UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO HÍBRIDO MULTIOBJETIVO PARA O PROBLEMA *FLEXIBLE JOB SHOP*

LUIZ CARLOS FELIX CARVALHO

Uberlândia - Minas Gerais
2015

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUIZ CARLOS FELIX CARVALHO

ALGORITMO HÍBRIDO MULTIOBJETIVO PARA O PROBLEMA *FLEXIBLE JOB SHOP*

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial.

Orientadora:
Prof[a]. Dr[a]. Márcia Aparecida Fernandes

Uberlândia, Minas Gerais
2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

C331a
2015

Carvalho, Luiz Carlos Felix, 1983-
Algoritmo híbrido multiobjetivo para o problema flexible job shop / Luiz Carlos Felix Carvalho. - 2015.
157 f. : il.

Orientadora: Márcia Aparecida Fernandes.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Computação evolutiva - Teses. 3. Flexible Job Shop - Teses. 4. Otimização combinatória - Teses. I. Fernandes, Márcia Aparecida. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Algoritmo híbrido multiobjetivo para o problema *Flexible Job Shop***" por **Luiz Carlos Felix Carvalho** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 15 de Janeiro de 2015

Orientadora: _____

Profª. Drª. Márcia Aparecida Fernandes
Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Anderson Rodrigues dos Santos
Universidade Federal de Uberlândia

_____

Prof. Dr. Anderson da Silva Soares
Universidade Federal de Goiás

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Janeiro de 2015

Autor:	**Luiz Carlos Felix Carvalho**
Título:	**Algoritmo híbrido multiobjetivo para o problema *Flexible Job Shop***
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

_____

Autor

Dedicatória

Dedico este trabalho à minha mãe, minha esposa, minha irmã e, em especial, ao meu futuro bebê. Que Deus possa sempre nos unir e nos abençoar.

Agradecimentos

Primeiramente, agradeço a Deus por estar ao meu lado nos momentos mais difíceis, tanto neste trabalho, quanto em qualquer outra atividade que exerço.

À minha esposa e ao meu futuro bebê. À minha esposa por ser meu refúgio, por me apoiar em todas as minhas decisões e por me ensinar todos os dias um novo significado da palavra amor. Ao meu futuro bebê, que embora tenha algumas semanas de existência, ele me dá forças para os momentos finais deste trabalho.

À minha mãe, que me educou e, desde minha infância, me ensinou e me incentivou a estudar e, com sua simples presença e amor, sempre me entusiasma a buscar novos conhecimentos.

À minha irmã e seu esposo, que sempre estão presentes em minha vida. Aos meus sobrinhos, que transformam momentos simples em oportunidades especiais através de uma alegria verdadeira.

À professora Márcia, pelos ensinamentos, compreensão, oportunidade, conselhos, paciência e dedicação neste trabalho, que, com certeza, não seria concretizado sem sua orientação. Ao professor Paulo, por aplicar seus conhecimentos ao trabalho, mesmo sem vínculo formal.

"Porque quando estou fraco, então sou forte."
(2 Coríntios 12:10b)
"Há últimos que serão os primeiros, e há primeiros que serão os últimos."
(Lucas 13:30)
"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis."
(José de Alencar)

Resumo

Flexible Job Shop (FJSP) é um importante problema de otimização combinatorial, que tem sido largamente pesquisado através de técnicas da Computação Evolutiva. Algoritmos baseados em *Particle Swarm Optimization* (PSO), por exemplo, têm apresentado bons resultados para esse problema, porém tendem a convergir prematuramente. Por outro lado, Algoritmos Genéticos (AGs) têm a capacidade de tratar um amplo espaço de busca, mas não garantem a convergência. O aspecto multiobjetivo desse problema tem sido tratado por ambas as técnicas. Essas características inspiraram este trabalho, que apresenta um algoritmo híbrido e multiobjetivo, baseado em PSO, operadores genéticos e definições do ótimo de Pareto por meio do procedimento *Fast Non-dominated Sorting* (FNS). Enquanto as características do PSO garantem a convergência, os operadores genéticos melhoram a exploração do espaço de busca. Contudo, com o objetivo de obter melhores resultados que aqueles apresentados na literatura, duas significantes inovações foram introduzidas, produzindo um novo algoritmo PSO, denominado PSO com Diversidade (DIPSO), que se mostrou eficiente para tratar o FJSP. A primeira inovação foi tomar o melhor global como o *Front-1* produzido pelo ótimo de Pareto. Assim, o melhor global não é apenas um indivíduo da população como na proposta original do PSO, mas um conjunto contendo todas as melhores soluções encontradas durante a execução do algoritmo. A segunda inovação foi a criação de um operador de cruzamento, que introduziu diversidade na população e permitiu ampliar a exploração do espaço de busca. Essas duas inovações permitiram encontrar novas soluções em três das quinze instâncias do FJSP, melhorar resultados anteriormente apresentados na literatura e reproduzir diversas soluções conhecidas, demonstrando a eficiência de DIPSO para tratar o FJSP. Observa-se ainda que DIPSO é uma nova proposta para algoritmos da área de Computação Evolutiva, que pode ser utilizada em outros problemas dessa natureza. Além da abordagem descrita em DIPSO, outras propostas foram investigadas, tais como a introdução de novos objetivos para tratar o FJSP e a utilização de novas técnicas ainda não exploradas.

Palavras chave: algoritmos evolutivos, *flexible job shop problem*, *particle swarm optimization*, algoritmo genético, *non-dominated sorting genetic algorithm*, algoritmo evolutivo multiobjetivo em tabelas.

Abstract

Flexible Job Shop (FJSP) is an important combinatorial optimization problem that has been widely researched by means evolutionary computation techniques. Algorithms based on Particle Swarm Optimization (PSO), for example, have shown good results for this problem, but tend to converge prematurely. Moreover, Genetic Algorithms (GAs) have the ability to deal with a large search space, but do not guarantee convergence. The multiobjective aspect of this problem has been considered by both of these techniques. These features have inspired this work, which presents a hybrid and multi-objective algorithm based on PSO, genetic operators and Pareto optimal settings through the procedure Fast Non-dominated Sorting (FNS). While the PSO characteristics guarantee convergence, genetic operators improve the exploitation of the search space. However, in order to reach better results than those presented in the literature, two significant innovations were introduced and a new PSO algorithm was obtained, that it is denominated PSO using Diversity (DIPSO), which is efficient to cope with FJSP. The first innovation was to take the global best as the Front-1 produced by Pareto optimal. Then, the global best is not just an individual of the population as in the original PSO, but a set containing all the best solutions found during the algorithm execution. The second innovation was the development of a crossover operator, which introduced diversity in the population and allowed to expand exploration of the search space. These two innovations allowed to find new solutions in three of the fifteen FJSP instances, improve results previously reported in the literature and reproduce several known solutions, demonstrating the DIPSO efficiency in addressing FJSP. It has been also observed that DIPSO is a new proposal of evolutionary algorithms, which can be used in other problems of this nature. In addition to the approach described in DIPSO, other proposals were investigated, such as the introduction of new objectives for the FJSP and the use of new techniques not yet explored.

Keywords: evolutionary algorithms, flexible job shop problem, particle swarm optimization, genetic algorithm, non-dominated sorting genetic algorithm, multi-objective evolutionary algorithms with many tables.

Sumário

Referências Bibliográficas 105

Lista de Abreviaturas e Siglas

AE	Algoritmo Evolutivo
AEMT	Algoritmo Evolutivo Multiobjetivo em Tabelas
AG	Algoritmo Genético
AGMT	Algoritmo Genético Multiobjetivo em Tabelas
BLG	Busca Local Guiada
BT	Busca Tabu
DIPSO	PSO com Diversidade
DX	*Diversity Crossover* (Cruzamento de Diversidade)
FIPS	*Fully Informed Particle Swarm* (PSO completamente informado)
FJSP	*Flexible Job Shop Problem* (Problema *Job Shop* Flexível)
FNS	*Fast Non-dominated Sorting*
IPOX	*Improved Precedence Operation Crossover*
JSP	Problema *Job Shop*
NSGA	*Non-dominated Sorting Genetic Algorithm*
NSGA II	*Non-dominated Sorting Genetic Algorithm* II
NSND	Novas soluções não dominadas
P-FJSP	FJSP parcial
PSO	*Particle Swarm Optimization*
SEA	*Simple Evolutionary Algorithm*
SNDL	Soluções não dominadas da literatura
T-FJSP	FJSP total

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O problema de escalonamento *Job Shop* (*Job Shop Problem*, JSP) tem motivado o interesse de um significante número de pesquisadores em diferentes áreas e tem ganhado importância através de diferentes aplicações do mundo real, tais como, sistemas de manufatura, planejamento de produções, projetos de computadores, logística, etc. [Cheng et al. 1996]. Esse problema consiste em, dado um conjunto de máquinas e um conjunto de tarefas, ou *jobs* (onde cada *job* é uma sequência de operações), encontrar um sequenciamento que combine as máquinas com as operações dos *jobs* para que essas sejam executadas em um período de tempo sem interrupção, de modo a otimizar um ou mais objetivos. Assim, o problema é encontrar um escalonamento dos *jobs* nas máquinas disponíveis [Cheng et al. 1996].

JSP é um problema complexo de combinatória, sendo provado pertencer à classe NP-Completo [Lenstra e Rinnooy 1979], para duas ou mais máquinas. Diferentes abordagens têm sido propostas para encontrar soluções para esse problema, muitas das quais baseadas em métodos como *branch and bound* [Nababan et al. 2008], programação dinâmica [Gromicho et al. 2012], programação inteira [Pan 1997], algoritmos evolutivos [Pezzella et al. 2008], bem como técnicas híbridas [Zhang et al. 2009], em que duas ou mais técnicas são associadas para compor um algoritmo.

O problema *Job Shop* flexível (*Flexible Job Shop Problem*, FJSP) é uma extensão do JSP, que permite que cada operação seja processada em mais de uma máquina. Essa característica introduz duas dificuldades adicionais ao JSP: associar adequadamente cada operação a uma máquina; e determinar o sequenciamento das operações e seus respectivos tempo de início de processamento em cada máquina [Ho e Tay 2004]. Consequentemente, o FJSP é mais próximo de modelos de aplicações do mundo real, onde são observadas características como a disponibilidade de mais de um recurso para realizar uma tarefa.

A fim de aproximar as pesquisas realizadas aos problemas do mundo real, o FJSP é tratado em diversos trabalhos de forma multiobjetivo. Nesse contexto, os objetivos mais utilizados nos trabalhos com FJSP multiobjetivo são o *makespan*, a carga de trabalho total e a carga de trabalho máxima. O primeiro controla a duração do escalonamento,

enquanto os outros avaliam a quantidade de trabalho das máquinas. Assim, podemos considerar que esses últimos controlam a qualidade da solução, visto que medem o tempo de trabalho dos recursos disponíveis.

Devido à natureza combinatorial desse problema, diversos algoritmos baseados em heurística têm sido propostos [Hsu et al. 2002] [Neifar et al. 2006]. Esses algoritmos buscam por soluções adequadas (sub-ótimas) sem, contudo, demandar alto tempo de processamento. Merecem destaque trabalhos envolvendo algoritmos evolutivos, como Algoritmos Genéticos (AGs) [Wang et al. 2010] [Gen et al. 1994] e *Particle Swarm Optimization* (PSO) [Zhang et al. 2009] [Jia et al. 2007] [Ling-li et al. 2009] [Xiao-hong et al. 2010].

O algoritmo proposto neste trabalho é um algoritmo evolutivo baseado em PSO e operadores genéticos. Tanto o PSO quanto os AGs são algoritmos evolutivos com eficiência comprovada nesse e em outros problemas [Heris e Oskoei 2014] [Hu et al. 2011] [Wang et al. 2010] [Gen et al. 1994] [Zhang et al. 2009]. A eficácia do algoritmo é testada com problemas já conhecidos e utilizados em outros trabalhos. As novas soluções que foram encontradas e as outras que foram reproduzidas mostram que o algoritmo é equivalente a outras técnicas estudadas neste trabalho. Além disso, a diversidade observada na evolução da população a cada geração possibilita acessar posições no espaço de busca ainda não exploradas.

Este trabalho também analisou outras abordagens relacionadas com o FJSP, como por exemplo, o uso de dois novos objetivos que observam a ociosidade das máquinas e a utilização de novas técnicas, ainda não exploradas com o FJSP.

1.1 Objetivos

O objetivo deste trabalho é desenvolver um algoritmo para tratar o FJSP com múltiplos objetivos, FJSP multiobjetivo, baseado em técnicas da computação evolutiva e que seja eficiente, isto é, que seja capaz de melhorar soluções já existentes e também apresentar novas soluções. Essas técnicas são utilizadas devido à capacidade de manusear um grande número de soluções candidatas, sendo apropriadas para problemas de combinatória, como é o caso do FJSP. Entretanto, algumas destas possuem convergência rápida, então, para contornar este aspecto, o algoritmo que este trabalho propõe, considera utilizar algoritmos híbridos, pois enquanto um pode manter a diversidade, o outro favorece a convergência. Assim, utilizou-se um algoritmo baseado em PSO associado a operadores genéticos. Portanto, outros objetivos são que o algoritmo proposto evite convergência prematura, mantenha a diversidade da população e, consequentemente, explore o espaço de busca de forma eficiente, apresentando boas soluções, quando comparado a outros trabalhos da literatura.

1.2 Organização

A apresentação do trabalho está organizada como se segue.

O Capítulo 2 define problemas de escalonamento e apresenta algumas classes, destacando o JSP e o FJSP. Além disso, uma caracterização de problemas multiobjetivo é apresentada juntamente com a definição de ótimo de Pareto. O capítulo é finalizado com a formulação da abordagem multiobjetivo para o FJSP.

O Capítulo 3 descreve técnicas da computação evolutiva utilizadas no desenvolvimento de algoritmos para o FJSP, tais como: Algoritmos Genéticos, *Non-dominated Sorting Genetic Algorithm* II, *Particle Swarm Optimization* e Algoritmo Evolutivo Multiobjetivo em Tabelas.

O Capítulo 4 faz a revisão da literatura, apresentando trabalhos com propostas para tratar o FJSP multiobjetivo. Em particular, foram considerados trabalhos envolvendo o aspecto multiobjetivo e algoritmos híbridos que utilizam computação evolutiva.

O Capítulo 5 descreve o algoritmo proposto neste trabalho, que se baseia nas técnicas *Particle Swarm Optimization* e operadores genéticos. Suas principais características são apresentadas e, nos experimentos, discutidas. Além disso, é apresentado um comparativo dos resultados obtidos com outros algoritmos da literatura.

O Capítulo 6 introduz novas abordagens para o FJSP multiobjetivo. Além disso, novos objetivos, tais como a ociosidade total e a ociosidade efetiva total são utilizados, bem como, novas técnicas para tratar o problema. O Capítulo 7 apresenta as conclusões do trabalho desenvolvido e algumas possibilidades de trabalhos futuros são listadas.

Capítulo 2

Otimização Multiobjetivo para o FJSP

Problemas de escalonamento possuem diversas possibilidades de aplicações, tais como sistemas de produção e manufatura, planejamento de produções, projeto de computadores e processamento de informações. De forma geral, um problema de escalonamento consiste em, através de um conjunto de tarefas e de um conjunto de recursos, ordenar a execução das tarefas e identificar quais tarefas cada recurso irá executar, observando um ou mais objetivos de otimização. Dentre os objetivos mais utilizados estão o *makespan* (tempo total para execução do escalonamento) e a carga de trabalho total (somatório da carga de trabalho de todas máquinas).

Assim, os problemas de escalonamento são caracterizados como problemas de otimização, cujas soluções são aquelas que apresentam os melhores valores para os objetivos considerados. Um problema de otimização multiobjetivo considera simultaneamente mais de um objetivo. Espera-se que as soluções satisfaçam todos os objetivos. Nota-se então, a dificuldade de obter convergência em algoritmos para este tipo de problema, pois os objetivos podem conflitarem entre si, isto é, uns podem ser de minimização e outros de maximização.

Para formular o problema tratado neste trabalho, o FJSP multiobjetivo, na seção 2.1, o problema de escalonamento é definido, nas seções 2.2 e 2.3, o JSP e o FJSP são, respectivamente, apresentados e a seção 2.4 descreve os problemas de otimização multiobjetivo. Por fim, a seção 2.5 apresenta a formulação do problema tratado neste trabalho, o FJSP multiobjetivo.

2.1 Problemas de Escalonamento

O problema de escalonamento de tarefas (*jobs*) é definido por [Bagchi 1999] como sendo um processo de otimização em que recursos limitados são alocados ao longo do tempo entre atividades paralelas e sequenciais. De acordo com [Lawler et al. 1993], o problema consiste em encontrar a alocação ótima, ao longo do tempo, entre recursos e determinadas tarefas. Do ponto de vista computacional, o escalonamento consiste em:

determinar uma ordem de execução de *jobs* (sequência de operações) e, se necessário, indicando o recurso (máquina) que será responsável por processá-los.

Esses problemas são de natureza combinatorial cuja resolução é computacionalmente complexa. Apesar de existirem problemas de escalonamento polinomiais, os problemas do mundo real se apresentam como NP-Hard, em que o tempo computacional aumenta exponencialmente, conforme o aumento do tamanho do problema, o que torna inviável a enumeração das soluções [Bagchi 1999].

Consequentemente, em problemas como esses, é comum utilizar algoritmos aproximados com tempo de execução razoável [Bagchi 1999]. Busca-se, assim, por soluções adequadas em um baixo tempo computacional. Nesse contexto, os algoritmos evolutivos podem ser considerados como uma boa opção para encontrar bons resultados para problemas de otimização (e, consequentemente, para problemas de escalonamentos), pois conseguem lidar com problemas tipicamente complexos em tempo computacional polinomial.

Diversos trabalhos têm proposto classificações para o problema de escalonamento [Graham et al. 1979], [Maccarthy e Liu 1993], [Lenstra e Rinnooy 1979] e [Guimarães 2007], pois existem diferentes tipos de características que podem ser consideradas. [Vaca 1995], por exemplo, considera restrições tecnológicas dos *jobs*, critérios de programação, natureza do escalonamento, entre outras.

Entre as características apresentadas por [Graham et al. 1979] para classificação de problema de escalonamento, cita-se o tipo de fluxo em que os *jobs* serão executados. Considerando os problemas com múltiplas máquinas, destacam-se os seguintes fluxos:

- ***Flow Shop*:** todos os *jobs* são submetidos a um fluxo de trabalho unidirecional e processados sequencialmente na mesma ordem [Bagchi 1999]. Assim, independente das características de cada *job*, todos devem ser executados na máquina 1, depois na máquina 2 e assim por diante [Guimarães 2007].
- ***Flow Shop* flexível:** o processamento do *job* é dividido em estágios. Em cada estágio, existe mais de uma possibilidade de máquina para execução dos *jobs*. Todos os *jobs* possuem a mesma ordem de execução em estágios, ou seja, todo *job* inicia a execução pelo estágio 1, depois o estágio 2 e assim por diante.
- ***Job Shop*:** diferente do *Flow Shop*, nessa classe os *jobs* não são processados sequencialmente na mesma ordem, isto é, cada *job* possui uma ordem específica de execução, o que permite que ocorram diferentes sequências de associações entre os *jobs* e as máquinas. Porém, cada *job* possui uma sequência de máquinas a seguir. Assim, um *job* pode iniciar a execução na máquina 1, enquanto outro *job* inicia a execução na máquina 3.
- **Open Shop:** cada *job* possui uma ordem de execução específica, semelhante ao *Job Shop*, porém a ordem de execução das operações não está estabelecida.

- ***Job Shop* flexível:** extensão do *Job Shop*, pois acrescenta a característica de que, cada passo de execução do *job* pode ser realizado em mais de uma máquina. Portanto, apesar de possuir uma ordem de execução, como no *Job Shop*, possui a característica, do Open Shop, que não há uma ordem de máquinas determinada para a execução do *job*.

Este trabalho tem como foco o *Job Shop* flexível, que, como citado, é uma extensão do *Job Shop*. Nas seções seguintes, esses problemas são apresentados.

2.2 *Job Shop*

O *Job Shop* (JSP) é um problema de escalonamento em que as operações dos *jobs* são dadas em sequências individuais, passando por todas as máquinas, não existindo uma ordem de execução única a ser seguida por todos os *jobs*. Como visto em [Guimarães 2007] e [Cheng et al. 1996], o problema é formulado da seguinte forma: dados m máquinas diferentes e n *jobs* diferentes para serem escalonados, sendo cada *job* uma sequência de operações, onde cada operação tem seu respectivo tempo de processamento. Observa-se assim que muitas restrições podem ser impostas nesse tipo de problema relativas às máquinas, *job* e operações. [Bagchi 1999] cita algumas restrições relacionadas ao JSP, tais como:

- Duas operações de qualquer *job* não podem ser executadas simultaneamente;
- Não é permitido interromper a execução de uma operação;
- Nenhum *job* é processado duas vezes em uma mesma máquina;
- Cada *job* é processado até o fim, porém pode haver esperas e atrasos entre a execução das operações;
- Os *jobs* podem ser iniciados em qualquer tempo, desde que não tenha um tempo de início pré-determinado;
- *Jobs* devem esperar até que a próxima máquina esteja livre;
- Nenhuma máquina pode executar mais de uma operação ao mesmo tempo;
- Os tempos de configuração das máquinas são independentes da ordem de execução e estão inclusos no tempo de execução;
- Há somente um tipo de cada máquina;
- Máquinas podem ficar ociosas durante o período de escalonamento;
- Máquinas estão disponíveis em qualquer momento;
- As restrições de processamento são conhecidas e imutáveis.

Um exemplo do problema JSP pode ser visto na Tabela 2.1, em que é apresentada uma instância com duas máquinas (M_1 e M_2), dois *jobs* (J_1 e J_2) e cada *job* possui duas operações ($O_{j,1}$ e $O_{j,2}$, onde j é o *job* J_j). Considerando que $maqOp_{ji}$ e t_{ji} são, respectivamente, a máquina de execução e o tempo de execução da operação O_{ji}, os dados da tabela mostram que: $maqOp_{1,1} = M_2$ e $t_{1,1} = 9$; $maqOp_{1,2} = M_1$ e $t_{1,2} = 5$; $maqOp_{2,1} = M_1$ e $t_{2,1} = 1$; e, $maqOp_{2,2} = M_2$ e $t_{2,2} = 7$.

Tabela 2.1: Exemplo de JSP

Job	Operação 1		Operação 2	
	Máquina	Tempo	Máquina	Tempo
J_1	M_2	9	M_1	5
J_2	M_1	1	M_2	7

2.3 *Job Shop* Flexível

Como observado na descrição do JSP, uma operação é executada em apenas uma máquina. Nesse sentido, o *Job Shop* flexível (*Flexible Job Shop Problem*, FJSP) é uma extensão do JSP, pois permite que uma operação seja executada em mais de uma máquina. Assim, deve-se determinar a máquina em que cada operação de cada *job* será executada e também a ordem de execução dessas operações. Portanto, o FJSP é considerado mais complexo do que o JSP [Jansen et al. 2000]. Quanto ao número de máquinas disponíveis para execução de cada operação do *job*, o FJSP é classificado como parcial (P-FJSP), se operações podem ser processadas por um subconjunto do conjunto de máquinas, ou total (T-FJSP), se todas as operações podem ser processadas por todas máquinas.

Todas as restrições listadas para o JSP, na seção 2.2, podem ser consideradas para o FJSP, exceto a que cita que nenhum *job* pode ser processado duas vezes em uma mesma máqiuna. No FJSP não há essa restrição devido à flexibilidade de qualquer operação ser executada por qualquer máquina, ou seja, uma máquina pode executar qualquer quantidade de operações de um mesmo *job*, respeitando apenas o número de operações do *job*.

Na Tabela 2.2, é apresentado um exemplo de instância do problema T-FJSP. Conforme a definição, é necessário informar os tempos que as máquinas gastariam para executar todas as operações. No exemplo, tem-se um problema com dez *jobs* (J_1, J_2, ..., J_{10}) e dez máquinas (M_1, M_2, ..., M_{10}), sendo que cada *job* possui três operações (O_{j1}, O_{j2} e O_{j3}, onde j é o *job* J_j). Cada linha da Tabela 2.2 apresenta o tempo de execução da operação O_{ji} (t_{ji}), para cada máquina M_k, onde $k = 1, \ldots m$. Nesse exemplo, $n = 10$ e $m = 10$, por isso, o problema é denominado 10×10 de acordo com o *benchmark* de [Kacem et al. 2002c].

Tabela 2.2: Exemplo de problema T-FJSP - problema 10×10 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
J_2	$O_{2,1}$	2	10	4	5	9	8	4	15	8	4
	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
J_3	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4
J_4	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
J_5	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
J_6	$O_{6,1}$	8	9	10	8	4	2	7	8	3	10
	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
J_7	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
J_8	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
	$O_{8,2}$	8	3	10	7	5	13	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
J_9	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
J_{10}	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

Na Tabela 2.3, é apresentado um exemplo de instância do problema P-FJSP, o que significa que nem todas as máquinas executam todas as operações. No exemplo, o problema possui oito *jobs* (J_1, J_2, ..., J_8) e oito máquinas (M_1, M_2, ..., M_8), sendo que cada *job* possui até quatro operações (O_{j1}, O_{j2}, O_{j3} e O_{j4}, onde j é o *job* J_j). Cada linha da Tabela 2.3 apresenta o tempo de execução da operação O_{ji} (t_{ji}), para cada máquina M_k, onde $k \ldots m$. Esse problema é denominado por 8×8 e, assim como o problema 10×10, foi obtido do *benchmark* de [Kacem et al. 2002c].

Para apresentar uma solução para o FJSP, utiliza-se o diagrama de Gantt, que facilita as visualizações das tarefas associadas a cada máquina, da ordem de execução das tarefas e do tempo gasto para execução do escalonamento como um todo e de cada tarefa. Neste

Tabela 2.3: Exemplo de problema P-FJSP - problema 8×8 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
J_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
J_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
J_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

diagrama são representadas graficamente as operações associadas às máquinas ao longo do tempo, indicando o tempo de início e fim das operações. Na Figura 2.1, uma solução para o problema 10×10 de [Kacem et al. 2002c] é apresentada, através do diagrama de Gantt. Nessa solução, observa-se que:

- A máquina M_1 é responsável por executar as operações $O_{1,1}$, $O_{7,1}$ e $O_{2,1}$;
- Um *job* utiliza pelo menos duas máquinas para executar, por exemplo: o J_1 utiliza as máquinas M_1, M_3 e M_4, que executam, respectivamente, $O_{1,1}$, $O_{1,2}$ e $O_{1,3}$;
- O valor do *makespan*, dessa solução, é sete e é determinado pelas máquinas M_2 e M_{10};
- O somatório da carga de trabalho de todas máquinas é 42;
- A máquina com maior carga de trabalho é a M_9, que possui seis unidades de tempo

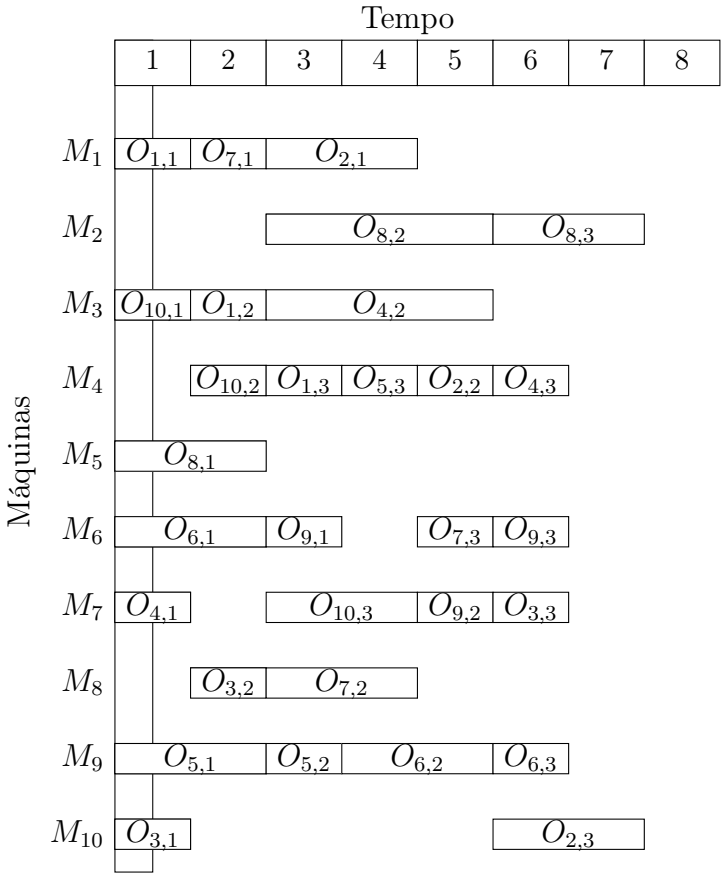

Figura 2.1: Exemplo de solução para o problema 10×10 de [Kacem et al. 2002c]

alocadas.

Esses três últimos valores são denominados, respectivamente, como *makespan*, carga de trabalho total e carga de trabalho máxima, que são utilizados, neste trabalho, como objetivos do FJSP e são definidos na seção 2.5.

2.4 Otimização Multiobjetivo

As soluções de um problema de otimização são aquelas pertencentes ao espaço de busca do problema e cujos valores da função de otimização são mínimos ou máximos, dependendo do sentido da otimização. Assim, uma definição para esse tipo de problema, de acordo com [Boyd e Vandenberghe 2004], é dada pela Equação 2.1.

$$\begin{aligned} &\min f(x) \\ \text{sujeito a} \quad &g_i(x) \leq b_i, \quad i = 1, \ldots, m \end{aligned} \tag{2.1}$$

onde, $x = (x_1, x_2, \ldots, x_n)$ é o vetor das variáveis do problema, f é a função objetivo que se deseja otimizar (minimizar), as funções g_i, $i = 1, \ldots, m$, são as restrições do

problema e as constantes $b_1, \ldots, b_m$ são os limites das restrições. Assim, um vetor x^* é uma solução para o problema (conforme descrito na Equação 2.1), se para qualquer $z \in \mathbb{R}^n$, $g_1(z) \leq b_1, \ldots, g_m(z) \leq b_m$ e $f(z) \geq f(x^*)$. Essa é uma definição para o problema de minimização. Para considerar um problema de maximização, deve-se negar a função objetivo.

A Equação 2.1 apresenta uma única função objetivo. Porém, um problema de otimização pode ser formado por mais de uma função objetivo, $f(x) = \{f_1(x), f_2(x), \ldots, f_{numobj}(x)\}$, onde cada $f_i(x)$, para $i = 1, \ldots numobj$, é uma função objetivo e $numobj$ é o número de objetivos, caracterizando assim o problema de otimização multiobjetivo. Muitos problemas de otimização podem considerar o aspecto multiobjetivo, os problemas JSP e FJSP são dois exemplos. Por isso, muitas pesquisas têm aplicado o aspecto multiobjetivo aos problemas, como podemos ver em [Cheng et al. 2011], [dao-er ji et al. 2013] e [Cheng et al. 2009], que trataram o JSP e em [Zhang et al. 2009], [Xia e Wu 2005] e [Chiang e Lin 2013] que trataram o FJSP. Nos trabalhos que trataram o FJSP, os objetivos considerados são a minimização do *makespan*, da carga de trabalho total e da carga de trabalho máxima.

Para classificar uma solução de um problema multiobjetivo, existem algumas abordagens, como:

- **Abordagem de Objetivo Único:** seleciona-se um objetivo e outros possíveis objetivos são transformados em restrições aplicadas às soluções.
- **Método da Função Utilitária:** transforma-se o problema multiobjetivo em um problema com um objetivo através de uma função auxiliar. Tipicamente, é aplicada a soma ponderada dos objetivos, indicando a importância de cada objetivo.
- **Ótimo de Pareto:** são aplicadas as definições de ótimo de Pareto e dominância, para comparar as soluções. Ao fim da execução do algoritmo o conjunto ótimo de Pareto conterá as soluções para o problema.

2.4.1 Ótimo de Pareto

Em um problema multiobjetivo, é comum ocorrerem objetivos conflitantes, pois um objetivo pode influenciar um outro e, além disso, dois objetivos podem ter sentidos opostos de otimização. Por exemplo, duas soluções conhecidas para o problema 10×10 de [Kacem et al. 2002c], considerando os objetivos *makespan* (M), carga de trabalho total (WT) e carga de trabalho máxima (W), são $M = 8, WT = 42, W = 5$ e $M = 7, WT = 42, W = 6$. Observa-se que, enquanto a carga de trabalho total é igual entre as duas soluções, a segunda solução possui *makespan* menor do que a primeira, porém possui a maior carga de trabalho máxima. Nesse caso, diminuir o *makespan* foi possível somente com o aumento da carga de trabalho máxima.

