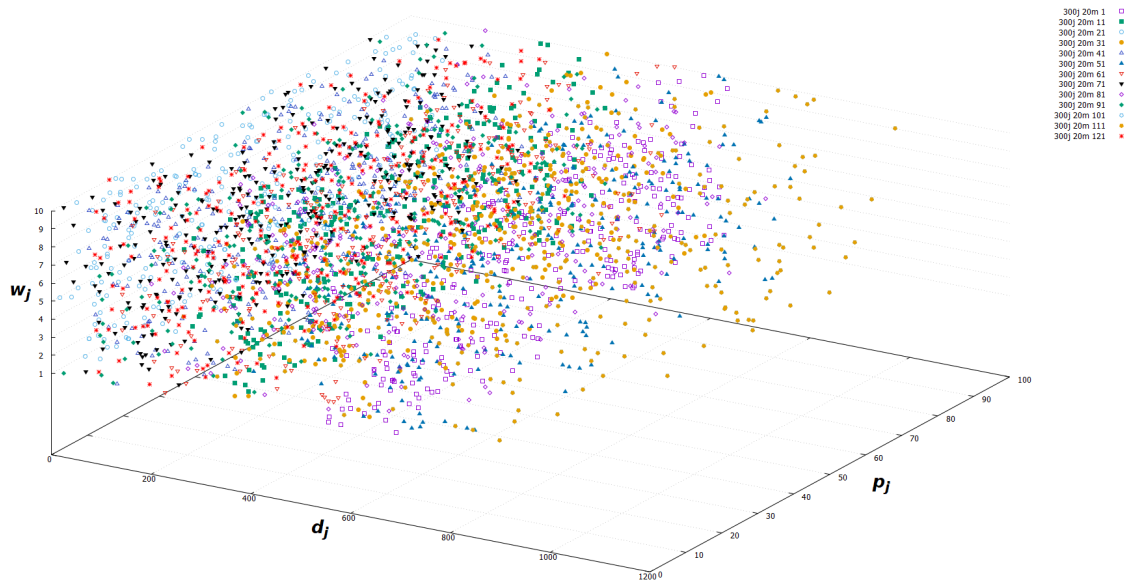


Figura 5.3: Dispersão de tarefas de cada uma das instâncias analisadas para o problema $P||\sum w_j T_j$ para 300 tarefas e 20 máquinas

Através da Figura 5.3 é possível verificar o aglomeração das tarefas de cada uma das instâncias, em diferentes pontos. Isso porque cada uma das instâncias está em uma classe diferenciada de complexidade. De forma a contribuir com pesquisas e trabalhos futuros, bem como permitir a replicação dos resultados, todas as instâncias envolvidas estão disponíveis no endereço <http://algox.icomp.ufam.edu.br/index.php/benchmark-instances/weighted-tardiness-scheduling>.

5.3 Observações da execução

Durante a execução do método foi possível comprovar um comportamento da abordagem heurística, que seria a convergência da solução enquanto da execução do algoritmo genético. Essa convergência pode ser notada pelo gráfico apresentado pela Figura 5.4, que apresenta o rastreamento das soluções, isto é, a descoberta das soluções (valores de *fitness*) dentro do tempo. No gráfico em questão o rastreamento executado leva em consideração as 10 repetições realizadas.

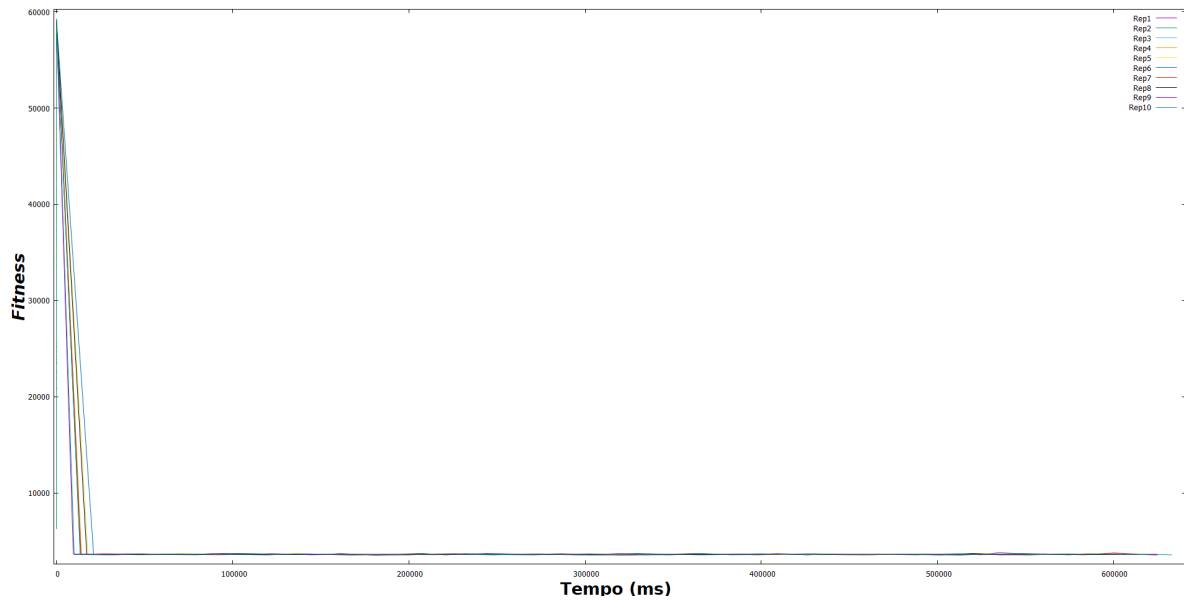


Figura 5.4: Rastreo das soluções encontradas para o problema $P || \sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 1

Este comportamento acaba por ser comum as demais instâncias, mesmo variando a quantidade de máquinas, quantidade de tarefas ou complexidade da instância. Tal observação foi percebida ao efetuar o rastreo de todas as instâncias. Nas Figuras 5.5 e 5.6, mesmo apresentando uma subida no gráfico que em determinados pontos, existe uma queda muito rápida no início do gráfico, que trata do ponto onde as soluções obtidas através dos operadores genéticos passam a tomar o lugar das soluções randômicas.

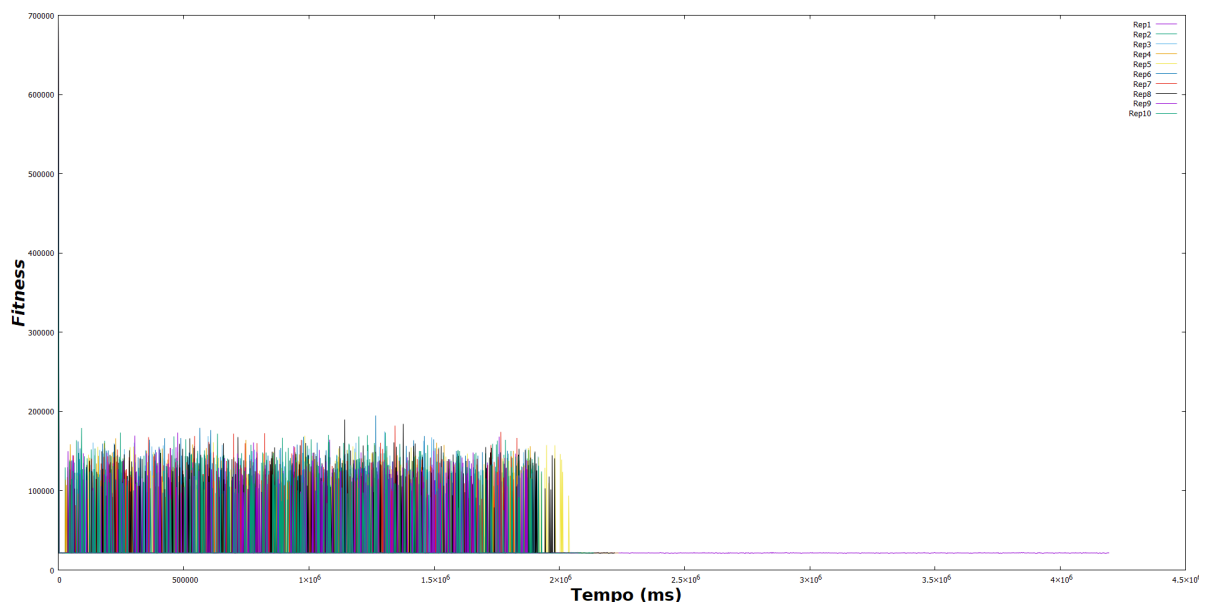


Figura 5.5: Rastreo das soluções encontradas para o problema $P || \sum w_j T_j$ para 300 tarefas e 2 máquinas - instância 1

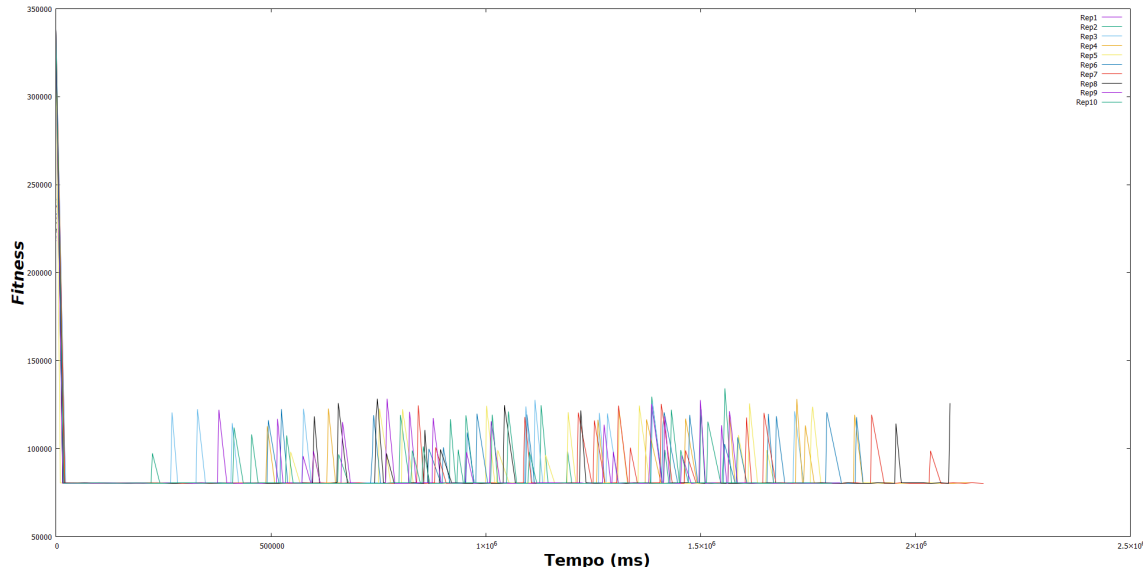


Figura 5.6: Rastreamento das soluções encontradas para o problema $P||\sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 116

Ao remover do gráfico gerado a área responsável pela geração da população inicial, é possível avaliar que o comportamento do algoritmo acaba por produzir soluções muito próximas, como pode ser avaliado pela Figura 5.7.

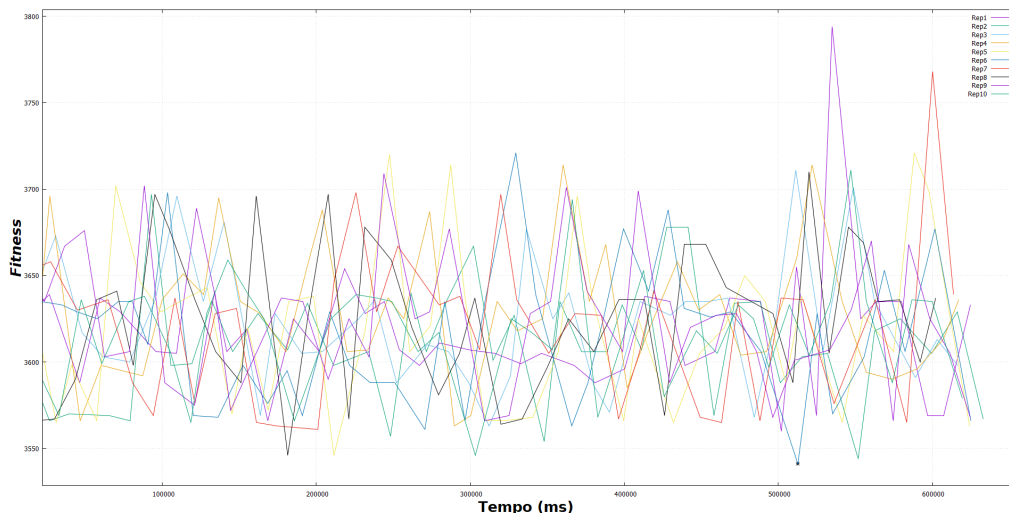


Figura 5.7: Rastreamento das soluções para o problema $P||\sum w_j T_j$ para 300 tarefas e 20 máquinas - instância 1 - sem população inicial

Como diferencial de modelos já apresentados, foram realizadas otimizações e ajustes no código fonte implementado que permitiram explorar soluções maiores e obter resultados em menor tempo. Citando as modificações mais relevantes, é possível citar o ajuste na alocação de memória, realizado nas operações de cruzamento e mutação onde, para cada tarefa processada a memória era alocada dinamicamente. Como implementação de

ajustes, a memória necessária para o processamento das tarefas de mutação e cruzamento é alocada uma única vez no início da execução e liberada ao final da execução. Isso evita a alocação custos de tempo com alocação dinâmica, além de evitar que ocorram problemas por não haver memória disponível durante a execução. Com o apoio da ferramenta PVS-Studio, alguns problemas com alocação de memória também foram corrigidos.

Foram adicionados novos parâmetros configuráveis, permitindo maior flexibilidade e refinamento do método através da parametrização, além de ajustes no código de forma a permitir a utilização de ferramentas de otimização de parâmetros. Como ajuste inicial, o código foi preparado para atuar e responder a análise da ferramenta *iRace* (apresentada no Capítulo 2.4). A utilização da ferramenta bem como a flexibilização dos parâmetros permitiu que o método pudesse se adaptar melhor as instâncias, além de obter melhores resultados em um menor espaço de tempo.

Outra opção adicionada a implementação atual foi o recurso de gerar gráficos de rastreamento das soluções encontradas. Isso permitiu avaliar a qualidade das soluções geradas ao longo da execução do algoritmo. A forma de alocação dos arcos foi levemente ajustado, possibilitando o armazenamento de um número maior de arcos. Como as modificações possibilitaram o consumo menor de recursos, ajustes na função de cruzamento do AG e no armazenamento de arcos foram realizados. Na função de cruzamento, em sua implementação inicial, era gerada apenas uma nova solução e, após a modificação, são geradas duas novas soluções, ambas passam pelo processo de busca local e, caso uma delas tenha um valor melhor da FO, este se torna o limite superior da execução. Como resultado, foi possível avaliar e registrar diversas boas soluções, antes eram ignoradas. Com relação aos ajustes no registro dos arcos, este ocorreu devido ao baixo número de arcos armazenados, o que não possibilitava a exploração melhor do método. Na implementação original os arcos eram coletados apenas ao final de cada geração (apenas a melhor solução) ou ao término da execução do AG (toda a população era registrada). Nos ajustes realizados foi codificado para que, cada nova solução encontrada no decorrer da execução da heurística, bem como toda a população de cada geração seria armazenada. Isso permitiu que a formulação pudesse explorar um espaço maior de soluções, retornando bons resultados, mesmo para grandes instâncias.

Durante a fase de experimentação foi substituído o operador de mutação e de cruzamento (este pelos métodos de cruzamento em um ponto, e o método de cruzamento em

dois pontos), como também a aplicação de mutação não apenas quanto a diversidade genética está baixa, mas também, em situações onde praticamente não se consegue explorar novos espaços de soluções (execução inicia quando as soluções não sofrem modificação em um dado número de execuções). Dessa forma, também buscando atingir uma certa diversidade, fugindo de ótimos locais, o operador de inversão também foi implementado, porém, mesmo quando sua execução ocorria, não houve mudança representativa na solução, devido ao fato da busca local ser aplicada após cada execução de um dos operadores genéticos.

Também foi implementada uma versão de *branch-and-bound* para teste e comparação com a formulação, porém, este demonstrou uma queda de performance para grandes instâncias (acima de 150 tarefas), o que acarretou em sua remoção do código principal, mantendo apenas a formulação.