Nesse contexto, aplicar uma forma de avaliação que permita buscar não somente uma solução, mas um conjunto de soluções, é uma possibilidade para apresentar soluções com qualidade semelhantes e que satisfaça as restrições do problema. Assim, a definição de ótimo de Pareto pode ser aplicada de forma a construir um conjunto de soluções que satisfaça a necessidade de um problema multiobjetivo, não desconsiderando soluções que não são completamente ineficientes. Nas Definições 2.1 e 2.2 apresentamos as definições relacionadas à dominância de soluções, que são necessárias para definição do ótimo de Pareto, apresentada na Definição 2.3.

Definição 2.1. (Relação de Dominância) Sejam S um conjunto de soluções, x e y duas soluções que são diferentes e pertencem ao conjunto S, $f(x) = \{f_1(x), f_2(x), f_3(x), \ldots, f_{numobj}(x)\}$ funções objetivo, onde $numobj$ é o número de objetivos. Então, x domina y, se e somente se:

- **Minimização:** $f_1(x) \leq f_1(y),\ f_2(x) \leq f_2(y), \ldots, f_{numobj}(x) \leq f_{numobj}(y)$ e para pelo menos um objetivo $c, 1 \leq c \leq numobj,\ f_c(x) < f_c(y)$.
- **Maximização:** $f_1(x) \geq f_1(y),\ f_2(x) \geq f_2(y), \ldots, f_{numobj}(x) \geq f_{numobj}(y)$ e para pelo menos um objetivo $c, 1 \leq c \leq numobj,\ f_c(x) > f_c(y)$.

Definição 2.2. Conjuntos não dominados são subconjuntos obtidos a partir do conjunto S, que contém soluções que não são dominados por outras soluções. Assim, Front 1 (F_1) é o conjunto de todas soluções que não são dominadas por qualquer outra solução de S. Front 2 (F_2) é o conjunto de todas soluções que não são dominadas por qualquer outra solução de SF_1 ($S - F_1$), e assim por diante.

Definição 2.3. O conjunto ótimo de Pareto é um subconjunto de S, que contém soluções que não são dominadas por qualquer outra. Em outras palavras, o conjunto F_1.

2.5 Formulação do FJSP Multiobjetivo

Formalmente, FJSP multiobjetivo é formulado da seguinte maneira: dados n *jobs* e m máquinas, onde cada *job* é uma sequência de operações em ordem pré-definida e considerando que, todas máquinas estão disponíveis no instante de tempo $t = 0$, deseja-se obter a minimização de três objetivos: 1) *makespan* (M), 2) carga de trabalho total das máquinas (WT) e 3) carga de trabalho máxima das máquinas (W). Sendo os índices e variáveis do problema, j: índice do *job*, onde $1 \leq j \leq n$; k: índice da máquina, onde $1 \leq k \leq m$; i: índice da operação de um *job*; n_j: número de operações do *job* j; $N = \sum_{j=1}^{n} n_j$: número de operações, os objetivos são dados por:

- ***Makespan*:** é a quantidade de tempo que o escalonamento necessita para executar, ou seja, é o maior tempo final de execução, entre todas máquinas. A Equação 2.2 define o *makespan*, onde t_k é o tempo de execução final da máquina k.

$$M = max_{1 \leq k \leq m} t_k \tag{2.2}$$

- **Carga de trabalho total:** é o somatório do tempo de trabalho de todas as máquinas, não incluindo o tempo ocioso. A Equação 2.3 define a carga de trabalho total, onde W_k é a carga de trabalho, somatório do tempo de trabalho, da máquina k.

$$WT = \sum_{1 \leq k \leq m} W_k \tag{2.3}$$

- **Carga de trabalho máxima:** é a maior carga de trabalho entre todas as máquinas. A Equação 2.4 o define.

$$W = max_{1 \leq k \leq m} W_k \tag{2.4}$$

Assim, minimizar cada um desses objetivos, significa buscar soluções com um menor tempo de término de execução do escalonamento, uma menor carga de trabalho total e uma menor carga de trabalho individual, respectivamente.

A solução desse problema é uma sequência de execução de *jobs*, observando todos os objetivos de otimização. A fim de tratar o aspecto multiobjetivo, [Zhang et al. 2009] utilizou o método da função utilitária com a função de soma ponderada, atribuindo um peso para cada objetivo. Entretanto, considerar cada objetivo individualmente é mais efetivo para trabalhar com soluções de aspecto multiobjetivo, pois cada objetivo será comparado entre si, evitando que a soma de valores que possuem significados diferentes possam anular uma boa solução. Por esse motivo, o ótimo de Pareto é utilizado neste trabalho para avaliar soluções candidatas.

2.6 Considerações Finais

Neste capítulo, foram apresentados o FJSP, o conceito de otimização multiobjetivo e a formulação do FJSP multiobjetivo. O objetivo deste trabalho é tratar o FJSP multiobjetivo, conforme formulado, que é um dos problemas mais complexos de escalonamento, considerando a classificação apresentada.

Capítulo 3

Algoritmos Evolutivos

Desde a década de 1930, sendo mais intensos a partir da década de 1960, são registrados estudos que envolvem definições de Algoritmos Evolutivos (AEs). Inicialmente, os AEs eram ferramentas de simulação computacional e, posteriormente, passaram a ser adotados como algoritmos de otimização. Atualmente, as técnicas evolutivas são motivo de destaque em problemas de otimização complexos, pois têm conseguido resultados satisfatórios com tempo de execução aceitável.

Assim, apresentamos neste capítulo todas as técnicas evolutivas presentes neste trabalho. Na seção 3.1 apresentamos os principais passos dos AE. A seção 3.2 descreve os Algoritmos Genéticos. A seção 3.3, descreve um Algoritmo Genético multiobjetivo baseado nas definições de ótimo de Pareto, denominado *Non-dominated Sorting Genetic Algorithm* II. A seção 3.4 apresenta o algoritmo *Particle Swarm Optimization*, que é um algoritmo baseado em troca de conhecimento local e global. Na seção 3.5, uma nova técnica evolutiva multiobjetivo, denominada Algoritmo Evolutivo Multiobjetivo em Tabelas, é apresentada.

3.1 Visão Geral

Algoritmos Evolutivos são técnicas computacionais baseadas na teoria da evolução de Darwin, que afirma que a evolução de uma população de seres vivos ocorre através da seleção natural. Essa, por sua vez, é um processo que indica que os seres vivos com características favoráveis possuem maior possibilidade de sobrevivência. A estrutura geral de um AE aplica essas características da teoria da evolução. Um AE, geralmente, possui os seguintes passos: 1) inicialização da população, 2) criação de novos indivíduos, 3) seleção para formar uma nova população. Os passos 2 e 3 determinam a evolução da população e são repetidos até que um critério de parada seja alcançado. Cada repetição desses passos é chamada de geração. É comum que, após a execução de um certo número de gerações, boas soluções dominem a população, indicando a convergência do algoritmo.

Os AEs têm permitido a exploração de problemas complexos, como os problemas

de otimização multiobjetivo e, consequentemente, o FJSP. Nas seções que se seguem, abordaremos as técnicas evolutivas estudadas nesta pesquisa: Algoritmos Genéticos, *Non-dominated Sorting Algortihm* II, *Particle Swarm Optimization* e o Algoritmo Evolutivo Multiobjetivo em Tabelas.

3.2 Algoritmos Genéticos

Algoritmos Genéticos (AGs) são técnicas baseadas na teoria da seleção natural e da reprodução genética [Goldberg 1989] [Bagchi 1999]. Um AG é um método de busca e otimização, assim como *Simulated Annealing* e Busca Tabu, que ganhou popularidade em tratar problemas complexos de otimização visitando um grande número de soluções do espaço de busca [Bagchi 1999].

Entre as técnicas de computação evolutiva, os AGs são os mais utilizados. De modo geral, AGs começam o processo gerando uma população inicial, denominada no Algoritmo 1 como *pop*, de tamanho *tamPop*. Em seguida, cada indivíduo da população inicial é avaliado através da função de aptidão, que, de acordo com a teoria da evolução de Darwin, mede a adaptação do indivíduo ao ambiente. Caso algum indivíduo atenda ao critério de parada do algoritmo, o mesmo é apresentado como solução. Caso contrário, o algoritmo evolui a população inicial, com operadores genéticos, tais como: cruzamento, seleção e mutação. Esse processo de evolução é repetido várias vezes e termina quando um certo critério de parada é atingido [Spillman 1993] [Bagchi 1999]. No Algoritmo 1, um AG é representado.

Algoritmo 1: Algoritmo Genético

```
1  início
2      inicializar população pop com tamPop indivíduos;
3      avaliar indivíduos de pop;
4      while não atingir critério de parada do
5          selecionar percRepr indivíduos para formar pares de indivíduos para
           reprodução;
6          para cada par de indivíduos, aplica-se um operador de cruzamento,
           formando dois novos indivíduos;
7          aplicar operador de mutação aos novos indivíduos, considerando a
           probabilidade probMut;
8          avaliar novos indivíduos;
9          selecionar tamPop melhores indivíduos, considerando a população atual e os
           novos indivíduos;
10     end
11 fim
```

A aplicação do operador de cruzamento ocorre para que dois indivíduos da população gerem dois novos indivíduos. Esse processo simula a reprodução genética. Nos AGs, existe um parâmetro que indica a quantidade de indivíduos que serão selecionados para serem utilizados no processo de reprodução, aqui chamado de *percRepr*, que é uma porcentagem a ser aplicada sobre a quantidade de indivíduos da população. Assim *percRepr* indivíduos são selecionados e agrupados em pares. Cada par será utilizado para executar o cruzamento e criar dois novos indivíduos, que serão adicionados na população.

Outro parâmetro presente nos AGs é a probabilidade de um novo indivíduo ser submetido à mutação, $probMut$. Portanto, ao ser gerado, há a probabilidade de $probMut$ de um indivíduo passar pelo processo de mutação. Esse operador tem o objetivo de manter a diversidade da população, através de uma pequena alteração nos indivíduos selecionados. O próximo passo, após a mutação, é realizar a avaliação dos novos indivíduos para que os indivíduos possam ser comparados e, assim, realizar a seleção de $tamPop$ indivíduos na população, descartando os excedentes. Esse processo representa a seleção natural.

Ao realizar essa seleção, é comum manter um subconjunto dos melhores avaliados da população atual, para formar a população da próxima geração. Essa técnica é denominada de elitismo, pois considera um subconjunto de elite no processo de seleção.

A proposta original de AG consiste em otimizar um único objetivo e os indivíduos eram representados por um vetor de caracteres 0s e 1s. Entretanto, o sucesso da aplicação desses algoritmos para problemas de diferentes características ampliou o uso para o tratamento de problemas com representações mais complexas e com múltiplos objetivos[1]. Um AG multiobjetivo foi desenvolvido por [Srinivas e Deb 1994] e denominado *Non-dominated Sorting Genetic Algorithm* (NSGA), onde o ótimo de Pareto segue a definição apresentada na seção 2.4.1.

No entanto, apesar de simples, esse algoritmo possui alta complexidade computacional. De fato, o procedimento possui complexidade $O(MN^3)$, sendo M, o número de objetivos e N, o tamanho da população [Deb et al. 2002]. Além disso, o NSGA não implementa elitismo e requer a especificação de um parâmetro de compartilhamento (nomeado de σ_{share}), isto é, o NSGA possui uma função de compartilhamento, que garante sua diversidade ao longo da evolução da população [Srinivas e Deb 1994]. Assim, o desempenho do NSGA depende diretamente desse parâmetro. Tais limitações motivaram o desenvolvimento do *Non-dominated Sorting Genetic Algorithm* II (NSGA II), descrito a seguir.

3.3 *Non-dominated Sorting Genetic Algorithm* II

[Deb et al. 2002] trabalharam as limitações do NSGA para apresentar um novo algoritmo com menor complexidade computacional, utilizando elitismo e sem a necessidade

[1]FJSP é um exemplo de problema em que esses avanços permitiram resultados significantes.

de um parâmetro para manter a diversidade da população.

Para diminuir a complexidade do NSGA, o procedimento *Non-dominated Sorting* foi substituído pelo *Fast Non-dominated Sorting* (FNS). No Algoritmo 2, o pseudocódigo de FNS é apresentado e possui complexidade $O(MN^2)$. O algoritmo tem duas variáveis que se destacam, $S[\ p\ ]$ e $n[\ p\ ]$, onde $p \in P$ e P é o conjunto de soluções; $S[\ p\ ]$ é o conjunto das soluções dominadas por p e $n[\ p\ ]$ é a quantidade de soluções que domina p. Assim, ao fim de cada iteração de **for**($p \in P$), se $n[\ p\ ]= 0$, p é um elemento que não é dominado e faz parte do conjunto F_1. Depois de executar esse laço, são verificadas, repetidamente, as soluções dominadas pelos elementos do conjunto F_1, a fim de se obter o conjunto F_2, até encontrar-se todos os conjuntos de dominância.

Algoritmo 2: *Fast Non-dominated Sorting*

```
   Entrada: P: conjunto de soluções
 1 início
 2     for p ∈ P do
 3         S[p]= ∅;
 4         n[p]= 0;
 5         for q ∈ P do
 6             if p domina q then
 7                 S[p]= S[p]∪{q};
 8             else
 9                 if q domina p then
10                     n[p]= n[p]+1;
11             end
12         end
13         if n[p]= 0 then
14             rank[p]= 1;
15             F_1 = F_1 ∪ {p};
16     end
17     i = 1;
18     while F_i ≠ ∅ do
19         Q = ∅ ;
20         for p ∈ F_i do
21             for q ∈ S[p] do
22                 n[q]= n[q]−1;
23                 if n[q]= 0 then
24                     rank[q]= i + 1;
25                     Q = Q ∪ {q};
26             end
27         end
28         i = i + 1;
29         F_i = Q;
30     end
31 fim
```

No NSGA II, [Deb et al. 2002] substituem a função de compartilhamento pela com-

paração da *crowding-distance* e, consequentemente, descartam o parâmetro σ_{share}. No Algoritmo 3, é mostrado como calcular a *crowding-distance* dos pontos de um conjunto. Esse processo é usado, no NSGA II, para classificar soluções que estão em um mesmo conjunto de dominância, $F_1, F_2, \ldots, F_i$, onde i é a quantidade de conjuntos de dominância.

O cálculo de *crowding-distance* é baseado na distância média entre os pontos próximos ao ponto em questão. Sejam os pontos x_a, x, x_p, onde $x_a < x < x_p$ e nos intervalos [x_a, x] e [x, x_p] não existe outro ponto representado. Se o ponto x é o ponto a qual deseja-se efetuar o cálculo, então, os pontos a serem considerados para a distância média são x_a e x_p. Na Equação 3.1, apresentamos o cálculo da distância média (dm) de dois pontos de um conjunto qualquer.

$$dm = (x_a - x_p)/(x_1 - x_w) \tag{3.1}$$

onde, x_a, x_p, x_1 e x_w pertencem a um conjunto S de tamanho w, tal que, $S = \{x_1, \ldots, x_a, \ldots, x_p, \ldots, x_w\}$ e S está ordenado por algum critério. O cálculo da distância média é utilizado para determinar a *crowding-distance* de uma solução. Portanto, para realizar o cálculo da *crowding-distance* é necessário ter um conjunto ordenado, para cada objetivo, dos valores da aptidão. Assim, *crowding-distance* é o somatório da distância média entre os valores imediatamente anterior e posterior ao valor da aptidão de cada objetivo da solução em questão. Esse cálculo é observado no Algoritmo 3, no laço **for**($m \in M$).

Algoritmo 3: Calcular *crowding-distance*

Entrada: F: conjunto de soluções

```
1  início
2      M : objetivos_problema;
3      tam = quantidadeElementos(F);
4      for ind ∈ F do
5          ind.dist = 0;
6      end
7      for m ∈ M do
8          F = ordenar(F, m);
9          F[1].dist = ∞;
10         F[tam].dist = ∞;
11         difMaxMin = maxValor(m) − minValor(m);
12         for cont = 2 to (tam − 1) do
13             difAntPos = F[cont + 1].valorObj(m) − F[cont − 1].valorObj(m);
14             ind.dist = ind.dist + (difAntPos/difMaxMin);
15         end
16     end
17 fim
```

No NSGA II, o elitismo utiliza o operador *crowded-comparison*, que compara os indivíduos observando a classificação de dominância (*rank*) e a *crowding-distance*. Se os

indivíduos possuírem *ranks* diferentes, aquele que tiver o menor *rank* será melhor avaliado. Caso os indivíduos possuírem *rank* iguais, aquele que possuir a *crowding-distance* menor, será melhor avaliado.

O Algoritmo 4, mostra como o NSGA II evolui a população, ou seja, o pseudocódigo do laço principal, como definido por [Deb et al. 2002]. O primeiro passo do algoritmo é classificar a dominância da população, através do algoritmo FNS (Algoritmo 2) aplicado à combinação da população atual, P_t, e dos indivíduos recém gerados, Q_t. Em seguida, um laço é executado para iniciar a formação da nova população, P_{t+1}. Nesse laço, o cálculo da *crowding-distance* é realizado para os conjuntos de dominância, F_i, que, após a avaliação, são adicionados à P_{t+1}. O limite desse laço é $|P_{t+1}| + |F_i| \leq N$, sendo que N é o tamanho da população.

Em outras palavras, o processo ocorre até a última iteração que não ultrapassa o tamanho máximo da população. Assim, são descartadas avaliações desnecessárias de indivíduos que não serão aproveitados em P_{t+1}. Para preencher o restante de P_{t+1}, a *crowding-distance* de F_i é calculada e é realizada a ordenação, através do operador *crowded-comparison*, para que os indivíduos melhores avaliados de F_i sejam adicionados em P_{t+1}. Finalizando, um novo conjunto de soluções, Q_{t+1}, é construído através dos operadores genéticos de seleção, cruzamento e mutação. A operação de seleção, quando realizada, utiliza o método de torneio, que, para comparar os indivíduos, aplica o *crowded-comparison*.

Algoritmo 4: NSGA II

```
1  início
2      R_t : P_t ∪ Q_t;
3      F = fast-non-dominated-sort(R_t);
4      P_{t+1} = ∅;
5      i = 1;
6      while |P_{t+1}| + |F_i| ≤ N do
7          calcularCrowdingDistance(F_i);
8          P_{t+1} = P_{t+1} ∪ F_i;
9          i = i + 1;
10     end
11     calcularCrowdingDistance(F_i);
12     Ordenar(F_i, operador crowded-comparison);
13     P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)];
14     Q_{t+1} = construirNovaPopulacao(P_{t+1});
15     t = t + 1;
16 fim
```

3.4 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) [Kennedy e Eberhart 1995] é originado das teorias da vida artificial e da inteligência de enxames. Além disso, essa técnica está relacionada com algoritmos evolutivos, tais como AGs e programação evolutiva. Algoritmos PSO são definidos por uma população de partículas (indivíduos), sendo que cada partícula possui posição e velocidade. A evolução do algoritmo é determinada pela atualização desses dois atributos. Em cada passo de evolução, as novas posições e velocidades das partículas são calculadas levando em consideração as seguintes posições: melhor global e melhor local.

A posição de melhor global (g_{best}) está relacionada com a memória global do algoritmo e representa a melhor posição visitada, dentre todas as posições visitadas por todas partículas, em um determinado tempo de execução. A posição de melhor local (p_{best}) está relacionada com a memória da partícula e representa a melhor posição que a partícula visitou. Essas duas variáveis são importantes para a convergência do algoritmo e podem indicar o estado da população: se, na maioria das partículas, g_{best} é melhor que p_{best}, então há diversidade; caso contrário, dependendo da iteração analisada, pode-se concluir que há convergência prematura.

O Algoritmo 5 apresenta o pseudo-código para o PSO, com base na formulação de [Martínez e Gonzalo 2009]. Nesse algoritmo, tem-se uma população inicial com b partículas com posições (${x_i}^0$) e velocidades (v_i) aleatórias, para $i = 1, \ldots b$. Em cada passo $c+1$, as novas posições e velocidades são calculadas pela Equação 3.2.

$$\begin{aligned} {v_i}^{c+1} &= \omega {v_i}^c + \phi_1(g_{best} - {x_i}^c) + \phi_2(p_{best} - {x_i}^c) \\ {x_i}^{c+1} &= {x_i}^c + {v_i}^{c+1} \end{aligned} \tag{3.2}$$

onde, $\phi_1 = r_1 * a_g$, $\phi_2 = r_2 * a_l$, $\omega \in \Re$ significa a inérica, a_g e $a_l \in \Re$ são, respectivamente, as constantes de aceleração global (fator social) e local (fator cognitivo), r_1 e r_2 são números aleatórios uniformemente distribuídos em $(0, 1)$, ϕ_1 e ϕ_2 são acelerações aleatórias global e local. A posição e velocidade de cada partícula são atualizadas levando em consideração a função objetivo.

A inércia ω desempenha um papel fundamental provendo um balanceamento no processo de exploração. Esse parâmetro determina a taxa de contribuição da velocidade anterior da partícula para a nova velocidade que está sendo calculada no passo corrente [Bansal et al. 2011].

Além disso, [Bansal et al. 2011] descrevem quinze fórmulas diferentes para calcular a inérica durante as iterações do PSO e há diversas outras funções que podem ajustar esse parâmetro, como pode ser visto em [Bansal et al. 2011], [Nickabadi et al. 2011] e [Qin et al. 2006]. [Shi e Eberhart 1998] usaram a fórmula definida na Equação 3.3. Eles

Algoritmo 5: PSO

```
1 início
2   inicializar população com b partículas associadas a posições (x_i^0) e velocidades
    (v_i) aleatórias, i = 1, ... b;
3   while não atingir critério de parada do
4     avaliar todas as partículas da população;
5     avaliar p_best de toda partícula e, caso necessário, atualizar;
6     identificar a melhor partícula da população e, caso necessário, atualizar g_best;
7     atualizar a velocidade e a posição de cada partícula, através da Equação 3.2;
8   end
9 fim
```

demonstraram empiricamente que bons valores para ω_{max} e ω_{min} são 0.9 e 0.5, respectivamente. Os parâmetros que representam os fatores social e cognitivo, a_g e a_l, foram definidos como $a_g = a_l = 2$, de acordo com [Bansal et al. 2011].

$$\omega = \omega_{max} - c\frac{\omega_{max} - \omega_{min}}{c_{max}} \tag{3.3}$$

onde, ω_{max} e ω_{min} são, respectivamente, a inércia máxima e a mínima permitida no algoritmo. A iteração corrente é dada por c enquanto que o número de iterações é dado por c_{max}.

3.5 Algoritmo Evolutivo Multiobjetivo em Tabelas

O Algoritmo Evolutivo Multi-objetivo em Tabelas (AEMT) foi proposto por [dos Santos 2009]. Trata-se de um algoritmo que utiliza subpopulações em tabelas para manter as melhores soluções de cada objetivo considerado no problema, a fim de utilizar essa informação para fazer a análise multi-objetivo. O número de subpopulações proposto é igual ao número de objetivos do problema, *numObj*, mais uma. Assim, existem *numObj* subpopulações em que cada uma armazena os y (tamanho definido para os conjuntos de subpopulações) melhores indivíduos de um determinado objetivo do problema. A outra subpopulação armazena os y melhores indivíduos considerando uma função de agregação, como a soma ponderada dos objetivos, utilizada por [dos Santos 2009].

Para efetuar a troca de informações entre os indivíduos, o algoritmo utiliza dois operadores de cruzamento, sendo que a cada utilização é feita uma escolha aleatória para definir qual utilizar. No Algoritmo 6, o pseudocódigo de AEMT é apresentado. A cada iteração, um operador de cruzamento é executado, dois novos indivíduos são gerados e logo após avaliados. Esses novos indivíduos são inseridos em todas as subpopulações, porém para sobreviverem para a próxima iteração, necessitam estar entre os y melhores indivíduos em pelo menos uma subpopulação. Desta forma, a cada iteração os melhores indivíduos de cada objetivo são mantidos na população. Ao fim da execução, os indivíduos que es-

tiverem em todas as subpopulações são considerados como resposta pelo algoritmo, pois, em relação as soluções visitadas, são as melhores opções considerando todos objetivos simultaneamente.

Algoritmo 6: Algoritmo AEMT

```
1  início
2      inicializar população pop;
3      avaliar indivíduos;
4      seja no o número de objetivos considerados no problema;
5      gerar no subpopulações sbpop a partir de pop, com os y melhores indivíduos de
       cada objetivo;
6      while não atingir critério de parada do
7          selecionar dois elementos, e1 e e2, aleatoriamente e de qualquer
           subpopulação;
8          escolher operador op entre os dois operadores definidos;
9          ne = op(e1, e2);
10         avaliar os novos indivíduos ne;
11         fazer sbpop[noa] = sbpop[noa] ∪ ne, para 1 ≤ noa ≤ no;
12         redefinir sbpop[noa] selecionando os y melhores indivíduos de sbpop[noa],
           onde 1 ≤ noa ≤ no ;
13     end
14 fim
```

Brasil [Brasil 2012] adicionou uma nova subpopulação ao AEMT para aumentar a diversidade da população e o número de soluções visitadas. A nova subpopulação consiste em armazenar os indivíduos não dominados, segundo o critério de não dominância do NSGA II. Assim, esse novo algoritmo, denominado AEMT$_{ND}$, possui as seguintes subpopulações:

1. Uma subpopulação para cada objetivo (presente no AEMT);
2. Uma subpopulação para a função de agregação, no trabalho de [dos Santos 2009] foi utilizado a soma ponderada dos objetivos (presente no AEMT);
3. Uma subpopulação para os indivíduos não dominados por nenhum outro da população.

3.6 Considerações Finais

Neste capítulo, além de descrever, de forma geral, os algoritmos evolutivos, foram descritos os AGs, o NSGA II, o algoritmo PSO, o AEMT e o AEMT$_{ND}$. Neste trabalho,

todos esses algoritmos foram utilizados. AGs, PSO e NSGA II foram utilizados para formar um algoritmo PSO híbrido para tratar o FJSP multiobjetivo, enquanto o AEMT e o AEMT$_{ND}$ foram utilizados em experimentos adicionais.

Capítulo 4

Trabalhos Relacionados

Como descrito no Capítulo 3, técnicas evolutivas têm sido largamente empregadas em problemas de otimização multiobjetivo. No contexto desta pesquisa, ou seja, em relação ao problema de *Job Shop* flexível multiobjetivo, destacam-se técnicas de Algoritmo Genético [Wang et al. 2010] [Gen et al. 1994] e *Particle Swarm Optimization* [Zhang et al. 2009] [Jia et al. 2007] [Ling-li et al. 2009] [Xiao-hong et al. 2010]. Além disso, [Zhang et al. 2009], [Ling-li et al. 2009], [Xiao-hong et al. 2010], [Ho e Tay 2007] apresentam algoritmos híbridos, isto é, definidos pela combinação de diferentes técnicas, buscando assim, melhoria dos resultados até então obtidos, tanto do ponto de vista da solução propriamente dita, quanto em relação aos aspectos do algoritmo, como por exemplo a convergência prematura.

Métodos de buscas locais [Zhang et al. 2009] [Ho e Tay 2007], regras baseadas em características do problema [Chiang e Lin 2013] e técnicas de diversificação da população [Xiao-hong et al. 2010] são exemplos de variações que são aplicadas aos algoritmos evolutivos, a fim de obter alguns benefícios como aumentar a exploração do espaço de busca ou a redução da convergência prematura da população.

Alguns trabalhos que utilizaram essas técnicas são descritos nas seções 4.1, 4.2 e 4.3, respectivamente. Além disso, são apresentados os principais *benchmarks* utilizados nos trabalhos descritos.

4.1 Buscas Locais

Vários trabalhos da literatura associam um método de busca local a alguma outra técnica, como por exemplo, o PSO [Zhang et al. 2009] [Moslehi e Mahnam 2011]. Essa associação permite aumentar a diversidade da população, visitar pontos do espaço de busca que são complicados de serem explorados, dividir responsabilidades no algoritmo, aumentar a velocidade de convergência, evitar mínimos locais, entre outras possibilidades. Nesses últimos trabalhos, em todos os casos, um método de busca local está associado a uma técnica evolutiva, o que mostra que esta combinação é possível nos mais diversos

contextos.

No trabalho descrito por [Ho e Tay 2007], foi apresentado um algoritmo evolutivo associado a um método de busca local guiada (BLG). A introdução desse método, ao invés de um método de busca aleatória, é justificado para acelerar a convergência. Para sua utilização, [Ho e Tay 2007] apresentaram uma série de definições e teoremas para serem aplicados na BLG, para justificar os movimentos realizados na busca. No algoritmo, a utilização da BLG é feita usando os melhores elementos da população, porém o resultado da busca é aplicado para substituir os piores elementos, o que possibilita, segundo os autores, manter a diversidade da mesma. Tanto a seleção dos melhores elementos, quanto a seleção dos piores, são realizadas por meio de parâmetros do algoritmo.

[Zhang et al. 2009] utilizaram a busca Tabu (BT) associada ao PSO. O algoritmo apresentado tem um laço para evolução da população, que inclui a cada passo desse, um determinado número de passos da BT. Assim, ela foi adaptada para que seja uma busca local para cada partícula da população. Na utilização da BT, o método de vizinhança aplicado permite uma redução do tamanho do conjunto de vizinhos de uma solução e provê, ao mesmo tempo, que os melhores vizinhos estejam neste conjunto. Esse método de vizinhança utilizado foi apresentado por [Mastrolilli e Gambardella 2000] e permite diminuir a memória utilizada pelo algoritmo no processamento, o que, consequentemente, torna a execução do mesmo mais rápida.

A utilização de um método de busca local para refinar a busca e focar em uma parte especial do espaço de busca é a justificativa utilizada no trabalho de [Moslehi e Mahnam 2011]. Nesse, é combinado a busca local com o PSO. Assim, há uma divisão de responsabilidades no algoritmo, pois, enquanto a busca local tem o objetivo de refinar e focar a busca, o PSO tem o objetivo de realizar uma busca extensa para determinar a parte do espaço de busca em que tem maior possibilidade de conter ótimos globais.

Diferentemente das outras associações citadas, o trabalho elaborado por [qing Li et al. 2010], não combina um método de busca local com alguma técnica evolutiva. A associação realizada no trabalho é feita com algoritmos de buscas semelhantes. A representação do indivíduo utilizada na pesquisa de [qing Li et al. 2010] é dividida em dois vetores: 1) X_m: um que tem a responsabilidade de determinar qual máquina executa uma determinada operação e, 2) X_{op}: que é responsável por ordenar a execução das operações. Assim, para cada vetor da representação, há uma combinação de técnicas distintas para realizar a busca. Em X_m, BT é combinada com uma busca local a fim de que essa, possa fazer um movimento na solução, de modo que encontre uma máquina mais adequada para a respectiva operação. No caso de X_{op}, um método de busca de vizinhança variável (BVV) é elaborado. Esse, é aplicado com três modificações na solução, que podem ser inserções ou uma mudança baseada na vizinhança.

4.2 Características do Problema

Conhecimento específico do problema, como o tempo de execução de uma operação em uma determinada máquina ou a quantidade de operações que estão associadas a uma determinada máquina em uma possível solução, pode ser considerado para acelerar a convergência do algoritmo, construir indivíduos ou, em casos específicos, manter a diversidade da população. São exemplos de trabalhos que utilizaram essa possibilidade, [Kacem et al. 2002a], [Kacem et al. 2002b], [Wang et al. 2010], [qing Li et al. 2010] e [Chiang e Lin 2013].

Nos trabalhos de [Kacem et al. 2002a], [Kacem et al. 2002b] e [Wang et al. 2010] são utilizadas características do problema para formar a população inicial. [Wang et al. 2010] cita que a população inicial tem um papel muito importante no desempenho do algoritmo e, assim, justifica o uso de dados do problema. [Kacem et al. 2002a], [Kacem et al. 2002b] utilizaram o tempo de processamento e a carga de trabalho das máquinas, enquanto que, [Wang et al. 2010], usaram somente o tempo de processamento. Geralmente, uma população inicial formada aleatoriamente, possui várias soluções que estão distantes da solução ótima. Além disso, em ambos os trabalhos, a formação dirigida da população tem o objetivo de diminuir esta distância.

[Chiang e Lin 2013] usam conhecimento específico do problema para formar a população inicial e construir o algoritmo de mutação. Na construção da população inicial, as regras utilizadas foram divididas em dois grupos: roteamento (cinco regras) e sequenciamento (três regras). As regras de roteamento estão relacionadas com a determinação de qual máquina executará uma determinada operação e, as regras de sequenciamento, relacionadas com a ordem de execução das operações. A geração de um novo indivíduo é feita através de dois métodos, um de cada grupo. Para realizar a mutação, cinco regras foram adotadas, sendo que somente uma delas é escolhida para executar o processo. As regras definidas possuem o objetivo de diminuir a carga de trabalho, reduzir o *makespan*, manter a diversidade e diminuir a carga de trabalho de uma máquina sobrecarregada. Portanto, em cada caso, diferentes dados do problema são aplicados para satisfazer a necessidade respectiva.

Outra característica associada ao operador de mutação é que ele é executado sempre que um indivíduo repetido está presente na população. Assim, a diversidade na população é mantida e, consequentemente, não é necessário um parâmetro probabilístico para determinar a execução da mutação.