5.4 Análise dos resultados

Foram analisados um total de 575 instâncias. Todas as instâncias com menos de 200 tarefas foram parametrizadas para executar o método por pelo menos 2 horas (1 hora para a heurística e 1 hora para o método exato), sendo o procedimento repetido 10 vezes. De forma semelhante, as instâncias com 200 tarefas ou mais, foram parametrizadas para executarem por 4 horas (2 horas para o método aproximado e 2 horas para o método exato). Se considerar cada repetição uma nova instância, foi realizada a execução do método sobre 5750 instâncias (575 instâncias únicas, cada uma passando por 10 repetições), sendo um montante estimado de 11500 horas (considerando uma média de 2 horas por instância).

Todo o processo de experimentação foi realizado levando em conta os parâmetros obtidos através do uso da ferramenta *iRace*, sendo possível avaliar a qualidade do método gerado ao compará-lo com outras instâncias existentes na literatura. A execução do método, tanto em sua parte heurística, quanto em sua parte exata passaram por ajustes e otimização de código, o que permitiu alcançar bons resultados em períodos de tempo menores, além de se mostrar competitivo frente a outros métodos existentes na literatura.

Um resumo dos resultados obtidos pode ser visualizado pela Tabela 5.1.

Tabela 5.1: Resumo de Resultados Obtidos.

n	m	Quantidade de Instâncias				Perc.Ganhos / Empates	Teve Comparações	
		Total	sm	si	sp			
40	2	25	0	25	0	100.00	Sim	
	4	25	0	25	0	100.00		
	10	25	0	25	0	100.00		
50	2	25	0	25	0	100.00		
	4	25	0	25	0	100.00		
	10	25	0	25	0	100.00		
100	2	25	0	24	1	96.00		
	4	25	0	25	0	100.00		
	10	25	8	16	1	96.00		
150	2	25	25	0	0	100.00		Não
	4	25	25	0	0	100.00		
	10	25	25	0	0	100.00		
200	2	25	25	0	0	100.00		
	4	25	25	0	0	100.00		
	10	25	25	0	0	100.00		
300	2	25	10	4	11	56.00	Sim	
	4	25	10	4	11	56.00		
	10	25	11	4	10	60.00		
	20	25	11	4	10	60.00		
500	2	25	25	0	0	100.00	Não	
	4	25	25	0	0	100.00		
	10	25	25	0	0	100.00		
	20	25	25	0	0	100.00		

Na Tabela 5.1 é possível avaliar a quantidade total de instâncias avaliadas, a quantidade de instâncias com resultado melhor que o existente na literatura (sm), a quantidade de soluções idênticas (si), a quantidade de soluções com resultado inferior ao existente na literatura (sp), o percentual de soluções competitivas (que tiveram solução superior ou similar a existente) bem como, o indicador se houve ou não a comparação com outros resultados. Acerca dos resultados obtidos, cabe destaque as seguintes informações, dentre as quais não estão sendo consideradas as repetições:

- Total de instâncias avaliadas: **575**

- Total de instâncias onde foi atingido o ótimo ou teve resultado equivalente ao melhor resultado disponível (considerando empates, ganhos e novos resultados): **531** (92%)
- Total de instâncias onde foi obtido resultado melhor que o disponível na literatura (apenas os que melhoraram sobre todos os resultados disponíveis): **52** (9%)
- Total de instâncias com dados para comparação: **325**
- Total de instâncias com dados para comparação onde o resultado obtido foi melhor ou equivalente ao melhor da literatura: **260** (86%)

Dentre os diversos resultados obtidos, cabe destacar a execução para ambientes com 100 tarefas e 4 máquinas (apresentado na Tabela A.8), onde foi possível comparar o híbrido a outros três métodos disponíveis na literatura. Os resultados obtidos pelo híbrido foram iguais ou superiores a todas as instâncias comparadas. Já para o ambiente de 100 tarefas e 10 máquinas (Tabela A.9), ficou abaixo em apenas 1 caso, sendo que nessas instâncias, a formulação já começa a apresentar resultados melhorando o valor obtido com o AG do método híbrido, ocorrendo a melhora em ambientes com 150 tarefas e 2 (Tabela A.10), 4 (Tabela A.11) e 10 máquinas (Tabela A.12); com 200 tarefas e 2 (Tabela A.13), 4 (Tabela A.14) e 10 máquinas (Tabela A.15); com 300 tarefas e 2 (Tabela A.16) e 4 (Tabela A.17); e ambientes com 500 tarefas e 2 (Tabela A.20) e 4 (Tabela A.21). Cabe destaque para os ambientes de 300 tarefas com 2 (Tabela A.16) e 4 máquinas (Tabela A.17), onde a formulação do método híbrido foi capaz de melhorar o resultado obtido pelo AG em 12 (48%) e 13 (52%) instâncias respectivamente. É relevante o fato também que o método foi capaz de oferecer melhora também em instâncias maiores, com 500 tarefas, o que corrobora com a eficácia e eficiência do híbrido.

Como análise, foi executada fora do período de experimentos instâncias em ambientes com 500 tarefas e 30 máquinas. Nesse modelo, mesmo obtendo resultados, a formulação não foi capaz de oferecer resultados melhores. Dessa forma, e conforme observado durante a fase de experimentação, o método é eficiente, porém, cabem melhoras em cenários onde existe o aumento do número de máquinas.

5.4.1 Comparação dos resultados com métodos da literatura

As instâncias com 40, 50 e 100 tarefas, em ambientes com 2, 4 e 10 máquinas puderam ser comparadas com os resultados obtidos por Della Croce et al. (2012) (Dellacroce), Pessoa et al. (2010) (BCP) e Kramer and Subramanian (2015) (UILS), um total de 225 instâncias com estas características. Destas, em todas as instâncias avaliadas com 40 e 50 tarefas, o método proposto obteve o resultado ótimo, também obtido pelos demais métodos.

Para as 25 instâncias com 100 tarefas e 2 máquinas, o resultado ótimo não foi alcançado em apenas uma instância, cujo melhor resultado foi obtido pelo método BCP, apresentado na Tabela 5.2 (resumida), onde é destacado em vermelho os métodos que obtiveram os melhores resultados, e destacado em amarelo quando a formulação do método exato - MIP - consegue melhorar o resultado obtido pela heurística GLS.

Tabela 5.2: Comparativo entre métodos para instâncias com 100 tarefas e 2 máquinas.

Instância	Híbrido		Dellacroce	BCP	UILS
	GLS	MIP			
6	30665	30665	30665	30665	30665
16	209100	209100	209100	209100	209100
26	92	92	92	92	92
36	56671	56671	56671	56671	56671
46	422831	422831	422831	422831	422831
56	5047	5047	5047	5047	5047
66	126513	126513	126513	126512	126513
76	0	0	0	0	0
86	36581	36581	36581	36581	36581
96	254194	254194	254194	254194	254194
106	0	0	0	0	0
111	84228	84220	84220	84250	84220
116	191191	191186	191186	191186	191186

Para as instâncias de 100 tarefas e 4 máquinas apresentou um excelente resultado, sendo o único dentre os quatro métodos comparados a apresentar os melhores resultados para todas as 25 instâncias avaliadas, conforme demonstrado no Gráfico 5.8.

As análises que originaram os dados do gráfico podem ser vistos na Tabela 5.3. Nesta tabela são apresentados os resultados obtidos por cada uma das instâncias/métodos.

Tabela 5.3: Comparativo entre métodos para instâncias com 100 tarefas e 4 máquinas.

Instância	Híbrido		Dellacroce	BCP	UILS
	GLS	MIP			
1	2001	2001	2001	2001	2001
6	16893	16893	16893	16893	16893
11	50232	50232	50232	50236	50232
16	110219	110219	110221	110222	110219
21	237392	237392	237392	237392	237392
26	141	141	141	141	141
31	7130	7130	7130	7130	7130
36	30791	30791	30791	30791	30791
41	126185	126185	126185	126185	126185
46	219536	219536	219536	219536	219536
51	0	0	0	0	0
56	3076	3076	3076	3076	3076
61	24856	24856	24856	24868	24856
66	67967	67967	67967	67967	67970
71	170689	170689	170689	170689	170691
76	0	0	0	0	0
81	819	819	819	819	819
86	21282	21282	21290	21299	21286
91	70606	70606	70609	70606	70608
96	133587	133587	133588	133587	133587
101	0	0	0	0	0
106	0	0	0	0	0
111	46709	46709	46711	46747	46719
116	101546	101546	101561	101546	101551
121	127618	127618	127619	127618	127619

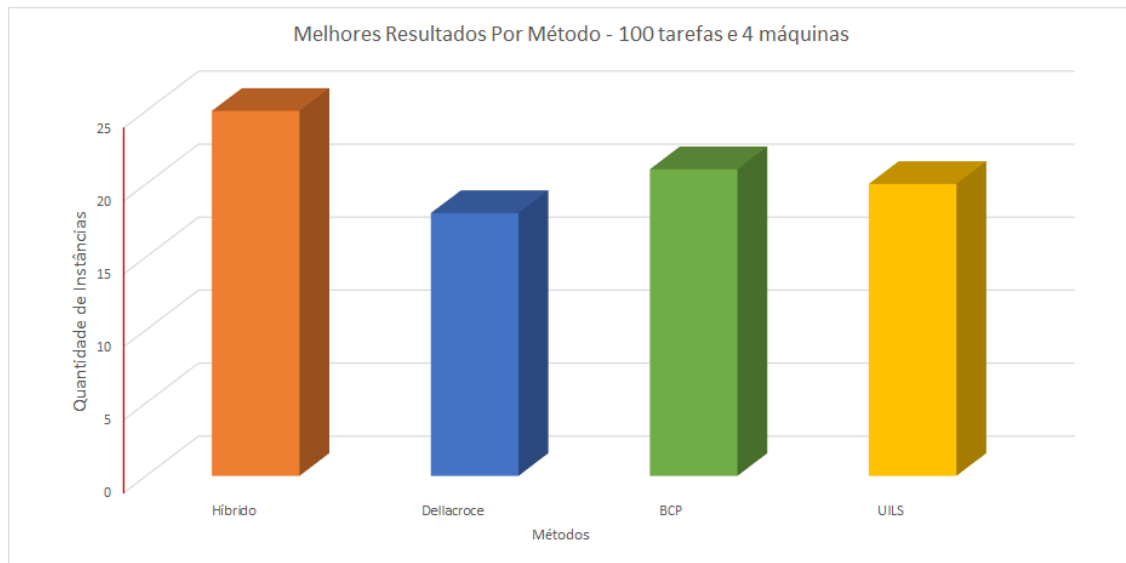


Figura 5.8: Melhores Resultados para Instância de 100 tarefas e 4 máquinas.

Para as instâncias de 100 tarefas e 10 máquinas, a comparação foi realizada apenas com o método de [Della Croce et al. \(2012\)](#), por ser o único que apresentou resultados para este conjunto de instâncias. Dentre as 25 instâncias analisadas, o método proposto foi capaz de obter os melhores resultados em 24 instâncias, conforme pode ser avaliado na Tabela 5.4.

Analisando os resultados obtidos para grandes instâncias e comparando com resultados existentes na literatura, o método proposto se mantém competitivo, além de haver uma maior proporção de soluções onde a formulação exata melhora os resultados da heurística do método. Para instâncias com 300 tarefas, em ambientes com 2, 4, 10 e 20 máquinas, as soluções obtidas pelo método híbrido puderam ser comparadas com as soluções obtidas por [Della Croce et al. \(2012\)](#) conforme pode ser avaliado na Tabela 5.5. Neste cenário complexo, o híbrido não apenas se manteve competitivo, como também, passou a apresentar um maior número de soluções onde a formulação matemática melhora o resultado obtivo pelo GLS.

5.4.2 Resumo de resultados obtidos

Como forma de representar a efetividade do método proposto, foi calculado um indicador capaz de avaliar o método. Tal indicador é dado pelo número de instâncias onde o método obteve o melhor resultado conhecido, dividido pela quantidade de instâncias avaliadas. Esse cálculo se torna necessário uma vez que nem todos os métodos avaliaram as mesmas

Tabela 5.4: Comparativo entre métodos para instâncias com 100 tarefas e 10 máquinas.

Instância	Híbrido		Dellacroce
	GLS	MIP	
1	1193	1193	1193
6	8641	8641	8641
11	24026	24026	24026
16	51627	51627	51627
21	105422	105422	105422
26	248	248	248
31	3789	3789	3789
36	15610	15610	15614
41	59474	59474	59474
46	97789	97789	97789
51	0	0	0
56	1992	1992	1992
61	13131	13129	13089
66	33210	33210	33210
71	76960	76960	76962
76	0	0	0
81	1308	1308	1316
86	12832	12832	12835
91	35714	35714	35716
96	61476	61476	61476
101	0	0	0
106	32	32	32
111	25167	25167	25193
116	48126	48126	48137
121	59310	59306	59315

Tabela 5.5: Comparativo entre métodos para instâncias com 300 tarefas e 2, 4, 10 e 20 máquinas

Instância	2 máquinas			4 máquinas			10 máquinas			20 máquinas		
	Híbrido		Dellacroce	Híbrido		Dellacroce	Híbrido		Dellacroce	Híbrido		Dellacroce
	GLS	MIP		GLS	MIP		GLS	MIP		GLS	MIP	
1	21429	21429	18576	11500	11500	10130	5479	5479	5069	3531	3531	3458
6	214326	214326	244811	110837	110837	126481	48854	48854	55588	28261	28260	32109
11	662310	662310	721631	339065	339065	369359	145441	145441	158089	81208	81207	88237
16	1664089	1664088	1861524	846710	846710	946632	356564	356564	398013	194312	194312	216081
21	2891957	2891956	2879379	1466355	1466354	1459646	611434	611434	608205	327093	327093	325044
26	176	176	22	189	189	36	167	167	110	335	335	258
31	110836	110836	116331	58101	58101	60656	26553	26553	27496	16068	16068	16623
36	649749	649749	556315	332686	332681	285151	142622	142622	122928	79723	79723	69194
41	1544545	1544545	1404153	788303	788303	718399	335403	335403	307843	186047	186047	171724
46	2505832	2505831	2663370	1273097	1273097	1352353	533645	533645	566299	287857	287857	305007
51	0	0	0	0	0	0	0	0	0	0	0	0
56	41066	41066	43074	22155	22155	23068	10857	10857	11489	7581	7581	7889
61	385883	385879	366023	198922	198922	190325	87387	87387	85132	50816	50816	50837
66	1221091	1221071	1039242	626714	626714	534484	271519	271519	232778	154103	154103	133639
71	1897516	1897515	2118511	966168	966168	1078051	408165	408165	454322	222846	222846	247195
76	0	0	0	0	0	0	0	0	0	0	0	0
81	2623	2623	2591	2189	2189	2182	2175	2175	2318	2611	2611	2806
86	378765	378728	278559	200106	200106	146941	94774	94774	69849	61324	61324	46944
91	920641	920543	938877	473659	473657	481976	206724	206724	208656	118889	118889	119631
96	1520269	1520230	1433978	776493	776470	733190	331363	331363	313375	183553	183553	174566
101	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0
111	128069	128011	146147	70214	70214	79011	37756	37756	39755	28821	28821	28446
116	569252	569252	468343	296329	296329	244917	133297	133297	112286	80154	80154	69999
121	1035960	1035937	1039943	532180	532180	535470	230852	230852	233269	131656	131656	133432
Ganhos	14		15	14	14	15	14	14	15	14	15	14

instâncias. Com base nesse indicador, e levando em consideração os quatro métodos analisados foi construído o Gráfico 5.9.

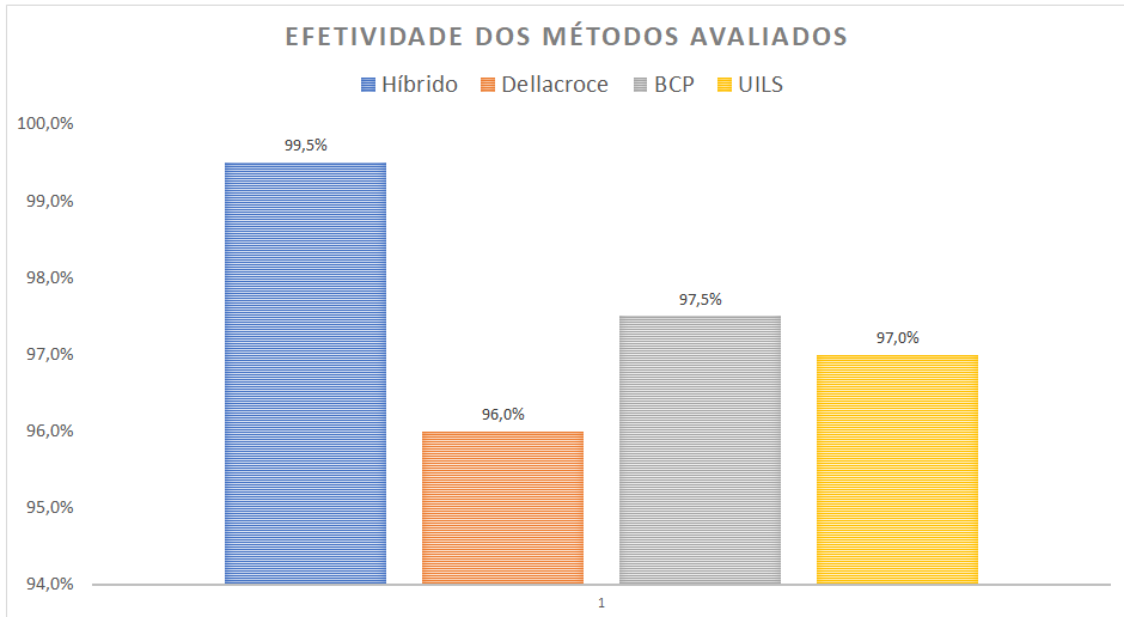


Figura 5.9: Efetividade dos 4 métodos para 200 instâncias.

Neste gráfico foi levado em consideração o indicador para a mesma quantidade de instâncias executadas (200 instâncias). De forma semelhante o índice foi construído, porém levando em consideração apenas um dos métodos, cujas instâncias puderam ser comparadas, sendo o resultado desta comparação, apresentado no Gráfico 5.9.

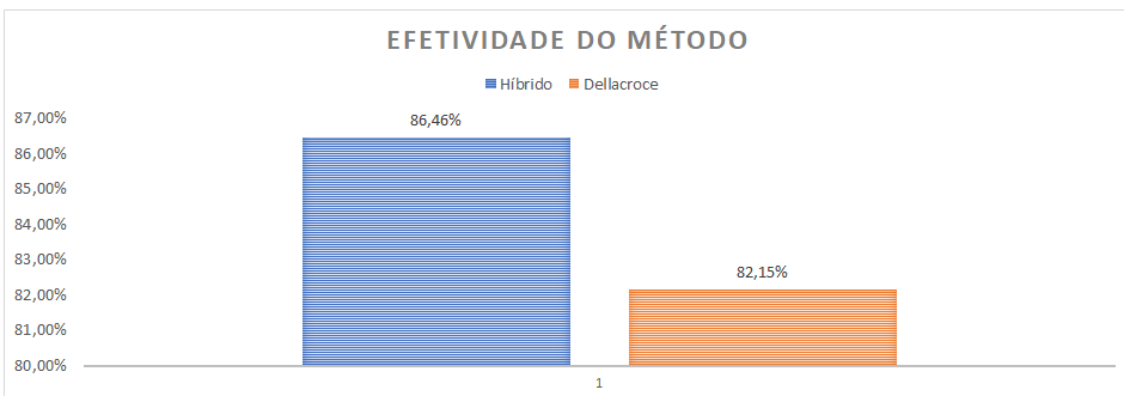


Figura 5.10: Efetividade do método para 325 instâncias.

Capítulo 6

Conclusões

Nesta dissertação foi abordado o problema de escalonamento relacionado a minimização das penalidades sofridas na ocorrência de atraso em um ambiente de máquinas paralelas. Fazendo parte de um conjunto especial de problemas (NP-Difícil) a abordagem foi construída buscando obter melhores resultados, principalmente quando aplicado a grandes instâncias.

Por se tratar de uma situação amplamente aplicada (ambientes de produção dentre outros), diversas abordagens tratando deste cenário. Para instâncias menores, onde existe uma quantidade razoável de tarefas e máquinas, abordagens exatas puderam ser aplicadas, enquanto que em instâncias maiores (maior quantidade de tarefas e máquinas), devido a natureza do problema, foram aplicados métodos aproximados (heurísticas e meta-heurísticas). O trabalho realizado buscou fazer uso de ambas as abordagens, elegendo um conjunto de soluções através dos métodos aproximados (utilizando algoritmos genéticos com busca local), reduzindo assim o conjunto de soluções a ser avaliado utilizando um método exato (utilizando uma formulação baseada na Arc-Time Indexed (Pessoa et al., 2010)).

A abordagem aplicada, além de oferecer uma alternativa viável para o que foi proposto, ainda oferece uma alternativa de implementação, possibilitando outras combinações de heurísticas, meta-heurísticas e formulações, permitindo avanços ainda maiores nesta área.

6.1 Considerações finais

A presente dissertação apresenta conceitos e trabalhos relacionados à problemas de escalonamento, em especial, na minimização de atraso ponderado em máquinas paralelas. Com base na pesquisa, foi possível perceber que existe uma grande quantidade de métodos, abordagens e resultados nessa área, porém, devido a sua natureza e importância em ambientes industriais e ambientes reais de produção, ainda existe a possibilidade de melhorar nos métodos já existentes, principalmente, quanto à utilização de modelos híbridos. Outros modelos puderam ser avaliados, como o trabalho de [Setio \(2014\)](#), ao utilizar a heurística conhecida por *Ant Colony Optimization* (Otimização de Colônia de Formigas).

A aplicação de novos métodos bem como a combinação de métodos existentes (heurísticas, meta-heurísticas e métodos exatos) permite que seja realizada uma comparação com soluções ótimas já encontradas e conceituadas, dando espaço a novas implementações, eficientes e mais viáveis computacionalmente, embora não sendo uma implementação recente ([Srinivasan, 1971](#)) ([Quian, 2006](#)) ([Mensendiek et al., 2015](#)), foi possível constatar como soluções geradas respostas próximas do ótimo (em casos com muitas tarefas e máquinas, conseguindo um resultado ótimo em casos menores). Contudo cabe destacar a complexidade do problema proposto, que propõe realizar a minimização do atraso, mas também tratar restrições complexas como o *release date* e o tempo de *setup*.

Durante elaboração desta dissertação foram avaliados outros trabalhos relacionados, alguns aplicando heurísticas como a *Tabu Search* de [Lee et al. \(2013\)](#) na busca da minimização do atraso ponderado, como também o tratamento de heurísticas de escalonamento preemptivo (em máquinas paralelas uniformes) ([Lushchakova, 2012](#)).

Outras propostas puderam ser avaliadas, como a aplicação de modelos híbridos (heurísticas GRASP, VNS¹-VND² e GRASP-VND) ([João et al., 2014](#)) ([Borges De Freitas et al., 2008](#)), que muito enriqueceram a dissertação, não apenas por tratar de um trabalho relacionado, mas por permitir avaliar a aplicação da heurística em um modelo que trata também de antecipação, e tempo ocioso.

Também foram realizados experimentos utilizando a heurística ILS (*Iterated Local Search*). Após a obtenção de resultados, os mesmos não se mostraram plenamente satisfatórios. Dessa forma, outra abordagem foi aplicada, quando foi aplicado o Algoritmo

¹Variable Neighborhood Search ou Busca em vizinhança variável

²Variable Neighborhood Descent ou Vizinhança Variável Descendente

Genético com Busca Local como heurística, alimentando um método exato. Para resolução do método exato foi utilizada uma ferramenta de propósito geral(*solver*), chamado IBM Ilog Cplex [IBM \(2016\)](#), que utiliza como resolução o método *Branch-and-Cut*. Em determinados casos, o modelo matemático aplicado junto ao *solver* CPLEX, utilizando os arcos gerados a partir do algoritmo genético, consegue melhorar o resultado obtido pela heurística. Em outros casos, o *solver* é utilizado como forma de provar a otimalidade da solução.

6.2 Trabalhos futuros

Um estudo interessante poderia utilizar a mesma metodologia empregada nesta dissertação, abordando problemas com outras restrições tais como preempção (*pmtn*), tempos de preparação (*setup*), datas de liberação (*release date*) e datas máximas de conclusão (*deadline*), combinando ainda a outros ambientes de processamento como máquinas de propósito geral (*mpm*) e ambientes *flow shop*, *open shop* e *job shop*. Outra aplicação interessante seria a adaptação da metodologia híbrida aplicada na resolução de problemas multiobjetivo. Uma implementação promissora a ser avaliada seria a construção de um método híbrido, combinando a heurística Evolução Diferencial em substituição ao AG utilizado neste trabalho. Esta sugestão se mostra interessante pois faz com que as substituições sejam realizadas mediante uma ponderação de valores posicionais que talvez permitam explorar melhor o ambiente de soluções em um mesmo tempo.

Referências

- Amorim, R. X. D. (2013). Um estudo sobre formulações matemáticas e estratégias algorítmicas para problemas de escalonamento em máquinas paralelas com penalidades de antecipação e atraso. Master's thesis, Universidade Federal do Amazonas-UFAM, Manaus.
- Arenales, M., Morabito, R., Armentano, V. A., and Yanasse, H. (2007). *Pesquisa Operacional: As Disciplinas da Execução da Estratégia*. Elsevier.
- Balaprakash, P., Birattari, M., and Stutzle, T. (2007). “Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement. *Hybrid Metaheuristics*.
- Beale, E. and Small, R. (1965). Mixed integer programming by a branch and bound technique. *Proceedings of 3th IFIP Congress*, pages 450–451.
- Bilyk, A. and Mönch, L. (2010). A variable neighborhood search approach for planning and scheduling of jobs on unrelated parallel machines. *Journal of Intelligent Manufacturing*, pages 1–15.
- Blum, C. (2012). Hybrid metaheuristics in combinatorial optimization: A tutorial. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7505 LNCS(6):1–10.
- Borges De Freitas, L. M., Alfredo, F., and Montané, T. (2008). Metaheurísticas VNS-VND e GRASP-VND Para Problemas de Roteamento de Veículos com Coleta e Entrega Simultânea. *SPOLM 2*.
- Brucker, P. (2007). *Scheduling Algorithms*, volume 93. , 5a. edition.

- Congram, R. K. and Potts, C. N. (1998). Or-library - benchmark instances the single-machine total weighted tardiness problem. Disponível em: <http://people.brunel.ac.uk/~mastjbj/jeb/info.html>.
- Cormen, T. (2017). *Desmistificando algoritmos*. Elsevier Editora Ltda.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). *Algoritmos - Teoria e Prática*. Campus, New York, second ed edition.
- Dakin, R. J. (1965). A tree search algorithm for mixed integer programming problems. *Computer Journal*, pages 250–255.
- de Freitas, R., Pessoa, A., Uchoa, E., and Poggi, M. (2008). Heuristic algorithm for the parallel machine total weighted tardiness scheduling problem. *Relatórios de pesquisa em engenharia de produção*, 8:1–12.
- de Freitas Rodrigues, R. (2009). *Caracterizações e algoritmos para problemas clássicos de escalonamento*. PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro.
- Della Croce, F., Garaix, T., and Grosso, A. (2012). Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. *Computers and Operations Research*, 39(6):1213–1217.
- Demirel, T., Özkir, V., Çetin Demirel, N., and Taşdelen, B. (2011). A genetic algorithm approach for minimizing total tardiness in parallel machine scheduling problems. *Proceedings of the World Congress on Engineering*, 2.
- Dourado, M. C., de Freitas, R., and Szwarcfiter, J. L. (2010). Escalonamento em máquinas paralelas para minimizar o atraso ponderado de tarefas em tempos iguais. *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*, 1:1–10.
- Du, J. and Leung, J. Y. (1990). Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research*, 15:483–495.
- Emmons, H. (1969). One-Machine Sequencing to Minimize Certain Functions of Job Tardiness. *Operations Research*, 17(4):701–715.
- Fang, K. T. and Lin, B. M. T. (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers and Industrial Engineering*, 64(1):224–234.

- Feili, H., Haddad, H., and Ghanbari, P. (2012). Two hybrid algorithms for single-machine total weighted tardiness scheduling problem with sequence-dependent setup. *American Journal of Scientific Research*, pages 22–29.
- Fisher, Ronald A.; Yates, F. (1963). Statistical tables for biological, agricultural and medical research. .
- Goldbarg, M. and Luna, H. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. CAMPUS - RJ.
- Goldbarg, M. C., Goldbarg, E. G., and Loureiro, H. P. (2016). *Otimização Combinatória e Meta-Heurísticas: Algoritmos e Aplicações*. Elsevier, Rio de Janeiro, first ed edition.
- Gomory, R. E. (1960). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, pages 275–278.
- Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey.
- Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA.
- IBM, C. (2016). Ibm cplex optimizer - united states. <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html>.
- João, J. P., Arroyo, J. E. C., Villadiego, H. M. M., and Gonçalves, L. B. (2014). Hybrid GRASP heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. *Electronic Notes in Theoretical Computer Science*, 302:53–72.
- Kramer, A. and Subramanian, A. (2015). A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems. *CoRR*, abs/1509.02384.
- Kravchenko, S. and Werner, F. (2009). *Minimizing a separable convex function on parallel machines with preemptions*. Univ., Fak. für Mathematik.

- Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica* 28, pages 497–520.
- Lawler, E. L. (1977). A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Annals of Discrete Mathematics*, 1(C):331–342.
- Lee, I. S. (2013). Minimizing total tardiness for the order scheduling problem. *International Journal of Production Economics*, 144(1):128–134.
- Lee, J. H., Yu, J. M., and Lee, D. H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: Minimizing total tardiness. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2081–2089.
- Lee, Y. H., Bhaskaran, K., and Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, 29(1):45–52.
- Lenstra, J., Rinnooy Kan, A. H. G., and Brucker, P. (1977). Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1(8):343–362.
- Liaw, C.-f. (2016). A branch-and-bound algorithm for identical parallel machine total tardiness scheduling problem with preemption. , 1015(August).
- Liu, Y., Dong, H., Lohse, N., and Petrovic, S. (2014). An Investigation into Minimising Total Electricity Consumption , Total Electricity Cost and Total Weighted Tardiness in Job Shops When the Rolling Blackout Policy is Applied. , 65.
- López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package: Iterated racing for automatic algorithm configuration. , 3.
- Lushchakova, I. N. (2012). Preemptive scheduling of two uniform parallel machines to minimize total tardiness. *European Journal of Operational Research*, 219(1):27–33.
- Martí, R. and Reinelt, G. (2011). *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Applied Mathematical Sciences. Springer Berlin Heidelberg.
- Melián, B., Moreno Perez, J., and Marcos Moreno-Vega, J. (2003). Metaheuristics: A global view. *Inteligencia Artificial*, 7(19):7–28.

- Mensendiek, A., Gupta, J. N. D., and Herrmann, J. (2015). Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. *European Journal of Operational Research*, 243(2):514–522.
- Montalvão, E. H. R. and Pessoa, A. A. (2016). Single machine total weighted tardiness problem with budgeted uncertainty set on the processing times. *SBPO*, pages 2313–2323.
- Müller-Merbach, H. (1981). Heuristics and their design: a survey. *European Journal of Operational Research*, 8(1):1–23.
- Nannen, V. and Eiben, A. E. (2006). A Method for Parameter Calibration and Relevance. *Genetic and Evolutionary Computation, GECCO'06*, pages 183–190.
- Nannen, V. and Eiben, A. E. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. *IJCAI International Joint Conference on Artificial Intelligence*, pages 975–980.
- Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-city symmetric single machine scheduling problems with asymmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.*, 6(1):1–7.
- Pessoa, A. and Uchoa, E. (2011). Integrando programação inteira e mista e planilhas de cálculo.
- Pessoa, A., Uchoa, E., de Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3-4):259–290.
- Pinedo, M. L. (2010). *Scheduling - Theory, Algorithms and Systems*. Springer, New York, fourth edition.
- Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems: Fifth edition*. Springer, New York, 5th edition.
- Prado, D. (2010). *Programação Linear*. Indg Tecnologia, Rio de Janeiro, sixth edition.