4.3 Diversificação da População

É comum a utilização de operadores genéticos, cruzamento ou mutação, para manter a diversidade na população [Zhang et al. 2009], [Ling-li et al. 2009], [Niu et al. 2008],

[Chiang e Lin 2013], [Xiao-hong et al. 2010] e [Jia et al. 2007]. Em todos esses casos, as técnicas centrais adotadas não usavam os operadores genéticos em seus respectivos algoritmos. Porém, para manter a diversidade da população, os operadores foram incluídos e formaram um algoritmo híbrido que é capaz de evitar mínimos locais e evitar a rápida convergência.

Deve-se observar que a rápida convergência para uma solução não é, necessariamente, uma boa característica de uma técnica evolutiva no tratamento de problemas complexos, pois o indivíduo encontrado pode ser um mínimo local. Essa característica é observada na aplicação de PSO [Jia et al. 2007]. Para aumentar a diversidade da população e evitar a convergência em mínimos locais, a utilização dos operadores genéticos é observada nos trabalhos desenvolvidos por [Jia et al. 2007], [Zhang et al. 2009] e [Xiao-hong et al. 2010]. [Jia et al. 2007] aplica dois operadores de mutação em seu algoritmo que utiliza um PSO Completamente Informado (*Fully Informed Particle Swarm*, FIPS). Essa técnica foi desenvolvida por [Mendes et al. 2004] e se baseia em PSO, diferenciando-se por utilizar informação de vários indivíduos para evoluir a população e não apenas alguns indivíduos específicos. [Zhang et al. 2009] e [Xiao-hong et al. 2010] utilizam cruzamento e mutação associados ao PSO. [Zhang et al. 2009], que utilizam uma representação de indivíduo com dois vetores, semelhante a de [qing Li et al. 2010], adotam dois operadores de cruzamento para serem aplicados de forma distinta nos vetores do indivíduo e a mutação é aplicada apenas no vetor de sequenciamento das operações.

O trabalho apresentado por [Niu et al. 2008], mostra uma redefinição do PSO, com a utilização dos operadores genéticos. Assim como nos casos citados anteriormente, a convergência rápida e a dificuldade com mínimos locais, também é relatada por [Niu et al. 2008]. Aliado ao fato de que o PSO não permite uma aplicação direta aos problemas de escalonamento, [Niu et al. 2008] utilizaram os operadores genéticos para construir uma nova equação de mudança de posição da partícula no espaço de busca, como pode ser visto na Equação 4.1.

De acordo com a Equação 4.1, as trocas de informações entre a partícula atual (P_k) e os melhores local (p_{best}) e global (g_{best}) ocorrem através do operadores de cruzamento e mutação. Assim, a nova posição (P_{k+1}) da partícula é dada pelo melhor ($selectBest$) entre as três partículas P_1, P_2 e P_3, sendo que P_1 é o resultado do cruzamento entre a partícula atual e o melhor global, P_2 é o resultado do cruzamento entre a partícula atual e o melhor local e P_3 é o resultado da mutação da partícula atual.

$$\begin{aligned} P_{k+1} &= selectBest(P_1, P_2, P_3), \text{onde} \\ P_1 &= cruzamento(P_k, g_{best}), \\ P_2 &= cruzamento(P_k, p_{best}), \\ P_3 &= mutacao(P_k) \end{aligned} \tag{4.1}$$

Também são encontrados na literatura, algoritmos evolutivos para o FJSP. São exemplos dessa aplicação, [Ho e Tay 2007] e [Chiang e Lin 2013]. Além da associação de algoritmo evolutivo e busca local guiada, que já foi discutida anteriormente, [Ho e Tay 2007] utiliza o elitismo, de forma que as soluções não dominadas não sejam desconsideradas na próxima geração. Em [Chiang e Lin 2013], foi construído um algoritmo evolutivo, denominado *Simple Evolutionary Algorithm* (SEA), que requer apenas dois parâmetros para ser executado: o tamanho da população e o número máximo de gerações. Esse algoritmo utiliza conhecimento do problema para gerar a população inicial e no operador de mutação, os quais já foram apresentados neste capítulo. Na evolução da população, SEA [Chiang e Lin 2013] utiliza dois operadores de cruzamento que são escolhidos aleatoriamente e um processo de mutação. Para aplicar o cruzamento, é realizada uma seleção por torneio e a mutação é aplicada a todo indivíduo repetido que for gerado.

Os resultados apresentados pelo SEA [Chiang e Lin 2013] aumentaram a quantidade de soluções dos problemas de [Brandimarte 1993], mostrando uma maior diversidade desse quando comparado a vários outros trabalhos ([Wang et al. 2010], [qing Li et al. 2010], [Xing et al. 2009] etc.). Além desses resultados, um estudo sobre o comportamento dos objetivos amplamente utilizados nas pesquisas (*makespan*, carga de trabalho total e carga de trabalho máxima) é apresentado. Segundo as conclusões do trabalho, há um conflito entre *makespan* e carga de trabalho total e entre carga de trabalho máxima e carga de trabalho total.

Nos trabalhos analisados, o aspecto multiobjetivo pode ser visto de duas variáveis: através da soma ponderada ou do ótimo de Pareto. [qing Li et al. 2010] e [Xing et al. 2009] utilizam a soma ponderada, enquanto em [Wang et al. 2010] e [Ho e Tay 2007] são utilizadas as definições do ótimo de Pareto. A utilização de soma ponderada requer a definição de valores associados a cada objetivo, além da possibilidade de boas soluções serem superadas na avaliação por outras que não são necessariamente as melhores. Porém, a definição das melhores soluções é um cálculo rápido. Utilizar as definições de ótimo de Pareto é uma maneira mais justa de avaliar uma solução, pois os conceitos de dominância são aplicados e uma boa solução não é desprezada. No entanto, o custo computacional para realizar esse processo é considerável, visto que a comparação engloba toda a população e é aplicada em cada iteração.

Muitos trabalhos da literatura, que tratam o FJSP, utilizam como *benchmarks* os problemas apresentados por [Kacem et al. 2002c] e [Brandimarte 1993], como é visto em [Zhang et al. 2009], [Wang et al. 2010], [Ho e Tay 2007], [qing Li et al. 2010], [Chiang e Lin 2013] e [Moslehi e Mahnam 2011]. A utilização dessas bases de dados é interessante, pois torna possível a comparação de resultados com outros trabalhos semelhantes. Em [Kacem et al. 2002c] são encontrados cinco problemas e em [Brandimarte 1993] há um conjunto de dez problemas, onde, em alguns casos, o tamanho do problema (número de *jobs*, número de operações por *job* e número de máquinas) gera uma complexidade muito

maior do que aqueles presentes em [Kacem et al. 2002c].

4.4 Considerações Finais

Observamos que é justificável a combinação de técnicas para tratamento de problemas complexos, como o FJSP. Sabe-se que o PSO possui uma rápida convergência e mínimos locais podem impedir ou evitar a convergência adequada da população. Assim, a associação desse com operadores genéticos é uma opção válida que viabiliza o uso do PSO em problemas complexos.

Capítulo 5

DIPSO: um Algoritmo PSO com Diversidade para o FJSP

Conforme visto anteriormente, FJSP é um problema complexo, por isso, em vários trabalhos, observa-se a utilização de algoritmos evolutivos em propostas para o FJSP, pois são algoritmos com complexidade polinomial que possuem resultados satisfatórios em problemas de otimização [Zhang et al. 2009] [Jia et al. 2007] [Xiao-hong et al. 2010] [Wang et al. 2010] [Chiang e Lin 2013]. Nesse trabalho, o FJSP foi considerado com aspecto multiobjetivo e, para tanto, três objetivos foram utilizados: *makespan*, a carga de trabalho total e a carga de trabalho máxima. Assim, foi proposto um algoritmo multiobjetivo híbrido baseado em PSO para o FJSP multiobjetivo, denominado DIPSO (PSO com diversidade).

A fim de atender os objetivos propostos neste trabalho, isto é, evitar convergência prematura e melhoria das soluções, duas novas características foram introduzidas:

- Um operador de cruzamento que mantém a diversidade da população, denominado DX (Diversity Crossover: cruzamento de diversidade)
- O melhor global (g_{best}) em DIPSO é um conjunto com as melhores soluções[1] e não apenas uma melhor solução.

Os experimentos foram realizados com *benchmarks* encontrados na literatura e utilizados pela maioria dos trabalhos discutidos nesta pesquisa. Assim, possibilitou-se a comparação dos resultados do algoritmo DIPSO com outros trabalhos da literatura. Essa comparação mostrou que grande parte dos resultados da literatura foram reproduzidos e novas soluções foram encontradas em três instâncias, sendo que em duas, as novas soluções apresentaram melhoria em relação às soluções da literatura.

Neste capítulo apresentamos DIPSO. Na seção 5.1, uma visão geral do algoritmo proposto é apresentada. A seção 5.2 apresenta a representação da partícula e a seção 5.3

[1]Solução representada pela partícula ou indivíduo.

descreve os operadores genéticos utilizados. Na seção 5.4 é explicado como o algoritmo evolui a população. A seção 5.6 apresenta e discute os resultados dos experimentos realizados com o algoritmo DIPSO.

5.1 Visão Geral

O algoritmo *Particle Swarm Optmization* com diversidade (DIPSO) é um algoritmo evolutivo multiobjetivo híbrido baseado em PSO, que utiliza operadores genéticos (cruzamento e mutação) a fim de evitar a convergência prematura, que é uma característica do algoritmo PSO. Além disso, utiliza o procedimento FNS (Algoritmo 2) do algoritmo NSGA II, para efetuar as comparações entre as soluções, considerando o aspecto multiobjetivo.

O algoritmo DIPSO possui duas características que se destacam: a modificação da variável g_{best} para armazenar um conjunto de partículas e a implementação de dois operadores de cruzamento. A utilização do conjunto visa aumentar o número de soluções para qual o algoritmo converge, já que um algoritmo evolutivo baseado em uma população finita tende a convergir para apenas uma solução [Mahfoud 1995] e uma ou mais soluções podem ser consideradas como resposta para um problema de otimização multiobjetivo. Os operadores de cruzamento têm objetivos distintos, um tem o objetivo de evoluir a população e o outro visa manter a diversidade populacional. Os passos do algoritmo são apresentados a seguir e detalhados nas próximas seções.

1. **Inicialização:** a população inicial é gerada aleatoriamente. Cada partícula é avaliada e sua respectiva variável p_{best} é inicializada. O conjunto F_1, que são as partículas não dominadas da população, é obtido, por meio do procedimento FNS, e atribuído ao conjunto g_{best}.
2. **Evolução:** toda partícula passa por um processo para alterar sua posição no espaço de busca. Esse processo utiliza os operadores genéticos para efetuar a troca de informação entre a partícula atual, a respectiva partícula p_{best} e uma outra selecionada do conjunto g_{best}.
3. **Seleção de p_{best} e g_{best}:** toda partícula tem sua nova posição avaliada. Cada melhor local (p_{best}) é atualizado verificando a partícula a qual é associado e o melhor global (g_{best}) é atualizado utilizando o conjunto F_1 e o g_{best} atual.
4. **Finalização:** o algoritmo é interrompido quando o número máximo de gerações é atingido. Caso contrário o algoritmo volta ao passo 2.

5.2 Representação da Partícula

[Cheng et al. 1996] apresentam e discutem sobre representações de um indivíduo para problemas de escalonamento *Job Shop* usando AG. Por exemplo, existem representações que são baseadas em operação, outras baseadas em *jobs* e assim por diante. Em [Gen et al. 1994], pode ser vista uma representação do primeiro tipo. Devido à possibilidade de recuperação imediata do escalonamento e a facilidade em calcular a aptidão do indivíduo, [Ling-li et al. 2009], [Xiao-hong et al. 2010] e [Wang et al. 2010] adotam essa representação. Outra característica importante dessa representação é que ela permite manter a viabilidade dos indivíduos obtidos após as operações de cruzamento e mutação.

Devido a essas características, a representação da partícula em DIPSO é conforme descrita em [Gen et al. 1994], sendo que um indivíduo (ou partícula) é identificado por dois vetores de tamanho N, X_{op} e X_{mach}, onde N é o número total de operações. O primeiro vetor contém as operações dos *jobs* de acordo com sua ordem de execução. Cada posição do vetor X_{op} é representada por um inteiro j, onde j refere-se ao *job* correspondente e $j = 1, \ldots n$. Assim, as n_j operações do j^{th} *job* são representadas por n_j posições em X_{op} com valor j. A primeira posição em X_{op} indica a primeira operação e a última indica a última operação do *job* j. O segundo vetor contém a máquina em que cada operação será executada. Assim, cada posição do vetor X_{mach} contém um valor k, $1 \leq k \leq m$, que indica a máquina em que a operação da posição correspondente em X_{op} é executada.

A Figura 5.1 mostra um exemplo dessa representação para um problema com $n = 4$ e $m = 5$, sendo que cada *job* contém duas operações. Observa-se que as operações do *job* $j = 1$ são representadas na primeira e terceira posições de X_{op} e ambas são executadas na máquina três, como pode ser visto em X_{mach}. Analogamente, observa-se que: as máquinas dois e quatro executam as operações do $job = 2$; as máquinas um e cinco executam as operações do $job = 3$; as máquinas um e dois executam as operações do $job = 4$.

X_{op}:	1	3	1	4	2	2	4	3
X_{mach}:	3	1	3	1	2	4	2	5

Figura 5.1: Representação da Partícula

Supondo, para esse exemplo, que para cada operação i, $i = 1, \ldots 8$, tenha tempo de processamento igual a uma unidade de tempo para toda máquina k, $k = 1, \ldots 5$, o escalonamento da Figura 5.1 é representado através do diagrama de Gantt na Figura 5.2.

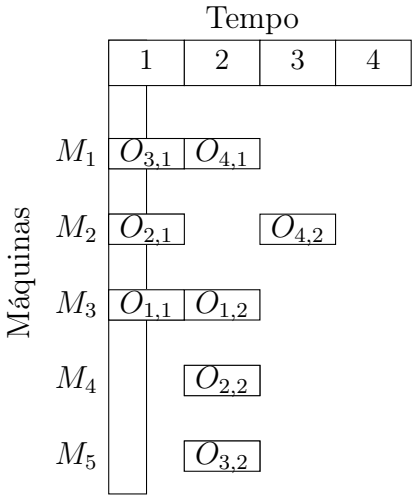

Figura 5.2: Diagrama de Gantt para o exemplo da Figura 5.1

5.3 Operadores Genéticos

Como discutido anteriormente sobre a rápida convergência do PSO, algumas propostas têm introduzido operadores genéticos para evitar mínimos locais. Por outro lado, operadores de cruzamento podem também acelerar a convergência. Nesse contexto, este trabalho propõe o uso de dois operadores de cruzamento, um responsável pela convergência e o outro responsável por diversificar a população.

O primeiro operador é aplicado sobre X_{op} e X_{mach} de maneira análoga, para que toda operação continue associada à mesma máquina após a operação. Assim, se a operação i do *job* j estava sendo executada pela máquina k, após a execução desse cruzamento, a operação i continuará a ser executada pela máquina k. Em contrapartida, o segundo operador, denominado DX e proposto neste trabalho, é aplicado diferentemente sobre cada vetor, o que permite a geração de uma maior diversidade, produzindo alterações na associação entre operações e máquinas. Essa característica, introduzida no DIPSO, objetiva manter a diversidade na população para controlar a convergência do PSO.

Na Figura 5.3, um exemplo do primeiro cruzamento é apresentado. Considere duas partículas P_1 e P_2 com $n = 4$ *jobs* e os respectivos vetores X_{op} e X_{mach}. Esse operador seleciona aleatoriamente $\lceil \frac{n}{2} \rceil$ *jobs*. Considere o conjunto J_1 contendo os *jobs* sorteados, e o conjunto J_2, os *jobs* restantes. No exemplo, $J_1 = \{2, 4\}$ e $J_2 = \{1, 3\}$. As operações dos *jobs* selecionados são posicionadas na partícula resultante, C_1, na mesma posição em que aparecem em P_1. Da mesma forma, as máquinas associadas com essas operações (vetor X_{mach}) são mantidas nas mesmas posições. As posições restantes de C_1 serão ocupadas pelas operações dos *jobs* de J_2, de acordo com a ordem de P_2.

O mesmo procedimento é efetuado para obter a partícula resultante C_2, a diferença é que os *jobs* não selecionados, J_2, são considerados primeiro. Assim, as operações dos *jobs* de J_2 são posicionadas em C_2, nas mesmas posições em que aparecem em P_2, e as posições restantes são ocupadas pelas operações dos *jobs* de J_1 na mesma ordem em que

aparecem em P_1. Seguindo a definição desse operador, as máquinas associadas com as operações que foram transferidas de C_2 para o vetor X_{op}, também são transferidas para o vetor X_{mach} de C_2. Esse operador de cruzamento é definido por [Gen et al. 1994] e é denominado *Improved Precedence Operation Crossover* (IPOX).

$$J_1 = \{2, 4\},\ J_2 = \{1, 3\}$$

P_1:	X_{op}:	1	3	1	4	2	2	4	3	P_2:	X_{op}:	3	4	2	4	3	2	1	1
	X_{mach}:	3	1	3	1	2	4	2	5		X_{mach}:	5	5	1	1	4	3	1	2
					↓	↓	↓	↓				↓				↓		↓	↓
C_1:	X_{op}:	3	3	1	4	2	2	4	1	C_2:	X_{op}:	3	4	2	2	3	4	1	1
	X_{mach}:	5	4	1	1	2	4	2	2		X_{mach}:	5	1	2	4	4	2	1	2

Figura 5.3: Operador de cruzamento IPOX

O segundo operador de cruzamento também é aplicado sobre ambos vetores, X_{op} e X_{mach}, entretanto ocorre em duas fases distintas, uma para cada vetor. A primeira fase ocorre no vetor das operações, isto é, em X_{op}. Sejam P_1 e P_2 duas partículas e um intervalo aleatoriamente selecionado entre a primeira e a última posições de P_1. A partícula resultante, C_1, recebe todos os valores que pertencem a esse intervalo nas mesmas posições de P_1 e as posições que não estão presentes no intervalo são preenchidas com as operações restantes, de acordo com a ordem em que aparecem em P_2. A segunda partícula resultante, C_2, é gerada analogamente, recebendo todos os valores não pertencentes ao intervalo selecionado, nas mesmas posições de P_1, e as posições restantes são preenchidas com as operações restantes de acordo com a ordem em que aparecem em P_2. Em relação ao vetor X_{mach}, na segunda fase o procedimento de cruzamento de um ponto é utilizado. Assim, uma posição aleatória é selecionada, formando os intervalos $[0, Z[$ e $[Z, 7]$, onde Z é a posição selecionada. Ocorre, então, a troca de informações entre os vetores, considerando os intervalos formados.

Esse procedimento em duas fases permite a geração de novas partículas com modificação nas máquinas que executarão a operação e, consequentemente, gera alterações nos tempos de início e fim na execução das operações aumentando a diversidade da população. As figuras 5.4 e 5.5 mostram as duas fases desse operador, denominado DX. A Figura 5.4 mostra um exemplo da primeira fase do DX com $n = 4$ *jobs*, sendo que cada um contém duas operações. Supondo que $[0, 3]$ seja o intervalo selecionado. As posições que pertencem a esse intervalo serão copiadas de P_1 para C_1. Nesse caso, uma operação de cada *job* foi adicionada em C_1 de acordo com a ordem $1, 3, 2$ e 4. A segunda operação de cada *job* é incluída em C_1 na ordem em que aparece em P_2. A partícula C_2 é obtida de maneira análoga, sendo o intervalo $[4, 7]$ copiado de P_1 para C_2 e as posições restantes (de C_2) são completadas com as operações restantes, de acordo com a ordem em que aparecem em P_2. A Figura 5.5 mostra um exemplo da segunda fase de DX, onde a posição 4 foi selecionada

e os intervalos de P_1 e P_2 formados entre a posição selecionada e a última são trocadas entre as partículas.

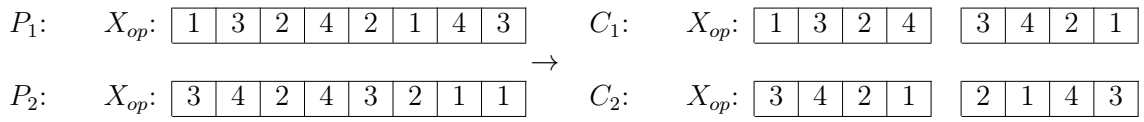

Figura 5.4: Operador de cruzamento: DX - Primeira fase

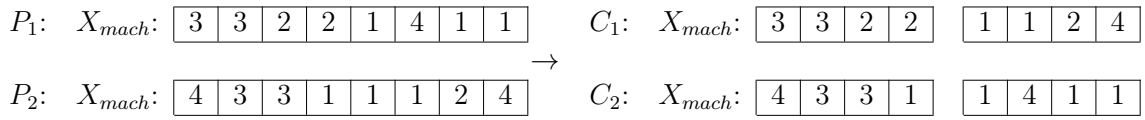

Figura 5.5: Operador de cruzamento: DX - Segunda fase

Assim como o operador de cruzamento DX, a mutação é realizada independentemente em X_{op} e X_{mach}. Em X_{op}, o operador de mutação seleciona duas posições aleatórias, pos_1 e pos_2, e executa $X_{op}[pos_1] = X_{op}[pos_2]$. Em seguida, para cada posição pos, tal que $pos_1 < pos <= pos_2$, executa-se $X_{op}[pos] = X_{op}[pos - 1]$. Em X_{mach}, os valores de duas posições aleatórias são modificados aleatoriamente, sendo que esses novos valores devem ser diferentes dos antigos e menores ou iguais a m. A Figura 5.6 mostra um exemplo desse operador.

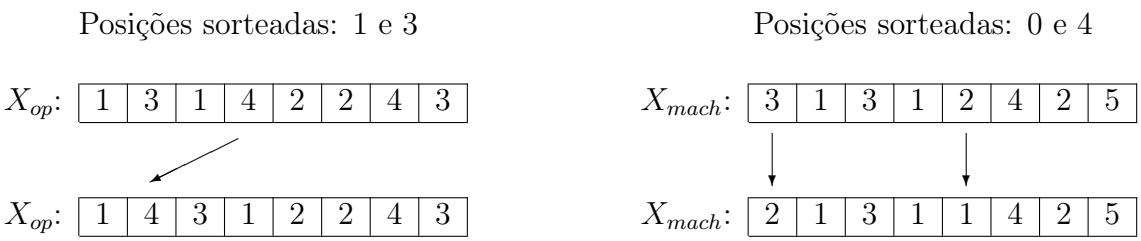

Figura 5.6: Mutação

É importante observar que todos os operadores adotados, especialmente aqueles aplicados sobre X_{op}, preservam o número e a ordem das operações do *job*. Dessa maneira, nenhum processo de correção de partículas é aplicado evitando, assim, o aumento no tempo de execução do algoritmo proposto.

5.4 Evolução da População

Para evoluir a população, o algoritmo DIPSO percorre todas partículas e modifica suas posições segundo uma fórmula que utiliza dois elementos chaves: p_{best} e g_{best}. O primeiro é a memória local de uma partícula, na qual é armazenada a melhor posição visitada pela

respectiva partícula. O segundo é a memória global da população. No DIPSO, ela é representada por um conjunto, que possui as melhores posições visitadas considerando toda população, o que difere do PSO, que possui apenas um elemento para representar a memória global. Essa alteração foi efetuada pois deseja-se tratar um problema multiobjetivo e, neste contexto, apontar uma única solução como melhor global não é coerente com os objetivos do algoritmo, ou seja, encontrar várias soluções não dominadas em vez de apenas uma. Assim, alterar g_{best} de uma partícula para um conjunto aumenta a possibilidade de convergência do algoritmo, visto que toda solução não dominada poderá influenciar a troca de posição de uma partícula.

A fórmula utilizada para realizar a mudança de posição de uma partícula é baseada naquela apresentada na Equação 4.1, sendo reescrita conforme Equação 5.1 devido a introdução da função *selectOne*. Conforme já apresentado, g_{best} é o conjunto das soluções não dominadas de cada iteração. Sendo assim, a função *selectOne* é para selecionar aleatoriamente um elemento de g_{best}. As demais funções e parâmetros da Equação 5.1 são conforme descritos na Equação 4.1

$$\begin{aligned} P_{k+1} &= selectBest(P_1, P_2, P_3), onde \\ P_1 &= cruzamento(P_k, selectOne(g_{best})), \\ P_2 &= cruzamento(P_k, p_{best}), \\ P_3 &= mutacao(P_k) \end{aligned} \tag{5.1}$$

Entretanto, como o algoritmo DIPSO utiliza dois operadores de cruzamento, IPOX e DX, realiza-se uma escolha probabilística para determinar qual operador utilizar na execução da fórmula. Para realizar essa escolha, são utilizadas as probabilidades p_{cc} e p_{dx}, onde $p_{cc} + p_{dx} = 100\%$. A primeira refere-se à probabilidade de executar o operador de cruzamento de IPOX (Figura 5.3) e a segunda refere-se à probabilidade de executar o operador de cruzamento DX (Figuras 5.4 e 5.5). Foi observado que, quanto maior o valor de p_{cc}, mais rapidamente o algoritmo converge e, quanto maior p_{dx}, maior a diversidade aplicada na população. A cada execução da Equação 5.1 é realizado o processo de escolha do operador de cruzamento, de modo que, a cada iteração do algoritmo, os dois operadores de cruzamento sejam utilizados segundo sua proporção, garantindo que parte da população seja direcionada para convergência e outra parte seja diversificada.

A cada mudança de posição de uma partícula é necessário atualizar p_{best} e a cada iteração do algoritmo, g_{best} precisa ser atualizado. Ambas atualizações utilizam as definições do ótimo de Pareto. Na atualização de p_{best}, é verificado se a nova posição da partícula domina p_{best} atual. Caso domine, p_{best} é atualizada com a nova posição da partícula em questão. Se p_{best} atual dominar a nova posição, nenhuma modificação é feita. Porém, se nenhuma dessas posições dominarem a outra, p_{best} é atualizada com a nova posição. Para

realizar a atualização de g_{best}, o procedimento FNS é utilizado a fim de obter o conjunto F_1, de soluções não dominadas, a partir de pop', onde, considerando pop como a população no momento da operação, $pop' = pop \cup g_{best}$. Esse procedimento aplicado à atualização de g_{best} é também utilizado após a inicialização da população para instanciar o primeiro conjunto g_{best}, porém como g_{best} estará vazio, tem-se $pop' = pop$.

Como já apresentado, DIPSO utiliza o procedimento FNS para toda comparação entre as soluções, exceto quando somente duas soluções são comparadas. Devido ao grande esforço computacional para a execução do procedimento FNS, o algoritmo não aplica o procedimento sobre todo indivíduo. Como é típico de populações de algoritmos evolutivos, no decorrer do processamento, a população converge para algumas soluções gerando indivíduos repetidos. Além disso, no caso do FJSP, pode ocorrer de indivíduos diferentes (escalonamentos diferentes) terem a mesma aptidão. O FNS efetua comparações a fim de determinar os conjuntos ordenados de não dominância (*rank* de Pareto) e todo seu processamento observa somente a aptidão do indivíduo. Assim, não é necessário avaliar toda a população, no caso da atualização do g_{best}, que é o momento em que se recorre ao FNS com maior número de indivíduos a serem analisados. Dessa maneira, antes de aplicar o FNS, é feita a seleção de elementos, de modo a selecionar uma partícula que represente cada aptidão. Após esse processo, a informação do *rank* obtida pelas partículas é repassada para toda a população.

5.5 O Algoritmo DIPSO

No Algoritmo 7, apresentamos o DIPSO. Nesse pseudocódigo temos o resumo do que foi descrito nas seções anteriores.

- Inicialização aleatória da população inicial, com a avaliação das partículas e determinação de p_{best} e g_{best};
- A mudança de posição das partículas através da função *selectBest*, que utiliza os operadores genéticos, p_{best} e g_{best};
- Possibilidade de aplicar-se a mutação, segundo uma probabilidade;
- Atualização de p_{best} e g_{best}, sendo que para a g_{best} utiliza-se o procedimento FNS;
- A evolução da população, que é formada por esses três últimos itens.

Algoritmo 7: Algoritmo DIPSO

```
1  início
2      inicializar população pop com tamPop partículas;
3      avaliar as partículas de pop;
4      executar o procedimento FNS, obtendo F_1;
5      determinar p_best de cada partícula como a própria partícula;
6      determinar g_best como F_1;
7      while não atingir critério de parada do
8          for cada partícula P_k de pop do
9              cruzamento = selecionar operador de cruzamento, segundo
               probabilidades p_cc e p_dx;
10             P_1 = cruzamento(P_k, selectOne(g_best));
11             P_2 = cruzamento(P_k, p_best);
12             P_3 = mutacao(P_k);
13             P_{k+1} = selectBest(P_1, P_2, P_3);
14             aplicar mutação, segundo probabilidade;
15             avaliar partícula;
16             atualizar p_best da partícula;
17             pop' = pop ∪ g_best;
18             atualizar g_best por meio do processo FNS, utilizando pop';
19         end
20     end
21 fim
```

5.6 Experimentos e Resultados

Os experimentos foram projetados para observar as principais características de DIPSO e para comparar seus resultados com os de outros algoritmos presentes na literatura. Nesses experimentos, o FJSP multiobjetivo foi tratado considerando o *makespan*, a carga de trabalho total e a carga de trabalho máxima, assim como outros trabalhos que trataram esse problema, [qing Li et al. 2010], [Wang et al. 2010] e [Zhang et al. 2009]. Todos experimentos foram realizados em um computador com processador Intel Core I7 (2GHz) com 8 GB de memória RAM.

5.6.1 Comparação das principais características de DIPSO

O algoritmo DIPSO possui duas características importantes: 1) utilização de dois operadores de cruzamento distintos (um caracterizado por convergir a população e o outro por manter ou elevar a diversidade presente na população); e 2) o uso de um conjunto

para representar g_{best}. Para mostrar o comportamento do algoritmo com e sem essas características, quatro experimentos são apresentados nas próximas seções, sendo que, em cada um aplica-se combinações diferentes dessas características.

Os problemas utilizados nesses experimentos foram retirados de [Kacem et al. 2002c]. Os experimentos utilizaram os problemas 4×5, 8×8, 10×7 and 10×10. Cada experimento foi executado trinta vezes e os resultados apresentados são o conjunto F_1 em que o algoritmo convergiu.

Além dos resultados, características da evolução do algoritmo são analisadas, tais como diversidade e convergência. A taxa de convergência apresentada nos experimentos consideram a soma do número de elementos repetidos no resultado final. Os parâmetros utilizados nesses experimentos são apresentados na Tabela 5.1. Os parâmetros p_{cc} e p_{dx} são considerados apenas quando os experimentos utilizam as dois operadores de cruzamento simultaneamente. Múltiplas execuções foram realizadas variando os parâmetros para verificar o comportamento do algoritmo. Assim, baseado nos resultados, os valores dos parâmetros foram determinados. Valores altos para o tamanho da população, número de iterações e mutação, são para evidenciar as diferenças das características que são analisadas. Ao comparar DIPSO com outros algoritmos, parâmetros menores serão apresentados.

Tabela 5.1: Parâmetros para comparação das características de DIPSO

p_{cc}	p_{dx}	Mutação	Tamanho da população	Número de gerações
80%	20%	23%	1000	1400

Primeiro Experimento

Para avaliar as principais características de DIPSO, esse experimento não levou em consideração essas características, isto é, utilização de dois operadores de cruzamento e o uso de um conjunto para representar g_{best}. O algoritmo foi executado com os dois operadores de cruzamento, um de cada vez e somente uma partícula foi utilizada em g_{best} por geração. Assim, g_{best} não armazenou F_1 para evoluir a população. Para determinar a partícula, o método *selectOne* é usado para escolher um elemento de F_1. A Tabela 5.2 mostra os resultados usando somente o cruzamento IPOX (Figure 5.3).

Tabela 5.2: Resultados: primeiro experimento

	4×5			8×8		10×7		10×10	
M	11	12	13	16	15	12	11	8	8
WT	32	32	33	73	75	60	61	41	42
W	10	8	7	13	12	12	11	7	6

Os resultados na Tabela 5.2 são similares àqueles obtidos por [Zhang et al. 2009], [Xia e Wu 2005], [Wang et al. 2010] e [Ho e Tay 2007], apresentados nas Tabelas 5.3, 5.4, 5.5 e 5.6. Porém, os resultados desses algoritmos da literatura mostram que existem mais soluções possíveis de serem encontradas. Em relação ao comportamento do algoritmo nesse experimento, pode-se destacar que 95% das execuções do algoritmo resultaram na convergência em apenas uma solução e a taxa de convergência de todas execuções ficou entre 30% e 40%.

Tabela 5.3: Resultados da literatura - problema 4×5

	PSO + TS [Zhang et al. 2009]	MOGA [Wang et al. 2010]			SEA [Chiang e Lin 2013]			
M	11	11	11	12	11	11	12	13
WT	32	32	34	32	32	34	32	33
W	10	10	9	8	10	9	8	7

Tabela 5.4: Resultados da literatura - problema 8×8

	PSO + TS [Zhang et al. 2009]		PSO + SA [Xia e Wu 2005]		MOGA [Wang et al. 2010]			MOEA-GLS [Ho e Tay 2007]			
M	14	15	16	15	15	15	16	16	15	14	16
WT	77	75	73	75	81	75	73	73	75	77	77
W	12	12	13	12	11	12	13	13	12	12	11

Tabela 5.5: Resultados da literatura - problema 10×7

	TS + VNS [qing Li et al. 2010]		SEA [Chiang e Lin 2013]		
M	11	11	12	11	11
WT	61	62	60	61	62
W	11	10	12	11	10

Tabela 5.6: Resultados da literatura - problema 10×10

	PSO + TS [Zhang et al. 2009]	PSO + SA [Xia e Wu 2005]	MOGA [Wang et al. 2010]				MOEA-GLS [Ho e Tay 2007]			
M	7	7	8	8	7	7	8	8	7	7
WT	43	44	41	42	45	42	41	42	43	42
W	6	6	7	5	5	6	7	5	5	6

Utilizando o operador de cruzamento DX (figuras 5.4 e 5.5), os resultados não se mostraram adequados, sendo que a taxa de convergência é baixa, ficando entre 2% a 8%. Porém, na maioria dos experimentos a taxa foi menor do que 5%. No caso dos problemas 8×8 e 10×10, foi observado que o número de soluções visitadas foi 50%

maior do que quando o algoritmo é executado com o operador de cruzamento IPOX. Isso mostra a diversidade que é gerada com a utilização de DX e também justifica a taxa de convergência do algoritmo. A única instância em que os resultados foram semelhantes aos do IPOX foi o problema 4×5, que é mais simples. Assim, os resultados considerados para comparação, com os outros experimentos, são aqueles obtidos com o IPOX.

Segundo Experimento

A avaliação do primeiro experimento considerou somente um operador de cruzamento. No segundo experimento, ambos os operadores de cruzamento são considerados no algoritmo, de acordo com $p_{cc} = 80\%$ (probabilidade de IPOX) e $p_{dx} = 20\%$ (probabilidade de DX). A variável g_{best} continua armazenando uma única partícula. A Tabela 5.7 apresenta os resultados deste experimento. As soluções encontradas são similares àquelas mostradas no experimento anterior, porém, com o aumento do número de soluções do problema 10×10, pode-se concluir que a diversidade aumentou.

Tabela 5.7: Resultados: segundo experimento

	4×5			8×8		10×7		10×10		
M	11	12	13	16	15	12	11	8	8	7
WT	32	32	33	73	75	60	61	41	42	42
W	10	8	7	13	12	12	11	7	5	6

A introdução do cruzamento de diversidade (DX), junto ao IPOX, aumentou o número de soluções diferentes visitadas em uma média de 21%, quando comparado com o primeiro experimento, que adotou somente IPOX. Devido a esse aumento na diversidade, a habilidade de evitar mínimos locais também aumentou. Em relação à convergência, destaca-se que em 70% das execuções o algoritmo convergiu durante as primeiras 250 gerações e, em relação ao problema 4×5, 80% das execuções convergiram até a geração de número 25.

Terceiro Experimento

Neste experimento, foi considerado somente o operador de cruzamento IPOX e g_{best} foi tratado como um conjunto. Os resultados deste experimento estão mostrados na Tabela 5.8, em que similaridades com os outros experimentos podem ser observadas. Entretanto, quando comparado com o segundo experimento, três novas soluções foram identificadas. Além disso, o algoritmo apresenta convergência para mais de uma solução, em uma única execução, diferente dos primeiros experimentos. Em particular, as execuções do algoritmo com o problema 4×5, em 80% das execuções, apresentou convergência para as quatro soluções apresentadas na Tabela 5.8.

Tabela 5.8: Resultados: terceiro experimento

	4×5				8×8			10×7			10×10		
M	11	12	13	11	16	15	14	12	11	11	8	8	7
WT	32	32	33	34	73	75	77	60	61	62	41	42	42
W	10	8	7	9	13	12	12	12	11	10	7	5	6

Quarto Experimento: DIPSO

Nos experimentos anteriores, cada característica do algoritmo DIPSO foi avaliada separadamente. Neste quarto experimento, ambas as características foram consideradas, mostrando que essas particularidades do algoritmo introduzem mais diversidade na evolução da população. Um número maior de soluções, não dominadas pelas soluções encontradas na literatura, são apresentadas como pode ser visto na Tabela 5.9. O número de partículas diferentes, que são geradas durante a evolução do algoritmo, se manteve semelhante ao do segundo experimento.

Tabela 5.9: Resultados do quarto experimento: DIPSO

	4×5				8×8				10×7			10×10			
M	11	12	13	11	16	15	14	16	12	11	11	8	8	7	7
WT	32	32	33	34	73	75	77	77	60	61	62	41	42	42	43
W	10	8	7	9	13	12	12	11	12	11	10	7	5	6	5

Em 78% das execuções, a convergência do algoritmo aconteceu antes das primeiras 250 gerações e em todas as execuções, o algoritmo convergiu para mais que uma solução não dominada pela literatura. Em particular, toda execução do algoritmo para o problema 4×5 convergiu para todas soluções da Tabela 5.9. Além disso, em todas execuções desse problema, na geração de número 25, pelo menos três soluções, das quatro apresentadas, já estavam no conjunto de convergência do algoritmo. Em relação aos problemas 8×8 e 10×10, 68% das execuções convergiram antes da geração de número 250.

Como citado, o algoritmo DIPSO converge, em uma mesma execução, para mais de uma solução. Essa é uma característica a ser destacada, pois um algoritmo evolutivo baseado em uma população finita tende a convergir para apenas uma solução [Mahfoud 1995].

Assim, observa-se que os resultados obtidos por DIPSO são semelhantes aos da literatura, visto nas Tabelas 5.3, 5.4, 5.5 e 5.6. Na próxima seção, os resultados de DIPSO são comparados com outros algoritmos da literatura e uma análise da comparação é apresentada.

5.6.2 Análise e Comparação de Resultados

Nesta seção, uma análise dos resultados encontrados pelo algoritmo DIPSO é apresentada. Nessa análise, comparou-se o DIPSO com outros algoritmos da literatura, baseando-se na análise realizada no trabalho de [Chiang e Lin 2013]. Para representar uma solução, utilizou-se uma representação de 3-upla (X, Y, Z), onde X é o valor do *makespan*, Y da carga de trabalho total e Z da carga de trabalho máxima.

Problemas e algoritmos

Para realizar os experimentos com o algoritmo DIPSO, foram utilizados os *benchmarks* de [Kacem et al. 2002c] (KC) e de [Brandimarte 1993] (BM), que juntos formam um conjunto de quinze problemas. Os problemas de [Kacem et al. 2002c] são denominados por 4×5, 8×8, 10×7, 10×10 e 15×10, que fazem uma analogia à quantidade de *jobs* (n) e de máquinas (m), $n \times m$. O conjunto de problemas de [Brandimarte 1993] são denominados por Mk1, Mk2, ..., Mk10. Estes são problemas que variam na quantidade de *jobs* e máquinas e vão desde 10×6 (Mk1) a 20×15 (Mk10).

Tabela 5.10: Algoritmos

Algoritmo	Aptidão	Número de parâmetros	Soluções visitadas	Número de execuções
TS + VNS [qing Li et al. 2010]	Soma ponderada	10	-	20
MOGA [Wang et al. 2010]	Dominância	8	200×200	10
PSO + TS [Zhang et al. 2009]	Soma ponderada	10	100×50	-
PSO + SA [Xia e Wu 2005]	Soma ponderada	12	100×50	-
SEA [Chiang e Lin 2013]	Dominância	2	200×200	10
DIPSO	Dominância	5	-	30

Para verificar a qualidade dos resultados, foram considerados vários trabalhos da literatura, com diversas técnicas para resolução. A Tabela 5.10 apresenta esses trabalhos, mostrando algumas de suas características para melhor classificá-los. Na coluna "Algoritmo", destacamos o nome do algoritmo e a referência do trabalho que o apresentou; em "Aptidão", a forma como calcula-se a aptidão de um indivíduo é apresentada; a coluna "Número de Parâmetros" mostra a quantidade de parâmetros que são necessários para executar o algoritmo; em "Soluções Visitadas" está a quantidade de soluções visitadas pelo algoritmo em uma execução; e em "Número de Execuções" é mostrada a quantidade de execuções efetuadas nos experimentos. Nos dados da coluna "Soluções Visitadas" não são consideradas as soluções visitadas por técnicas de busca locais.

Na última linha da tabela, os dados referente ao algoritmo DIPSO são apresentados. Em relação ao número de soluções visitadas, para cada conjunto de problemas foram utilizados parâmetros diferentes, o que torna essa informação dependente do problema.

Parametrização

A Tabela 5.11 apresenta os parâmetros utilizados para a execução dos problemas. Adotou-se parâmetros diferentes para cada conjunto de problemas ([Kacem et al. 2002c] e [Brandimarte 1993]). Essa distinção é devido ao fato que os dois conjuntos de problemas possuem características específicas. Os problemas de [Kacem et al. 2002c] são T-FJSP (exceto o 8×8) e cada *job*, em todas as instâncias de problemas, possui no máximo quatro operações. Já os problemas de [Brandimarte 1993] são todos P-FJSP e os *jobs*, considerando todas as instâncias de problemas, possuem entre três e quinze operações. Essa distinção justifica a diferenciação no tamanho da população e no número de gerações. A diferença no parâmetro da mutação é justificada pelo fato dos problemas de [Brandimarte 1993] possuírem uma quantidade pequena de máquinas que executam as operações dos *jobs*. Como o operador de mutação, que é aplicado em X_{mach}, depende dessa característica, uma porcentagem alta não obtém a diversidade esperada, já que não há um grande número de máquinas para efetuar as duas trocas aleatórias de posições no vetor, definidas no operador de mutação.

Tabela 5.11: Parâmetros - DIPSO

Conjunto de problemas	p_{cc}	p_{dx}	Mutação	Tamanho da população	Número de gerações
[Kacem et al. 2002c]	80%	20%	5%	400	800
[Brandimarte 1993]	70%	30%	1%	2000	1400

Análise de desempenho

Dos quinze problemas analisados, em apenas dois o algoritmo não chegou em soluções ótimas (conjunto de soluções encontradas na literatura e não dominadas por nenhuma outra), em quatro o algoritmo encontrou todas as soluções (4×5, 10×7, 10×10 e Mk1), em três o DIPSO encontrou novas soluções (Mk1, Mk2 e Mk10) e em dois problemas soluções anteriores foram dominadas por novas (Mk1 e Mk2).

As Tabelas 5.12, 5.13 e 5.14 mostram as novas soluções encontradas nos problemas Mk1, Mk2 e Mk10. Na Tabela 5.12, analisamos as soluções encontradas no problema Mk1 comparando as soluções de DIPSO com o conjunto SNDL (Soluções não Dominadas da Literatura), que é formado pelas soluções não dominadas da literatura. Esse conjunto foi obtido a partir do trabalho de [Chiang e Lin 2013], que apresentaram uma lista de soluções não dominadas para cada problema de [Kacem et al. 2002c] e [Brandimarte 1993], através de uma análise que considera oito trabalhos.

Na Tabela 5.12, as soluções de SNDL que estão em negrito indicam que as mesmas foram dominadas por alguma solução do conjunto de soluções de DIPSO. No conjunto de soluções de DIPSO, as soluções em negrito indicam que são novas soluções. Assim a

solução (42, 156, 40) de DIPSO domina (42, 157, 40) de SNDL e a solução (43, 154, 40) de DIPSO domina as soluções (43, 155, 40) e (44, 154, 40) de SNDL. Na Tabela 5.13 mostramos as soluções de Mk2. As soluções que estão em negrito nessa tabela possuem o mesmo significado que na Tabela 5.12. Assim a solução (28, 150, 26) de DIPSO domina (29, 150, 26) de SNDL. Na Tabela 5.14, algumas soluções de Mk10 são analisadas e, analogamente às Tabelas 5.12 e 5.13, as soluções que estão em negrito no conjunto de soluções de DIPSO são novas, mas não dominam nenhuma outra solução de SNDL, ou seja, aumentam o número de soluções ótimas. A quantidade de soluções que o problema Mk10 possui é muito grande que, incluindo as soluções novas de DIPSO, são 146, o que inviabiliza apresentar uma tabela com todas.

Tabela 5.12: Principais resultados de DIPSO - Mk1

Mk1					
SNDL			DIPSO		
M	WT	W	M	WT	W
40	162	38	40	162	38
40	164	37	40	164	37
40	167	36	40	167	36
41	160	38	41	160	38
41	163	37	41	163	37
42	**157**	**40**	**42**	**156**	**40**
42	158	39	42	158	39
42	165	36	42	165	36
43	**155**	**40**	**43**	**154**	**40**
44	**154**	**40**	45	153	42
45	153	42			

Tabela 5.13: Principais resultados de DIPSO - Mk2

Mk2					
SNDL			DIPSO		
M	WT	W	M	WT	W
26	151	26	31	141	31
27	145	27	**28**	**150**	**26**
28	144	28	30	142	30
29	143	29	29	143	29
29	**150**	**26**	33	140	33
30	142	30	28	144	28
31	141	31	27	146	27
33	140	33	28	145	27

As Tabelas 5.15, 5.16, 5.17 e 5.18 são formadas por dados quantitativos e qualitativos, coletados do conjunto de soluções de DIPSO. Quantitativos, pois possuem uma contagem

Tabela 5.14: Principais resultados de DIPSO - Mk10

Results - Mk10					
SNDL			DIPSO		
M	WT	W	M	WT	W
214	2053	210	**303**	**1915**	**204**
214	2082	204	**279**	**1883**	**212**
217	2064	207	**270**	**1877**	**215**
224	1980	219	**281**	**1887**	**211**
225	1930	209	262	1877	216
	...			...	
263	1942	199	252	1903	212
264	1853	236	263	1866	224
270	1849	270	246	1870	225
280	1848	280	272	1900	209
290	1847	290	253	1871	220

de soluções e, qualitativos, pois as melhores soluções de cada objetivo foram catalogadas. Assim, na Tabela 5.15 apresentamos um comparativo de DIPSO com outros trabalhos da literatura. NSND (Novas Soluções não Dominadas) corresponde ao conjunto de soluções não dominadas que considera as soluções de SNDL e as soluções de DIPSO. Ou seja, em NSND, as novas soluções apresentadas por DIPSO estão contempladas e as soluções que estas dominam são desconsideradas. A coluna "Total de Soluções não Dominadas" mostra o número de soluções não dominadas que determinado problema possui, considerando o conjunto NSND. As demais colunas são reservadas para os dados das soluções do DIPSO e dos outros algoritmos, onde cada coluna representa um trabalho ou algoritmo específico.

Para cada problema e cada algoritmo, é apresentada a relação $\frac{d}{s}$, em que s é o número de soluções encontradas pelo algoritmo específico e d é o número de soluções encontradas pelo algoritmo que participam de NSND. Observando os dados apresentados na Tabela 5.15, verifica-se que todos os trabalhos conseguiram bons resultados com o conjunto de problemas de [Kacem et al. 2002c], porém com os problemas de [Brandimarte 1993] destacamos os algoritmos SEA [Chiang e Lin 2013] e o DIPSO, pois foram os trabalhos que conseguiram apresentar o maior número de soluções.

Nas Tabelas 5.16, 5.17 e 5.18 são mostrados os melhores resultados de todos os algoritmos para cada problema, em cada objetivo. Na primeira coluna, é mostrado o melhor valor encontrado para o respectivo problema no determinado objetivo. Nas outras colunas, os melhores resultados encontrados pelos algoritmos relacionados para o objetivo e problema específicos são exibidos. Na última linha, é apresentada a quantidade de melhores resultados encontrados por cada algoritmo.

Analisando os dados das Tabelas 5.16, 5.17 e 5.18, observamos que entre os algoritmos que utilizaram PSO, DIPSO possui o melhor desempenho. Em todos os problemas de

Tabela 5.15: Número de soluções não dominadas

Problema	Total de soluções não dominadas	PSO+TS	PSO+SA	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	4	1/1	-	2/2	3/3	4/4	4/4
KC 8×8	5	2/2	2/2	2/2	3/3	4/4	4/4
KC 10×7	3	-	-	2/2	-	3/3	3/3
KC 10×10	4	0/1	0/1	3/3	4/4	4/4	4/4
KC 15×10	2	0/1	0/1	2/2	1/3	2/2	0/2
BM Mk1	10	-	-	1/1	1/4	8/11	10/10
BM Mk2	8	-	-	1/1	4/6	6/9	6/8
BM Mk3	24	-	-	0/1	7/10	17/17	16/18
BM Mk4	28	-	-	1/1	6/10	20/24	10/23
BM Mk5	11	-	-	1/1	5/5	10/10	9/11
BM Mk6	110	-	-	0/1	7/10	103/110	0/78
BM Mk7	15	-	-	0/1	7/7	13/16	12/19
BM Mk8	10	-	-	0/1	5/5	8/9	8/9
BM Mk9	39	-	-	1/1	4/9	64/64	1/81
BM Mk10	146	-	-	1/1	3/18	138/138	4/59

Tabela 5.16: *Makespan* - melhores soluções

Problema	Melhor valor encontrado	PSO+TS	PSO+SA	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	11	11	-	11	11	11	11
KC 8×8	14	14	15	14	15	14	14
KC 10×7	11	-	-	11	-	11	11
KC 10×10	7	7	7	7	7	7	7
KC 15×10	11	11	12	11	11	11	12
BM Mk1	40	-	-	40	40	40	40
BM Mk2	26	-	-	26	26	26	27
BM Mk3	204	-	-	204	204	204	204
BM Mk4	60	-	-	61	60	61	66
BM Mk5	172	-	-	172	173	173	173
BM Mk6	60	-	-	65	60	65	75
BM Mk7	139	-	-	140	139	140	143
BM Mk8	523	-	-	523	523	523	523
BM Mk9	310	-	-	310	310	311	336
BM Mk10	214	-	-	214	214	225	245
Total	-	4	1	12	12	9	7

[Kacem et al. 2002c][2], DIPSO só não encontrou o melhor resultado em um objetivo de um problema (*makespan* do 15×10), enquanto os demais (PSO + TS [Zhang et al. 2009] e PSO + SA [Xia e Wu 2005]) não conseguiram o melhor valor em mais de uma oportunidade. Considerando todos os trabalhos, observa-se que DIPSO e MOGA [Wang

[2]Consideramos apenas os problemas de [Kacem et al. 2002c], pois os outros trabalhos que utilizaram PSO não consideraram os problemas de [Brandimarte 1993] em seus experimentos.

Tabela 5.17: Carga de trabalho total - melhores soluções

Problema	Melhor valor encontrado	PSO+TS	PSO+SA	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	32	32	-	32	32	32	32
KC 8×8	73	75	73	75	73	73	73
KC 10×7	60	-	-	61	-	60	60
KC 10×10	41	43	44	42	41	41	41
KC 15×10	91	93	91	91	91	91	91
BM Mk1	153	-	-	167	154	153	153
BM Mk2	140	-	-	151	140	140	140
BM Mk3	812	-	-	852	847	812	812
BM Mk4	324	-	-	366	331	324	324
BM Mk5	672	-	-	687	676	672	672
BM Mk6	330	-	-	398	330	331	333
BM Mk7	649	-	-	695	657	649	649
BM Mk8	2484	-	-	2524	2484	2484	2484
BM Mk9	2210	-	-	2294	2259	2210	2211
BM Mk10	1847	-	-	2053	1854	1847	1850
Total	-	1	2	2	7	14	12

Tabela 5.18: Carga de trabalho máxima - melhores soluções

Problema	Melhor valor encontrado	PSO+TS	PSO+SA	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	7	10	-	8	8	7	7
KC 8×8	11	12	12	12	11	11	11
KC 10×7	10	-	-	10	-	10	10
KC 10×10	5	6	6	5	5	5	5
KC 15×10	10	11	11	10	10	10	10
BM Mk1	36	-	-	36	36	36	36
BM Mk2	26	-	-	26	26	26	26
BM Mk3	133	-	-	204	133	204	204
BM Mk4	37	-	-	61	54	60	60
BM Mk5	172	-	-	172	173	173	172
BM Mk6	50	-	-	62	54	50	54
BM Mk7	138	-	-	140	138	143	141
BM Mk8	497	-	-	523	497	524	523
BM Mk9	299	-	-	301	299	299	299
BM Mk10	196	-	-	210	204	196	204
Total	-	0	0	6	9	10	9

et al. 2010] são equivalentes e que SEA [Chiang e Lin 2013] possui melhores resultados do que DIPSO e MOGA [Wang et al. 2010], porém, apesar de seus resultados serem melhores, eles são próximos aos de DIPSO e MOGA [Wang et al. 2010]. Além disso, considerando apenas o *makespan*, o trabalho de [qing Li et al. 2010] (TS + VNS) possui bons resultados, apesar de não se destacar na Tabela 5.15.

Discussão - desempenho multiobjetivo

Os trabalhos analisados possuem diversas técnicas associadas: PSO, GA, buscas locais, aleatoriedade, conhecimento específico do problema, entre outras. Na Tabela 5.15, os algoritmos que mais se destacam, apresentando o maior número de soluções não dominadas, são o DIPSO e o SEA [Chiang e Lin 2013]. Esses algoritmos possuem características distintas quanto ao tratamento do problema. O SEA [Chiang e Lin 2013] é um algoritmo evolutivo que utiliza conhecimento de caraterísticas do problema para gerar a população inicial e para realizar a operação de mutação. Esse fato auxilia a busca pelas soluções em todos os problemas, principalmente no conjunto de [Brandimarte 1993].

SEA [Chiang e Lin 2013] é um algoritmo simples de ser usado, pois possui apenas dois parâmetros a serem determinados: o tamanho da população e o número máximo de gerações. Entretanto essa simplicidade não se destaca pela dificuldade apresentada na implementação, que requer conhecimento específico do problema. Essa característica também aumenta a dificuldade em encapsular o algoritmo, pois retira a possibilidade de aplicação direta em outros problemas, sendo necessária a implementação de métodos eficientes interpretando o problema em específico para substituição e adequação do algoritmo, assim como aqueles utilizados para o FJSP.

Em contrapartida, DIPSO é um algoritmo que não utiliza qualquer tipo de conhecimento específico do problema em nenhuma operação, não utiliza métodos de busca local e a avaliação das possíveis soluções são realizadas com as definições do ótimo de Pareto, aplicando os conceitos de dominância. Os operadores de cruzamento e mutação presentes no DIPSO são aleatórios, seguindo apenas os algoritmos pré-estabelecidos. Mesmo sem observar dados do problema, DIPSO conseguiu obter uma grande quantidade de soluções e, em alguns casos, melhorar soluções da literatura, que foram apresentadas pelo próprio SEA [Chiang e Lin 2013]. Tal fato mostra que sua característica de alta diversidade permite apresentar um grande número de soluções e com qualidade, superando o fato de outros algoritmos utilizarem buscas locais e conhecimento específico do problema. Assim, apesar de não possuir a maior quantidade de soluções não dominadas, é capaz de explorar pontos do espaço de busca que não tinham sido visitados antes.

Considerando que DIPSO é um PSO híbrido, assim como PSO + TS [Zhang et al. 2009] e PSO + SA [Xia e Wu 2005], podemos comparar a execução desses algoritmos. PSO + TS [Zhang et al. 2009] e PSO + SA [Xia e Wu 2005] realizaram experimentos somente com os problemas de [Kacem et al. 2002c]. Considerando somente esse conjunto de problemas, DIPSO apresentou como solução quinze das dezoito soluções não dominadas da literatura, o que representa 83% de alcance do conjunto de soluções não dominadas conhecidas.

Tanto PSO + TS [Zhang et al. 2009] quanto PSO + SA [Xia e Wu 2005] não executaram todos os problemas de [Kacem et al. 2002c] e, para os problemas que executaram,

não conseguiram alcançar uma grande variedade de resultados. Considerando somente os problemas executados para cada algoritmo, PSO + TS [Zhang et al. 2009] conseguiram alcançar 20% (3 de 15) das soluções não dominadas e PSO + SA [Xia e Wu 2005], 18% (2 de 11). Tais resultados mostram que as alterações que foram aplicadas no PSO para construir o DIPSO trouxeram maior capacidade de varredura do espaço de busca.

Discussão - desempenho mono-objetivo

Apesar deste trabalho tratar o problema com característica de ser multiobjetivo, é importante avaliar o comportamento das soluções de cada objetivo individualmente. As Tabelas 5.19, 5.20 e 5.21 fazem a contagem do número soluções não dominadas apresentadas pelos algoritmos TS + VNS [qing Li et al. 2010], MOGA [Wang et al. 2010], SEA [Chiang e Lin 2013] e DIPSO, porém considerando apenas aquelas em que o melhor valor de cada objetivo foi encontrado. Na coluna "Melhor valor encontrado", é indicado o melhor resultado conhecido nos trabalhos analisados para o determinado objetivo e respectivo problema. A coluna "Número de soluções com o melhor valor" mostra o número de soluções não dominadas que são conhecidas na literatura, que possuem o melhor valor. As demais colunas apresentam a contagem do número de soluções não dominadas, que possuem o melhor valor, que foram encontradas pelo respectivo algoritmo. Na última linha da tabela é mostrado o somatório das colunas de contagem de melhores resultados e sua respectiva porcentagem. Assim, na última linha da coluna "Número de soluções com o melhor valor", é apresentado a quantidade total de soluções com o melhor valor encontrado.

Tabela 5.19: Número de soluções para o melhor *makespan*

Problema	Melhor valor encontrado	Número de soluções com o melhor valor	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	11	2	1	2	2	2
KC 8×8	14	1	1	0	1	1
KC 10×7	11	2	2	-	2	2
KC 10×10	7	2	2	2	2	2
KC 15×10	11	2	2	1	2	0
BM Mk1	40	3	1	0	3	3
BM Mk2	26	1	1	1	0	0
BM Mk3	204	5	0	4	1	0
BM Mk4	60	1	0	1	0	0
BM Mk5	172	1	1	0	0	0
BM Mk6	60	1	0	1	0	0
BM Mk7	139	1	0	1	0	0
BM Mk8	523	2	0	2	0	0
BM Mk9	310	2	1	1	0	0
BM Mk10	214	2	1	1	0	0
Total	-	28	13 (46.4%)	17 (60.7%)	13 (46.4%)	10 (35.71%)

Analisando as Tabelas 5.19, 5.20 e 5.21, observa-se que o algoritmo SEA [Chiang e Lin 2013] possui seu melhor desempenho nos valores relacionados com carga de trabalho

Tabela 5.20: Número de soluções para a melhor carga de trabalho total

Problema	Melhor valor encontrado	Número de soluções com o melhor valor	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	32	2	2	2	2	2
KC 8×8	73	1	0	1	1	1
KC 10×7	60	1	0	0	1	1
KC 10×10	41	1	0	1	1	1
KC 15×10	91	1	1	1	1	1
BM Mk1	153	1	0	0	1	1
BM Mk2	140	1	0	1	1	1
BM Mk3	812	1	0	0	1	1
BM Mk4	324	1	0	0	1	1
BM Mk5	672	1	0	0	1	1
BM Mk6	330	1	0	1	0	0
BM Mk7	649	1	0	0	1	1
BM Mk8	2484	1	0	1	1	1
BM Mk9	2210	1	0	0	1	0
BM Mk10	1847	1	0	0	1	0
Total	-	16	3 (18.75%)	8 (50%)	15 (93.7%)	13 (81.2%)

Tabela 5.21: Número de soluções para a melhor carga de trabalho máxima

Problema	Melhor valor encontrado	Número de soluções com o melhor valor	TS+VNS	MOGA	SEA	DIPSO
KC 4×5	7	1	0	0	1	1
KC 8×8	11	2	0	1	1	1
KC 10×7	10	1	1	0	1	1
KC 10×10	5	2	2	2	2	2
KC 15×10	10	1	1	0	1	1
BM Mk1	36	2	1	0	2	2
BM Mk2	26	2	1	1	0	1
BM Mk3	133	1	0	1	0	0
BM Mk4	37	1	0	0	0	0
BM Mk5	172	1	1	0	0	0
BM Mk6	50	3	0	0	3	0
BM Mk7	138	1	0	1	0	0
BM Mk8	497	1	0	1	0	0
BM Mk9	299	2	0	1	1	0
BM Mk10	196	3	0	0	3	0
Total	-	24	7 (29.1%)	8 (33.3%)	15 (62.5%)	9 (37.5%)

máxima e carga de trabalho total, enquanto o MOGA [Wang et al. 2010] possui no *makespan*. O algoritmo de [qing Li et al. 2010] (TS + VNS) apresenta resultados adequados no *makespan*, porém não produz uma boa diversidade em suas soluções (por exemplo, em todos os problemas de [Brandimarte 1993] apresentou apenas uma solução para cada problema, sendo que a quantidade de soluções não dominadas conhecidas destes problemas variam entre 8 e 146). Além disso, não apresentou bons resultados quanto à carga de trabalho total e máxima.

Apesar de não superar todos os outros algoritmos em nenhum objetivo, os valores

apresentados pelo DIPSO são promissores. Se compararmos o DIPSO com o MOGA [Wang et al. 2010], verificamos que o DIPSO é superior na carga de trabalho total e máxima, enquanto o MOGA [Wang et al. 2010] é superior no *makespan*, mostrando a equivalência das técnicas. Se compararmos o SEA [Chiang e Lin 2013] com DIPSO, verificamos que o SEA possui melhores resultados em todos os objetivos, porém essa é uma técnica que utiliza conhecimento específico do problema, como já analisado anteriormente, enquanto o DIPSO utiliza trocas de informações aleatórias.

Lista de soluções não dominadas

Em [Chiang e Lin 2013] uma tabela com as soluções não dominadas para os problemas de [Kacem et al. 2002c] e de [Brandimarte 1993] é apresentada. Nessa tabela, estão soluções de diversos trabalhos a fim de mostrar as soluções conhecidas para cada problema. Como novas soluções foram encontradas pelo algoritmo DIPSO, apresentamos a Tabela 5.22, considerando a adição dessas novas soluções e a exclusão das soluções dominadas. As novas soluções encontradas pelo DIPSO estão destacadas em negrito.

5.7 Considerações Finais

Neste capítulo, o algoritmo DIPSO foi apresentado e aplicado ao FJSP multiobjetivo. Os resultados encontrados por DIPSO foram promissores, pois boa parte das soluções já conhecidas foram encontradas, alguns resultados foram melhorados e novos resultados foram obtidos. Além disso, esta pesquisa resultou na publicação de um artigo [Carvalho e Fernandes 2014], no 2014 IEEE Congress on Evolutionary Computation (CEC). No próximo capítulo, novos estudos para o FJSP são apresentados, a fim de iniciar novas pesquisas para o problema.

Tabela 5.22: Nova lista de soluções não dominadas para o FJSP (com resultados de DIPSO)

KC 4 × 5			KC 8 × 8			KC 10 × 7			KC 10 × 10			KC 15 × 10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	14	77	12	11	61	11	7	42	6	11	91	11
11	34	9	15	75	12	11	62	10	7	43	5	11	93	10
12	32	8	15	81	11	12	60	12	8	41	7			
13	33	7	16	73	13				8	42	5			
			16	77	11									

Mk1			Mk2			Mk3			Mk4			Mk5		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
40	162	38	26	151	26	204	850	204	60	390	59	172	687	172
40	164	37	27	145	27	204	855	199	61	366	61	173	683	173
40	167	36	28	144	28	204	871	144	61	372	60	175	682	175
41	160	38	**28**	**150**	**26**	204	882	135	62	360	61	178	680	178
41	163	37	29	143	29	204	884	133	62	366	60	179	679	179
42	**156**	**40**	30	142	30	210	848	210	63	354	62	183	677	183
42	158	39	31	141	31	213	844	213	63	357	61	185	676	185
42	165	36	33	140	33	213	850	199	63	360	60	191	675	191
43	**154**	**40**				221	842	221	64	353	62	197	674	197
45	153	42				221	847	199	65	348	63	203	673	203
							...			...		209	672	209

Mk6			Mk7			Mk8			Mk9			Mk10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
62	424	55	139	693	139	523	2524	515	310	2294	301	224	1980	219
65	396	57	140	686	138	523	2534	497	311	2265	308	225	1930	209
65	397	56	143	683	143	524	2519	524	311	2267	304	225	1937	205
65	417	54	144	673	144	533	2514	533	311	2271	303	226	1943	204
66	382	65	150	669	150	542	2509	542	311	2272	302	227	1918	215
66	383	64	151	667	151	551	2504	551	311	2273	301	228	1913	219
66	386	62	156	664	156	560	2499	560	311	2274	299	**270**	**1877**	**215**
66	387	61	157	662	157	569	2494	569	312	2262	310	**279**	**1883**	**212**
66	388	60	161	660	161	578	2489	578	312	2264	308	**281**	**1887**	**211**
66	392	58	162	659	162	587	2484	587	312	2266	307	**303**	**1915**	**204**
	...		166	657	166					...			...	
			175	655	175									
			187	653	187									
			202	651	202									
			217	649	217									

Capítulo 6

Outras Abordagens com o FJSP

Neste trabalho, durante o desenvolvimento do algoritmo DIPSO, foi observada a possibilidade de adotar novas abordagens para tratar o FJSP, tanto com a utilização de novos objetivos, quanto com a implementação de novos algoritmos evolutivos.