- Quian, B. e. a. (2006). Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. *International Journal of Advanced Manufacturing Technology*, 35:1014–1027.
- Rodrigues, R. D. F. (1996). Times Assíncronos para a Resolução de Problemas de Otimização Combinatória com Múltiplas Funções Objetivo. Master's thesis, Universidade Estadual de Campinas - UNICAMP.
- Rudolf, M. (2016). *Parameter tuning for numerical optimization algorithms*. PhD thesis, Czech Technical University in Prague.
- Sen, T., Sulek, J. M., and Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey.
- Setio, K. I. (2014). Metaheurística para minimização do atraso total ponderado em problemas de sequenciamento de tarefas. *Guaratinguetá*.
- Shafiei Nikabadi, M. and Naderi, R. (2016). A hybrid algorithm for unrelated parallel machines scheduling. *International Journal of Industrial Engineering Computations*, 7(4).
- Sivapragasam, S. and Suppiah, Y. (2017). Minimizing Total Weighted Tardiness in Identical Parallel Machine with Sequence Dependent Setup Time Using Genetic Algorithm. *Journal of Telecommunication, Electronic and Computer Engineering good*, 9(1):89–93.
- Srinivasan, V. (1971). A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly*, 18(3):317–327.
- Tanaka, S. and Fujikuma, S. (2011). A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *Journal of Scheduling*, pages 1–15.
- Xiao, W.-Q. and Li, C.-L. (2002). Approximation algorithms for common due date assignment and job scheduling on parallel machines. *IIE Transactions*, 34:467–477.
- Yang, X. (2008). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Apêndice A

Tabela Completa de Resultados

Aqui estão dispostas as tabelas contendo os resultados completos da execução do método para $P||\sum w_j T_j$. Nessas tabelas é possível visualizar os resultados de execução para todas as instâncias (25 instâncias diferentes em cada tabela). Estão apresentados os seguintes resultados

1. Instâncias com 40 tarefas - resultados para 2, 4 e 10 máquinas.
2. Instâncias com 50 tarefas - resultados para 2, 4 e 10 máquinas.
3. Instâncias com 100 tarefas - resultados para 2, 4 e 10 máquinas.
4. Instâncias com 150 tarefas - resultados para 2, 4 e 10 máquinas.
5. Instâncias com 200 tarefas - resultados para 2, 4 e 10 máquinas.
6. Instâncias com 300 tarefas - resultados para 2, 4, 10 e 20 máquinas.
7. Instâncias com 500 tarefas - resultados para 2, 4, 10 e 20 máquinas.

Tabela A.1: Resultados para $P||\sum w_j T_j$ com 40 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	606	606	6,429	0,046	6,375	6,474	6,62	0	0,012	0	40	0,011	606	606	606	606	606
6	3886	3886	8,423	0,035	8,01	8,458	8,756	0	0,013	0	40	0,008	3886	3886	3886	3886	3886
11	9617	9617	8,554	0,043	7,971	8,597	9,117	0	0,009	0	40	0,009	9617	9617	9617	9617	9617
16	38356	38356	12,322	0,042	10,416	12,365	13,879	0	0,01	0	40	0,012	38356	38356	38356	38356	38356
21	41048	41048	12,084	0,039	10,451	12,123	12,945	0	0,013	0	40	0,005	41048	41048	41048	41048	41048
26	87	87	8,771	0,041	8,54	8,812	9,298	0	0,018	0	40	0,007	87	87	87	87	87
31	3812	3812	8,555	0,046	8,146	8,601	9,06	0	0,087	0	40	0,007	3812	3812	3812	3812	3812
36	10713	10713	10,209	0,041	8,356	10,25	11,435	0	0,1	0	40	0,007	10713	10713	10713	10713	10713
41	30802	30802	27,111	0,149	18,758	27,26	34,772	0	0,148	0	40	0,007	30802	30802	30802	30802	30802
46	34146	34146	14,094	0,479	10,332	14,573	26,948	0	0,026	0	40	0,052	34146	34146	34146	34146	34146
51	0	0	0,387	0,02	0,283	0,407	0,512	0	0	0	40	0,006	0	0	0	0	0
56	1279	1279	7,728	0,044	7,232	7,771	8,89	0	0,017	0	40	0,006	1279	1279	1279	1279	1279
61	11488	11488	24,122	0,687	17,066	24,809	32,382	0	0,057	0	40	0,011	11488	11488	11488	11488	11488
66	35279	35279	23,148	0,091	15,748	23,239	26,733	0	0,015	0	40	0,007	35279	35279	35279	35279	35279
71	47952	47952	25,545	0,1	17,972	25,645	32,383	0	0,029	0	40	0,009	47952	47952	47952	47952	47952
76	0	0	0,307	0,023	0,273	0,33	0,484	0	0	0	40	0,005	0	0	0	0	0
81	574,8	574,8	19,987	0,386	14,689	20,373	25,567	9	5,247	0	40	0,018	571	571	571	571	571
86	6048	6048	22,646	0,263	18,062	22,908	26,367	0	0,202	0	40	0,005	6048	6048	6048	6048	6048
91	26075	26075	19,605	0,17	16,242	19,775	21,979	0	0,026	0	40	0,008	26075	26075	26075	26075	26075
96	66116	66116	23,325	0,282	15,651	23,607	29,873	0	0,017	0	40	0,011	66116	66116	66116	66116	66116
101	0	0	0,33	0,025	0,3	0,355	0,566	0	0	0	40	0,009	0	0	0	0	0
106	0	0	0,425	0,024	0,328	0,448	0,586	0	0	0	40	0,005	0	0	0	0	0
111	17936	17936	16,175	0,045	13,893	16,22	21,625	0	0,015	0	40	0,008	17936	17936	17936	17936	17936
116	25870	25870	13,322	0,05	12,476	13,372	15,46	1	0,392	0	40	0,01	25870	25870	25870	25870	25870
121	64516	64516	15,993	0,071	14,713	16,064	18,423	0	0,05	0	40	0,025	64516	64516	64516	64516	64516

Tabela A.2: Resultados para $P||\sum w_j T_j$ com 40 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	439	439	16,03	0,054	14,546	16,085	18,409	0	0,039	0	40	0,006	439	439	439	439	439
6	2374	2374	15,736	0,041	15,075	15,776	17,511	0	0,061	0	40	0,006	2374	2374	2374	2374	2374
11	5737	5737	15,217	0,043	14,583	15,26	16,647	0	0,054	0	40	0,011	5737	5737	5737	5737	5737
16	21493	21493	16,173	0,044	14,808	16,216	18,473	0	0,036	0	40	0,008	21493	21493	21493	21493	21493
21	22793	22793	18,006	0,044	17,314	18,05	19,279	0	0,015	0	40	0,006	22793	22793	22793	22793	22793
26	88	88	16,275	0,049	15,336	16,325	17,116	0	0,016	0	40	0,012	88	88	88	88	88
31	2525	2525	16,288	0,045	15,937	16,333	16,675	0	0,191	0	40	0,01	2525	2525	2525	2525	2525
36	6420	6420	16,265	0,079	15,428	16,344	17,665	13	5,214	0	40	0,008	6420	6420	6420	6420	6420
41	17685	17685	18,342	0,048	16,297	18,39	28,488	0	0,472	0	40	0,039	17685	17685	17685	17685	17685
46	19124	19124	13,974	0,043	12,613	14,017	16,82	0	0,088	0	40	0,008	19124	19124	19124	19124	19124
51	0	0	0,703	0,02	0,614	0,723	0,777	0	0,024	0	40	0,008	0	0	0	0	0
56	826	826	15,067	0,044	14,522	15,111	17,117	0	0,095	0	40	0,005	826	826	826	826	826
61	7357	7357	23,606	0,093	21,521	23,699	26,64	0	0,115	0	40	0,007	7357	7357	7357	7357	7357
66	20251	20251	29,02	0,046	20,553	29,066	38,592	0	0,22	0	40	0,008	20251	20251	20251	20251	20251
71	26740	26740	26,58	0,068	23,718	26,648	28,681	0	0,026	0	40	0,006	26740	26740	26740	26740	26740
76	0	0	0,416	0,023	0,377	0,44	0,576	0	0	0	40	0,007	0	0	0	0	0
81	564	564	22,345	0,156	20,023	22,501	26,321	6	4,532	0	40	0,064	564	564	564	564	564
86	4725	4725	27,864	0,056	22,783	27,92	36,055	1	1,311	0	40	0,005	4725	4725	4725	4725	4725
91	15569	15569	25,882	0,083	22,731	25,965	28,839	8	4,991	0	40	0,011	15569	15569	15569	15569	15569
96	36266	36266	19,997	0,069	19,254	20,066	22,508	2	1,472	0	40	0,061	36266	36266	36266	36266	36266
101	0	0	1,483	1,262	0,515	2,746	7,304	0	0	0	40	0,009	0	0	0	0	0
106	0	0	1,752	1,056	0,588	2,808	9,641	0	0,019	0	40	0,006	0	0	0	0	0
111	11263,1	11263,1	19,468	0,057	16,212	19,525	29,324	0	0,091	0	40	0,01	11263	11263	11263	11263	11263
116	15566	15566	20,789	0,051	17,642	20,84	32,978	2	0,824	0	40	0,008	15566	15566	15566	15566	15566
121	35751	35751	19,657	0,048	17,832	19,704	24,753	0	0,047	0	40	0,01	35751	35751	35751	35751	35751

Tabela A.3: Resultados para $P||\sum w_j T_j$ com 40 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	501,1	501,1	13,385	0,135	12,724	13,52	16,131	4	4,215	0	40	0,021	501	501	501
6	1584,1	1584,1	15,648	0,098	14,536	15,746	18,752	3	3,686	0	40	0,007	1584	1584	1584
11	3929	3929	15,233	0,074	13,301	15,307	18,585	1	1,669	0	40	0,006	3929	3929	3929
16	11987	11987	10,551	0,043	9,397	10,595	11,245	0	0,118	0	40	0,006	11987	11987	11987
21	12060	12060	11,209	0,05	9,844	11,259	15,23	0	0,105	0	40	0,007	12060	12060	12060
26	208,5	208,5	13,794	0,098	11,197	13,892	17,648	18	10,52	0	40	0,011	208	208	208
31	2136	2136	13,348	0,047	11,386	13,396	15,909	0	0,172	0	40	0,01	2136	2136	2136
36	4940	4940	17,03	0,049	12,302	17,079	24,838	0	0,142	0	40	0,008	4940	4940	4940
41	10304	10304	9,53	0,044	8,7	9,574	10,17	0	0,192	0	40	0,006	10304	10304	10304
46	10348	10348	8,894	0,042	8,224	8,936	9,34	0	0,165	0	40	0,009	10348	10348	10348
51	251,7	251,7	11,276	0,098	10,321	11,373	12,273	1	0,943	0	40	0,008	242	242	242
56	722,7	722,7	11,907	0,062	10,415	11,969	12,879	11	5,842	0	40	0,011	722	722	722
61	5769	5769	16,239	0,062	14,764	16,301	18,498	0	0,914	0	40	0,008	5769	5769	5769
66	11643	11643	20,786	0,055	19,215	20,841	23,23	0	0,111	0	40	0,005	11643	11643	11643
71	14290	14290	14,695	0,055	13,449	14,75	16,971	0	0,118	0	40	0,02	14290	14290	14290
76	0	0	0,616	0,021	0,489	0,637	0,911	0	0,035	0	40	0,005	0	0	0
81	1415,4	1415,4	14,369	0,139	13,626	14,508	15,986	10	7,741	0	40	0,011	1415	1415	1415
86	4919	4919	19,96	0,06	17,987	20,019	22,382	0	0,631	0	40	0,01	4919	4919	4919
91	9661	9661	16,593	0,05	14,835	16,644	17,798	0	0,076	0	40	0,008	9661	9661	9661
96	18580	18580	14,82	0,057	13,87	14,878	16,346	0	0,081	0	40	0,008	18580	18580	18580
101	120	120	11,029	0,051	9,39	11,08	13,895	0	0	0	40	0,011	120	120	120
106	808	808	10,536	0,045	8,559	10,582	13,718	0	0,434	0	40	0,011	808	808	808
111	7766	7766	12,593	0,051	10,17	12,644	16,136	4	3,169	0	40	0,007	7766	7766	7766
116	9854	9854	14,977	0,045	12,085	15,022	19,41	0	0,2	0	40	0,009	9854	9854	9854
121	18814	18814	9,901	0,049	8,929	9,95	11,302	0	0,029	0	40	0,012	18814	18814	18814

Tabela A.4: Resultados para $P||\sum w_j T_j$ com 50 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	1268	1268	20,736	0,049	17,691	20,785	23,963	0	0,023	0	50	0,007	1268	1268	1268	1268	1268
6	14272	14272	15,791	0,048	13,868	15,839	17,845	0	0,069	0	50	0,009	14272	14272	14272	14272	14272
11	23028	23028	17,935	0,049	15,621	17,984	20,706	0	0,06	0	50	0,01	23028	23028	23028	23028	23028
16	46072	46072	21,758	0,05	18,801	21,808	25,36	0	0,043	0	50	0,008	46072	46072	46072	46072	46072
21	111069	111069	26,94	0,047	26,166	26,988	29,477	0	0,104	0	50	0,01	111069	111069	111069	111069	111069
26	26	26	14,582	0,049	14,415	14,63	15,148	0	0,157	0	50	0,01	26	26	26	26	26
31	5378	5378	18,872	0,052	17,809	18,924	21,896	0	0,026	0	50	0,012	5378	5378	5378	5378	5378
36	18956	18956	17,416	0,049	16,463	17,465	18,275	0	0,296	0	50	0,009	18956	18956	18956	18956	18956
41	38058	38058	32,59	1,223	23,175	33,814	45,293	0	0,138	0	50	0,144	38058	38058	38058	38058	38058
46	82105	82105	43,099	0,798	28,708	43,897	61,54	0	0,042	0	50	0,008	82105	82105	82105	82105	82105
51	0	0	0,573	0,025	0,56	0,598	0,662	0	0	0	50	0,01	0	0	0	0	0
56	761,4	761,4	13,837	0,048	12,876	13,885	16,694	0	0,024	0	50	0,009	761	761	761	761	761
61	13682	13682	35,43	0,431	28,128	35,861	47,867	0	0,078	0	50	0,008	13682	13682	13682	13682	13682
66	40907	40907	47,903	0,914	34,602	48,817	64,698	0	0,034	0	50	0,007	40907	40907	40907	40907	40907
71	78532	78532	46,95	0,14	35,918	47,09	54,838	0	0,016	0	50	0,011	78532	78532	78532	78532	78532
76	0	0	0,615	0,026	0,558	0,641	0,691	0	0	0	50	0,007	0	0	0	0	0
81	542	542	25,561	2,687	23,839	28,248	36,44	0	0,241	0	50	0,071	542	542	542	542	542
86	12557	12557	49,7	0,329	37,221	50,029	62,656	0	0,178	0	50	0,008	12557	12557	12557	12557	12557
91	47349	47349	43,346	0,408	34,142	43,754	56,525	0	0,026	0	50	0,007	47349	47349	47349	47349	47349
96	92822	92822	48,388	0,479	34,961	48,867	62,141	0	0,569	0	50	0,007	92822	92822	92822	92822	92822
101	0	0	0,546	0,026	0,44	0,572	0,709	0	0	0	50	0,008	0	0	0	0	0
106	0	0	0,527	0,025	0,524	0,552	0,588	0	0	0	50	0,01	0	0	0	0	0
111	15564	15564	38,712	0,062	32,555	38,774	50,054	0	0,778	0	50	0,016	15564	15564	15564	15564	15564
116	19608	19608	21,843	0,105	20,315	21,949	25,068	1	0,873	0	50	0,009	19608	19608	19608	19608	19608
121	41696	41696	36,964	0,051	27,828	37,015	41,224	0	0,051	0	50	0,01	41696	41696	41696	41696	41696