Neste capítulo, são apresentadas essas abordagens, que não têm o objetivo de apresentar resultados concretos neste trabalho, porém buscam iniciar novas abordagens com o problema. Na seção 6.1, os novos objetivos propostos, a ociosidade total e a ociosidade efetiva total, são descritos. A seção 6.2 apresenta a aplicação de novos algoritmos ao FJSP. A seção 6.3 descreve os experimentos realizados com os novos objetivos e com os novos algoritmos.

6.1 Novos Objetivos para o FJSP Multiobjetivo

Tradicionalmente, os trabalhos que tratam o FJSP multiobjetivo utilizam como objetivos a minimização do *makespan*, da carga de trabalho total e da carga de trabalho máxima, como podemos observar em [qing Li et al. 2010], [Chiang e Lin 2013], [Carvalho e Fernandes 2014] e [Wang et al. 2010]. [Chiang e Lin 2013] apresentaram um estudo em que concluem que existe incompatibilidade entre o *makespan* e a carga de trabalho total e entre a carga de trabalho máxima e a carga de trabalho total. Tal fato motiva a combinação de outros objetivos para analisar o comportamento do problema, como pode ser visto em [Jia et al. 2007], que utilizaram o atraso[1] combinado com o *makespan*. Neste trabalho, um estudo com a ociosidade foi elaborado e é apresentado a seguir.

6.1.1 Ociosidade total

Em um escalonamento, o tempo em que um recurso fica parado, ocioso, aguardando a próxima tarefa a ser executada, é contabilizado no *makespan*. Considerando uma solução de um escalonamento qualquer, por exemplo, a solução apresentada na Figura 2.1, se

[1] Atraso de um *job* é a diferença entre o tempo de término do *job* e seu tempo de entrega.

a ociosidade existente nos recursos reduzir e o tempo de execução gasto com as tarefas permanecer o mesmo, haverá redução do *makespan*, ou seja, a execução do escalonamento será mais rápida. Nesse contexto, aplicou-se a minimização da ociosidade total (OT) como objetivo do FJSP. Na Equação 6.1, a ociosidade total é definida como sendo o tempo em que uma máquina fica sem executar uma tarefa, considerando o tempo em que o escalonamento está em execução.

$$OT = \sum_{1 \leq k \leq m} (M - W_k) \tag{6.1}$$

Como definido no Capítulo 2, m é a quantidade de máquinas que o problema possui, M é o *makespan* do escalonamento e W_k é a carga de trabalho da máquina k. Portanto, OT é a soma do tempo em que cada máquina não está executando uma tarefa, ou seja, o tempo em que a máquina está ociosa até o fim do escalonamento.

6.1.2 Ociosidade efetiva total

Na definição de ociosidade total, o período em que uma máquina está ociosa e não tem mais nenhuma tarefa a ser executada por ela é contabilizado como ociosidade. Porém, esse tempo em que uma máquina está esperando o escalonamento ser finalizado e não tem mais responsabilidades no escalonamento, pode não ser considerado como ociosidade, pois uma máquina, nesse estado, pode ser liberada para outras atividades. Nesse contexto, definimos a ociosidade efetiva total.

A ociosidade efetiva total (OET) é o tempo em que uma máquina está aguardando uma tarefa para a executar, considerando o período até finalizar sua última tarefa, ou seja, não considera o período que uma máquina não tem mais nenhuma tarefa para ser executada e fica esperando a finalização do escalonamento. Na Equação 6.2, OET é definida como o tempo em que todas máquinas ficaram paradas aguardando a próxima tarefa.

$$OET = \sum_{1 \leq k \leq m} (t_k - W_k) \tag{6.2}$$

No Capítulo 2, t_k é definido como o tempo de execução final da máquina k, isto é, o tempo que é finalizado a última atividade da máquina no escalonamento. Assim, subtrair W_k de t_k significa obter o tempo ocioso da máquina até t_k, pois W_k é o tempo em que a máquina fica trabalhando, executando alguma tarefa. Portanto, nesse objetivo, pretende-se tratar somente o tempo em que a máquina fica aguardando uma tarefa e não o período em que ela já pode estar liberada do escalonamento.

6.2 AEMTs para o FJSP

No trabalho de [dos Santos 2009], o AEMT foi utilizado para o problema de reconfiguração de redes de energia e os resultados apresentados mostraram que o mesmo é capaz de explorar o espaço de busca, pois encontrou soluções de alta qualidade, em um espaço de tempo pequeno. Outra característica do AEMT é a capacidade de trabalhar com múltiplos objetivos conflitantes utilizando uma estratégia de baixo custo computacional [dos Santos 2009]. Esses fatos motivaram o estudo do AEMT associado ao FJSP. Nos estudos realizados, foram considerados os mesmos objetivos antes analisados com o PSO: minimização do *makespan* (Equação 2.2), da carga de trabalho total (Equação 2.3) e da carga de trabalho máxima (Equação 2.4).

6.2.1 Aplicando o AEMT ao FJSP

Como apresentado no Capítulo 3, o AEMT usa subpopulações para lidar com os vários objetivos do problema. Portanto, aplicando os objetivos do problema e, como definido no AEMT, utilizando uma função de agregação, teremos quatro subpopulações:

- ***Makespan*:** os melhores indivíduos, considerando apenas o *makespan*;
- **Carga de trabalho total:** os melhores indivíduos, considerando apenas a carga de trabalho total;
- **Carga de trabalho máxima:** os melhores indivíduos, considerando apenas a carga de trabalho máxima;
- **Função de agregação:** os melhores indivíduos, considerando o resultado da função de soma ponderada, definida na Equação 6.3;

A representação do indivíduo, os dois operadores de cruzamento e o operador de mutação aplicados foram os mesmos daqueles utilizados no DIPSO, ou seja, definidos respectivamente nas figuras, 5.1, 5.3, 5.4, 5.5 e 5.6. A Equação 6.3 mostra a função de soma ponderada aplicada.

$$F(c) \quad = \quad w_1 \cdot F_1(c) + w_2 \cdot F_2(c) + w_3 \cdot F_3(c) \tag{6.3}$$

onde, c é um indivíduo, $F(c)$ é a função de aptidão, w_1 e $F_1(c)$ são, respectivamente, o peso do *makespan* na soma ponderada e seu valor, w_2 e $F_2(c)$ são, respectivamente, o peso da carga de trabalho total na soma ponderada e seu valor e, w_3 e $F_3(c)$ são, respectivamente, o peso da carga de trabalho máxima na soma ponderada e seu valor.

6.2.2 Aplicando o AEMT$_{ND}$ ao FJSP

O AEMT$_{ND}$ é uma variação do AEMT, como pode ser visto no Capítulo 3, e foi apresentado por [Brasil 2012]. A diferencia entre o AEMT$_{ND}$ e o AEMT é que o primeiro possui uma subpopulação a mais. Essa subpopulação, denominada ND, armazena os indivíduos da população que participam do conjunto F_1 (Definição 2.1), ou seja, as soluções não dominadas, representando o conjunto ótimo de Pareto (Definição 2.3). Portanto, as subpopulações participantes do AEMT$_{ND}$ são:

- ***Makespan*:** os melhores indivíduos considerando apenas o *makespan*;
- **Carga de trabalho total:** os melhores indivíduos considerando apenas a carga de trabalho total;
- **Carga de trabalho máxima:** os melhores indivíduos considerando apenas a carga de trabalho máxima;
- **Função de agregação:** os melhores indivíduos considerando o resultado da função de soma ponderada, definida na Equação 6.3;
- **ND:** soluções não dominadas da população.

A representação do indivíduo e os operadores de cruzamento e mutação utilizados são os mesmos presentes no DIPSO e AEMT.

6.2.3 Desenvolvendo um AG multiobjetivo em tabelas para o FJSP

O AEMT apresenta uma técnica diferenciada para lidar com os múltiplos objetivos de um problema, em que a avaliação multiobjetivo utiliza subpopulações. Após o estudo realizado com o FJSP, utilizando AG, PSO e AEMT, a possibilidade de utilizar o processo de avaliação multiobjetivo do AEMT em outras técnicas foi analisada. Assim, implementamos um algoritmo genético multiobjetivo em tabelas (AGMT), que é um AG tradicional, que utiliza o tratamento multiobjetivo do AEMT. No Algoritmo 8, o AGMT é apresentado.

O AGMT é um algoritmo multiobjetivo e a avaliação dos indivíduos é caracterizada pela formação de subpopulações, como no AEMT. Assim como no AG, a população inicial é criada com um tamanho determinado e em seguida os indivíduos são avaliados (linhas 2 e 3). Com a população inicializada, as subpopulações são criadas com os melhores indivíduos, considerando cada objetivo individualmente e com os melhores na avaliação da função de agregação (linhas 4 e 5). Com as principais estruturas inicializadas, a evolução da população é iniciada com a seleção dos pares para reprodução (linha 7), em que o parâmetro *percRepr* é uma porcentagem aplicada à quantidade total de indivíduos presentes em todas subpopulações. A seleção é feita utilizando as subpopulações e não existe

Algoritmo 8: Algoritmo AGMT

```
1  início
2      inicializar população pop com tamPop indivíduos;
3      avaliar todos indivíduos de pop, considerando todos os objetivos;
4      seja no o número de objetivos considerados no problema;
5      gerar (no + 1) subpopulações sbpop a partir de pop, com os y melhores
       indivíduos de cada objetivo e com y melhores indivíduos considerando a função
       de agregação;
6      while não atingir critério de parada do
7          selecionar percRepr indivíduos de sbpop (de qualquer subpopulação),
           formando pares de indivíduos para reprodução;
8          para cada par de indivíduos, escolher aleatoriamente o operador de
           cruzamento a ser utilizado e executar, formando dois novos indivíduos;
9          aplicar operador de mutação aos novos indivíduos, considerando a
           probabilidade probMut;
10         avaliar novos indivíduos, considerando todo os objetivos;
11         seja ne o conjunto dos novos indivíduos gerados;
12         fazer sbpop[noa] = sbpop[noa] ∪ ne, para 1 ≤ noa ≤ no + 1;
13         redefinir sbpop[noa] selecionando os y melhores indivíduos de sbpop[noa],
           para 1 ≤ noa ≤ no + 1 ;
14     end
15 fim
```

nenhuma regra que determine em qual subpopulação deve ser feita a seleção, ou seja, a escolha da subpopulação é aleatória. Nas linhas 8 e 9, os operadores de cruzamento e mutação são executados e, na linha 10, os novos indivíduos são avaliados em todos objetivos do problema. Na sequência, esses novos indivíduos são adicionados em todas as subpopulações e, em seguida, os melhores indivíduos de cada subpopulação são selecionados, de modo que o tamanho de cada subpopulação permaneça o mesmo que o determinado inicialmente (linhas 11, 12 e 13). Ao fim, entende-se como resposta os indivíduos que estiverem presentes em todas subpopulações.

Assim como no AEMT e no AEMT$_{ND}$, o AGMT utiliza dois operadores de cruzamento e a escolha para determinar qual operador executar é feita aleatoriamente. A representação do indivíduo e os operadores de cruzamento e mutação utilizados são os mesmos presentes no DIPSO, AEMT e AEMT$_{ND}$.

Além do AGMT, outro AG implementado foi o AGMT$_{ND}$, que possui a mesma lógica do AGMT, porém possui uma subpopulação a mais, assim como o AEMT$_{ND}$ em relação ao AEMT. Essa subpopulação armazena os indivíduos não dominados, ou seja, o conjunto F_1.

6.3 Experimentos e Resultados

Os experimentos realizados com as abordagens deste capítulo foram realizados em um computador com processador Intel Core I7 (2GHz) com 8 GB de memória RAM. Para realização dos experimentos foram utilizadas as instâncias de [Kacem et al. 2002c], ou seja: 4×5, 8×8, 10×7, 10×10 e 15×10, que fazem uma analogia à quantidade de *jobs* (n) e de máquinas (m), $n \times m$. Nos experimentos com a ociosidade total e a ociosidade efetiva total, foi utilizado o problema 10×10, combinando vários objetivos e realizando as execuções com o algoritmo DIPSO. Nos experimentos com os algoritmos AEMT, AEMT$_{ND}$, AGMT e AGMT$_{ND}$, todos problemas de *benchmark* de [Kacem et al. 2002c] foram utilizados, sendo tratados de forma multiobjetivo, considerando o *makespan*, a carga de trabalho total e a carga de trabalho máxima. As seções a seguir apresentam os resultados desses experimentos, com análise e comparação dos resultados entre eles e com o conjunto NSND. Os resultados apresentados são os melhores encontrados após trinta execuções de cada problema.

6.3.1 Experimentos com os novos objetivos

Nos experimentos com OT e OET, implementou-se várias combinações com os objetivos definidos anteriormente para o FJSP. Em todas combinações, o *makespan* foi considerado, pois é o principal afetado nas alterações de OT e OET. A ideia de aplicar tanto OT quanto OET foi baseada na lógica de que, ao diminuir o tempo de ociosidade das máquinas, pode-se determinar uma diminuição do *makespan*, pois a ociosidade é somada no cálculo do *makespan*. Na Tabela 6.1, são apresentados os parâmetros utilizados para executar o algoritmo DIPSO com as combinações de objetivos com OT e OET.

Tabela 6.1: Parâmetros para DIPSO com OT e OET

Problema	p_{cc}	p_{dx}	Mutação	Tamanho da população	Número de gerações
10×10	80	20	2	1.000	1.400

Ociosidade total

As combinações de objetivos utilizadas nesse experimento foram as seguintes (sendo, todos eles, minimização): M e OT; M, W e OT; M, WT e OT; M, W, WT e OT. Os resultados desse experimento são apresentados na Tabela 6.2. Para mostrar a relação entre os objetivos, os gráficos das Figuras 6.1, 6.2 e 6.3 foram montados. Em todos os gráficos, o conjunto de dados utilizado, para criar a curva, foi retirado do experimento com os quatro objetivos: M, WT, W e OT. Entretanto, cada gráfico mostra a relação de OT com um dos outros objetivos, isoladamente. Assim, o eixo x representa as soluções, onde

cada solução é identificada por um valor, e o eixo y representa os valores dos objetivos considerados em cada gráfico.

Tabela 6.2: Soluções do problema 10×10 - combinações com OT

M	WT	W	OT	M	WT	OT	M	W	OT	M	OT
7	42	6	28	7	42	28	9	7	23	13	14
7	43	5	27	7	43	27	9	8	18	14	9
7	44	5	26	7	44	26	9	9	13	15	8
7	45	5	25	7	45	25	10	10	12		
7	46	5	24	7	46	24					
7	47	5	23	7	47	23					
7	48	5	22	7	48	22					
7	49	6	21	7	49	21					
7	50	6	20	7	50	20					
7	51	6	19	7	51	19					
7	52	6	18	7	52	18					
7	53	6	17	7	53	17					
7	54	6	16	7	54	16					
7	55	6	15	7	55	15					
7	56	7	14	7	56	14					
7	57	7	13	7	57	13					
8	41	7	39	8	41	39					
8	42	5	38								

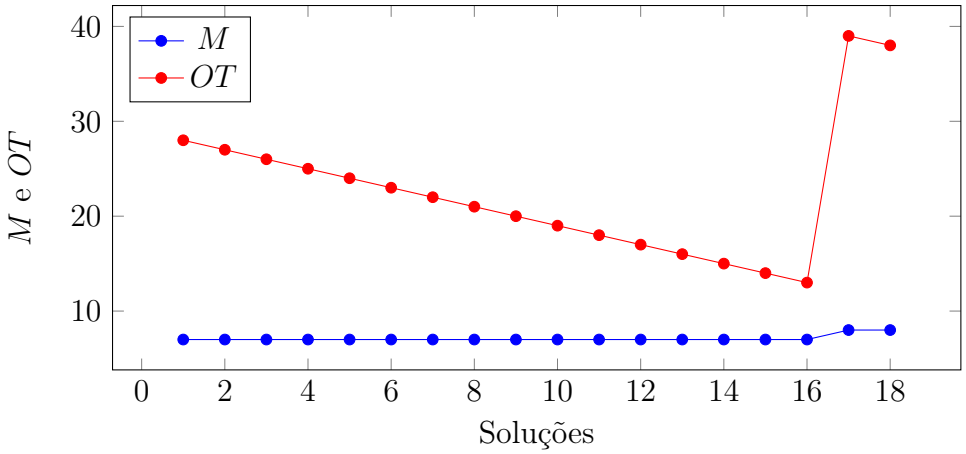

Figura 6.1: Resultados de M e OT

Observando os dados da Tabela 6.2 e dos gráficos das Figuras 6.1, 6.2 e 6.3, concluiu-se que quanto maior o *makespan*, maior a ociosidade total. Isto é verificado quando o *makespan* varia de sete para oito. Fixando o *makespan*, verifica-se a diminuição da ociosidade total com o aumento da carga de trabalho total, que ocorre devido a definição de ociosidade total, ou seja, quando uma máquina não está trabalhando (carga de trabalho),

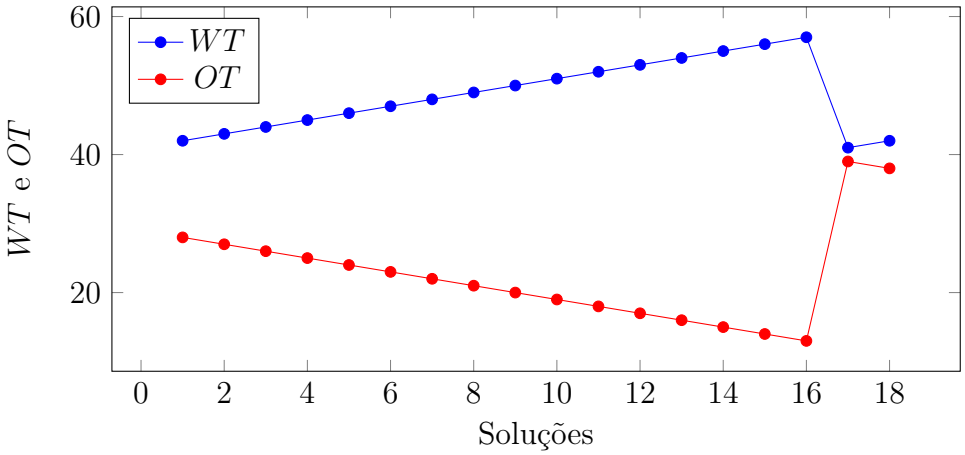

Figura 6.2: Resultados de WT e OT

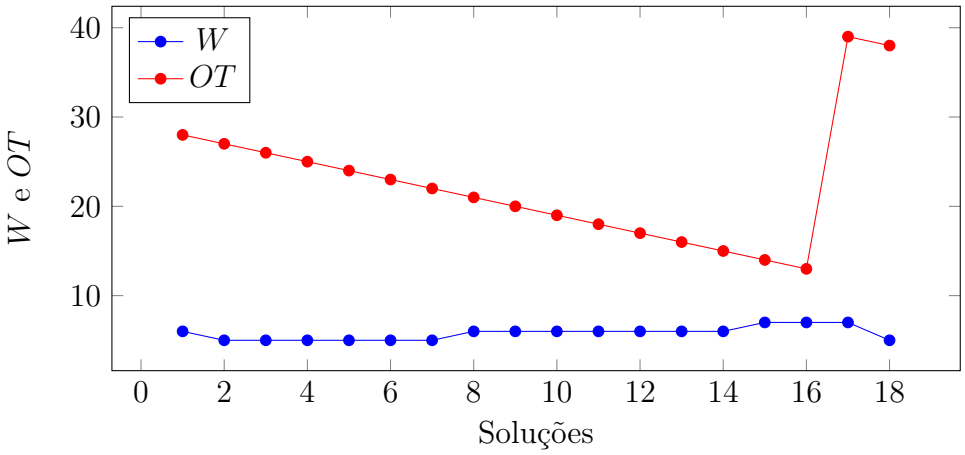

Figura 6.3: Resultados de W e OT

ela está ociosa. Assim, com o *makespan* fixo, quanto maior a carga de trabalho total, menor a ociosidade total. Quanto à carga de trabalho máxima, assim como a relação da carga de trabalho total com a ociosidade total, sua relação com a ociosidade total é inversamente proporcional, ou seja, quanto maior a carga de trabalho máxima, menor a ociosidade total.

Os resultados obtidos relacionando M, W, WT, OT e M, WT, OT foram adequados pois, no experimento com M, W, WT, OT, os quatro resultados conhecidos na literatura foram encontrados: $(M = 7, WT = 42, W = 6)$, $(M = 7, WT = 43, W = 5)$, $(M = 8, WT = 41, W = 7)$ e $(M = 8, WT = 42, W = 5)$. No experimento com M, WT, OT, três soluções conhecidas na literatura foram encontradas: $(M = 7, WT = 42)$, $(M = 7, WT = 43)$ e $(M = 8, WT = 41)$.

Ociosidade efetiva total

Esse experimento utilizou as seguintes combinações de objetivos (sendo, todos eles, minimização): M e OET; M, W e OET; M, WT e OET; M, W, WT e OET. Na Tabela

Tabela 6.3: Soluções do problema 10×10 - combinações com OET

M	WT	W	OET	M	WT	OET	M	W	OET	M	OET
21	55	21	0	12	49	0	11	11	0	10	0
20	56	20	0	12	45	1	10	7	4	9	3
19	62	19	0	11	41	7	9	8	2	8	4
12	44	12	3	11	59	0	9	7	5		
11	45	10	2	10	41	8	9	9	1		
10	46	10	2	10	46	1	8	7	6		
10	42	7	9	10	43	3	8	6	8		
10	41	9	7	9	41	10	8	8	3		
10	42	6	10	9	42	4					
10	41	7	12	8	44	2					
10	43	5	11	8	43	4					
9	41	8	8	7	42	5					
9	45	8	3	7	44	4					
9	47	9	1								
9	48	8	2								
9	42	6	11								
9	44	6	9								
8	43	6	11								
8	42	7	11								
8	45	5	11								
8	42	8	6								
8	42	6	12								
8	42	5	13								
7	43	7	4								
7	45	6	7								
7	47	5	11								

6.3, os resultados dos experimentos com OET são apresentados. Assim como nos gráficos de OT, os gráficos das Figuras 6.4, 6.5 e 6.6 mostram os resultados dos experimentos onde executamos DIPSO com quatro objetivos: M, WT, W e OET. Porém, cada gráfico mostra a relação de OET com um dos outros objetivos, isoladamente. A representação dos eixos x e y é análoga à representação utilizada nos gráficos de OT (Figuras 6.1, 6.2 e 6.3).

Nos resultados apresentados, tem-se a solução $(M = 8, WT = 42, W = 5)$, que é um resultado apresentado pelo DIPSO quando este é executado com os três objetivos tradicionais (M, WT, W). Nesses resultados, essa solução é apresentada com OET igual a treze. Diminuir esse valor de ociosidade, mantendo a carga de trabalho, significaria possivelmente em melhorar o *makespan*. Porém, os resultados dos experimentos, nos mostram que diminuir a ociosidade efetiva total, em benefício dos outros objetivos, não é simples. Para toda combinação de objetivos, o comportamento é semelhante, quanto menor for o valor da ociosidade, o valor do *makespan* tende a ser maior.

Os gráficos das Figuras 6.4, 6.5 e 6.6, que mostram a relação de OET com todos os

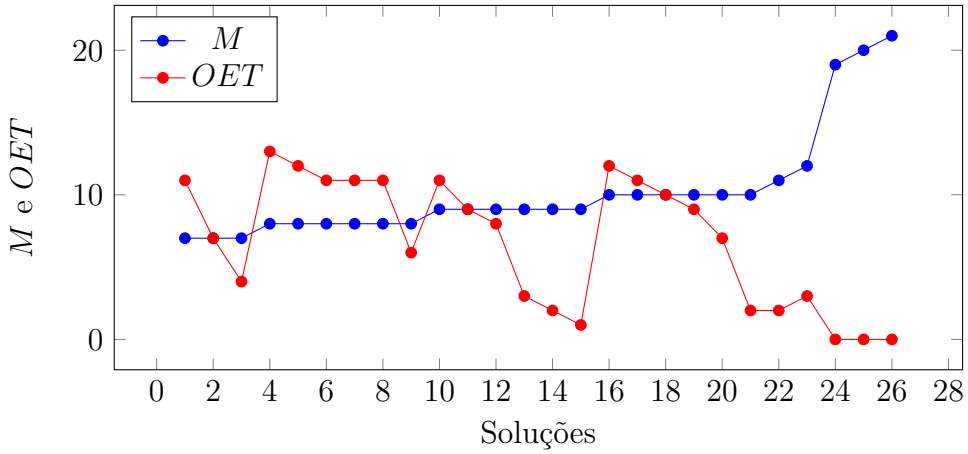

Figura 6.4: Resultados de M e OET

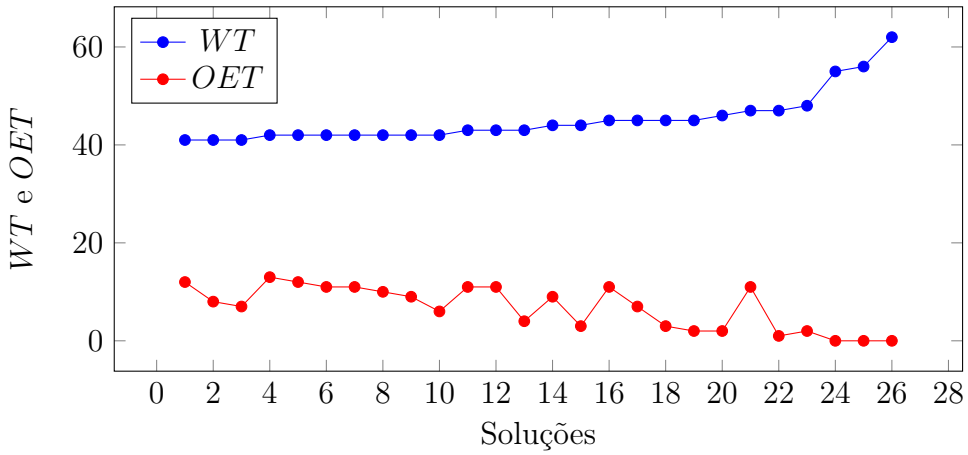

Figura 6.5: Resultados de WT e OET

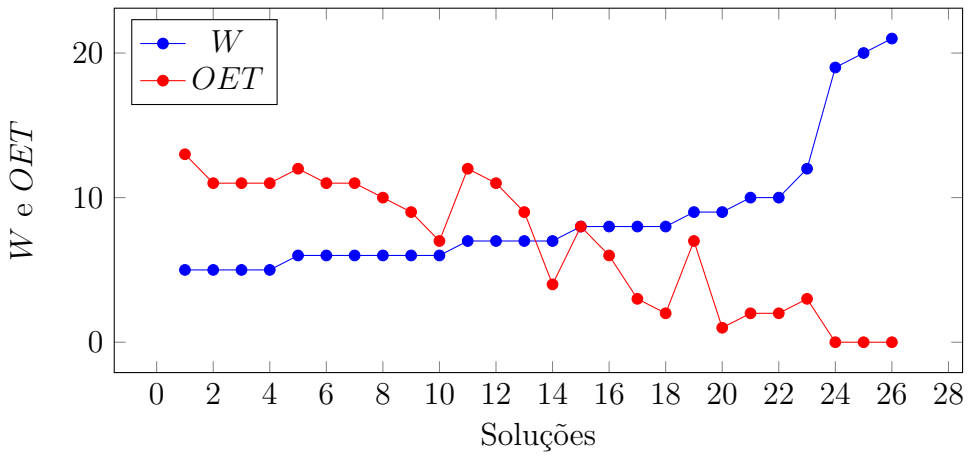

Figura 6.6: Resultados de W e OET

outros objetivos, permitem visualizar melhor a característica de que, em geral, quanto pior o valor do *makespan*, carga de trabalho total ou carga de trabalho máxima, melhor o valor da ociosidade efetiva total. Isto é justificado pelas dependências de cada objetivo. O cálculo dos valores da carga de trabalho (total e máxima) depende somente do tempo gasto

para as máquinas executarem seu escalonamento de operações, ou seja, dependem somente do tempo. O cálculo do *makespan* depende do mesmo tempo gasto pelas máquinas com as operações e da ociosidade entre as operações. O cálculo da ociosidade possui uma dependência complexa. Ele não possui relacionamento direto com o tempo gasto pelas máquinas na execução e depende da possibilidade de formação de escalonamento que o problema oferece, ou seja, do sequenciamento possível de ser construído. Assim, tentar minimizar um parâmetro que depende da organização do escalonamento como um todo, tende a penalizar todos os outros parâmetros, aumentando a complexidade envolvida entre eles e dificultando a análise do espaço de busca pelo algoritmo.

6.3.2 Experimentos com o AEMT

Os parâmetros utilizados no AEMT foram distintos para cada problema, exceto para os pesos da função de agregação, w_1, w_2 e w_3, que foram, respectivamente, 0.8, 0.05 e 0.15, e são os mesmos valores utilizados por [qing Li et al. 2010]. Na Tabela 6.4 são apresentados todos parâmetros utilizados para cada problema: tamanho da população, tamanho das subpopulações, número de gerações e os pesos da função de agregação. Nos parâmetros configurados, destacam-se o número de gerações e o tamanho da população, que estão configurados com valores altos.

Tabela 6.4: Parâmetros para execução do AEMT

Problema	Tamanho da população	Tamanho das subpopulações	Número de gerações	w_1	w_2	w_3
4×5	300	100	2.000	0,8	0,05	0,15
8×8	10.000	200	20.000	0,8	0,05	0,15
10×7	10.000	250	23.000	0,8	0,05	0,15
10×10	10.000	250	30.000	0,8	0,05	0,15
15×10	10.000	400	40.000	0,8	0,05	0,15

O número de gerações, configurado de 2000 (4×5) a 40000 (15×10), é justificado pela lógica do AEMT, que, a cada geração, visita apenas duas novas soluções no espaço de busca, independente dos tamanhos da população e das subpopulações. Portanto, com um número de gerações igual a ng, o AEMT visitará $2ng$ novas soluções na evolução da população. Por outro lado, um AG visita um número de soluções proporcional à quantidade de indivíduos selecionados para reprodução, a cada geração. Assim, se o número de indivíduos selecionados para reprodução é igual ao tamanho da população, o número de soluções visitadas na evolução do algoritmo é igual a $tamPop \times ng$, considerando $tamPop$ como o tamanho da população de um AG. Portanto, um AG com $tamPop = 200$ e $ng = 20$, visita o mesmo número de soluções na evolução da população, do que um AEMT com $ng = 2000$.

O tamanho da população, que está configurado entre 300 (4×5) e 10000 (8×8, 10×7, 10×10 e 15×10), no AEMT, é considerado apenas na inicialização da população, pois, depois de inicializada, a população é dividida em subpopulações e somente essas são consideradas para seleção e avaliação dos indivíduos nas demais etapas do algoritmo. Assim, um valor alto para o tamanho da população, no AEMT, é utilizado para formar um conjunto de indivíduos em que os melhores participarão de alguma subpopulação e os excedentes são desconsiderados, o que aumenta a qualidade dos indivíduos que participarão da evolução da população.

Um parâmetro importante para o algoritmo é o tamanho das subpopulações. Nos experimentos realizados, foi verificado que, quanto menor o valor desse parâmetro, melhor. Porém ele deve ser grande o suficiente para manter a diversidade que a complexidade do problema necessita. Aumentar o tamanho das subpopulações significa aumentar o número de soluções com qualidade inferior que serão consideradas para evolução da população. Assim, a qualidade das novas soluções visitadas tende diminuir e, como o número de soluções visitadas no AEMT não é proporcional ao tamanho das subpopulações, é necessário aumentar o número de gerações para aumentar o número de soluções visitadas.

Tabela 6.5: Resultados dos experimentos com AEMT

KC 4×5			KC 8×8			KC 10×7			KC 10×10			KC 15×10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	14	82	14	11	63	11	8	46	7	15	116	15
12	32	8				12	61	12	9	45	7	16	109	15

Tabela 6.6: Soluções do conjunto NSND para o *benchmark* de [Kacem et al. 2002c]

KC 4×5			KC 8×8			KC 10×7			KC 10×10			KC 15×10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	14	77	12	11	61	11	7	42	6	11	91	11
11	34	9	15	75	12	11	62	10	7	43	5	11	93	10
12	32	8	15	81	11	12	60	12	8	41	7			
13	33	7	16	73	13				8	42	5			
			16	77	11									

Os resultados obtidos nesse experimento, tratando o FJSP multiobjetivo com o AEMT, são apresentados na Tabela 6.5. Na Tabela 6.6, as soluções para os problemas do *benchmark* de [Kacem et al. 2002c], pertencentes ao conjunto NSND, apresentado no Capítulo 5, são listadas. Comparando esses dois conjuntos de soluções, verificamos que o AEMT se comportou bem na maioria dos problemas, porém somente no 4×5 conseguiu soluções compatíveis com as presentes no conjunto NSND. Nos problemas 8×8, 10×7 e 10×10, o AEMT obteve soluções próximas àquelas do conjunto NSND e no problema 15×10, o

mais complexo desse *benchmark*, a diferença entre os resultados foi grande, chegando a 25 unidades para a carga de trabalho total.

6.3.3 Experimentos com o AEMT$_{ND}$

Nos experimentos realizados com o AEMT$_{ND}$, houve a necessidade de algumas alterações na parametrização do algoritmo. Na Tabela 6.7, são apresentados os parâmetros utilizados para execução do AEMT$_{ND}$. Comparando com os parâmetros de AEMT, verifica-se que, no problema 4×5, houve diminuição do número de gerações, nos problemas 10×7 e 10×10, o tamanho da subpopulação foi reduzido e o número de gerações no problema 10×7 foi aumentado. As alterações apresentadas são justificadas pelas diferenças entre os algoritmos, em que o AEMT$_{ND}$ possui uma subpopulação a mais, que armazena os indivíduos não dominados da população, que propõe um novo comportamento na evolução da população.

Tabela 6.7: Parâmetros para execução do AEMT$_{ND}$

Problema	Tamanho da população	Tamanho das subpopulações	Número de gerações	w_1	w_2	w_3
4×5	300	100	1.500	0.8	0.05	0.15
8×8	10.000	200	20.000	0.8	0.05	0.15
10×7	10.000	200	30.000	0.8	0.05	0.15
10×10	10.000	200	30.000	0.8	0.05	0.15
15×10	10.000	400	40.000	0.8	0.05	0.15

A Tabela 6.8 mostra os resultados obtidos nos experimentos com o $AEMT_{ND}$. No problema 4×5, foram encontradas três soluções, porém uma delas não faz parte do conjunto NSND. Nos problemas 8×8, 10×7 e 10×10, os resultados foram próximos aos do conjunto NSND e no problema 15×10, obtivemos a maior diferença de soluções, quando comparamos com o conjunto NSND.

Tabela 6.8: Resultados dos experimentos com AEMT$_{ND}$

4×5			8×8			10×7			10×10			15×10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	17	80	14	12	63	11	8	43	6	17	114	16
12	34	9										16	124	15
13	33	7										18	113	17

Portanto, os resultados encontrados por AEMT$_{ND}$ foram semelhantes aos do AEMT e não obtivemos melhores resultados como [Brasil 2012] obteve, quando propôs o algoritmo, modificando o AEMT. Em alguns problemas, 10×7 e 10×10, foi proposto novos parâmetros a fim de obter resultados semelhantes ao AEMT, pois com a mesma parametrização não foi possível, mas não foi obtido melhoria significativa.

6.3.4 Experimentos com o AGMT

Os experimentos realizados com o AGMT foram executados com a parametrização apresentada na Tabela 6.9. Como a cada geração, o número de soluções visitadas é proporcional ao tamanho da geração, o número de gerações, em todos problemas, é consideravelmente menor do que aqueles apresentados para o AEMT e AEMT$_{ND}$. Em todos os problemas, a porcentagem de seleção para reprodução é de 50%, o que significa que, a cada geração, são gerados a quantidade igual a metade do número de indivíduos presentes em todas subpopulações. Portanto, no problema 4×5, a cada geração, são gerados 200 novos indivíduos, pois são quatro subpopulações com cem indivíduos cada.

Tabela 6.9: Parâmetros para execução do AGMT

Problema	Taxa de Reprodução (%)	Probabilidade de Mutação (%)	Tamanho da população	Tamanho das subpopulações	Número de gerações	w_1	w_2	w_3
4×5	50	7	300	100	300	0.8	0.05	0.15
8×8	50	7	10.000	200	700	0.8	0.05	0.15
10×7	50	7	10.000	200	1.000	0.8	0.05	0.15
10×10	50	10	10.000	150	1.000	0.8	0.05	0.15
15×10	50	7	10.000	300	2.000	0.8	0.05	0.15

Na Tabela 6.10, os resultados obtidos nos experimentos com o AGMT são apresentados. Em relação aos experimentos com o AEMT e AEMT$_{ND}$, houve muita melhora, de modo que das quatorze soluções encontradas para todos problemas, onze estão presentes no conjunto NSND. Nos problemas 4×5, 8×8 e 10×7, todas soluções encontradas estão presentes em NSND. No problema 10×10, das três encontradas, uma solução não pertence ao conjunto NSND e no 15×10, as duas soluções apresentadas não pertencem ao NSND, porém as duas são próximas às soluções de NSND.

Tabela 6.10: Resultados dos experimentos com AGMT

4×5			8×8			10×7			10×10			15×10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	16	73	13	11	62	10	8	41	7	12	92	11
11	34	9	15	75	12	12	60	12	8	44	5	12	91	12
12	32	8	14	77	12	11	61	11	7	42	6			

6.3.5 Experimentos com o AGMT$_{ND}$

Na Tabela 6.11, os parâmetros utilizados para execução do AGMT$_{ND}$ são apresentados. Comparando eles com os parâmetros de AGMT, tem-se que, no problema 4×5, houveram reduções nos tamanhos da população e das subpopulações. A redução no tamanho das subpopulações, apesar de ser um benefício para diminuir o consumo de recursos na execução do algoritmo, foi realizada por necessidade, ou seja, sem a realização dela, os

resultados apresentados não seriam encontrados. Isto é justificado pela sensibilidade no tamanho das subpopulações do AEMT, já discutida na seção 6.3.2.

Tabela 6.11: Parâmetros para execução do $AGMT_{ND}$

Problema	Taxa de Reprodução (%)	Probabilidade de Mutação (%)	Tamanho da população	Tamanho das subpopulações	Número de gerações	w_1	w_2	w_3
4×5	50	7	100	33	300	0.8	0.05	0.15
8×8	50	7	10.000	200	700	0.8	0.05	0.15
10×7	50	7	10.000	200	1.000	0.8	0.05	0.15
10×10	50	10	10.000	150	1.000	0.8	0.05	0.15
15×10	50	7	10.000	300	2.000	0.8	0.05	0.15

Os resultados dos experimentos com o $AGMT_{ND}$ são apresentados na Tabela 6.12. Eles são semelhantes aos resultados obtidos com o AGMT, sendo os resultados do AGMT melhores em todos problemas, exceto no problema 4×5, em que os resultados são idênticos. No problema 8×8 e 10×7, o AGMT obteve um resultado a mais que o $AGMT_{ND}$. No problema 10×10, não se consegue determinar qual algoritmo foi melhor se compararmos os resultados apenas entre eles, pois se aplicarmos as regras de dominância em um conjunto com as soluções dos dois algoritmos juntas, cada algoritmo teria duas soluções não dominadas. Porém, se compararmos esses resultados com o conjunto NSND, observamos que o AGMT encontrou duas soluções presentes nele e o $AGMT_{ND}$ encontrou apenas uma. Portanto, os resultados do AGMT são melhores. No problema 15×10, todos os resultados do AGMT são melhores que os do $AGMT_{ND}$.

Tabela 6.12: Resultados dos experimentos com $AGMT_{ND}$

4×5			8×8			10×7			10×10			15×10		
M	WT	W	M	WT	W	M	WT	W	M	WT	W	M	WT	W
11	32	10	16	73	13	12	60	12	8	41	7	13	94	11
11	34	9	15	75	12	11	61	11	8	42	6	13	91	12
12	32	8							8	43	5	15	93	11

6.3.6 Discussão dos resultados das novas técnicas aplicadas ao FJSP

Entre os quatro algoritmos (AEMT, $AEMT_{ND}$, AGMT e $AGMT_{ND}$), aquele que conseguiu os melhores resultados foi o AGMT, pois conseguiu o maior número de soluções e o maior número de soluções referenciadas no conjunto NSND. A utilização de uma subpopulação com as soluções não dominadas, característica do $AEMT_{ND}$ e do $AGMT_{ND}$, não resultaram em melhoras que justifiquem a utilização, quando comparados com as versões dos algoritmos que não possuem essa subpopulação, AEMT e AGMT. Porém, é necessário ressaltar, que não houve nenhum experimento com outros operadores de cruzamento e mutação, esse utilizado somente no AGMT e $AGMT_{ND}$, o que pode mudar

o comportamento dos algoritmos. Neste trabalho, não efetuamos nenhum experimento com esses algoritmos que envolva outro operador, além daqueles apresentados no Capítulo 5. Portanto, pode-se concluir que, com esses operadores, os algoritmos AEMT$_{ND}$ e AGMT$_{ND}$ não obtiveram resultados que justifiquem a utilização deles em relação ao AEMT e AGMT.

Tabela 6.13: Número de soluções visitadas na evolução dos algoritmos

Problema	AEMT	AEMT$_{ND}$	AGMT	AGMT$_{ND}$
4×5	4.000	3.000	60.000	24.600
8×8	40.000	40.000	280.000	350.000
10×7	46.000	60.000	400.000	500.000
10×10	60.000	60.000	300.000	375.000
15×10	80.000	80.000	1.200.000	1.500.000

Tabela 6.14: Tempo de execução dos algoritmos (segundos)

Problema	AEMT	AEMT$_{ND}$	AGMT	AGMT$_{ND}$
4×5	0,8	1,8	1,3	2,5
8×8	9,5	150,2	7,2	29,1
10×7	15,1	112,2	9,5	43,6
10×10	17,3	482,3	7,8	32,6
15×10	58,8	445,2	49,9	212,5

Na Tabela 6.13 são apresentados a quantidade de soluções visitadas na evolução da população de cada algoritmo, para cada problema e na Tabela 6.14 são apresentados os tempos de execução. Pela estrutura do algoritmo e pelos parâmetros utilizados, os AGs (AGMT e AGMT$_{ND}$) visitam um número de soluções maior, em um tempo menor, considerando uma comparação com AEMT e AEMT$_{ND}$, respectivamente. Assim, os resultados apresentados pelos AGs, melhores que dos AEMTs (AEMT e AEMT$_{ND}$), são justificados pelo número de soluções visitadas. Uma experimento, realizado com o problema 8×8, foi modelar o AEMT de modo que ele tivesse o mesmo número de soluções visitadas do que o AGMT. Os resultados iniciais apresentados não tiveram melhorias e o experimento foi descartado. Além disso, um experimento com todos problemas, implicaria em implementar uma versão para o problema 15×10 e, seguindo a lógica da relação número de soluções visitadas por tempo de execução, esse experimento teria o tempo de execução alto, principalmente no AEMT$_{ND}$.

6.4 Considerações Finais

Neste capítulo, foram apresentadas novas abordagens para o FJSP, novos objetivos e novos algoritmos para tratar o FJSP. Os novos objetivos foram apresentados e experimentos foram realizados para validar as possibilidades. Em relação aos algoritmos, foi

apresentado como utilizamos o AEMT, AEMT$_{ND}$, AGMT e o AGMT$_{ND}$ para tratar o FJSP e foi apresentado e discutido os experimentos realizados com esses algoritmos. Os resultados com AGMT foram promissores e novos experimentos devem ser realizados para melhorar o algoritmo. Os outros algoritmos, principalmente os AEMTs, necessitam de um estudo aprofundado para verificar a viabilidade deles tratando o FJSP, avaliando novos operadores (cruzamento e mutação).

Capítulo 7

Conclusões e Trabalhos Futuros

Neste trabalho apresentamos o algoritmo DIPSO aplicado ao FJSP multiobjetivo, considerando a minimização do *makespan*, carga de trabalho total e carga de trabalho máxima. Além disso, elaboramos um estudo complementar visando novas abordagens com o FJSP multiobjetivo. Nesse caso, elaborou-se uma nova combinação de objetivos e utilizou-se novas técnicas para tratar o problema.

DIPSO é um algoritmo PSO multiobjetivo híbrido que utiliza os operadores de AG, cruzamento e mutação, para evoluir a população. DIPSO utiliza dois operadores de cruzamento: IPOX e DX. Proposto neste trabalho, o operador DX introduz diversidade à população, o que permite uma melhor análise do espaço de busca, enquanto que IPOX, proposto por [Gen et al. 1994], possui características que auxiliam a população na convergência. Outra particularidade de DIPSO é a utilização de um conjunto como memória global do algoritmo, ou seja, ao invés da variável g_{best} armazenar somente uma partícula a cada geração, ela armazena um conjunto de partículas, que são as partículas não dominadas que o algoritmo conhece até a geração em questão.

A combinação de tais características formaram um algoritmo eficiente, capaz de evitar mínimos locais e fazer uma melhor análise do espaço de busca. Tal fato é comprovado com os experimentos comparativos dos resultados do algoritmo quando utiliza e quando não utiliza as características citadas. Assim, com esses experimentos, ficou clara a importância da utilização dessas particularidades, pois foi possível comparar o comportamento de DIPSO com e sem a utilização de cada característica. Além disso, os resultados que foram alcançados pelo algoritmo DIPSO foram adequados, pois reproduziram diversas soluções já conhecidas na literatura e apresentaram novas soluções. No caso de duas instâncias (Mk1 e Mk2), essas novas soluções dominaram soluções presentes na literatura. Assim, fez-se necessária a atualização da tabela de soluções não dominadas conhecidas, para os problemas testados, considerando a adição das novas soluções e a exclusão de soluções conhecidas que foram dominadas.

As novas abordagens com o FJSP contemplaram a combinação de novos objetivos e dois novos foram apresentados: ociosidade total e ociosidade efetiva total. Nos experi-

mentos realizados, esses novos objetivos foram combinados com os objetivos tradicionais do problema: *makespan*, carga de trabalho total e carga de trabalho máxima. Os experimentos mostraram que a combinação entre a ociosidade total e os outros objetivos é compatível, porém a utilização da ociosidade total sem a carga de trabalho total não trouxe bons resultados. Além disso, a ociosidade efetiva total é incompatível com o *makespan*. Quanto menor o *makespan*, a ociosidade efetiva total tende a ser maior e a complexidade existente na avaliação da ociosidade efetiva não favorece a associação.

Além dos novos objetivos, novas técnicas foram aplicadas ao FJSP multiobjetivo. AEMT, AEMT$_{ND}$, AGMT e AGMT$_{ND}$. Os dois primeiros são algoritmos encontrados na literatura e os dois últimos são AGs modificados para utilizar a forma de avaliação de soluções multiobjetivo, que o AEMT e o AEMT$_{ND}$ utilizam. Nos experimentos realizados, ficou evidente a dificuldade de AEMT e AEMT$_{ND}$ evoluírem a população e apresentar resultados semelhantes aos da literatura. Entretanto, o AGMT e o AGMT$_{ND}$ conseguiram apresentar resultados presentes na literatura sem nenhum estudo específico. Destaca-se, também, que o AGMT apresentou os melhores resultados entre eles, pois conseguiu o maior número de soluções presentes na literatura e, se comparado entre eles, apresenta o maior número de soluções não dominadas.

É importante ressaltar que essas novas abordagens são apenas o início de um trabalho. Assim, seus resultados não são para eliminar possibilidades de trabalho com elas, pois não foi realizado nenhum estudo específico com essas novas abordagens, neste trabalho. Apesar de que nem todos os experimentos relacionados com as novas abordagens trouxeram bons resultados, acreditamos que essas abordagens podem ser evoluídas e trazer resultados interessantes.

A fim de estender este trabalho, temos a possibilidade de criar novos operadores sobre o algoritmo DIPSO para observar a diversidade e convergência da população. Outra possibilidade é realizar um estudo aprofundado com os novos objetivos apresentados e com as novas técnicas aplicadas ao FJSP multiobjetivo. No caso das novas técnicas, pode-se, também, realizar experimentos com operadores diferentes daquele utilizado neste trabalho.

Referências Bibliográficas

[Bagchi 1999] Bagchi, T. (1999). *Multiobjective Scheduling by Genetic Algorithms*. Springer US.

[Bansal et al. 2011] Bansal, J., Singh, P., Saraswat, M., Verma, A., Jadon, S., e Abraham, A. (2011). Inertia Weight strategies in Particle Swarm Optimization. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pp. 633 – 640.

[Boyd e Vandenberghe 2004] Boyd, S. e Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Brandimarte 1993] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183.

[Brasil 2012] Brasil, C. R. S. (2012). *Algoritmo evolutivo de muitos objetivos para predição ab initio de estrutura de proteínas*. Doutorado, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.

[Carvalho e Fernandes 2014] Carvalho, L. e Fernandes, M. (2014). Multi-Objective Flexible Job-Shop Scheduling Problem with DIPSO: More Diversity, Greater Efficiency. In Coello Coello, C. A. (editor), *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, pp. 282 – 289, Beijing, China.

[Cheng et al. 2009] Cheng, H.-C., Chiang, T.-C., e Fu, L.-C. (2009). Multiobjective job shop scheduling using memetic algorithm and shifting bottleneck procedure. In *Computational Intelligence in Scheduling, 2009. CI-Sched '09. IEEE Symposium on*, pp. 15–21.

[Cheng et al. 2011] Cheng, H.-C., Chiang, T.-C., e Fu, L.-C. (2011). A two-stage hybrid memetic algorithm for multiobjective job shop scheduling. *Expert Systems with Applications*, 38(9):10983 – 10998.

[Cheng et al. 1996] Cheng, R., Gen, M., e Tsujimura, Y. (1996). A Tutorial Survey of Job-Shop Scheduling Problems using Genetic Algorithms - I. Representation. *Computers & Industrial Engineering*, 30(4):983 – 997.

[Chiang e Lin 2013] Chiang, T.-C. e Lin, H.-J. (2013). A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics*, 141(1):87 – 98. Meta-heuristics for manufacturing scheduling and logistics problems.

[dao-er ji et al. 2013] dao-er ji, R. Q., Wang, Y., e Wang, X. (2013). Inventory based two-objective job shop scheduling model and its hybrid genetic algorithm. *Applied Soft Computing*, 13(3):1400 – 1406. Hybrid evolutionary systems for manufacturing processes.

[Deb et al. 2002] Deb, K., Pratap, A., Agarwal, S., e Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182 – 197.

[dos Santos 2009] dos Santos, A. C. (2009). *Algoritmo Evolutivo Computacionalmente Eficiente para Reconfiguração de Sistemas de Distribuição*. Doutorado, Escola de Engenharia de São Carlos - Universidade de São Paulo.

[Gen et al. 1994] Gen, M., Tsujimura, Y., e Kubota, E. (1994). Solving job-shop scheduling problems by genetic algorithm. In *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, volume 2, pp. 1577 – 1582 vol.2.

[Goldberg 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Graham et al. 1979] Graham, R., Lawler, E., Lenstra, J., e Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In P.L. Hammer, E. J. e Korte, B. (editores), *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, volume 5 de *Annals of Discrete Mathematics*, pp. 287 – 326. Elsevier.

[Gromicho et al. 2012] Gromicho, J. A., van Hoorn, J. J., da Gama, F. S., e Timmer, G. T. (2012). Solving the job-shop scheduling problem optimally by dynamic programming. *Computers & Operations Research*, 39(12):2968 – 2977.

[Guimarães 2007] Guimarães, K. F. (2007). Escalonamento Genético FJSP com tempo de configuração dependente de sequência. Mestrado, Faculdade de Computção - Universidade Federal de Uberlândia.

[Heris e Oskoei 2014] Heris, J. e Oskoei, M. (2014). Modified genetic algorithm for solving n-queens problem. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pp. 1–5.

[Ho e Tay 2004] Ho, N. B. e Tay, J. C. (2004). Genace: An Efficient Cultural Algorithm for Solving the Flexible Job-Shop Problem. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pp. 1759 – 1766.

[Ho e Tay 2007] Ho, N. B. e Tay, J. C. (2007). Using Evolutionary Computation and Local Search to Solve Multi-objective Flexible Job Shop Problems. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pp. 821 – 828.

[Hsu et al. 2002] Hsu, T., Dupas, R., Jolly, D., e Goncalves, G. (2002). Evaluation of mutation heuristics for solving a multiobjective flexible job shop by an evolutionary algorithm. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 5, pp. 6 pp. vol.5–.

[Hu et al. 2011] Hu, X., Wang, L., e Zhong, Y. (2011). An improved particle swarm optimization algorithm for site index curve model. In *Business Management and Electronic Information (BMEI), 2011 International Conference on*, volume 3, pp. 838–842.

[Jansen et al. 2000] Jansen, K., Mastrolilli, M., e Solis-Oba, R. (2000). Approximation Algorithms for Flexible Job Shop Problems. In Gonnet, G. e Viola, A. (editores), *LATIN 2000: Theoretical Informatics*, volume 1776 de *Lecture Notes in Computer Science*, pp. 68–77. Springer Berlin Heidelberg.

[Jia et al. 2007] Jia, Z., Chen, H., e Tang, J. (2007). A New Multi-objective Fully-Informed Particle Swarm Algorithm for Flexible Job-Shop Scheduling Problems. In *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*, pp. 191 – 194.

[Kacem et al. 2002a] Kacem, I., Hammadi, S., e Borne, P. (2002a). Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-shop Scheduling Problems. *Trans. Sys. Man Cyber Part C*, 32(1):1–13.

[Kacem et al. 2002b] Kacem, I., Hammadi, S., e Borne, P. (2002b). Correction to "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems". *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(2):172–172.

[Kacem et al. 2002c] Kacem, I., Hammadi, S., e Borne, P. (2002c). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60(3 - 5):245 – 276.

[Kennedy e Eberhart 1995] Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pp. 1942–1948 vol.4.

[Lawler et al. 1993] Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., e Shmoys, D. B. (1993). Sequencing and Scheduling: Algorithms and Complexity. In Graves, S., Kan, A. H. G. R., e Zipkin, P. (editores), *Logistic of Production and Inventory*, chapter 9, pp. 445–522. Handbooks in Operationals Research and Management Science, North Holland, Amsterdam.

[Lenstra e Rinnooy 1979] Lenstra, J. e Rinnooy, K. (1979). Computational Complexity of Discrete Optimization Problems. In *Discrete Optimization I Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium*, volume 4 de *Annals of Discrete Mathematics*, pp. 121 – 140.

[Ling-li et al. 2009] Ling-li, Z., Feng-Xing, Z., e Xiao-hong, X. (2009). Mathematical Model and Hybrid Particle Swarm Optimization for Flexible Job-Shop Scheduling Problem. *Genetic and Evolutionary Computation Conference*, pp. 731 – 736.

[Maccarthy e Liu 1993] Maccarthy, B. L. e Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79.

[Mahfoud 1995] Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. Doutorado, Graduate College - University of Illinois at Urbana-Champaign.

[Martínez e Gonzalo 2009] Martínez, J. L. F. e Gonzalo, E. G. (2009). The PSO family: deduction, stochastic analysis and comparison. *Swarm Intelligence*, 3(4):245 – 273.

[Mastrolilli e Gambardella 2000] Mastrolilli, M. e Gambardella, L. M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1):3–20.

[Mendes et al. 2004] Mendes, R., Kennedy, J., e Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, 8(3):204–210.

[Moslehi e Mahnam 2011] Moslehi, G. e Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14 – 22.

[Nababan et al. 2008] Nababan, E. B., Hamdan, A. R., Abdullah, S., e Zakaria, M. (2008). Branch and bound algorithm in optimizing job shop scheduling problems. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 1, pp. 1–5.

[Neifar et al. 2006] Neifar, F., Gzara, M., Moukrim, A., e Loukil, T. (2006). Hybrid Evolutionary Algorithm With Insertion Heuristics For The Flexible Job Shop Problem. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pp. 1211–1216.

[Nickabadi et al. 2011] Nickabadi, A., Ebadzadeh, M. M., e Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing*, 11(4):3658 – 3670.

[Niu et al. 2008] Niu, Q., Jiao, B., e Gu, X. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Applied Mathematics and Computation*, 205(1):148 – 158.

[Pan 1997] Pan, C.-h. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, 28(1):33–41.

[Pezzella et al. 2008] Pezzella, F., Morganti, G., e Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, 35(10):3202 – 3212. Part Special Issue: Search-based Software Engineering.

[Qin et al. 2006] Qin, Z., Yu, F., Shi, Z., e Wang, Y. (2006). Adaptive Inertia Weight Particle Swarm Optimization. In *Artificial Intelligence and Soft Computing - ICAISC 2006*, volume 4029 de *Lecture Notes in Computer Science*, pp. 450 – 459.

[qing Li et al. 2010] qing Li, J., ke Pan, Q., e Liang, Y.-C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 59(4):647 – 662.

[Shi e Eberhart 1998] Shi, Y. e Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73.

[Spillman 1993] Spillman, R. (1993). Genetic Algorithms: Nature's way to search for the best. *Dr. Dobb's Jounal*, pp. 26 – 30.

[Srinivas e Deb 1994] Srinivas, N. e Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.*, 2(3):221–248.

[Vaca 1995] Vaca, O. C. L. (1995). Um algoritmo evolutivo para a programação de projetos multimodos com nivelamento de recursos limitados. Mestrado, Universidade Federal de Santa Catarina, Florianópolis, Brasil.

[Wang et al. 2010] Wang, X., Gao, L., Zhang, C., e Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 51(5 - 8):757 – 767.

[Xia e Wu 2005] Xia, W. J. e Wu, Z. M. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 48(2):409 – 425.

[Xiao-hong et al. 2010] Xiao-hong, X., Ling-li, Z., e Yue-wen, F. (2010). Hybrid particle swarm optimization for flexible job-shop scheduling problem and its implementation. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pp. 1155 – 1159.

[Xing et al. 2009] Xing, L.-N., Chen, Y.-W., e Yang, K.-W. (2009). An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 20(3):283–293.

[Zhang et al. 2009] Zhang, G., Shao, X., Li, P., e Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309 – 1318.

Apêndices

Apêndice A

Instâncias dos problemas FJSP

Este apêndice apresenta as instâncias de problemas dos *benchmarks* utilizados neste trabalho: [Kacem et al. 2002c] e [Brandimarte 1993].

A.1 [Kacem et al. 2002c]

Tabela A.1: Problema 4×5 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Tabela A.2: Problema 8×8 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
J_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	10	-	9	9	9	7	6	4
	$O_{6,3}$	11	5	9	10	11	-	10	-
J_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
J_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

Tabela A.3: Problema 10×7 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7
J_1	$O_{1,1}$	1	4	6	9	3	5	2
	$O_{1,2}$	8	9	5	4	1	1	3
	$O_{1,3}$	4	8	10	4	11	4	3
J_2	$O_{2,1}$	6	9	8	6	5	10	3
	$O_{2,2}$	2	10	4	5	9	8	4
J_3	$O_{3,1}$	15	4	8	4	8	7	1
	$O_{3,2}$	9	6	1	10	7	1	6
	$O_{3,3}$	11	2	7	5	2	3	14
J_4	$O_{4,1}$	2	8	5	8	9	4	3
	$O_{4,2}$	5	3	8	1	9	3	6
	$O_{4,3}$	1	2	6	4	1	7	2
J_5	$O_{5,1}$	7	1	8	5	4	3	9
	$O_{5,2}$	2	4	5	10	6	4	9
	$O_{5,3}$	5	1	7	1	6	6	2
J_6	$O_{6,1}$	8	7	4	56	9	8	4
	$O_{6,2}$	5	14	1	9	6	5	8
	$O_{6,3}$	3	5	2	5	4	5	7
J_7	$O_{7,1}$	5	6	3	6	5	15	2
	$O_{7,2}$	6	5	4	9	5	4	3
	$O_{7,3}$	9	8	2	8	6	1	7
J_8	$O_{8,1}$	6	1	4	1	10	4	3
	$O_{8,2}$	11	13	9	8	9	10	8
	$O_{8,3}$	4	2	7	8	3	10	7
J_9	$O_{9,1}$	12	5	4	5	4	5	5
	$O_{9,2}$	4	2	15	99	4	7	3
	$O_{9,3}$	9	5	11	2	5	4	2
J_{10}	$O_{10,1}$	9	4	13	10	7	6	8
	$O_{10,2}$	4	3	25	3	8	1	2
	$O_{10,3}$	1	2	6	11	13	3	5

Tabela A.4: Problema 10×10 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
J_2	$O_{2,1}$	2	10	4	5	9	8	4	15	8	4
	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
J_3	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4
J_4	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
J_5	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
J_6	$O_{6,1}$	8	9	10	8	4	2	7	8	3	10
	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
J_7	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
J_8	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
	$O_{8,2}$	8	3	10	7	5	13	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
J_9	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
J_{10}	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

Tabela A.5: Problema 15×10 de [Kacem et al. 2002c]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
J_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
J_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	$O_{3,4}$	7	3	12	1	6	5	8	3	5	2
J_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
J_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
J_6	$O_{6,1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6,2}$	1	3	6	5	4	7	5	4	6	5
J_7	$O_{7,1}$	1	4	2	5	3	6	9	8	5	4
	$O_{7,2}$	2	1	4	5	2	3	5	4	2	5
J_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2
J_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	10	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
	$O_{9,4}$	9	8	7	4	5	8	7	4	56	2
J_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
J_{11}	$O_{11,1}$	1	2	3	6	5	2	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4

Tabela A.6: Problema 15 × 10 de [Kacem et al. 2002c] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
J_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
J_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
J_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5
	$O_{15,3}$	4	5	2	3	5	2	8	4	7	5
	$O_{15,4}$	6	2	11	14	2	3	6	5	4	8

A.2 [Brandimarte 1993]

Tabela A.7: Problema Mk1 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6
J_1	$O_{1,1}$	5	-	4	-	-	-
	$O_{1,2}$	-	1	5	-	3	-
	$O_{1,3}$	-	-	4	-	-	2
	$O_{1,4}$	1	6	-	-	-	5
	$O_{1,5}$	-	-	1	-	-	-
	$O_{1,6}$	-	-	6	3	-	6
J_2	$O_{2,1}$	-	6	-	-	-	-
	$O_{2,2}$	-	-	1	-	-	-
	$O_{2,3}$	2	-	-	-	-	-
	$O_{2,4}$	-	6	-	6	-	-
	$O_{2,5}$	1	6	-	-	-	5
J_3	$O_{3,1}$	-	6	-	-	-	-
	$O_{3,2}$	-	-	4	-	-	2
	$O_{3,3}$	1	6	-	-	-	5
	$O_{3,4}$	-	6	4	-	-	6
	$O_{3,5}$	1	-	-	-	5	-
J_4	$O_{4,1}$	1	6	-	-	-	5
	$O_{4,2}$	-	6	-	-	-	-
	$O_{4,3}$	-	-	1	-	-	-
	$O_{4,4}$	-	1	5	-	3	-
	$O_{4,5}$	-	-	4	-	-	2
J_5	$O_{5,1}$	-	1	5	-	3	-
	$O_{5,2}$	1	6	-	-	-	5
	$O_{5,3}$	-	6	-	-	-	-
	$O_{5,4}$	5	-	4	-	-	-
	$O_{5,5}$	-	6	-	6	-	-
	$O_{5,6}$	-	6	4	-	-	6
J_6	$O_{6,1}$	-	-	4	-	-	2
	$O_{6,2}$	2	-	-	-	-	-
	$O_{6,3}$	-	6	4	-	-	6
	$O_{6,4}$	-	6	-	-	-	-
	$O_{6,5}$	1	6	-	-	-	5
	$O_{6,6}$	3	-	-	2	-	-
J_7	$O_{7,1}$	-	-	-	-	-	1
	$O_{7,2}$	3	-	-	2	-	-
	$O_{7,3}$	-	6	4	-	-	6
	$O_{7,4}$	6	6	-	-	1	-
	$O_{7,5}$	-	-	1	-	-	-
J_8	$O_{8,1}$	-	-	4	-	-	2
	$O_{8,2}$	-	6	4	-	-	6

Tabela A.8: Problema Mk1 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6
	$O_{8,3}$	1	6	-	-	-	5
	$O_{8,4}$	-	6	-	-	-	-
	$O_{8,5}$	-	6	-	6	-	-
J_9	$O_{9,1}$	-	-	-	-	-	1
	$O_{9,2}$	1	-	-	-	5	-
	$O_{9,3}$	-	-	6	3	-	6
	$O_{9,4}$	2	-	-	-	-	-
	$O_{9,5}$	-	6	4	-	-	6
	$O_{9,6}$	-	6	-	6	-	-
J_{10}	$O_{10,1}$	-	-	4	-	-	2
	$O_{10,2}$	-	6	4	-	-	6
	$O_{10,3}$	-	1	5	-	3	-
	$O_{10,4}$	-	-	-	-	-	1
	$O_{10,5}$	-	6	-	6	-	-
	$O_{10,6}$	3	-	-	2	-	-