Tabela A.5: Resultados para $P||\sum w_j T_j$ com 50 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	785	785	33,019	0,065	27,218	33,084	47,828	0	0,648	0	50	0,014	785	785	785	785	785
6	8317	8317	34,491	0,052	26,021	34,544	49,411	0	0,291	0	50	0,009	8317	8317	8317	8317	8317
11	12882	12882	30,742	0,065	27,802	30,807	34,047	33	25,151	0	50	0,01	12879	12879	12879	12879	12879
16	25376	25376	36,776	0,058	24,891	36,835	58,18	0	0,033	0	50	0,011	25376	25376	25376	25376	25376
21	59440	59440	33,533	0,056	26,303	33,589	50,49	0	0,312	0	50	0,007	59440	59440	59440	59440	59440
26	54	54	28,962	0,055	26,772	29,017	31,093	0	0,099	0	50	0,009	54	54	54	54	54
31	3061	3061	27,142	0,049	25,608	27,191	28,278	0	0,101	0	50	0,008	3061	3061	3061	3061	3061
36	10796	10796	33,51	0,073	29,12	33,583	35,628	1	1,226	0	50	0,009	10796	10796	10796	10796	10796
41	21806	21806	26,391	0,078	23,786	26,469	31,041	0	0,546	0	50	0,01	21806	21806	21806	21806	21806
46	44455	44455	31,96	0,074	27,025	32,034	47,187	0	0,249	0	50	0,224	44455	44455	44455	44455	44455
51	0	0	0,634	0,026	0,55	0,66	0,822	0	0	0	50	0,005	0	0	0	0	0
56	570	570	28,189	0,056	26,715	28,245	31,554	1	1,144	0	50	0,008	570	570	570	570	570
61	7898	7898	55,147	0,492	47,59	55,639	60,47	8	14,397	0	50	0,031	7898	7898	7898	7898	7898
66	23138	23138	76,234	0,194	54,301	76,427	110,885	0	0,436	0	50	0,007	23138	23138	23138	23138	23138
71	42645	42645	62,464	0,221	54,206	62,686	70,362	0	0,351	0	50	0,016	42645	42645	42645	42645	42645
76	0	0	0,749	0,024	0,683	0,774	0,904	0	0	0	50	0,009	0	0	0	0	0
81	495	495	53,835	0,149	49,568	53,985	58,358	0	0,179	0	50	0,037	495	495	495	495	495
86	8369	8369	78,645	0,11	59,389	78,756	91,457	0	0,077	0	50	0,009	8369	8369	8369	8369	8369
91	26551	26551	63,038	0,149	56,837	63,187	68,818	2	2,997	0	50	0,01	26551	26551	26551	26551	26551
96	50326	50326	55,087	0,15	52,933	55,236	60,946	0	0,914	0	50	0,012	50326	50326	50326	50326	50326
101	0	0	1,381	0,151	1,201	1,532	1,889	0	0	0	50	0,018	0	0	0	0	0
106	0	0	1,677	0,312	1,157	1,989	2,98	0	0,099	0	50	0,014	0	0	0	0	0
111	10069,5	10069,5	40,097	0,097	33,581	40,195	55,638	24	21,614	0	50	0,01	10069	10069	10069	10069	10069
116	11552	11552	42,88	0,112	33,698	42,992	68,436	3	3	0	50	0,01	11552	11552	11552	11552	11552
121	23792	23792	50,18	0,096	43,389	50,276	59,938	2	3,256	0	50	0,008	23792	23792	23792	23792	23792

Tabela A.6: Resultados para $P||\sum w_j T_j$ com 50 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	559	559	42,499	0,153	39,528	42,652	48,707	2	6,141	0	50	0,053	559	559	559
6	4783,7	4783,7	42,523	0,294	38,162	42,818	51,204	18	44,838	0	50	0,014	4782	4782	4782
11	7082	7082	31,012	0,432	28,143	31,444	40,905	2	4,29	0	50	0,174	7082	7082	7082
16	13761	13761	29,635	0,119	25,829	29,753	32,191	0	0,701	0	50	0,039	13761	13761	13761
21	28666	28666	27,019	0,092	21,458	27,112	37,088	0	1,448	0	50	0,012	28666	28666	28666
26	200,1	200,1	38,866	0,179	31,021	39,045	52,839	7	13,206	0	50	0,006	200	200	200
31	1909,9	1909,8	39,781	0,311	32,916	40,092	48,382	12	21,413	0	50	0,017	1905	1905	1905
36	7152	7152	31,432	0,091	26,034	31,523	40,064	0	0,309	0	50	0,011	7152	7152	7152
41	12421	12421	30,117	0,065	22,015	30,181	36,994	0	1,219	0	50	0,012	12421	12421	12421
46	22154	22154	23,027	0,063	16,691	23,089	29,113	0	0,24	0	50	0,011	22154	22154	22154
51	52	52	31,003	0,09	24,479	31,092	40,146	4	7,467	0	50	0,01	52	52	52
56	521,1	521,1	33,216	0,135	26,915	33,351	44,152	9	12,841	0	50	0,009	521	521	521
61	5319	5319	38,761	0,196	32,006	38,956	47,208	0	0,375	0	50	0,054	5319	5319	5319
66	12895	12895	30,027	0,137	26,837	30,165	36,196	0	0,516	0	50	0,063	12895	12895	12895
71	21298	21298	31,861	0,283	28,271	32,144	36,628	0	0,215	0	50	0,201	21298	21298	21298
76	0	0	1,625	0,034	1,417	1,659	1,863	0	0,078	0	50	0,008	0	0	0
81	932,6	932,6	39,596	0,219	35,362	39,815	48,918	17	32,491	0	50	0,015	930	930	930
86	6690	6690	39,853	0,169	33,965	40,022	45,499	16	33,011	0	50	0,01	6690	6690	6690
91	14483	14483	34,652	0,265	30,5	34,917	40,309	3	6,724	0	50	0,053	14483	14483	14483
96	25100	25100	29,887	0,176	26,777	30,063	39,863	6	11,232	0	50	0,011	25100	25100	25100
101	70	70	25,289	0,109	18,952	25,399	31,258	0	0,045	0	50	0,006	70	70	70
106	840	840	31,119	0,248	22,956	31,367	39,015	5	6,504	0	50	0,035	840	840	840
111	7531	7531	37,145	0,113	26,019	37,258	50,236	1	1,811	0	50	0,012	7531	7531	7531
116	7015	7015	37,545	0,104	25,134	37,649	48,271	4	5,817	0	50	0,006	7015	7015	7015
121	13341	13341	18,789	0,112	17,242	18,902	20,866	4	3,769	0	50	0,008	13341	13341	13341

Tabela A.7: Resultados para $P||\sum w_j T_j$ com 100 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	3339	3339	89,236	0,111	86,252	89,348	92,753	0	1,15	0	256	0,008	3339	3339	3339	3339	3339
6	30665	30665	94,997	0,116	90,006	95,113	102,636	0	0,19	0	249	0,016	30665	30665	30665	30665	30665
11	93894	93894	96,048	0,102	87,591	96,149	104,958	0	0,39	0	241	0,018	93894	93894	93894	93894	93894
16	209100	209100	210,278	0,106	196,111	210,384	240,942	0	3,821	0	254	0,009	209100	209100	209100	209100	209100
21	457836	457836	184,762	0,086	179,814	184,849	186,633	0	1,35	0	276	0,01	457836	457836	457836	457836	457836
26	92	92	91,67	0,113	86,134	91,783	96,231	0	0,161	0	273	0,01	92	92	92	92	92
31	12735,8	12735,8	86,104	0,113	81,434	86,216	96,387	0	0,533	0	256	0,008	12729	12729	12729	12729	12729
36	56671	56671	112,467	0,102	108,076	112,568	121,885	0	0,5	0	245	0,012	56671	56671	56671	56671	56671
41	237964	237964	221,68	0,115	212,204	221,796	227,734	3	19,086	0	285	0,017	237964	237964	237964	237964	237964
46	422831	422831	192,5	0,136	187,524	192,636	196,777	9	44,013	0	262	0,02	422831	422831	422831	422831	422831
51	0	0	2,731	0,048	2,589	2,779	3,039	0	0	0	100	0,008	0	0	0	0	0
56	5094,5	5094,5	86,324	0,11	84,122	86,434	89,663	0	0,549	0	247	0,025	5047	5047	5047	5047	5047
61	45573	45573	133,619	0,118	126,733	133,737	140,822	0	2,362	0	235	0,016	45573	45573	45573	45573	45573
66	126513,4	126513,2	169,55	0,133	159,548	169,683	180,447	0	3,347	0	222	0,013	126513	126513	126513	126512	126513
71	327305	327305	167,933	0,115	152,784	168,047	185,215	2	10,365	0	255	0,013	327305	327305	327305	327305	327305
76	0	0	2,622	0,052	2,467	2,674	3,083	0	0	0	100	0,009	0	0	0	0	0
81	908	908	123,074	0,109	114,587	123,183	129,368	0	2,641	0	232	0,012	908	908	908	908	908
86	36581	36581	171,131	0,122	160,072	171,253	180,924	2	11,304	0	252	0,016	36581	36581	36581	36581	36581
91	129929,6	129929,6	163,42	0,107	157,222	163,527	172,086	36	148,016	0	237	0,019	129929	129929	129929	129931	129929
96	254194	254194	206,721	0,098	202,6	206,819	210,047	4	24,39	0	239	0,017	254194	254194	254194	254194	254194
101	0	0	2,694	0,043	2,507	2,737	2,915	0	0	0	100	0,006	0	0	0	0	0
106	0	0	2,719	0,043	2,532	2,762	2,935	0	0	0	100	0,008	0	0	0	0	0
111	84234,3	84230	197,802	0,133	187,337	197,935	211,604	13	74,407	0	258	0,02	84228	84220	84220	84250	84220
116	191195,5	191193	219,8	0,12	211,314	219,92	228,45	26	148,308	0	242	0,016	191191	191186	191186	191186	191186
121	242019,3	242019,2	173,335	0,121	165,068	173,456	181,698	7	33,293	0	249	0,015	242018	242018	242018	242018	242018

Tabela A.8: Resultados para $P||\sum w_j T_j$ com 100 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce	BCP	UILS
1	2001	2001	138,596	0,101	133,296	138,697	141,085	0	2,074	0	128	0,022	2001	2001	2001	2001	2001
6	16893	16893	139,84	0,094	135,447	139,934	146,923	0	0,596	0	124	0,021	16893	16893	16893	16893	16893
11	50236,8	50236,8	163,712	0,194	151,42	163,907	208,967	1	7,145	0	120	0,014	50232	50232	50232	50236	50232
16	110221,7	110221,7	175,524	0,179	154,985	175,703	180,342	8	43,059	0	127	0,009	110219	110219	110221	110222	110219
21	237392,2	237392,2	190,977	0,127	177,346	191,104	243,218	30	146,283	0	137	0,012	237392	237392	237392	237392	237392
26	141	141	173,655	0,107	168,379	173,762	177,434	0	1,074	0	136	0,009	141	141	141	141	141
31	7130	7130	156,624	0,101	147,844	156,725	203,682	0	1,469	0	127	0,01	7130	7130	7130	7130	7130
36	30791	30791	175,125	0,147	172,864	175,272	180,645	1	12,769	0	122	0,016	30791	30791	30791	30791	30791
41	126185,1	126185,1	213,298	0,194	203,425	213,492	223,569	23	126,594	0	141	0,017	126185	126185	126185	126185	126185
46	219536	219536	179,275	0,093	173,823	179,368	184,862	0	4,424	0	131	0,014	219536	219536	219536	219536	219536
51	0	0	3,742	0,044	3,563	3,786	3,934	0	0	0	100	0,007	0	0	0	0	0
56	3076	3076	150,46	0,127	147,998	150,587	153,079	4	20,522	0	123	0,009	3076	3076	3076	3076	3076
61	24872,5	24871,8	222,185	0,258	199,809	222,443	245,712	9	52,115	0	118	0,014	24856	24856	24856	24868	24856
66	67973,3	67973,1	214,489	0,173	207,63	214,662	225,82	33	188,667	1	222	0,026	67967	67967	67967	67967	67970
71	170693,6	170693,2	190,245	0,177	185,177	190,422	195,575	5	26,825	0	127	0,015	170689	170689	170689	170689	170691
76	0	0	3,435	0,045	3,151	3,479	3,668	0	0	0	100	0,005	0	0	0	0	0
81	819	819	229,112	0,101	222,402	229,213	235,637	0	1,311	0	116	0,012	819	819	819	819	819
86	21283,7	21283,7	235,8	0,276	227,158	236,076	243,608	7	45,178	0	125	0,028	21282	21282	21290	21299	21286
91	70606	70606	222,835	0,237	216,144	223,072	230,905	32	187,014	0	118	0,016	70606	70606	70609	70606	70608
96	133587,1	133587,1	201,281	0,207	192,912	201,488	211,697	17	91,668	0	119	0,013	133587	133587	133588	133587	133587
101	0	0	3,736	0,045	3,355	3,781	4,406	0	0	0	100	0,005	0	0	0	0	0
106	0	0	3,475	0,044	3,286	3,52	3,893	0	0	0	100	0,008	0	0	0	0	0
111	46714,7	46714,7	294,866	0,331	286,932	295,196	302,672	16	126,496	0	129	0,014	46709	46709	46711	46747	46719
116	101557,7	101555,8	229,643	0,319	210,248	229,962	277,296	17	98,044	0	121	0,016	101546	101546	101561	101546	101551
121	127619,1	127619,1	223,351	0,22	210,979	223,572	232,268	36	203,715	0	124	0,015	127618	127618	127619	127618	127619

Tabela A.9: Resultados para $P||\sum w_j T_j$ com 100 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	1193	1193	184,345	0,135	179,048	184,481	187,403	5	53,092	0	100	0,011	1193	1193	1193
6	8641	8641	185,65	0,13	177,123	185,78	217,106	0	1,522	0	100	0,01	8641	8641	8641
11	24035,8	24035,8	194,781	0,334	191,228	195,115	202,076	6	62,1	0	100	0,007	24026	24026	24026
16	51627,6	51627,6	139,419	0,189	130,861	139,608	156,462	6	50,126	0	100	0,02	51627	51627	51627
21	105422,1	105422,1	108,513	0,153	103,557	108,666	112,903	7	42,86	0	100	0,009	105422	105422	105422
26	249,8	249,8	173,034	0,107	165,116	173,142	192,385	0	2,997	0	100	0,006	248	248	248
31	3789	3789	179,588	0,108	165,244	179,697	202,093	1	11,916	0	100	0,007	3789	3789	3789
36	15614,4	15614,4	182,261	0,299	176,427	182,559	190,838	6	63,967	0	100	0,012	15610	15610	15614
41	59483	59483	131,001	0,193	115,78	131,194	150,166	1	14,923	0	100	0,008	59474	59474	59474
46	97789	97789	115,595	0,173	106,166	115,768	130,317	5	37,901	0	100	0,006	97789	97789	97789
51	0	0	5,951	0,045	5,125	5,996	7,009	0	0,254	0	100	0,006	0	0	0
56	1999,5	1999,5	204,002	0,275	189,756	204,278	217,915	4	45,191	0	100	0,007	1992	1992	1992
61	13135,8	13135	205,727	0,268	200,023	205,996	212,519	17	182,847	2	300	0,044	13131	13129	13089
66	33214,6	33214,6	186,117	0,365	168,126	186,482	194,58	13	130,685	0	100	0,013	33210	33210	33210
71	76960,1	76960,1	137,382	0,287	125,304	137,669	150,582	18	132,237	0	100	0,008	76960	76960	76962
76	0	0	6,102	0,066	5,512	6,168	6,91	0	0	0	100	0,005	0	0	0
81	1318,1	1317,8	222,24	0,38	216,758	222,619	233,288	16	187,509	0	100	0,012	1308	1308	1316
86	12843	12843	204,931	0,471	197,882	205,403	215,518	9	104,931	0	100	0,021	12832	12832	12835
91	35717,6	35717,6	180,776	0,365	156,728	181,141	208,171	3	34,399	0	100	0,013	35714	35714	35716
96	61476,2	61476,2	132,918	0,248	119,588	133,167	158,404	3	26,06	0	100	0,012	61476	61476	61476
101	0	0	5,973	0,054	5,493	6,027	6,493	0	0	0	100	0,005	0	0	0
106	32	32	118,6	0,112	116,723	118,713	120,728	0	0,39	0	100	0,007	32	32	32
111	25173	25172,4	201,84	0,454	198,792	202,294	205,421	13	141,171	0	100	0,012	25167	25167	25193
116	48129,1	48129	175,852	0,367	167,425	176,22	186,674	18	154,572	0	100	0,009	48126	48126	48137
121	59316	59315,4	170,228	0,38	151,888	170,608	185,425	18	159,997	0	100	0,02	59310	59306	59315