Tabela A.9: Problema Mk2 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6
J_1	$O_{1,1}$	3	2	3	5	3	6
	$O_{1,2}$	-	-	4	-	-	5
	$O_{1,3}$	1	6	3	3	6	5
	$O_{1,4}$	-	6	-	-	-	-
	$O_{1,5}$	-	-	-	-	6	3
	$O_{1,6}$	-	1	2	-	4	-
J_2	$O_{2,1}$	3	4	-	2	6	1
	$O_{2,2}$	-	-	-	-	6	3
	$O_{2,3}$	-	-	-	-	2	-
	$O_{2,4}$	-	4	3	-	-	-
	$O_{2,5}$	-	1	2	-	4	-
	$O_{2,6}$	3	2	3	5	3	6
J_3	$O_{3,1}$	1	6	3	3	6	5
	$O_{3,2}$	2	4	6	6	3	6
	$O_{3,3}$	-	1	2	-	4	-
	$O_{3,4}$	4	3	5	-	2	3
	$O_{3,5}$	5	4	3	1	5	3
	$O_{3,6}$	4	-	6	6	3	3
J_4	$O_{4,1}$	4	3	5	-	2	3
	$O_{4,2}$	3	4	-	2	6	1
	$O_{4,3}$	-	6	-	-	-	-
	$O_{4,4}$	1	6	3	3	6	5
	$O_{4,5}$	4	3	-	5	4	3
	$O_{4,6}$	5	4	3	1	5	3
J_5	$O_{5,1}$	2	4	6	6	3	6
	$O_{5,2}$	4	3	-	5	4	3
	$O_{5,3}$	-	-	-	3	-	-
	$O_{5,4}$	4	-	6	6	3	3
	$O_{5,5}$	3	4	-	2	6	1
	$O_{5,6}$	-	4	3	-	-	-
J_6	$O_{6,1}$	4	-	6	6	3	3
	$O_{6,2}$	-	-	4	-	-	5
	$O_{6,3}$	4	3	5	-	2	3
	$O_{6,4}$	2	4	6	6	3	6
	$O_{6,5}$	-	6	-	-	-	-
	$O_{6,6}$	3	4	-	2	6	1
J_7	$O_{7,1}$	5	4	3	1	5	3
	$O_{7,2}$	-	-	-	-	2	-
	$O_{7,3}$	2	4	6	6	3	6
	$O_{7,4}$	3	2	3	5	3	6

Tabela A.10: Problema Mk2 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6
	$O_{7,5}$	4	-	6	6	3	3
J_8	$O_{8,1}$	-	4	3	-	-	-
	$O_{8,2}$	4	3	5	-	2	3
	$O_{8,3}$	2	4	6	6	3	6
	$O_{8,4}$	4	-	6	6	3	3
	$O_{8,5}$	4	3	-	5	4	3
	$O_{8,6}$	3	4	-	2	6	1
J_9	$O_{9,1}$	-	6	-	-	-	-
	$O_{9,2}$	-	-	4	-	-	5
	$O_{9,3}$	3	4	-	2	6	1
	$O_{9,4}$	4	3	-	5	4	3
	$O_{9,5}$	-	4	3	-	-	-
J_{10}	$O_{10,1}$	-	-	-	3	-	-
	$O_{10,2}$	2	4	6	6	3	6
	$O_{10,3}$	4	-	6	6	3	3
	$O_{10,4}$	5	4	3	1	5	3
	$O_{10,5}$	-	-	-	-	6	3
	$O_{10,6}$	3	4	-	2	6	1

Tabela A.11: Problema Mk3 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	-	-	-	5	19	-	15	11
	$O_{1,2}$	-	-	18	5	-	-	-	-
	$O_{1,3}$	-	-	16	-	-	11	3	18
	$O_{1,4}$	-	1	19	-	7	-	2	-
	$O_{1,5}$	-	-	-	-	6	3	-	-
	$O_{1,6}$	-	-	-	5	2	-	-	18
	$O_{1,7}$	-	-	-	-	2	-	-	-
	$O_{1,8}$	17	-	-	-	-	-	-	-
	$O_{1,9}$	12	10	14	-	10	-	-	5
	$O_{1,10}$	-	-	-	-	-	2	15	19
J_2	$O_{2,1}$	-	-	16	-	-	11	3	18
	$O_{2,2}$	17	-	-	-	-	-	-	-
	$O_{2,3}$	-	1	-	13	-	-	-	-
	$O_{2,4}$	12	10	14	-	10	-	-	5
	$O_{2,5}$	9	18	13	11	-	18	-	-
	$O_{2,6}$	-	-	-	-	-	15	13	-
	$O_{2,7}$	-	-	-	5	19	-	15	11
	$O_{2,8}$	-	1	19	-	7	-	2	-
	$O_{2,9}$	7	-	-	11	-	13	-	3
	$O_{2,10}$	-	-	-	-	-	2	15	19
J_3	$O_{3,1}$	-	-	3	-	5	-	-	-
	$O_{3,2}$	-	1	19	-	7	-	2	-
	$O_{3,3}$	-	-	18	5	-	-	-	-
	$O_{3,4}$	-	-	-	-	6	3	-	-
	$O_{3,5}$	7	-	-	11	-	13	-	3
	$O_{3,6}$	-	-	-	-	-	2	15	19
	$O_{3,7}$	9	18	13	11	-	18	-	-
	$O_{3,8}$	-	-	-	5	2	-	-	18
	$O_{3,9}$	17	-	-	-	-	-	-	-
	$O_{3,10}$	-	1	-	13	-	-	-	-
J_4	$O_{4,1}$	-	-	18	5	-	-	-	-
	$O_{4,2}$	-	-	3	-	5	-	-	-
	$O_{4,3}$	9	18	13	11	-	18	-	-
	$O_{4,4}$	7	-	-	11	-	13	-	3
	$O_{4,5}$	-	-	-	-	-	15	13	-
	$O_{4,6}$	-	1	19	-	7	-	2	-
	$O_{4,7}$	-	-	-	-	2	-	-	-
	$O_{4,8}$	-	-	16	-	-	11	3	18
	$O_{4,9}$	17	-	-	-	-	-	-	-
	$O_{4,10}$	-	-	-	-	6	3	-	-
J_5									

Tabela A.12: Problema Mk3 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{5,1}$	-	-	-	-	-	15	13	-
	$O_{5,2}$	-	-	-	-	-	2	15	19
	$O_{5,3}$	-	-	-	-	2	-	-	-
	$O_{5,4}$	-	-	-	5	19	-	15	11
	$O_{5,5}$	9	18	13	11	-	18	-	-
	$O_{5,6}$	-	1	19	-	7	-	2	-
	$O_{5,7}$	-	-	-	5	2	-	-	18
	$O_{5,8}$	-	-	-	-	6	3	-	-
	$O_{5,9}$	-	-	3	-	5	-	-	-
	$O_{5,10}$	12	10	14	-	10	-	-	5
J_6	$O_{6,1}$	-	1	-	13	-	-	-	-
	$O_{6,2}$	-	-	-	-	-	15	13	-
	$O_{6,3}$	-	-	18	5	-	-	-	-
	$O_{6,4}$	-	-	16	-	-	11	3	18
	$O_{6,5}$	9	18	13	11	-	18	-	-
	$O_{6,6}$	12	10	14	-	10	-	-	5
	$O_{6,7}$	7	-	-	11	-	13	-	3
	$O_{6,8}$	-	-	-	5	19	-	15	11
	$O_{6,9}$	-	-	-	-	6	3	-	-
	$O_{6,10}$	-	-	3	-	5	-	-	-
J_7	$O_{7,1}$	12	10	14	-	10	-	-	5
	$O_{7,2}$	7	-	-	11	-	13	-	3
	$O_{7,3}$	-	1	-	13	-	-	-	-
	$O_{7,4}$	17	-	-	-	-	-	-	-
	$O_{7,5}$	-	-	-	-	-	15	13	-
	$O_{7,6}$	-	1	19	-	7	-	2	-
	$O_{7,7}$	-	-	-	-	2	-	-	-
	$O_{7,8}$	9	18	13	11	-	18	-	-
	$O_{7,9}$	-	-	18	5	-	-	-	-
	$O_{7,10}$	-	-	-	-	-	2	15	19
J_8	$O_{8,1}$	-	-	-	-	-	2	15	19
	$O_{8,2}$	17	-	-	-	-	-	-	-
	$O_{8,3}$	-	-	-	5	19	-	15	11
	$O_{8,4}$	-	-	-	-	-	15	13	-
	$O_{8,5}$	12	10	14	-	10	-	-	5
	$O_{8,6}$	7	-	-	11	-	13	-	3
	$O_{8,7}$	9	18	13	11	-	18	-	-
	$O_{8,8}$	-	1	-	13	-	-	-	-
	$O_{8,9}$	-	-	18	5	-	-	-	-
	$O_{8,10}$	-	-	3	-	5	-	-	-
J_9									

Tabela A.13: Problema Mk3 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{9,1}$	17	-	-	-	-	-	-	-
	$O_{9,2}$	12	10	14	-	10	-	-	5
	$O_{9,3}$	-	-	16	-	-	11	3	18
	$O_{9,4}$	-	-	-	-	-	2	15	19
	$O_{9,5}$	-	-	-	-	-	15	13	-
	$O_{9,6}$	7	-	-	11	-	13	-	3
	$O_{9,7}$	-	-	-	-	2	-	-	-
	$O_{9,8}$	-	1	-	13	-	-	-	-
	$O_{9,9}$	9	18	13	11	-	18	-	-
	$O_{9,10}$	-	-	-	5	19	-	15	11
J_{10}	$O_{10,1}$	17	-	-	-	-	-	-	-
	$O_{10,2}$	-	-	-	-	-	15	13	-
	$O_{10,3}$	-	-	-	5	2	-	-	18
	$O_{10,4}$	9	18	13	11	-	18	-	-
	$O_{10,5}$	7	-	-	11	-	13	-	3
	$O_{10,6}$	-	-	18	5	-	-	-	-
	$O_{10,7}$	-	-	-	-	6	3	-	-
	$O_{10,8}$	-	-	-	-	-	2	15	19
	$O_{10,9}$	-	-	16	-	-	11	3	18
	$O_{10,10}$	12	10	14	-	10	-	-	5
J_{11}	$O_{11,1}$	-	1	-	13	-	-	-	-
	$O_{11,2}$	-	-	-	-	-	2	15	19
	$O_{11,3}$	-	-	16	-	-	11	3	18
	$O_{11,4}$	-	-	18	5	-	-	-	-
	$O_{11,5}$	-	-	-	-	6	3	-	-
	$O_{11,6}$	17	-	-	-	-	-	-	-
	$O_{11,7}$	-	-	3	-	5	-	-	-
	$O_{11,8}$	-	-	-	5	2	-	-	18
	$O_{11,9}$	12	10	14	-	10	-	-	5
	$O_{11,10}$	9	18	13	11	-	18	-	-
J_{12}	$O_{12,1}$	7	-	-	11	-	13	-	3
	$O_{12,2}$	-	-	-	5	2	-	-	18
	$O_{12,3}$	-	-	16	-	-	11	3	18
	$O_{12,4}$	17	-	-	-	-	-	-	-
	$O_{12,5}$	9	18	13	11	-	18	-	-
	$O_{12,6}$	-	-	-	-	-	2	15	19
	$O_{12,7}$	-	-	-	-	2	-	-	-
	$O_{12,8}$	-	-	3	-	5	-	-	-
	$O_{12,9}$	-	-	-	5	19	-	15	11
	$O_{12,10}$	-	1	-	13	-	-	-	-
J_{13}									

Tabela A.14: Problema Mk3 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{13,1}$	12	10	14	-	10	-	-	5
	$O_{13,2}$	-	-	-	-	2	-	-	-
	$O_{13,3}$	-	-	18	5	-	-	-	-
	$O_{13,4}$	-	1	19	-	7	-	2	-
	$O_{13,5}$	-	-	-	-	-	15	13	-
	$O_{13,6}$	-	-	16	-	-	11	3	18
	$O_{13,7}$	-	-	-	5	19	-	15	11
	$O_{13,8}$	9	18	13	11	-	18	-	-
	$O_{13,9}$	-	-	-	-	6	3	-	-
	$O_{13,10}$	7	-	-	11	-	13	-	3
J_{14}	$O_{14,1}$	-	-	16	-	-	11	3	18
	$O_{14,2}$	-	-	-	5	2	-	-	18
	$O_{14,3}$	-	1	-	13	-	-	-	-
	$O_{14,4}$	-	1	19	-	7	-	2	-
	$O_{14,5}$	-	-	-	-	6	3	-	-
	$O_{14,6}$	-	-	18	5	-	-	-	-
	$O_{14,7}$	-	-	-	-	-	15	13	-
	$O_{14,8}$	-	-	-	-	2	-	-	-
	$O_{14,9}$	9	18	13	11	-	18	-	-
	$O_{14,10}$	17	-	-	-	-	-	-	-
J_{15}	$O_{15,1}$	12	10	14	-	10	-	-	5
	$O_{15,2}$	-	-	-	-	6	3	-	-
	$O_{15,3}$	-	-	-	-	-	15	13	-
	$O_{15,4}$	-	-	-	5	19	-	15	11
	$O_{15,5}$	-	-	16	-	-	11	3	18
	$O_{15,6}$	17	-	-	-	-	-	-	-
	$O_{15,7}$	9	18	13	11	-	18	-	-
	$O_{15,8}$	-	-	-	5	2	-	-	18
	$O_{15,9}$	-	-	18	5	-	-	-	-
	$O_{15,10}$	-	1	19	-	7	-	2	-

Tabela A.15: Problema Mk4 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	6	-	-	-	-	-	-	-
	$O_{1,2}$	6	-	-	-	-	-	9	-
	$O_{1,3}$	-	-	1	-	-	7	-	-
	$O_{1,4}$	-	-	-	2	-	-	5	-
	$O_{1,5}$	8	-	9	-	-	-	-	9
	$O_{1,6}$	-	3	2	8	-	-	-	-
	$O_{1,7}$	-	-	-	-	5	7	-	-
	$O_{1,8}$	-	-	-	7	-	1	-	-
J_2	$O_{2,1}$	-	-	-	-	-	1	-	-
	$O_{2,2}$	-	-	-	7	-	1	-	-
	$O_{2,3}$	6	-	-	-	-	-	-	-
	$O_{2,4}$	-	-	1	-	-	7	-	-
	$O_{2,5}$	-	3	2	8	-	-	-	-
	$O_{2,6}$	-	-	-	-	-	2	-	-
	$O_{2,7}$	-	-	-	-	-	-	2	-
J_3	$O_{3,1}$	-	-	-	-	-	1	-	-
	$O_{3,2}$	-	3	2	8	-	-	-	-
	$O_{3,3}$	-	-	2	4	-	-	1	-
	$O_{3,4}$	-	-	-	2	-	-	5	-
	$O_{3,5}$	7	-	7	-	-	-	-	-
	$O_{3,6}$	-	-	1	4	-	-	-	-
J_4	$O_{4,1}$	-	-	-	-	-	-	2	-
	$O_{4,2}$	6	-	-	-	-	-	-	-
	$O_{4,3}$	6	-	-	-	-	-	9	-
	$O_{4,4}$	-	-	1	-	-	7	-	-
	$O_{4,5}$	-	-	-	5	7	-	-	-
J_5	$O_{5,1}$	-	-	-	-	-	-	2	-
	$O_{5,2}$	6	-	-	-	-	-	9	-
	$O_{5,3}$	-	-	1	4	-	-	-	-
	$O_{5,4}$	8	-	9	-	-	-	-	9
	$O_{5,5}$	7	-	7	-	-	-	-	-
	$O_{5,6}$	-	3	2	8	-	-	-	-
	$O_{5,7}$	-	-	-	5	7	-	-	-
J_6	$O_{6,1}$	-	-	-	-	-	2	-	-
	$O_{6,2}$	-	-	1	4	-	-	-	-
	$O_{6,3}$	-	-	2	4	-	-	1	-
	$O_{6,4}$	-	-	-	7	-	1	-	-
	$O_{6,5}$	-	-	-	5	7	-	-	-
	$O_{6,6}$	8	-	9	-	-	-	-	9
	$O_{6,7}$	7	-	7	-	-	-	-	-

Tabela A.16: Problema Mk4 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{6,8}$	-	-	-	-	-	1	-	-
	$O_{6,9}$	6	-	-	-	-	-	9	-
J_7	$O_{7,1}$	-	-	-	-	5	7	-	-
	$O_{7,2}$	7	-	7	-	-	-	-	-
	$O_{7,3}$	-	-	-	7	-	1	-	-
	$O_{7,4}$	-	-	-	-	-	2	-	-
	$O_{7,5}$	-	-	1	-	-	7	-	-
J_8	$O_{8,1}$	-	-	-	5	7	-	-	-
	$O_{8,2}$	-	-	-	-	5	7	-	-
	$O_{8,3}$	-	3	2	8	-	-	-	-
	$O_{8,4}$	-	-	-	-	-	2	-	-
	$O_{8,5}$	-	-	-	-	-	1	-	-
	$O_{8,6}$	6	-	-	-	-	-	9	-
J_9	$O_{9,1}$	6	-	-	-	-	-	-	-
	$O_{9,2}$	6	-	-	-	-	-	9	-
	$O_{9,3}$	-	-	1	4	-	-	-	-
	$O_{9,4}$	8	-	9	-	-	-	-	9
	$O_{9,5}$	-	-	-	2	-	-	5	-
	$O_{9,6}$	-	-	-	7	-	1	-	-
	$O_{9,7}$	-	-	-	-	-	-	2	-
	$O_{9,8}$	7	-	7	-	-	-	-	-
	$O_{9,9}$	-	3	2	8	-	-	-	-
J_{10}	$O_{10,1}$	-	-	-	-	5	7	-	-
	$O_{10,2}$	6	-	-	-	-	-	-	-
	$O_{10,3}$	-	-	-	-	-	-	2	-
	$O_{10,4}$	-	-	-	5	7	-	-	-
	$O_{10,5}$	6	-	-	-	-	-	9	-
J_{11}	$O_{11,1}$	8	-	9	-	-	-	-	9
	$O_{11,2}$	6	-	-	-	-	-	-	-
	$O_{11,3}$	-	3	2	8	-	-	-	-
	$O_{11,4}$	-	-	-	2	-	-	5	-
J_{12}	$O_{12,1}$	-	-	-	2	-	-	5	-
	$O_{12,2}$	-	-	-	-	-	1	-	-
	$O_{12,3}$	6	-	-	-	-	-	-	-
	$O_{12,4}$	7	-	7	-	-	-	-	-
	$O_{12,5}$	8	-	9	-	-	-	-	9
	$O_{12,6}$	-	-	-	-	-	-	2	-
J_{13}	$O_{13,1}$	-	-	-	-	-	2	-	-
	$O_{13,2}$	-	-	1	-	-	7	-	-
	$O_{13,3}$	-	-	-	7	-	1	-	-

Tabela A.17: Problema Mk4 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{13,4}$	-	-	-	-	5	7	-	-
J_{14}	$O_{14,1}$	-	-	-	-	5	7	-	-
	$O_{14,2}$	-	-	-	-	-	1	-	-
	$O_{14,3}$	-	-	-	2	-	-	5	-
J_{15}	$O_{15,1}$	-	-	-	5	7	-	-	-
	$O_{15,2}$	-	-	-	-	-	-	2	-
	$O_{15,3}$	8	-	9	-	-	-	-	9
	$O_{15,4}$	-	3	2	8	-	-	-	-
	$O_{15,5}$	-	-	2	4	-	-	1	-
	$O_{15,6}$	6	-	-	-	-	-	-	-

Tabela A.18: Problema Mk5 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	-	7	5	-
	$O_{1,2}$	8	-	-	8
	$O_{1,3}$	6	5	-	-
	$O_{1,4}$	-	-	7	-
	$O_{1,5}$	-	6	-	5
	$O_{1,6}$	5	-	-	5
J_2	$O_{2,1}$	-	-	7	-
	$O_{2,2}$	6	5	-	-
	$O_{2,3}$	-	-	-	6
	$O_{2,4}$	-	6	-	5
	$O_{2,5}$	8	6	-	-
J_3	$O_{3,1}$	-	-	9	7
	$O_{3,2}$	-	7	5	-
	$O_{3,3}$	5	-	-	5
	$O_{3,4}$	8	-	-	8
	$O_{3,5}$	6	5	-	-
	$O_{3,6}$	-	-	-	6
	$O_{3,7}$	8	6	-	-
	$O_{3,8}$	-	-	6	9
J_4	$O_{4,1}$	5	-	-	5
	$O_{4,2}$	-	-	9	7
	$O_{4,3}$	8	-	-	8
	$O_{4,4}$	-	-	-	8
	$O_{4,5}$	8	6	-	-
	$O_{4,6}$	-	6	-	5
	$O_{4,7}$	-	-	-	6
J_5	$O_{5,1}$	5	-	7	-
	$O_{5,2}$	-	7	-	6
	$O_{5,3}$	-	-	9	7
	$O_{5,4}$	-	-	8	-
	$O_{5,5}$	-	7	5	-
	$O_{5,6}$	8	6	-	-
J_6	$O_{6,1}$	-	-	-	6
	$O_{6,2}$	-	6	-	5
	$O_{6,3}$	-	-	8	-
	$O_{6,4}$	5	-	7	-
	$O_{6,5}$	-	7	-	6
	$O_{6,6}$	-	-	-	8
	$O_{6,7}$	8	6	-	-
	$O_{6,8}$	8	-	-	8

Tabela A.19: Problema Mk5 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4
	$O_{6,9}$	5	-	-	5
J_7	$O_{7,1}$	-	-	8	-
	$O_{7,2}$	-	-	9	7
	$O_{7,3}$	6	5	-	-
	$O_{7,4}$	-	7	-	6
	$O_{7,5}$	-	-	7	-
J_8	$O_{8,1}$	5	-	7	-
	$O_{8,2}$	-	-	8	-
	$O_{8,3}$	-	-	9	7
	$O_{8,4}$	5	-	-	5
	$O_{8,5}$	-	-	7	-
	$O_{8,6}$	-	-	-	8
	$O_{8,7}$	-	-	6	9
	$O_{8,8}$	6	5	-	-
J_9	$O_{9,1}$	-	7	5	-
	$O_{9,2}$	-	-	-	8
	$O_{9,3}$	-	6	-	5
	$O_{9,4}$	6	5	-	-
	$O_{9,5}$	-	-	-	6
	$O_{9,6}$	8	-	-	9
	$O_{9,7}$	8	-	-	8
	$O_{9,8}$	8	6	-	-
	$O_{9,9}$	-	-	7	-
J_{10}	$O_{10,1}$	8	6	-	-
	$O_{10,2}$	8	-	-	8
	$O_{10,3}$	8	-	-	9
	$O_{10,4}$	-	-	6	9
	$O_{10,5}$	6	5	-	-
	$O_{10,6}$	-	-	8	-
	$O_{10,7}$	-	-	7	-
	$O_{10,8}$	-	-	-	6
	$O_{10,9}$	-	6	-	5
J_{11}	$O_{11,1}$	8	6	-	-
	$O_{11,2}$	8	-	-	8
	$O_{11,3}$	6	5	-	-
	$O_{11,4}$	-	-	7	-
	$O_{11,5}$	-	-	-	6
	$O_{11,6}$	-	-	8	-
	$O_{11,7}$	-	-	6	9
J_{12}	$O_{12,1}$	-	-	-	8

Tabela A.20: Problema Mk5 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4
	$O_{12,2}$	-	-	7	-
	$O_{12,3}$	-	-	9	7
	$O_{12,4}$	6	5	-	-
	$O_{12,5}$	-	-	8	-
	$O_{12,6}$	8	-	-	8
J_{13}	$O_{13,1}$	-	-	-	8
	$O_{13,2}$	-	-	6	9
	$O_{13,3}$	8	-	-	8
	$O_{13,4}$	-	7	-	6
	$O_{13,5}$	-	7	-	6
	$O_{13,6}$	8	6	-	-
	$O_{13,7}$	5	-	7	-
J_{14}	$O_{14,1}$	6	5	-	-
	$O_{14,2}$	5	-	7	-
	$O_{14,3}$	8	-	-	8
	$O_{14,4}$	8	6	-	-
	$O_{14,5}$	5	-	-	5
	$O_{14,6}$	-	7	-	6
	$O_{14,7}$	-	-	-	6
J_{15}	$O_{15,1}$	-	-	8	-
	$O_{15,2}$	8	-	-	9
	$O_{15,3}$	-	-	6	9
	$O_{15,4}$	-	-	7	-
	$O_{15,5}$	-	6	-	5
	$O_{15,6}$	8	6	-	-
	$O_{15,7}$	6	5	-	-

Tabela A.21: Problema Mk6 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
J_1	$O_{1,1}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{1,2}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{1,3}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{1,4}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{1,5}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{1,6}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{1,7}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{1,8}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{1,9}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{1,10}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{1,11}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{1,12}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{1,13}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{1,14}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{1,15}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
J_2	$O_{2,1}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{2,2}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{2,3}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{2,4}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{2,5}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{2,6}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{2,7}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{2,8}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{2,9}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{2,10}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{2,11}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{2,12}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{2,13}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{2,14}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{2,15}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
J_3	$O_{3,1}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{3,2}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{3,3}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{3,4}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{3,5}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{3,6}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{3,7}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{3,8}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{3,9}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{3,10}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-

Tabela A.22: Problema Mk6 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{3,11}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{3,12}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{3,13}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{3,14}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{3,15}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
J_4	$O_{4,1}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{4,2}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{4,3}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{4,4}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{4,5}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{4,6}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{4,7}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{4,8}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{4,9}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{4,10}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{4,11}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{4,12}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{4,13}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{4,14}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{4,15}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
J_5	$O_{5,1}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{5,2}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{5,3}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{5,4}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{5,5}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{5,6}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{5,7}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{5,8}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{5,9}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{5,10}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{5,11}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{5,12}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{5,13}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{5,14}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{5,15}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
J_6	$O_{6,1}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{6,2}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{6,3}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{6,4}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{6,5}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-

Tabela A.23: Problema Mk6 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{6,6}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{6,7}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{6,8}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{6,9}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{6,10}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{6,11}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{6,12}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{6,13}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{6,14}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{6,15}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
J_7	$O_{7,1}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{7,2}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{7,3}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{7,4}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{7,5}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{7,6}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{7,7}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{7,8}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{7,9}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{7,10}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{7,11}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{7,12}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{7,13}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{7,14}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{7,15}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
J_8	$O_{8,1}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{8,2}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{8,3}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{8,4}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{8,5}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{8,6}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{8,7}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{8,8}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{8,9}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{8,10}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{8,11}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{8,12}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{8,13}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{8,14}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{8,15}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
J_9																

Tabela A.24: Problema Mk6 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{9,1}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{9,2}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{9,3}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{9,4}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{9,5}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{9,6}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{9,7}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-
	$O_{9,8}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{9,9}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{9,10}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{9,11}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{9,12}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{9,13}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{9,14}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{9,15}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
J_{10}	$O_{10,1}$	1	-	-	-	-	-	-	2	-	-	-	-	-	-	-
	$O_{10,2}$	-	8	-	-	-	3	2	-	5	-	-	-	-	-	-
	$O_{10,3}$	-	-	-	5	-	-	-	9	-	2	-	-	-	-	-
	$O_{10,4}$	-	-	8	-	8	-	5	-	-	-	-	-	-	-	-
	$O_{10,5}$	-	-	7	-	-	-	5	-	3	-	-	-	-	-	-
	$O_{10,6}$	2	-	-	-	-	-	-	-	-	6	-	-	-	-	-
	$O_{10,7}$	-	-	-	-	6	-	9	-	-	-	-	-	-	-	-
	$O_{10,8}$	1	-	-	-	8	-	-	-	4	-	-	-	-	-	-
	$O_{10,9}$	4	7	-	-	-	-	4	-	1	4	-	-	-	-	-
	$O_{10,10}$	9	5	-	6	7	-	-	-	1	-	-	-	-	-	-
	$O_{10,11}$	3	5	8	-	-	-	-	8	-	9	-	-	-	-	-
	$O_{10,12}$	1	-	-	-	6	2	-	-	-	-	-	-	-	-	-
	$O_{10,13}$	8	1	7	2	8	-	-	-	-	-	-	-	-	-	-
	$O_{10,14}$	-	-	-	8	-	-	2	-	-	-	-	-	-	-	-
	$O_{10,15}$	2	-	-	-	-	-	-	-	7	-	-	-	-	-	-

Tabela A.25: Problema Mk7 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	15	4	-	-	-
	$O_{1,2}$	15	-	18	-	-
	$O_{1,3}$	-	4	-	-	-
	$O_{1,4}$	-	-	-	18	-
	$O_{1,5}$	7	7	8	5	2
J_2	$O_{2,1}$	3	-	-	-	13
	$O_{2,2}$	7	7	8	5	2
	$O_{2,3}$	15	4	-	-	-
	$O_{2,4}$	8	5	-	-	1
	$O_{2,5}$	3	-	2	-	13
J_3	$O_{3,1}$	9	18	3	19	1
	$O_{3,2}$	-	-	-	18	-
	$O_{3,3}$	-	-	9	11	-
	$O_{3,4}$	-	4	-	-	-
	$O_{3,5}$	-	-	14	19	12
J_4	$O_{4,1}$	15	4	-	-	-
	$O_{4,2}$	-	17	10	10	8
	$O_{4,3}$	5	2	13	-	18
	$O_{4,4}$	2	16	9	10	15
	$O_{4,5}$	6	-	15	-	-
J_5	$O_{5,1}$	3	-	2	-	13
	$O_{5,2}$	15	-	18	-	-
	$O_{5,3}$	9	18	3	19	1
	$O_{5,4}$	-	-	14	19	12
	$O_{5,5}$	-	-	-	5	-
J_6	$O_{6,1}$	7	7	8	5	2
	$O_{6,2}$	15	-	18	-	-
	$O_{6,3}$	15	-	-	-	7
	$O_{6,4}$	17	7	-	-	-
	$O_{6,5}$	15	4	-	-	-
J_7	$O_{7,1}$	-	-	-	5	-
	$O_{7,2}$	15	-	-	-	7
	$O_{7,3}$	15	4	-	-	-
	$O_{7,4}$	3	-	2	-	13
	$O_{7,5}$	-	17	15	6	7
J_8	$O_{8,1}$	-	17	15	6	7
	$O_{8,2}$	2	-	18	15	-
	$O_{8,3}$	-	14	19	14	15
	$O_{8,4}$	-	4	-	-	-
	$O_{8,5}$	17	7	-	-	-
J_9						

Tabela A.26: Problema Mk7 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5
	$O_{9,1}$	9	18	3	19	1
	$O_{9,2}$	-	17	15	6	7
	$O_{9,3}$	8	5	-	-	1
	$O_{9,4}$	-	14	19	14	15
	$O_{9,5}$	17	-	-	-	15
J_{10}	$O_{10,1}$	15	-	-	-	7
	$O_{10,2}$	-	17	10	10	8
	$O_{10,3}$	6	-	15	-	-
	$O_{10,4}$	-	-	-	5	-
	$O_{10,5}$	7	10	16	10	17
J_{11}	$O_{11,1}$	-	-	-	18	-
	$O_{11,2}$	8	5	-	-	1
	$O_{11,3}$	7	7	8	5	2
	$O_{11,4}$	15	-	-	-	7
	$O_{11,5}$	17	-	-	-	15
J_{12}	$O_{12,1}$	-	-	14	19	12
	$O_{12,2}$	-	17	10	10	8
	$O_{12,3}$	6	-	15	-	-
	$O_{12,4}$	7	7	8	5	2
	$O_{12,5}$	7	10	16	10	17
J_{13}	$O_{13,1}$	17	-	-	-	15
	$O_{13,2}$	-	-	-	18	-
	$O_{13,3}$	-	17	12	5	19
	$O_{13,4}$	2	-	18	15	-
	$O_{13,5}$	8	5	-	-	1
J_{14}	$O_{14,1}$	-	-	5	-	1
	$O_{14,2}$	2	-	18	15	-
	$O_{14,3}$	-	17	10	10	8
	$O_{14,4}$	15	-	18	-	-
	$O_{14,5}$	7	7	8	5	2
J_{15}	$O_{15,1}$	7	7	8	5	2
	$O_{15,2}$	-	-	5	-	1
	$O_{15,3}$	-	-	14	19	12
	$O_{15,4}$	7	10	16	10	17
	$O_{15,5}$	17	-	-	-	15
J_{16}	$O_{16,1}$	2	16	9	10	15
	$O_{16,2}$	-	-	9	11	-
	$O_{16,3}$	-	4	-	-	-
	$O_{16,4}$	15	-	-	-	7
	$O_{16,5}$	-	-	-	5	-
J_{17}						

Tabela A.27: Problema Mk7 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5
	$O_{17,1}$	7	7	8	5	2
	$O_{17,2}$	-	14	19	14	15
	$O_{17,3}$	2	-	18	15	-
	$O_{17,4}$	6	-	15	-	-
	$O_{17,5}$	9	18	3	19	1
J_{18}	$O_{18,1}$	-	4	-	-	-
	$O_{18,2}$	8	5	-	-	1
	$O_{18,3}$	-	-	5	-	1
	$O_{18,4}$	15	-	18	-	-
	$O_{18,5}$	15	-	-	-	7
J_{19}	$O_{19,1}$	3	-	2	-	13
	$O_{19,2}$	-	17	15	6	7
	$O_{19,3}$	5	2	13	-	18
	$O_{19,4}$	-	-	-	18	-
	$O_{19,5}$	3	-	-	-	13
J_{20}	$O_{20,1}$	-	-	-	5	-
	$O_{20,2}$	15	4	-	-	-
	$O_{20,3}$	-	-	-	18	-
	$O_{20,4}$	15	-	-	-	7
	$O_{20,5}$	2	16	9	10	15