Tabela A.10: Resultados para $P||\sum w_j T_j$ com 150 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	4242	4242	307,139	0,169	294,689	307,308	338,105	0	5,16	0	882	0,008	4242	4242
6	68489	68489	323,921	0,171	308,613	324,093	351,507	0	1,194	0	894	0,009	68489	68489
11	166737	166737	358,233	0,183	333,584	358,415	387,275	0	4,743	0	813	0,01	166737	166737
16	430423	430423	499,099	0,236	426,211	499,335	564,342	3	34,695	0	825	0,026	430423	430423
21	605867	605867	655,177	0,189	598,498	655,366	719,978	0	3,454	0	788	0,038	605867	605867
26	79	79	379,135	0,191	324,458	379,326	413,158	0	0,861	0	826	0,01	79	79
31	38516	38516	430,37	0,185	395,836	430,555	485,774	0	1,64	0	893	0,008	38516	38516
36	168345	168345	507,786	0,24	460,481	508,027	588,036	1	22,314	0	853	0,015	168345	168345
41	405611,2	405611,1	635,985	0,28	539,423	636,266	789,915	14	203,065	0	847	0,04	405608	405608
46	585673	585673	635,543	0,286	558,092	635,829	748,286	5	72,088	0	791	0,028	585673	585673
51	0	0	10,594	0,087	10,061	10,681	11,78	0	0	0	150	0,006	0	0
56	6983	6983	332,84	0,186	322,89	333,026	343,906	0	1,388	0	802	0,008	6983	6983
61	103094,1	103094,1	468,048	0,22	402,015	468,268	523,977	0	0,933	0	789	0,011	103090	103090
66	223516,2	223512,4	596,355	0,284	527,679	596,639	656,042	38	503,111	0	778	0,038	223507	223506
71	555470	555470	740,281	0,258	658,957	740,539	819,715	4	92,21	0	859	0,02	555470	555470
76	0	0	11,259	0,078	10,432	11,337	12,135	0	0	0	150	0,007	0	0
81	406	406	530,526	0,174	504,67	530,7	543,496	0	0,91	0	796	0,01	406	406
86	98504,5	98504,5	538,204	0,258	456,762	538,462	584,808	5	84,338	0	806	0,023	98486	98486
91	211272	211272	758,512	0,985	677,242	759,497	944,837	3	78,294	0	845	0,019	211272	211272
96	461960,6	461959,8	812,393	0,323	727,302	812,716	1034,498	11	266,369	0	808	0,031	461960	461959
101	0	0	10,922	0,081	10,246	11,004	11,855	0	0	0	150	0,005	0	0
106	0	0	12,329	0,083	11,658	12,412	13,437	0	0	0	150	0,006	0	0
111	76953	76952,5	682,698	0,297	620,88	682,995	839,112	10	222,656	0	888	0,028	76952	76952
116	94912,6	94912,2	505,104	0,273	480,296	505,377	524,643	33	419,466	0	775	0,062	94911	94911
121	319939	319939	764,895	0,222	607,197	765,118	942,179	0	4,614	0	888	0,007	319939	319939

Tabela A.11: Resultados para $P||\sum w_j T_j$ com 150 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	2518,9	2518,9	649,587	0,239	532,416	649,826	755,89	0	8,423	0	440	0,029	2516	2516
6	36532	36532	668,518	0,217	539,334	668,735	775,525	1	30,632	0	446	0,015	36532	36532
11	87303,6	87303,6	719,159	0,253	562,752	719,411	844,723	20	304,002	0	407	0,046	87303	87303
16	222612	222612	643,419	0,264	622,503	643,683	673,608	4	90,422	0	413	0,022	222612	222612
21	311874,6	311874,3	602,972	0,247	550,208	603,219	711,15	14	241,098	0	394	0,021	311873	311873
26	127	127	648,734	0,196	620,197	648,93	673,052	0	4,021	0	413	0,014	127	127
31	21226,3	21226,3	289,028	0,122	280,157	289,149	295,824	5	49,633	0	446	0,038	21210	21210
36	88441,7	88441,7	300,506	0,144	283,615	300,65	312,934	12	107,287	0	426	0,012	88434	88434
41	211522,2	211521,2	384,29	0,229	373,289	384,519	398,199	26	251,12	0	423	0,012	211455	211451
46	302425,8	302425,6	334,103	0,152	290,544	334,255	351,791	13	95,139	0	396	0,014	302422	302422
51	0	0	7,568	0,04	7,023	7,607	8,71	0	0	0	150	0,002	0	0
56	4165	4165	236,844	0,094	228,489	236,939	258,875	0	0,744	0	401	0,007	4165	4165
61	55040	55040	313,539	0,122	295,017	313,662	343,93	1	20,402	0	395	0,009	55040	55040
66	117696,6	117696	365,12	0,252	342,104	365,373	481,954	38	350,349	0	390	0,016	117689	117689
71	287307	287307	418,314	0,15	394,161	418,464	435,554	6	64,363	0	429	0,013	287307	287307
76	0	0	7,988	0,043	7,341	8,031	8,77	0	0	0	150	0,003	0	0
81	449	449	273,156	0,095	264,368	273,251	281,731	0	1,568	0	398	0,006	449	449
86	53829,7	53826,5	456,194	0,222	400,622	456,416	484,707	19	221,798	0	403	0,02	53717	53717
91	112611,5	112611,5	564,922	0,219	524,561	565,141	580,981	19	281,527	0	422	0,016	112607	112607
96	239911,4	239911,2	531,078	0,16	506,399	531,238	547,066	36	470,411	0	404	0,017	239907	239907
101	0	0	7,864	0,043	7,376	7,908	8,827	0	0	0	150	0,003	0	0
106	0	0	8,164	0,045	7,692	8,209	8,82	0	0	0	150	0,004	0	0
111	43634,2	43633,8	628,747	0,26	464,271	629,007	684,873	35	415,426	0	443	0,015	43629	43627
116	51496,3	51495,3	576,252	0,213	460,995	576,465	604,544	37	432,41	0	388	0,014	51457	51457
121	168168,2	168168,1	551,197	0,15	523,681	551,347	569,72	11	172,426	0	443	0,018	168167	168167

Tabela A.12: Resultados para $P||\sum w_j T_j$ com 150 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	1567	1567	356,414	0,084	331,185	356,498	388,838	0	1,421	0	175	0,004	1567	1567
6	17440,2	17440,2	373,731	0,176	350,186	373,907	414,764	7	146,838	0	177	0,008	17438	17438
11	39909,1	39909,1	363,79	0,166	338,814	363,957	398,161	0	3,446	0	163	0,009	39905	39905
16	98455,4	98454,2	385,325	0,444	311,367	385,769	420,776	8	141,432	0	165	0,009	98446	98438
21	135739,5	135739,5	285,012	0,315	272,734	285,327	303,402	11	171,505	0	158	0,006	135739	135739
26	187	187	443,865	0,088	391,795	443,953	471,873	0	13,016	0	165	0,006	187	187
31	10907	10907	321,59	0,178	289,805	321,768	333,364	6	96,935	0	177	0,004	10907	10907
36	40628,1	40628,1	371,312	0,289	319,075	371,601	389,993	13	219,048	0	170	0,008	40620	40620
41	95900,4	95900,1	367,485	0,479	320,747	367,964	559,802	13	368,686	0	169	0,011	95893	95893
46	132921,1	132921,1	261,181	0,282	231,493	261,463	311,277	9	140,657	0	159	0,01	132920	132920
51	0	0	13,349	0,04	11,349	13,389	14,675	0	0	0	150	0,003	0	0
56	2480,7	2480,7	298,821	0,118	270,557	298,939	329,732	2	33,153	0	161	0,008	2477	2477
61	26990,4	26989,8	597,5	1,02	544,848	598,52	648,374	13	414,81	0	159	0,006	26960	26960
66	55293,1	55293,1	540,711	1,184	504,776	541,894	580,379	19	502,893	0	157	0,003	55288	55288
71	127296,3	127296,3	454,868	2,68	410,688	457,547	495,093	9	248,919	0	171	0,01	127291	127291
76	0	0	14,71	0,038	12,258	14,748	15,997	0	0	0	150	0,004	0	0
81	665,2	665,2	612,831	0,448	476,691	613,279	700,771	0	15,143	0	159	0,01	576	576
86	28489,2	28489,2	613,138	1,3	498,066	614,438	680,916	16	434,135	0	161	0,301	28477	28477
91	54591,4	54591,4	564,744	0,603	536,008	565,347	606,503	15	456,862	0	168	0,007	54584	54584
96	107436,7	107436,5	527,459	1,337	486,918	528,796	589,788	15	410,158	0	162	0,01	107431	107431
101	0	0	21,116	3,487	19,436	24,604	38,801	0	0	0	150	0,394	0	0
106	0	0	22,508	2,108	18,114	24,616	40,852	0	0	0	150	0,014	0	0
116	26890	26890	424,602	0,473	393,399	425,075	606,288	13	276,89	0	156	0,005	26874	26874
111	24783,6	24783,6	457,758	0,393	421,808	458,151	647,029	12	274,767	0	176	0,008	24771	24771
121	78374,7	78374,7	335,423	0,508	321,671	335,931	361,684	14	241,244	0	176	0,007	78366	78366

Tabela A.13: Resultados para $P||\sum w_j T_j$ com 200 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	8936	8936	1478,828	26,463	905,341	1505,291	2292,8	0	12,602	0	1987	0,276	8936	8936
6	118059,5	118059,5	1551,313	9,668	961,688	1560,981	2713,529	0	17,828	0	2066	0,193	118058	118058
11	253173	253173	1379,678	26,722	959,028	1406,4	2035,311	0	5,992	0	1832	0,284	253173	253173
16	822852,3	822852,3	2205,56	21,646	1328,362	2227,206	3387,158	27	1395,643	0	2119	0,139	822852	822852
21	1202741	1202741	2605,769	15,345	1635,975	2621,114	3795,051	4	170,272	0	1997	0,056	1202741	1202741
26	94	94	1443,646	22,836	976,746	1466,482	2053,98	0	3,953	0	1924	0,272	94	94
31	65418	65418	1889,736	31,578	1292,151	1921,314	3079,003	2	81,662	0	2087	0,025	65418	65418
36	267878,1	267877,2	1994,385	22,169	1248,423	2016,554	3268,66	0	35,572	0	2044	0,01	267871	267871
41	697397,1	697397	1871,125	0,359	1743,853	1871,485	2032,826	2	106,281	0	1953	0,016	697397	697397
46	1118048	1118048	1495,025	0,354	1414,349	1495,379	1622,923	7	266,468	0	2010	0,073	1118048	1118048
51	0	0	106,449	45,927	114,736	152,375	260,23	0	0	0	200	1,337	0	0
56	18292	18292	1982,665	21,977	1315,445	2004,642	3097,942	0	10,123	0	2095	0,053	18292	18292
61	205966,1	205965,1	2059,591	20,626	1383,351	2080,217	3518,799	4	178,866	0	1978	0,026	205960	205960
66	544819,4	544818,5	2292,708	11,747	1393,297	2304,455	3674,23	34	1207,495	0	2060	0,094	544815	544815
71	866040,3	866040,3	2515,51	12,792	1361,176	2528,302	3775,531	9	312,237	0	2047	0,025	866039	866039
76	0	0	36,97	0,186	31,178	37,156	42,605	0	0	0	200	0,007	0	0
81	1375	1375	950,214	0,32	854,236	950,534	1102,057	1	36,652	0	2092	0,014	1375	1375
86	144824,6	144820,7	1440,283	0,4	1361,41	1440,683	1676,094	38	1327,819	0	1951	0,089	144813	144813
91	391326	391325,8	1693,636	0,425	1602,26	1694,062	1808,969	38	1539,765	0	1944	0,08	391324	391324
96	630537,3	630529,5	1267,04	0,478	1115,468	1267,517	1403,452	37	1231,176	0	1896	0,091	630514	630513
101	0	0	23,174	0,109	21,018	23,283	25,451	0	0	0	200	0,006	0	0
106	0	0	22,472	0,117	21,397	22,589	24,246	0	0	0	200	0,007	0	0
111	42409	42409	851,024	0,332	838,932	851,356	865,192	5	138,951	0	1996	0,033	42409	42409
116	309110,7	309110,3	1244,794	0,453	1145,535	1245,247	1429,318	34	1272,873	0	1945	0,033	309109	309109
121	493702,8	493697,9	1290,766	0,433	1185,722	1291,198	1462,586	26	826,647	0	1856	0,049	493682	493682

Tabela A.14: Resultados para $P||\sum w_j T_j$ com 200 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	4989	4989	608,24	0,186	498,873	608,426	936,417	1	28,996	0	993	0,012	4989	4989
6	62084	62084	484,146	0,156	431,683	484,301	522,652	4	60,063	0	1032	0,012	62083	62083
11	131362	131362	546,826	0,169	463,887	546,995	806,653	0	29,872	0	917	0,01	131354	131354
16	422146,6	422146,5	907,108	0,235	720,831	907,344	1059,48	36	715,976	0	1058	0,035	422078	422078
21	613999,9	613999,9	852,429	0,205	788,271	852,634	892,745	3	83,398	0	998	0,014	613999	613999
26	86	86	756,237	0,143	708,917	756,381	790,535	0	2	0	963	0,008	86	86
31	34990	34990	1136,563	1,623	772,279	1138,186	1424,87	7	272,921	0	1042	0,019	34990	34990
36	139184,9	139184,9	969,637	0,398	655,343	970,035	1215,579	28	491,2	0	1021	0,029	139184	139184
41	360317,1	360315	832,153	0,253	747,819	832,406	882,947	38	721,94	0	976	0,039	360307	360306
46	571742,5	571742,4	689,905	0,276	654,246	690,182	722,236	20	362,633	0	1005	0,03	571739	571739
51	0	0	13,255	0,067	11,669	13,322	14,54	0	0	0	200	0,005	0	0
56	10247	10247	554,404	0,272	527,759	554,676	604,911	3	59,517	0	1046	0,014	10247	10247
61	108713,6	108712,3	740,372	0,226	718,346	740,598	800,719	24	469,526	0	989	0,027	108698	108698
66	282748,9	282748,6	921,182	0,299	884,29	921,482	1042,427	21	536,926	0	1029	0,019	282740	282740
71	444816	444815,8	792,184	0,264	749,775	792,448	844,926	33	666,702	0	1023	0,033	444804	444803
76	0	0	15,188	0,065	13,72	15,253	17,093	0	0	0	200	0,004	0	0
81	1200,1	1199,7	810,379	0,239	754,456	810,618	939,175	17	372,48	1	1441	0,062	1197	1196
86	80036,5	80033	905,79	0,241	886,588	906,031	937,69	34	771,344	0	976	0,038	80031	80027
91	205874,6	205872,3	931,065	0,302	911,959	931,367	960,759	19	483,758	0	972	0,024	205860	205859
96	326001,4	325999,4	809,42	0,266	749,771	809,686	898,065	35	724,686	0	948	0,031	325975	325975
101	0	0	19,334	0,06	17,367	19,395	21,574	0	0	0	200	0,003	0	0
106	0	0	19,987	0,06	18,74	20,047	21,414	0	0	0	200	0,002	0	0
111	25170,7	25169,8	1092,673	0,251	1063,978	1092,924	1185,612	27	733,495	0	998	0,044	25163	25163
116	162627,4	162627,4	1000,087	0,272	948,246	1000,359	1049,669	19	490,242	0	973	0,033	162621	162621
121	256388,1	256386,7	935,352	0,269	890,855	935,621	957,921	39	925,608	0	929	0,032	256374	256369

Tabela A.15: Resultados para $P||\sum w_j T_j$ com 200 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	2553	2553	789,257	0,154	761,582	789,411	840,813	1	83,858	0	397	0,014	2553	2553
6	28515	28515	888,205	0,298	736,093	888,504	918,024	2	141,573	0	411	0,011	28515	28515
11	58510,1	58510,1	838,074	0,414	774,823	838,488	888,375	16	637,627	0	369	0,007	58508	58508
16	181802,9	181802,8	1014,325	0,313	787,704	1014,639	1102,654	16	657,762	0	422	0,012	181795	181795
21	261160,7	261160,7	772,07	0,358	733,604	772,427	801,149	14	555,813	0	399	0,011	261159	261159
26	203,5	203,5	1157,048	0,129	1052,956	1157,176	1234,042	0	7,636	0	386	0,007	202	202
31	16864	16864	949,13	0,298	878,644	949,429	994,569	17	752,649	0	415	0,011	16860	16860
36	62418,8	62418,8	1086,934	0,461	955,455	1087,396	1203,624	15	764,559	0	408	0,013	62382	62382
41	158430,9	158430,3	1113,412	9,927	957,226	1123,339	1670,342	14	1166,133	0	390	0,321	158419	158416
46	244549	244549	830,128	4,586	724,447	834,714	1031,536	19	939,64	0	401	0,395	244541	244541
51	0	0	33,367	0,066	31,181	33,433	38,58	0	0	0	200	0,002	0	0
56	5533	5533	789,597	0,29	759,961	789,886	874,645	2	125,391	0	417	0,01	5533	5533
61	50543,5	50543,5	1562,122	4,767	1397,758	1566,889	1649,707	13	1005,616	0	395	0,012	50523	50523
66	126122,1	126121,5	1487,353	7,921	1357,476	1495,274	1687,692	18	1275,927	0	411	0,011	126087	126081
71	192768,2	192768,2	1403,556	11,151	1311,45	1414,707	1678,881	13	944,194	0	408	0,02	192756	192756
76	0	0	36,449	0,077	32,945	36,526	39,643	0	0	0	200	0,004	0	0
81	1186,5	1186,5	1589,605	3,527	1368,227	1593,132	1876,258	10	734,849	0	416	0,009	1184	1184
86	42730,6	42730,1	1648,818	3,797	1380,932	1652,615	1771,747	16	1202,852	0	391	0,01	42714	42714
91	95306,6	95306,6	1560,077	4,086	1396,525	1564,163	1671,183	16	1126,082	0	389	0,063	95286	95286
96	144322,2	144322,1	1523,407	8,338	1285,852	1531,745	1787,058	19	1278,451	0	380	0,008	144303	144303
101	0	0	44,414	0,098	38,977	44,512	48,088	0	0	0	200	0,003	0	0
106	0	0	67,965	0,084	57,608	68,05	81,347	0	2,683	0	200	0,003	0	0
111	15977,6	15977,6	1067,454	0,438	1003,117	1067,892	1347,944	17	934,999	0	399	0,01	15960	15960
116	75979,5	75979,5	1044,022	0,825	1001,884	1044,847	1240,364	19	975,805	0	390	0,012	75940	75940
121	115320,9	115320,9	904,058	0,413	875,136	904,47	940,69	19	848,54	0	373	0,013	115307	115307

Tabela A.16: Resultados para $P||\sum w_j T_j$ com 300 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	21429	21429	3640,944	1,825	3642,769	3642,769	3642,769	13	2980,234	0	6674	0,21	21429	21429	18576
6	214326	214326	620,649	0,69	621,339	621,339	621,339	0	35,782	0	70012	0,012	214326	214326	244811
11	662310	662310	3770,722	0,337	3771,059	3771,059	3771,059	0	24,183	0	300	0,014	662310	662310	721631
16	1664089	1664088	3863,319	0,963	3864,282	3864,282	3864,282	2	2072,562	8	2680	0,288	1664089	1664088	1861524
21	2891990,9	2891956,7	1645,052	3,968	986,191	1649,02	7364,97	66	663,691	0	6640	0,908	2891957	2891956	2879379
26	176	176	812,628	12,029	442,36	824,657	3977,677	0	0	0	6613	0,352	176	176	22
31	110836	110836	605,708	0,671	606,057	606,379	606,701	2	81,944	0	68570	0,013	110836	110836	116331
36	649749	649749	607,157	1,142	606,173	608,298	610,424	4	184,315	0	134025	0,012	649749	649749	556315
41	1544567	1544567	645,225	0,478	638,725	645,703	652,681	5	322,96	0	65047	0,005	1544566	1544544	1404153
46	2505831	2505831	3686,981	0,777	3687,758	3687,758	3687,758	4	3583,971	0	300	0,011	2505832	2505831	2663370
51	0	0	42,642	0,133	41,544	42,775	44,005	0	0	0	300	0,012	0	0	0
56	41066	41066	3660,9	0,507	3661,407	3661,407	3661,407	0	80,288	0	300	0,011	41066	41066	43074
61	385883	385879	4054,906	0,768	4055,674	4055,674	4055,674	4	4038,383	5	1800	0,114	385883	385879	366023
66	1221091	1221071	4066,047	0,836	4066,883	4066,883	4066,883	7	3790,132	3	2752	0,167	1221091	1221071	1039242
71	1897516	1897515	3981,979	1,068	3983,047	3983,047	3983,047	0	618,265	5	1800	0,458	1897516	1897515	2118511
76	0	0	73,607	0,181	73,788	73,788	73,788	0	0	0	300	0,046	0	0	0
81	2623	2623	1298,451	10,302	505,387	1308,753	3662,348	1	12,577	0	6564	0,489	2623	2623	2591
86	378765	378728	3829,81	0,65	3830,46	3830,46	3830,46	1	1956,53	6	2017	0,173	378765	378728	278559
91	920641	920543	3665,129	1,903	3667,032	3667,032	3667,032	1	1326,906	9	3000	0,742	920641	920543	938877
96	1520269	1520230	4375,274	1,231	4376,505	4376,505	4376,505	4	4137,709	5	1800	0,485	1520269	1520230	1433978
101	0	0	563,525	0,133	563,658	563,658	563,658	0	0	0	300	0,01	0	0	0
106	0	0	576,324	0,131	576,455	576,455	576,455	0	0	0	300	0,006	0	0	0
111	128069	128011	3804,558	2,179	3806,737	3806,737	3806,737	3	3228,658	9	3000	0,417	128069	128011	146147
116	569252	569252	1851,144	0,073	1851,217	1851,217	1851,217	15	1768,887	0	5877	0,004	569252	569252	468343
121	1035960	1035937	3621,215	0,727	3621,942	3621,942	3621,942	1	1441,952	5	1685	0,123	1035960	1035937	1039943

Tabela A.17: Resultados para $P||\sum w_j T_j$ com 300 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	11506,5	11506,5	2100,313	0,376	1989,77	2100,689	2201,608	2	149,369	0	3337	0,015	11500	11500	10130
6	110848,1	110848,1	2026,644	0,451	1969,908	2027,095	2091,699	2	174,107	0	3499	0,021	110837	110837	126481
11	339065	339065	4054,106	1,326	4055,432	4055,432	4055,432	3	3147,925	0	300	0,007	339065	339065	369359
16	846711,1	846710,9	2654,472	0,579	2259,498	2655,051	3054,535	21	1332,171	0	3394	0,22	846710	846710	946632
21	1466359	1466354	1000,509	5,677	345,275	1006,186	3603,685	62	226,374	0	664256	1,025	1466355	1466349	1459646
26	80	80	1706,054	2,877	84,671	1708,931	4947,968	27	54,67	0	330818	0,034	189	75	36
31	58101	58101	4212,185	0,69	4212,875	4212,875	4212,875	0	929,192	0	300	0,008	58101	58101	60656
36	332686	332681	4456,252	0,911	4457,163	4457,163	4457,163	3	4124,996	7	2290	0,25	332686	332681	285151
41	788265	788262	2591,242	6,954	425,793	2598,196	11238,455	97	416,198	0	650808	0,937	788236	788221	718399
46	1273112,2	1273109,2	2228,476	6,449	375,074	2234,924	9650,294	6	28,143	0	680492	0,286	1273097	1273084	1352353
51	0	0	65,361	0,153	56,648	65,514	72,623	0	0	0	300	0,003	0	0	0
56	22192	22192	1677,87	6,466	260,213	1684,336	7361,697	45	112,902	0	678573	0,218	22142	22142	23068
61	198939,1	198936,6	504,513	2,777	468,954	507,29	801,014	40	326,735	0	3411207	1,323	198877	198869	190325
66	626732,1	626728,5	514,587	3,93	472,887	518,516	877,219	42	452,041	0	3441759	0,38	626637	626601	534484
71	966234,9	966230,5	512,971	3,112	471,78	516,083	888,794	45	455,157	0	3403349	0,42	966194	966180	1078051
76	0	0	78,595	0,185	71,81	78,78	87,989	0	0	0	300	0,004	0	0	0
81	2194,6	2192,8	2419,707	6,298	352,621	2426,005	10689,944	75	269,096	0	656745	0,387	2190	2189	2182
86	200239,4	200213,4	1867,427	5,049	528,29	1872,476	7201,677	79	417,185	0	644000	0,369	200206	200154	146941
91	473659	473657	5286,84	0,759	5287,599	5287,599	5287,599	2	3854,681	4	1500	0,119	473659	473657	481976
96	776501,4	776466,8	2568,242	11,721	434,274	2579,963	11123,703	19	86,086	0	648716	2,672	776453	776411	733190
101	0	0	106,213	3,912	90,675	110,125	194,93	0	0	0	300	0,145	0	0	0
106	0	0	115,894	15,02	90,028	130,915	187,745	0	0	0	300	0,005	0	0	0
111	70335,6	70317,2	1907,041	5,574	582,815	1912,615	7206,851	73	432,352	0	675778	0,62	70319	70269	79011
116	296329	296329	979,702	0,061	979,763	979,763	979,763	8	925,938	0	2944	0,012	296329	296329	244917
121	532180	532180	971,944	0,06	972,004	972,004	972,004	7	735,1	0	3190	0,008	532180	532180	535470

Tabela A.18: Resultados para $P||\sum w_j T_j$ com 300 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	5479	5479	983,666	0,043	983,709	983,709	983,709	1	185,093	0	1335	0,012	5479	5479	5069
6	48854	48854	922,6	0,062	922,662	922,662	922,662	6	771,349	0	1397	0,006	48854	48854	55588
11	145441	145441	959,39	0,042	959,432	959,432	959,432	6	849,716	0	1393	0,007	145441	145441	158089
16	356564	356564	906,178	0,047	906,225	906,225	906,225	5	779,323	0	1357	0,01	356564	356564	398013
21	611434	611434	975,118	0,038	975,156	975,156	975,156	3	470,927	0	1329	0,006	611425	611423	608205
26	167	167	982,121	0,037	982,158	982,158	982,158	0	316,206	0	1324	0,006	167	167	110
31	26554	26554	978,415	0,042	978,457	978,457	978,457	0	116,059	0	1370	0,007	26554	26554	27496
36	142622	142622	972,655	0,434	973,089	973,089	973,089	4	679,033	0	1340	0,033	142622	142622	122928
41	335495	335495	1006,939	0,041	1006,98	1006,98	1006,98	5	979,179	0	1303	0,008	335457	335427	307843
46	533636	533636	1028,74	0,044	1028,784	1028,784	1028,784	5	859,674	0	1359	0,007	533612	533610	566299
51	0	0	79,939	0,137	80,076	80,076	80,076	0	0	0	300	0,009	0	0	0
56	10857	10857	957,612	0,043	957,655	957,655	957,655	4	591,881	0	1357	0,011	10857	10857	11489
61	87387	87387	918,706	0,041	918,747	918,747	918,747	0	127,567	0	1364	0,006	87387	87387	85132
66	271547	271547	912,433	0,04	912,473	912,473	912,473	4	831,533	0	1375	0,006	271547	271547	232778
71	408165	408165	993,408	0,485	993,893	993,893	993,893	5	978,975	0	1360	0,072	408165	408165	454322
76	0	0	69,927	0,129	70,056	70,056	70,056	0	0	0	300	0,009	0	0	0
81	2175	2175	957,06	0,042	957,102	957,102	957,102	2	572,539	0	1315	0,007	2175	2175	2318
86	94774	94774	1037,797	0,043	1037,84	1037,84	1037,84	3	762,036	0	1290	0,008	94774	94774	69849
91	206724	206724	1006,278	0,043	1006,321	1006,321	1006,321	4	971,7	0	1397	0,01	206724	206724	208656
96	331363	331363	1043,762	0,041	1043,803	1043,803	1043,803	0	44,944	0	1299	0,007	331363	331363	313375
101	0	0	68,188	0,159	68,347	68,347	68,347	0	0	0	300	0,006	0	0	0
106	0	0	98,144	0,147	98,291	98,291	98,291	0	0	0	300	0,006	0	0	0
111	37756	37756	1070,011	0,037	1070,048	1070,048	1070,048	3	791,69	0	1351	0,006	37756	37756	39755
116	133297	133297	990,971	0,047	991,018	991,018	991,018	3	959,736	0	1185	0,008	133297	133297	112286
121	230852	230852	942,656	0,501	943,157	943,157	943,157	3	744,273	0	1279	0,031	230852	230852	233269

Tabela A.19: Resultados para $P||\sum w_j T_j$ com 300 jobs em 20 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP	Della Croce
1	3531	3531	977,585	0,035	977,62	977,62	977,62	3	915,526	0	668	0,006	3531	3531	3458
6	28290	28290	966,186	0,072	966,258	966,258	966,258	3	763,779	0	697	0,006	28290	28290	32109
16	194344	194344	955,606	0,037	955,643	955,643	955,643	2	864,063	0	678	0,008	194344	194344	216081
11	81207	81207	999,912	0,038	999,95	999,95	999,95	3	985,37	0	695	0,008	81207	81207	88237
21	327093	327093	984,67	0,038	984,708	984,708	984,708	3	901,429	0	666	0,011	327093	327093	325044
26	338	338	1034,24	0,368	1034,608	1034,608	1034,608	1	1034,224	0	663	0,012	338	338	258
31	16119	16119	948,313	0,034	948,347	948,347	948,347	3	811,849	0	684	0,005	16119	16119	16623
36	79734	79734	1119,418	0,074	1119,492	1119,492	1119,492	1	498,298	0	670	0,01	79734	79734	69194
41	186076	186076	1025,769	0,033	1025,802	1025,802	1025,802	2	861,247	0	653	0,004	186076	186076	171724
46	287857	287857	1201,531	0,046	1201,577	1201,577	1201,577	3	1180,938	0	679	0,009	287857	287857	305007
51	0	0	182,656	0,177	182,833	182,833	182,833	0	0	0	300	0,01	0	0	0
56	7600	7600	963,21	0,067	963,277	963,277	963,277	3	763,705	0	678	0,008	7600	7600	7889
61	50817	50817	1114,334	0,039	1114,373	1114,373	1114,373	2	1114,319	0	681	0,007	50817	50817	50837
66	154150	154150	1201,073	0,039	1201,112	1201,112	1201,112	2	1098,909	0	687	0,006	154150	154150	133639
71	222846	222846	1004,788	0,045	1004,833	1004,833	1004,833	2	765,827	0	680	0,01	222846	222846	247195
76	0	0	188,947	0,198	189,145	189,145	189,145	0	0	0	300	0,011	0	0	0
81	2611	2611	1069,361	0,036	1069,397	1069,397	1069,397	2	1069,342	0	659	0,005	2611	2611	2806
86	61324	61324	1001,76	0,033	1001,793	1001,793	1001,793	1	605,871	0	647	0,006	61324	61324	46944
91	118889	118889	1227,286	0,488	1227,774	1227,774	1227,774	2	1201,85	0	697	0,02	118889	118889	119631
96	183553	183553	1132,353	0,033	1132,386	1132,386	1132,386	0	311,371	0	652	0,006	183553	183553	174566
101	0	0	166,736	0,154	166,89	166,89	166,89	0	0	0	300	0,006	0	0	0
106	0	0	233,765	0,142	233,907	233,907	233,907	0	12,102	0	300	0,007	0	0	0
111	28821	28821	1121,433	0,032	1121,465	1121,465	1121,465	2	1030,063	0	676	0,005	28821	28821	28446
116	80154	80154	992,219	0,03	992,249	992,249	992,249	2	936,643	0	598	0,005	80154	80154	69999
121	131656	131656	930,607	0,389	930,996	930,996	930,996	2	847,748	0	642	0,015	131656	131656	133432

Tabela A.20: Resultados para $P||\sum w_j T_j$ com 500 jobs em 2 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	51721	51721	900,77	0,159	900,929	900,929	900,929	0	23,936	0	11909	0,009	51721	51721
6	538680	538680	7124,297	2,389	7126,686	7126,686	7126,686	0	1046,692	0	500	0,009	538680	538680
11	1908749	1908749	1042,686	0,155	1042,841	1042,841	1042,841	1	451,859	0	12037	0,006	1908749	1908749
16	4306842	4306842	7697,969	1,695	7699,664	7699,664	7699,664	1	6000,782	0	500	0,01	4306882	4306842
21	8612154	8612154	1063,453	0,146	1063,599	1063,599	1063,599	4	962,246	0	10503	0,006	8612154	8612154
26	234	234	987,24	0,162	987,402	987,402	987,402	0	22,156	0	13327	0,006	234	234
31	274098	274098	943,971	0,149	944,12	944,12	944,12	2	563,732	0	10562	0,007	274098	274098
36	1544136	1544136	1041,168	0,143	1041,311	1041,311	1041,311	4	857,662	0	10461	0,009	1544136	1544136
41	4317865	4317865	956,826	0,156	956,982	956,982	956,982	1	551,783	0	11493	0,006	4317865	4317865
46	6466537	6466537	904,52	0,153	904,673	904,673	904,673	3	712,833	0	11795	0,006	6466537	6466537
51	0	0	176,762	0,302	177,064	177,064	177,064	0	0	0	500	0,009	0	0
56	93593	93593	3604,656	1,15	3605,806	3605,806	3605,806	0	2639,907	0	3913	0,064	93593	93593
61	1031520,8	1031520,8	21692,751	0,81	21158,535	21693,562	22516,088	0	5140,446	0	5657	0,133	1031336	1031336
66	3353530,4	3353476,8	23364,485	1,028	22547,579	23365,513	24504,513	0	12769,735	0	4317	0,118	3353386	3353332
71	5785525,8	5785525,7	389,21	5,989	347,945	395,199	617,359	10	0	0	10708783	0,57	5785192	5785191
76	0	0	185,047	0,305	185,352	185,352	185,352	0	0	0	500	0,006	0	0
81	6703,7	6702,1	381,59	6,367	351,996	387,957	578,506	3	164,424	0	12126981	0,592	6700	6700
86	619852,1	619773	401,512	6,102	373,554	407,614	595,717	7	0	0	11545347	0,565	619731	619547
91	2021916,8	2021898,3	392,707	6,165	363,644	398,872	562,197	2	163,9	0	11542862	0,778	2021717	2021651
96	3408471,5	3408409,7	390,242	6,307	348,624	396,549	602,743	1	132,143	0	12842258	0,738	3408135	3408088
101	0	0	145,229	0,326	145,555	145,555	145,555	0	0	0	500	0,015	0	0
106	0	0	175,998	0,307	176,305	176,305	176,305	0	0	0	500	0,007	0	0
111	401344,9	401190,1	429,394	8,097	371,622	437,491	580,122	3	429,231	0	10486030	1,255	400863	400735
116	1259663,6	1259572,3	711,498	4,089	653,079	715,587	1186,433	14	495,94	0	11770566	0,554	1259309	1259309
121	2793626	2793626	940,12	0,176	940,296	940,296	940,296	2	713,841	0	11297	0,006	2793626	2793626

Tabela A.21: Resultados para $P||\sum w_j T_j$ com 500 jobs em 4 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	27271	27271	966,371	0,208	966,579	966,579	966,579	0	121,409	0	15521	0,015	27271	27271
6	275149	275149	1062,934	0,175	1063,109	1063,109	1063,109	3	933,375	0	16270	0,008	275149	275149
11	967482	967482	997,414	0,164	997,578	997,578	997,578	1	515,243	0	15457	0,009	967482	967482
16	2176202,5	2176202,5	398,529	6,082	366,081	404,611	585,457	5	0	0	15433211	0,65	2175706	2175706
21	4341616	4341616	1166,996	0,177	1167,173	1167,173	1167,173	2	1131,244	0	16218	0,007	4341616	4341616
26	218	218	1032,466	0,163	1032,629	1032,629	1032,629	0	51,071	0	14819	0,007	218	218
31	141362,1	141362,1	398,301	6,265	369,761	404,566	616,264	7	346,52	0	16189376	0,943	141325	141325
36	785695,4	785695,4	403,911	6,609	380,271	410,521	557,855	4	285,338	0	16240962	1,206	785403	785403
41	2187188,5	2187188,5	439,615	6,544	402,814	446,159	617,467	1	196,832	0	15727821	0,891	2186839	2186826
46	3265779	3265779	1236,911	0,169	1237,08	1237,08	1237,08	1	515,16	0	15578	0,007	3265779	3265779
51	0	0	223,495	0,318	223,813	223,813	223,813	0	0	0	500	0,009	0	0
56	49579	49579	1195,496	0,175	1195,671	1195,671	1195,671	3	1195,468	0	15192	0,006	49579	49579
61	527011	527011	970,234	0,174	970,408	970,408	970,408	1	946,091	0	15897	0,006	527011	527011
66	1702732,6	1702732,6	749,367	6,767	680,53	756,135	1220,835	6	432,83	0	16032281	1,312	1702507	1702507
71	2923444,4	2923435,8	2092,463	6,675	668,854	2099,137	14264,071	6	439,209	0	16118199	0,999	2923221	2923221
76	0	0	234,804	0,323	235,127	235,127	235,127	0	0	0	500	0,008	0	0
81	4512	4512	1202,051	0,172	1202,223	1202,223	1202,223	2	1122,682	0	15413	0,007	4512	4512
86	322286,1	322286,1	829,66	7,141	771,616	836,8	1218,404	1	219,857	0	15703290	1,383	322103	322103
91	1033155,4	1033150,6	899,476	5,903	822,751	905,379	1448,117	12	821,678	0	15703280	0,348	1032964	1032964
96	1728255	1728255	1172,938	0,154	1173,092	1173,092	1173,092	1	1098,825	0	15060	0,007	1728255	1728255
101	0	0	211,858	0,316	212,174	212,174	212,174	0	0	0	500	0,01	0	0
106	0	0	222,024	0,283	222,307	222,307	222,307	0	0	0	500	0,005	0	0
111	211703	211703	1415,868	0,186	1416,054	1416,054	1416,054	0	630,942	0	16228	0,007	211703	211703
116	646583	646583	1203,448	0,187	1203,635	1203,635	1203,635	1	908,193	0	15590	0,007	646583	646583
121	1419361	1419361	1076,51	0,174	1076,684	1076,684	1076,684	0	346,666	0	15826	0,006	1419361	1419361

Tabela A.22: Resultados para $P||\sum w_j T_j$ com 500 jobs em 10 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	12576	12576	1203,434	0,089	1203,523	1203,523	1203,523	1	819,263	0	6210	0,009	12576	12576
6	117058	117058	1505,078	0,758	1505,836	1505,836	1505,836	1	1125,597	0	6502	0,013	117058	117058
11	402960	402960	1388,083	0,097	1388,18	1388,18	1388,18	1	913,05	0	6186	0,006	402960	402960
16	897607	897607	1597,369	0,082	1597,451	1597,451	1597,451	1	1314,389	0	6176	0,006	897607	897607
21	1779663	1779663	1698,456	0,085	1698,541	1698,541	1698,541	0	232,016	0	6483	0,005	1779663	1779663
26	251	251	1755,269	0,651	1755,92	1755,92	1755,92	0	767,073	0	500	0,011	251	251
31	61899	61899	1294,343	0,088	1294,431	1294,431	1294,431	1	1016,032	0	6471	0,006	61899	61899
36	330172	330172	1550,428	0,687	1551,115	1551,115	1551,115	0	119,567	0	6490	0,011	330172	330172
41	909672	909672	1151,159	0,926	1152,085	1152,085	1152,085	0	136,889	0	500	0,309	909672	909672
46	1346009	1346009	970,438	0,909	971,347	971,347	971,347	0	620,272	0	500	0,296	1346009	1346009
51	0	0	450,353	0,299	450,652	450,652	450,652	0	0	0	500	0,009	0	0
56	23545	23545	1446,492	0,099	1446,591	1446,591	1446,591	1	1148,512	0	6083	0,012	23545	23545
61	224958	224958	936,489	0,628	937,117	937,117	937,117	0	936,465	0	500	0,012	224958	224958
66	713629	713629	1125,971	0,634	1126,605	1126,605	1126,605	0	1081,28	0	500	0,018	713629	713629
71	1206974	1206974	1175,401	0,661	1176,062	1176,062	1176,062	0	984,335	0	500	0,01	1206974	1206974
76	0	0	385,973	0,314	386,287	386,287	386,287	0	0	0	500	0,009	0	0
81	3868	3868	1525,108	0,658	1525,766	1525,766	1525,766	0	993,854	0	500	0,009	3868	3868
86	145608	145608	1615,892	0,643	1616,535	1616,535	1616,535	0	412,284	0	500	0,012	145608	145608
91	441599	441599	1345,104	0,66	1345,764	1345,764	1345,764	0	260,977	0	500	0,012	441599	441599
96	722056	722056	1144,954	0,697	1145,651	1145,651	1145,651	0	1144,925	0	500	0,012	722056	722056
101	0	0	401,67	0,31	401,98	401,98	401,98	0	0	0	500	0,007	0	0
106	0	0	422,098	0,316	422,414	422,414	422,414	0	0	0	500	0,006	0	0
111	100580	100580	1553,415	0,65	1554,065	1554,065	1554,065	0	894,407	0	500	0,01	100580	100580
116	280831	280831	1484,709	0,671	1485,38	1485,38	1485,38	0	756,293	0	500	0,01	280831	280831
121	596685	596685	1199,651	0,633	1200,284	1200,284	1200,284	0	940,235	0	500	0,011	596685	596685

Tabela A.23: Resultados para $P||\sum w_j T_j$ com 500 jobs em 20 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	2105	2105	1086,152	0,863	1087,015	1087,015	1087,015	0	937,372	0	957	0,023	2105	2105
6	64536	64536	7245,11	2,448	7247,558	7247,558	7247,558	0	1370,886	0	3246	0,141	64536	64536
11	215239	215239	8575,673	4,826	8580,499	8580,499	8580,499	1	2335,764	0	3095	0,533	215239	215239
16	471816	471816	7902,697	372,704	8275,401	8275,401	8275,401	3	4203,483	0	3090	89,202	471816	471816
21	926244	926244	7859,45	3,15	7862,6	7862,6	7862,6	0	992,025	0	3237	0,147	926244	926244
26	400	400	3104,646	0,697	3105,343	3105,343	3105,343	0	276,804	0	500	0,012	400	400
31	35647	35647	8967,468	5,299	8972,767	8972,767	8972,767	5	6873,58	0	3232	0,425	35647	35647
36	179165	179165	9192,333	4,746	9197,079	9197,079	9197,079	1	6018,502	0	3242	0,288	179165	179165
41	484842	484842	8249,073	2261,243	10510,316	10510,316	10510,316	1	6126,235	0	3145	513,467	484842	484842
46	706587	706587	7361,069	2268,728	9629,797	9629,797	9629,797	1	5241,869	0	3118	518,866	706587	706587
51	0	0	1550,409	1,439	1551,848	1551,848	1551,848	0	0	0	4849	0,29	0	0
56	14946	14946	8949,975	5,407	8955,382	8955,382	8955,382	0	1573,628	0	3046	0,4	14946	14946
61	124557,333	124557,333	7548,891	46,003	7403,132	7594,894	7728,452	8	3597,998	0	3177	0,359	124433	124433
71	635977	635977	8277,914	4,637	8282,551	8282,551	8282,551	0	3694,624	0	3219	0,358	635977	635977
76	0	0	1442,022	1,164	1443,186	1443,186	1443,186	0	0	0	4827	0,208	0	0
81	4068,333	4068,333	7980,958	5,822	7210,36	7986,78	9216,021	11	4798,997	0	3088	0,678	4001	4001
86	88091	88091	10665,545	6,026	10671,571	10671,571	10671,571	1	8624,142	0	3140	0,635	88091	88091
91	245500	245500	8751,113	68,695	8819,808	8819,808	8819,808	0	3240,778	0	3140	23,237	245500	245500
96	387676	387676	8704,891	4,855	8709,746	8709,746	8709,746	0	1115,313	0	3021	0,332	387676	387676
101	0	0	1490,713	1,651	1492,364	1492,364	1492,364	0	0	0	4810	0,261	0	0
106	0	0	1774,072	1,676	1775,748	1775,748	1775,748	0	0	0	4894	0,32	0	0
111	65488	65488	10240,112	4,748	10244,86	10244,86	10244,86	0	1379,459	0	3239	0,309	65488	65488
116	160788	160788	8997,374	4,746	9002,12	9002,12	9002,12	0	1442,48	0	3120	0,305	160788	160788
121	324596	324596	8434,555	1411,126	9845,681	9845,681	9845,681	0	3950,413	0	3163	552,89	324596	324596
125	391963	391963	3833,207	3,168	3836,375	3836,375	3836,375	0	868,57	0	6282	0,294	391963	391963

Tabela A.24: Resultados para $P||\sum w_j T_j$ com 500 jobs em 30 máquinas.

Inst num	Avg GLS Cost	Avg MIP Cost	Avg GLS Time	Avg MIP Time	Min Time	Avg Time	Max Time	Best GLS Iter	Best GLS Time	Best MIP Index	Best MIP Arcs	Best MIP Time	Best Cost GLS	Best Cost MIP
1	6102,25	6102,25	1885,626	34,112	1870,058	1919,738	1987,404	9	1167,959	0	2073	0,847	6096	6096
6	47119,25	47119,25	1910,601	29,216	1890,787	1939,817	1977,443	11	1727,995	0	2161	0,795	47104	47104
11	152829	152829	1912,043	61,458	1929,894	1973,501	2014,729	12	1711,043	0	2064	6,686	152816	152816
16	330464	330464	1869,714	4177,459	1809,59	6047,173	18529,9	9	1897,801	0	2062	2360,345	330373	330373
21	642209,5	642209,5	1869,872	3970,309	1859,132	5840,182	17689,35	9	1543,546	0	2157	2299,64	642204	642204
26	541	541	18541,17	10,91	18552,08	18552,08	18552,08	3	17740,66	0	1990	0,633	541	541
31	27125,25	27125,25	1880,755	33,863	1883,128	1914,618	1935,974	12	1859,676	0	2152	1,899	27075	27075
36	129122,5	129122,5	1912,372	31,504	1809,053	1943,877	2032,253	1	344,964	0	2159	0,677	129087	129087
41	343887,3	343887,3	1905,005	30,561	1848,066	1935,566	2033,761	2	672,338	0	2097	0,701	343870	343870
46	493933,8	493933,8	1967,225	139,144	1904,081	2106,369	2510,461	3	1014,338	0	2080	0,646	493901	493901
51	0	0	261,197	1,884	88,238	263,081	911,96	0	0	0	23356	0,411	0	0
56	12529,5	12529,5	1906,488	31,8	1873,954	1938,288	2026,455	6	1737,12	0	2034	0,757	12458	12458
61	92271,75	92271,75	9060,295	51,439	2046,417	9111,734	30077,21	2	728,631	0	2117	1,582	92174	92174
66	276643,4	276643,4	1216,312	65,078	1234,585	1281,39	1312,314	2	1130,704	0	2132	10,487	276595	276595
71	446491,2	446491,2	1103,268	37,313	1064,074	1140,581	1289,612	0	522,409	0	2143	0,861	446445	446445
76	0	0	354,54	3,201	128,279	357,741	997,613	0	0	0	23414	0,614	0	0
81	4378,667	4378,667	4975,188	70,75	3924,641	5045,938	7148,537	15	3561,445	0	2060	8,977	4317	4317
86	69824,4	69824,4	1234,465	35,171	1095,75	1269,637	1679,615	0	836,07	0	2095	0,651	69775	69775
91	181499,8	181499,8	1247,982	75,615	982,612	1323,596	1683,617	2	1310,318	0	2095	13,319	181379	181379
96	277030,8	277030,8	1220,937	39,112	1060,061	1260,049	1457,81	0	669,508	0	2018	0,743	276970	276970
101	0	0	339,693	3,221	159,908	342,914	1039,572	0	0	0	23789	0,613	0	0
106	0	0	382,013	3,445	137,346	385,458	977,112	0	234,583	0	24360	0,651	0	0
111	54883,6	54883,6	1510,408	34,578	1321,713	1544,986	1728,102	1	1680,69	0	2156	1,034	54854	54854
116	122135,8	122135,8	1058,508	39,275	1027,196	1097,782	1164,855	1	1112,458	0	2082	1,159	122022	122022
121	235276,6	235276,6	1104,75	37,826	966,156	1142,576	1380,63	0	547,273	0	2108	0,753	235200	235200