Tabela A.28: Problema Mk8 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	-	-	-	5	-	-	18	-	-	-
	$O_{1,2}$	-	-	-	-	7	-	7	-	-	-
	$O_{1,3}$	-	-	19	-	-	-	-	-	-	-
	$O_{1,4}$	-	-	-	-	-	-	14	-	-	-
	$O_{1,5}$	-	-	-	5	-	-	-	-	-	12
	$O_{1,6}$	10	-	-	-	-	-	-	-	-	-
	$O_{1,7}$	-	-	-	-	-	-	-	-	-	18
	$O_{1,8}$	-	-	-	-	-	-	10	19	-	-
	$O_{1,9}$	-	-	11	-	-	-	-	9	-	-
	$O_{1,10}$	-	-	5	-	-	-	-	12	-	-
J_2	$O_{2,1}$	-	5	-	-	-	-	-	-	-	-
	$O_{2,2}$	-	-	-	5	-	-	18	-	-	-
	$O_{2,3}$	-	-	5	-	-	-	-	12	-	-
	$O_{2,4}$	10	-	-	-	-	-	-	-	-	-
	$O_{2,5}$	-	-	-	-	-	-	-	-	-	19
	$O_{2,6}$	-	-	15	19	-	-	-	-	-	-
	$O_{2,7}$	-	-	-	-	-	-	14	-	-	-
	$O_{2,8}$	-	-	-	-	9	-	-	-	-	-
	$O_{2,9}$	-	-	-	-	14	-	-	-	5	-
	$O_{2,10}$	19	-	-	-	-	-	-	-	-	-
	$O_{2,11}$	-	-	-	-	-	-	10	19	-	-
	$O_{2,12}$	16	-	-	-	-	-	-	-	-	-
J_3	$O_{3,1}$	-	-	-	-	14	-	-	-	5	-
	$O_{3,2}$	19	-	-	-	-	-	-	-	-	-
	$O_{3,3}$	10	-	-	-	-	-	-	-	-	-
	$O_{3,4}$	-	-	19	-	-	-	-	-	-	-
	$O_{3,5}$	-	-	-	5	-	-	18	-	-	-
	$O_{3,6}$	-	-	-	5	-	-	-	-	-	12
	$O_{3,7}$	-	-	5	-	-	-	-	12	-	-
	$O_{3,8}$	-	-	-	-	-	-	-	-	-	10
	$O_{3,9}$	-	-	-	-	9	-	-	-	-	-
	$O_{3,10}$	7	-	-	-	-	-	-	-	-	-
	$O_{3,11}$	-	-	-	-	-	-	10	19	-	-
	$O_{3,12}$	10	-	-	-	-	-	-	-	-	-
	$O_{3,13}$	-	-	-	-	-	-	-	-	-	19
	$O_{3,14}$	-	-	-	-	-	-	-	-	-	18
J_4	$O_{4,1}$	-	-	-	-	-	-	-	-	-	10
	$O_{4,2}$	-	-	-	-	7	-	7	-	-	-
	$O_{4,3}$	-	-	-	-	-	-	14	-	-	-
	$O_{4,4}$	10	-	-	-	-	-	-	-	-	-

Tabela A.29: Problema Mk8 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{4,5}$	-	-	-	-	-	-	-	-	-	18
	$O_{4,6}$	-	-	15	-	-	-	13	-	-	-
	$O_{4,7}$	-	-	-	-	7	-	-	-	-	14
	$O_{4,8}$	-	-	11	-	-	-	-	9	-	-
	$O_{4,9}$	-	-	-	-	-	-	-	-	11	-
	$O_{4,10}$	-	-	-	-	9	-	-	-	-	-
J_5	$O_{5,1}$	-	-	-	-	9	-	-	-	-	-
	$O_{5,2}$	-	-	-	-	14	-	-	-	5	-
	$O_{5,3}$	-	-	-	5	-	-	18	-	-	-
	$O_{5,4}$	-	-	11	-	-	-	-	9	-	-
	$O_{5,5}$	10	-	-	-	-	-	-	-	-	-
	$O_{5,6}$	-	-	-	-	-	-	-	-	11	-
	$O_{5,7}$	7	-	-	-	-	-	-	-	-	-
	$O_{5,8}$	-	-	-	-	-	-	14	-	-	-
	$O_{5,9}$	-	-	-	5	-	-	-	-	-	12
	$O_{5,10}$	-	-	15	19	-	-	-	-	-	-
	$O_{5,11}$	-	-	-	-	-	-	-	18	-	-
	$O_{5,12}$	-	-	-	-	-	-	-	-	-	19
J_6	$O_{6,1}$	-	-	15	-	-	-	13	-	-	-
	$O_{6,2}$	-	-	19	-	-	-	-	-	-	-
	$O_{6,3}$	-	-	-	-	9	-	-	-	-	-
	$O_{6,4}$	-	-	-	-	-	-	-	-	-	19
	$O_{6,5}$	-	-	5	-	-	-	-	12	-	-
	$O_{6,6}$	-	-	-	5	-	-	18	-	-	-
	$O_{6,7}$	-	-	-	-	-	-	-	14	-	9
	$O_{6,8}$	-	-	-	5	-	-	-	-	-	12
	$O_{6,9}$	-	-	-	-	-	-	-	-	-	18
	$O_{6,10}$	7	-	-	-	-	-	-	-	-	-
J_7	$O_{7,1}$	10	-	-	-	-	-	-	-	-	-
	$O_{7,2}$	-	-	-	-	-	-	-	-	-	18
	$O_{7,3}$	7	-	-	-	-	-	-	-	-	-
	$O_{7,4}$	-	-	-	-	9	-	-	-	-	-
	$O_{7,5}$	-	-	-	-	-	-	-	14	-	9
	$O_{7,6}$	-	-	-	-	-	-	10	19	-	-
	$O_{7,7}$	-	-	15	19	-	-	-	-	-	-
	$O_{7,8}$	-	-	-	-	7	-	-	-	-	14
	$O_{7,9}$	-	-	-	-	-	-	-	18	-	-
	$O_{7,10}$	-	-	-	-	-	-	-	-	-	19
	$O_{7,11}$	19	-	-	-	-	-	-	-	-	-
	$O_{7,12}$	10	-	-	-	-	-	-	-	-	-
J_8											

Tabela A.30: Problema Mk8 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{8,1}$	10	-	-	-	-	-	-	-	-	-
	$O_{8,2}$	-	-	-	-	-	-	14	-	-	-
	$O_{8,3}$	10	-	-	-	-	-	-	-	-	-
	$O_{8,4}$	-	-	15	19	-	-	-	-	-	-
	$O_{8,5}$	-	-	-	-	14	-	-	-	5	-
	$O_{8,6}$	-	-	-	5	-	-	18	-	-	-
	$O_{8,7}$	-	-	19	-	-	-	-	-	-	-
	$O_{8,8}$	19	-	-	-	-	-	-	-	-	-
	$O_{8,9}$	-	-	-	5	-	-	-	-	-	12
	$O_{8,10}$	-	-	-	-	9	-	-	-	-	-
	$O_{8,11}$	-	-	-	-	-	-	-	-	-	19
J_9	$O_{9,1}$	-	-	-	-	-	-	10	19	-	-
	$O_{9,2}$	-	-	-	-	-	-	-	14	-	9
	$O_{9,3}$	19	-	-	-	-	-	-	-	-	-
	$O_{9,4}$	-	-	-	-	-	-	-	-	-	19
	$O_{9,5}$	-	-	-	-	7	-	-	-	-	14
	$O_{9,6}$	-	5	-	-	-	-	-	-	-	-
	$O_{9,7}$	-	-	-	5	-	-	-	-	-	12
	$O_{9,8}$	-	-	-	-	7	-	7	-	-	-
	$O_{9,9}$	16	-	-	-	-	-	-	-	-	-
	$O_{9,10}$	7	-	-	-	-	-	-	-	-	-
	$O_{9,11}$	-	-	-	-	-	-	-	-	11	-
	$O_{9,12}$	-	-	19	-	-	-	-	-	-	-
	$O_{9,13}$	10	-	-	-	-	-	-	-	-	-
	$O_{9,14}$	-	-	-	-	-	-	-	-	-	18
J_{10}	$O_{10,1}$	-	-	-	-	-	-	-	-	-	19
	$O_{10,2}$	-	-	-	-	7	-	-	-	-	14
	$O_{10,3}$	-	-	-	-	-	-	-	18	-	-
	$O_{10,4}$	-	-	11	-	-	-	-	9	-	-
	$O_{10,5}$	7	-	-	-	-	-	-	-	-	-
	$O_{10,6}$	10	-	-	-	-	-	-	-	-	-
	$O_{10,7}$	-	-	-	-	14	-	-	-	5	-
	$O_{10,8}$	-	-	15	19	-	-	-	-	-	-
	$O_{10,9}$	-	-	-	-	-	-	-	-	-	18
	$O_{10,10}$	-	-	19	-	-	-	-	-	-	-
	$O_{10,11}$	19	-	-	-	-	-	-	-	-	-
J_{11}	$O_{11,1}$	-	-	-	-	14	-	-	-	5	-
	$O_{11,2}$	10	-	-	-	-	-	-	-	-	-
	$O_{11,3}$	-	-	-	-	-	-	-	18	-	-
	$O_{11,4}$	-	-	15	19	-	-	-	-	-	-

Tabela A.31: Problema Mk8 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{11,5}$	-	-	-	-	-	-	10	19	-	-
	$O_{11,6}$	-	-	5	-	-	-	-	12	-	-
	$O_{11,7}$	-	-	11	-	-	-	-	9	-	-
	$O_{11,8}$	-	-	-	-	-	-	-	14	-	9
	$O_{11,9}$	-	-	-	-	-	-	-	-	-	10
	$O_{11,10}$	-	-	-	-	-	-	-	-	11	-
	$O_{11,11}$	-	-	19	-	-	-	-	-	-	-
J_{12}	$O_{12,1}$	-	-	-	-	-	-	-	-	-	19
	$O_{12,2}$	-	-	11	-	-	-	-	9	-	-
	$O_{12,3}$	-	-	-	-	7	-	7	-	-	-
	$O_{12,4}$	16	-	-	-	-	-	-	-	-	-
	$O_{12,5}$	-	-	-	-	-	-	14	-	-	-
	$O_{12,6}$	-	-	-	5	-	-	18	-	-	-
	$O_{12,7}$	-	-	-	5	-	-	-	-	-	12
	$O_{12,8}$	10	-	-	-	-	-	-	-	-	-
	$O_{12,9}$	-	-	-	-	-	-	-	18	-	-
	$O_{12,10}$	-	-	-	-	14	-	-	-	5	-
J_{13}	$O_{13,1}$	-	-	-	-	7	-	-	-	-	14
	$O_{13,2}$	-	-	-	-	-	-	-	-	-	19
	$O_{13,3}$	-	-	-	-	-	-	10	19	-	-
	$O_{13,4}$	-	-	15	19	-	-	-	-	-	-
	$O_{13,5}$	19	-	-	-	-	-	-	-	-	-
	$O_{13,6}$	-	-	-	-	-	-	-	18	-	-
	$O_{13,7}$	-	-	-	-	-	-	-	14	-	9
	$O_{13,8}$	-	-	11	-	-	-	-	9	-	-
	$O_{13,9}$	-	-	-	-	-	-	-	-	-	18
	$O_{13,10}$	-	-	-	-	14	-	-	-	5	-
	$O_{13,11}$	-	5	-	-	-	-	-	-	-	-
J_{14}	$O_{14,1}$	10	-	-	-	-	-	-	-	-	-
	$O_{14,2}$	-	-	-	-	7	-	7	-	-	-
	$O_{14,3}$	10	-	-	-	-	-	-	-	-	-
	$O_{14,4}$	-	-	-	-	-	-	-	-	11	-
	$O_{14,5}$	-	-	-	-	-	-	14	-	-	-
	$O_{14,6}$	-	-	15	-	-	-	13	-	-	-
	$O_{14,7}$	-	-	-	-	-	-	-	14	-	9
	$O_{14,8}$	16	-	-	-	-	-	-	-	-	-
	$O_{14,9}$	-	-	5	-	-	-	-	12	-	-
	$O_{14,10}$	-	-	-	-	14	-	-	-	5	-
	$O_{14,11}$	-	5	-	-	-	-	-	-	-	-
J_{15}	$O_{15,1}$	-	-	-	-	14	-	-	-	5	-

Tabela A.32: Problema Mk8 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{15,2}$	-	-	-	-	7	-	7	-	-	-
	$O_{15,3}$	-	-	-	-	-	-	14	-	-	-
	$O_{15,4}$	-	-	-	-	-	-	-	-	-	10
	$O_{15,5}$	-	-	-	-	-	-	10	19	-	-
	$O_{15,6}$	-	-	15	19	-	-	-	-	-	-
	$O_{15,7}$	-	-	-	5	-	-	18	-	-	-
	$O_{15,8}$	7	-	-	-	-	-	-	-	-	-
	$O_{15,9}$	-	-	11	-	-	-	-	9	-	-
	$O_{15,10}$	19	-	-	-	-	-	-	-	-	-
	$O_{15,11}$	-	-	-	-	-	-	-	18	-	-
J_{16}	$O_{16,1}$	-	5	-	-	-	-	-	-	-	-
	$O_{16,2}$	-	-	-	-	-	-	10	19	-	-
	$O_{16,3}$	-	-	-	-	-	-	-	-	-	10
	$O_{16,4}$	-	-	-	-	-	-	-	-	11	-
	$O_{16,5}$	-	-	-	-	-	-	-	18	-	-
	$O_{16,6}$	-	-	-	-	7	-	-	-	-	14
	$O_{16,7}$	-	-	-	-	14	-	-	-	5	-
	$O_{16,8}$	10	-	-	-	-	-	-	-	-	-
	$O_{16,9}$	19	-	-	-	-	-	-	-	-	-
	$O_{16,10}$	-	-	15	-	-	-	13	-	-	-
	$O_{16,11}$	-	-	-	-	-	-	-	14	-	9
J_{17}	$O_{17,1}$	-	-	-	-	-	-	-	-	-	10
	$O_{17,2}$	-	-	-	-	14	-	-	-	5	-
	$O_{17,3}$	-	-	-	-	9	-	-	-	-	-
	$O_{17,4}$	-	-	-	-	-	-	-	-	-	19
	$O_{17,5}$	10	-	-	-	-	-	-	-	-	-
	$O_{17,6}$	-	-	5	-	-	-	-	12	-	-
	$O_{17,7}$	-	5	-	-	-	-	-	-	-	-
	$O_{17,8}$	-	-	-	-	7	-	-	-	-	14
	$O_{17,9}$	10	-	-	-	-	-	-	-	-	-
	$O_{17,10}$	-	-	-	-	-	-	-	14	-	9
	$O_{17,11}$	-	-	15	-	-	-	13	-	-	-
	$O_{17,12}$	16	-	-	-	-	-	-	-	-	-
	$O_{17,13}$	-	-	-	-	-	-	14	-	-	-
J_{18}	$O_{18,1}$	-	-	15	-	-	-	13	-	-	-
	$O_{18,2}$	-	5	-	-	-	-	-	-	-	-
	$O_{18,3}$	-	-	-	-	-	-	-	-	-	19
	$O_{18,4}$	-	-	19	-	-	-	-	-	-	-
	$O_{18,5}$	-	-	-	-	-	-	-	18	-	-
	$O_{18,6}$	7	-	-	-	-	-	-	-	-	-

Tabela A.33: Problema Mk8 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{18,7}$	-	-	-	-	9	-	-	-	-	-
	$O_{18,8}$	-	-	-	-	-	-	14	-	-	-
	$O_{18,9}$	-	-	-	5	-	-	18	-	-	-
	$O_{18,10}$	10	-	-	-	-	-	-	-	-	-
	$O_{18,11}$	-	-	-	-	14	-	-	-	5	-
J_{19}	$O_{19,1}$	-	-	-	-	-	-	10	19	-	-
	$O_{19,2}$	-	5	-	-	-	-	-	-	-	-
	$O_{19,3}$	-	-	11	-	-	-	-	9	-	-
	$O_{19,4}$	-	-	-	-	-	-	-	-	11	-
	$O_{19,5}$	-	-	-	5	-	-	-	-	-	12
	$O_{19,6}$	-	-	-	-	-	-	-	-	-	18
	$O_{19,7}$	-	-	-	5	-	-	18	-	-	-
	$O_{19,8}$	-	-	-	-	-	-	-	14	-	9
	$O_{19,9}$	-	-	5	-	-	-	-	12	-	-
	$O_{19,10}$	-	-	-	-	-	-	-	-	-	19
J_{20}	$O_{20,1}$	-	-	-	-	-	-	-	-	-	18
	$O_{20,2}$	-	-	-	-	-	-	-	-	-	10
	$O_{20,3}$	-	-	-	-	-	-	14	-	-	-
	$O_{20,4}$	-	-	-	-	-	-	-	-	11	-
	$O_{20,5}$	-	-	15	-	-	-	13	-	-	-
	$O_{20,6}$	-	5	-	-	-	-	-	-	-	-
	$O_{20,7}$	-	-	-	-	-	-	-	14	-	9
	$O_{20,8}$	-	-	5	-	-	-	-	12	-	-
	$O_{20,9}$	-	-	-	-	9	-	-	-	-	-
	$O_{20,10}$	16	-	-	-	-	-	-	-	-	-

Tabela A.34: Problema Mk9 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	11	10	-	-	-	-	-	-	-	-
	$O_{1,2}$	-	-	-	-	-	-	-	17	-	-
	$O_{1,3}$	-	-	-	-	-	-	-	14	-	-
	$O_{1,4}$	10	-	-	-	-	-	-	-	-	-
	$O_{1,5}$	-	16	-	-	-	-	-	-	-	18
	$O_{1,6}$	-	12	-	-	-	-	-	-	6	-
	$O_{1,7}$	16	-	10	11	-	-	9	-	-	-
	$O_{1,8}$	7	-	-	-	19	-	-	-	-	-
	$O_{1,9}$	-	-	-	-	-	-	-	-	11	-
	$O_{1,10}$	-	-	-	16	-	-	-	-	-	-
	$O_{1,11}$	-	5	-	-	-	-	-	-	-	-
	$O_{1,12}$	-	-	-	6	-	16	9	14	9	-
J_2	$O_{2,1}$	-	-	-	-	-	-	-	17	-	-
	$O_{2,2}$	-	-	-	11	6	-	-	-	-	-
	$O_{2,3}$	11	10	-	-	-	-	-	-	-	-
	$O_{2,4}$	-	-	-	-	9	-	-	8	-	-
	$O_{2,5}$	-	16	11	-	-	-	-	-	-	-
	$O_{2,6}$	8	-	-	-	14	12	-	-	-	15
	$O_{2,7}$	-	8	-	-	-	10	5	15	-	-
	$O_{2,8}$	7	-	-	-	19	-	-	-	-	-
	$O_{2,9}$	16	-	10	11	-	-	9	-	-	-
	$O_{2,10}$	10	-	-	-	-	-	-	-	-	-
	$O_{2,11}$	16	-	11	7	-	-	17	-	-	-
	$O_{2,12}$	-	-	-	16	-	-	-	-	-	-
	$O_{2,13}$	-	-	11	-	8	-	11	-	17	-
J_3	$O_{3,1}$	-	8	-	-	-	10	5	15	-	-
	$O_{3,2}$	-	-	-	-	9	-	-	8	-	-
	$O_{3,3}$	-	16	-	-	-	-	-	-	-	18
	$O_{3,4}$	11	10	-	-	-	-	-	-	-	-
	$O_{3,5}$	-	-	-	6	-	16	9	14	9	-
	$O_{3,6}$	-	-	-	16	-	-	-	-	-	-
	$O_{3,7}$	7	-	-	-	19	-	-	-	-	-
	$O_{3,8}$	10	-	-	-	-	-	-	-	-	-
	$O_{3,9}$	-	-	-	11	6	-	-	-	-	-
	$O_{3,10}$	-	16	11	-	-	-	-	-	-	-
	$O_{3,11}$	-	-	14	-	-	-	-	-	-	-
J_4	$O_{4,1}$	8	-	-	-	14	12	-	-	-	15
	$O_{4,2}$	7	-	-	-	19	-	-	-	-	-
	$O_{4,3}$	6	-	-	11	-	-	-	16	15	-
	$O_{4,4}$	-	-	-	-	-	-	-	14	-	-

Tabela A.35: Problema Mk9 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{4,5}$	-	-	-	16	-	-	-	-	-	-
	$O_{4,6}$	-	-	-	-	-	-	-	17	-	-
	$O_{4,7}$	16	-	11	7	-	-	17	-	-	-
	$O_{4,8}$	-	8	-	-	5	-	-	13	-	6
	$O_{4,9}$	-	-	14	-	-	-	-	-	-	-
	$O_{4,10}$	16	-	10	11	-	-	9	-	-	-
	$O_{4,11}$	10	-	-	-	-	-	-	-	-	-
J_5	$O_{5,1}$	-	-	-	-	-	-	-	17	-	-
	$O_{5,2}$	-	-	-	16	-	-	-	-	-	-
	$O_{5,3}$	-	-	-	-	9	-	-	-	-	-
	$O_{5,4}$	-	8	-	-	5	-	-	13	-	6
	$O_{5,5}$	16	-	11	7	-	-	17	-	-	-
	$O_{5,6}$	-	16	-	-	-	-	-	-	-	18
	$O_{5,7}$	-	8	-	-	-	10	5	15	-	-
	$O_{5,8}$	-	-	-	-	-	-	-	14	-	-
	$O_{5,9}$	-	-	-	11	6	-	-	-	-	-
	$O_{5,10}$	-	5	-	-	11	-	13	-	-	10
	$O_{5,11}$	-	-	-	6	-	16	9	14	9	-
	$O_{5,12}$	-	-	-	-	9	-	-	8	-	-
	$O_{5,13}$	8	-	-	-	14	12	-	-	-	15
	$O_{5,14}$	7	-	-	-	19	-	-	-	-	-
J_6	$O_{6,1}$	-	5	-	-	11	-	13	-	-	10
	$O_{6,2}$	-	16	-	-	-	-	-	-	-	18
	$O_{6,3}$	10	-	-	-	-	-	-	-	-	-
	$O_{6,4}$	-	-	14	-	-	-	-	-	-	-
	$O_{6,5}$	-	-	-	-	9	-	-	-	-	-
	$O_{6,6}$	-	-	-	6	-	16	9	14	9	-
	$O_{6,7}$	-	-	-	-	-	-	-	17	-	-
	$O_{6,8}$	-	-	-	-	-	-	-	14	-	-
	$O_{6,9}$	-	5	-	-	-	-	-	-	-	-
	$O_{6,10}$	-	8	-	-	-	10	5	15	-	-
	$O_{6,11}$	6	-	-	11	-	-	-	16	15	-
J_7	$O_{7,1}$	-	-	-	-	-	-	-	14	-	-
	$O_{7,2}$	-	-	-	-	-	-	-	17	-	-
	$O_{7,3}$	-	-	-	-	9	-	-	8	-	-
	$O_{7,4}$	-	-	-	16	-	-	-	-	-	-
	$O_{7,5}$	10	-	-	-	-	-	-	-	-	-
	$O_{7,6}$	-	5	-	-	11	-	13	-	-	10
	$O_{7,7}$	-	5	-	-	-	-	-	-	-	-
	$O_{7,8}$	-	-	-	11	6	-	-	-	-	-

Tabela A.36: Problema Mk9 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{7,9}$	-	-	-	6	-	16	9	14	9	-
	$O_{7,10}$	6	-	-	11	-	-	-	16	15	-
	$O_{7,11}$	19	8	-	-	-	14	-	13	-	18
	$O_{7,12}$	-	8	-	-	-	10	5	15	-	-
	$O_{7,13}$	16	-	11	7	-	-	17	-	-	-
	$O_{7,14}$	-	16	-	-	-	-	-	-	-	18
J_8	$O_{8,1}$	10	-	-	-	-	-	-	-	-	-
	$O_{8,2}$	-	8	-	-	5	-	-	13	-	6
	$O_{8,3}$	-	-	-	-	9	-	-	-	-	-
	$O_{8,4}$	16	-	10	11	-	-	9	-	-	-
	$O_{8,5}$	-	-	-	-	-	-	-	-	11	-
	$O_{8,6}$	-	5	-	-	11	-	13	-	-	10
	$O_{8,7}$	-	8	-	-	-	10	5	15	-	-
	$O_{8,8}$	-	5	-	-	-	-	-	-	-	-
	$O_{8,9}$	19	8	-	-	-	14	-	13	-	18
	$O_{8,10}$	-	-	-	6	-	16	9	14	9	-
	$O_{8,11}$	11	10	-	-	-	-	-	-	-	-
	$O_{8,12}$	16	-	11	7	-	-	17	-	-	-
	$O_{8,13}$	-	-	-	11	6	-	-	-	-	-
J_9	$O_{9,1}$	-	-	-	-	-	-	-	17	-	-
	$O_{9,2}$	-	5	-	-	-	-	-	-	-	-
	$O_{9,3}$	10	-	-	-	-	-	-	-	-	-
	$O_{9,4}$	-	-	-	16	-	-	-	-	-	-
	$O_{9,5}$	-	-	-	11	6	-	-	-	-	-
	$O_{9,6}$	16	-	10	11	-	-	9	-	-	-
	$O_{9,7}$	19	8	-	-	-	14	-	13	-	18
	$O_{9,8}$	-	-	-	-	-	-	-	-	11	-
	$O_{9,9}$	-	12	-	-	-	-	-	-	6	-
	$O_{9,10}$	11	10	-	-	-	-	-	-	-	-
	$O_{9,11}$	-	-	-	-	9	-	-	8	-	-
J_{10}	$O_{10,1}$	-	-	-	16	-	-	-	-	-	-
	$O_{10,2}$	6	-	-	11	-	-	-	16	15	-
	$O_{10,3}$	-	-	14	-	-	-	-	-	-	-
	$O_{10,4}$	-	5	-	-	11	-	13	-	-	10
	$O_{10,5}$	-	-	-	-	-	-	-	-	11	-
	$O_{10,6}$	-	-	-	6	-	16	9	14	9	-
	$O_{10,7}$	-	-	-	11	6	-	-	-	-	-
	$O_{10,8}$	16	-	11	7	-	-	17	-	-	-
	$O_{10,9}$	11	10	-	-	-	-	-	-	-	-
	$O_{10,10}$	-	16	11	-	-	-	-	-	-	-

Tabela A.37: Problema Mk9 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{10,11}$	8	-	-	-	14	12	-	-	-	15
	$O_{10,12}$	10	-	-	-	-	-	-	-	-	-
J_{11}	$O_{11,1}$	-	-	-	-	-	-	-	-	11	-
	$O_{11,2}$	-	-	-	-	9	-	-	-	-	-
	$O_{11,3}$	19	8	-	-	-	14	-	13	-	18
	$O_{11,4}$	-	-	-	16	-	-	-	-	-	-
	$O_{11,5}$	6	-	-	11	-	-	-	16	15	-
	$O_{11,6}$	-	-	-	-	9	-	-	8	-	-
	$O_{11,7}$	16	-	10	11	-	-	9	-	-	-
	$O_{11,8}$	-	-	14	-	-	-	-	-	-	-
	$O_{11,9}$	10	-	-	-	-	-	-	-	-	-
	$O_{11,10}$	16	-	11	7	-	-	17	-	-	-
J_{12}	$O_{12,1}$	-	8	-	-	5	-	-	13	-	6
	$O_{12,2}$	6	-	-	11	-	-	-	16	15	-
	$O_{12,3}$	-	-	-	16	-	-	-	-	-	-
	$O_{12,4}$	-	12	-	-	-	-	-	-	6	-
	$O_{12,5}$	-	8	-	-	-	10	5	15	-	-
	$O_{12,6}$	16	-	10	11	-	-	9	-	-	-
	$O_{12,7}$	-	5	-	-	-	-	-	-	-	-
	$O_{12,8}$	-	-	-	-	-	-	-	14	-	-
	$O_{12,9}$	-	-	-	6	-	16	9	14	9	-
	$O_{12,10}$	-	-	-	11	6	-	-	-	-	-
	$O_{12,11}$	-	16	-	-	-	-	-	-	-	18
J_{13}	$O_{13,1}$	-	5	-	-	-	-	-	-	-	-
	$O_{13,2}$	-	-	14	-	-	-	-	-	-	-
	$O_{13,3}$	-	12	-	-	-	-	-	-	6	-
	$O_{13,4}$	-	-	-	-	9	-	-	-	-	-
	$O_{13,5}$	-	5	-	-	11	-	13	-	-	10
	$O_{13,6}$	16	-	11	7	-	-	17	-	-	-
	$O_{13,7}$	11	10	-	-	-	-	-	-	-	-
	$O_{13,8}$	-	-	-	-	-	-	-	17	-	-
	$O_{13,9}$	7	-	-	-	19	-	-	-	-	-
	$O_{13,10}$	10	-	-	-	-	-	-	-	-	-
	$O_{13,11}$	16	-	10	11	-	-	9	-	-	-
J_{14}	$O_{14,1}$	-	-	11	-	8	-	11	-	17	-
	$O_{14,2}$	10	-	-	-	-	-	-	-	-	-
	$O_{14,3}$	-	16	-	-	-	-	-	-	-	18
	$O_{14,4}$	11	10	-	-	-	-	-	-	-	-
	$O_{14,5}$	-	8	-	-	-	10	5	15	-	-
	$O_{14,6}$	6	-	-	11	-	-	-	16	15	-

Tabela A.38: Problema Mk9 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{14,7}$	-	-	-	16	-	-	-	-	-	-
	$O_{14,8}$	16	-	11	7	-	-	17	-	-	-
	$O_{14,9}$	16	-	10	11	-	-	9	-	-	-
	$O_{14,10}$	-	16	11	-	-	-	-	-	-	-
J_{15}	$O_{15,1}$	10	-	-	-	-	-	-	-	-	-
	$O_{15,2}$	6	-	-	11	-	-	-	16	15	-
	$O_{15,3}$	-	5	-	-	11	-	13	-	-	10
	$O_{15,4}$	19	8	-	-	-	14	-	13	-	18
	$O_{15,5}$	-	-	-	11	6	-	-	-	-	-
	$O_{15,6}$	-	12	-	-	-	-	-	-	6	-
	$O_{15,7}$	-	5	-	-	-	-	-	-	-	-
	$O_{15,8}$	-	8	-	-	5	-	-	13	-	6
	$O_{15,9}$	-	-	-	16	-	-	-	-	-	-
	$O_{15,10}$	-	16	11	-	-	-	-	-	-	-
	$O_{15,11}$	11	10	-	-	-	-	-	-	-	-
	$O_{15,12}$	-	8	-	-	-	10	5	15	-	-
J_{16}	$O_{16,1}$	-	-	-	-	-	-	-	17	-	-
	$O_{16,2}$	6	-	-	11	-	-	-	16	15	-
	$O_{16,3}$	-	-	14	-	-	-	-	-	-	-
	$O_{16,4}$	-	12	-	-	-	-	-	-	6	-
	$O_{16,5}$	-	-	-	-	-	-	-	14	-	-
	$O_{16,6}$	-	8	-	-	-	10	5	15	-	-
	$O_{16,7}$	16	-	10	11	-	-	9	-	-	-
	$O_{16,8}$	-	5	-	-	11	-	13	-	-	10
	$O_{16,9}$	8	-	-	-	14	12	-	-	-	15
	$O_{16,10}$	11	10	-	-	-	-	-	-	-	-
	$O_{16,11}$	-	-	-	16	-	-	-	-	-	-
	$O_{16,12}$	-	-	11	-	8	-	11	-	17	-
	$O_{16,13}$	7	-	-	-	19	-	-	-	-	-
	$O_{16,14}$	-	8	-	-	5	-	-	13	-	6
J_{17}	$O_{17,1}$	19	8	-	-	-	14	-	13	-	18
	$O_{17,2}$	-	-	-	-	-	-	-	-	11	-
	$O_{17,3}$	16	-	10	11	-	-	9	-	-	-
	$O_{17,4}$	-	-	-	-	-	-	-	17	-	-
	$O_{17,5}$	-	8	-	-	5	-	-	13	-	6
	$O_{17,6}$	-	-	-	11	6	-	-	-	-	-
	$O_{17,7}$	10	-	-	-	-	-	-	-	-	-
	$O_{17,8}$	-	8	-	-	-	10	5	15	-	-
	$O_{17,9}$	11	10	-	-	-	-	-	-	-	-
	$O_{17,10}$	-	16	-	-	-	-	-	-	-	18

Tabela A.39: Problema Mk9 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{17,11}$	16	-	11	7	-	-	17	-	-	-
	$O_{17,12}$	-	-	14	-	-	-	-	-	-	-
	$O_{17,13}$	7	-	-	-	19	-	-	-	-	-
J_{18}	$O_{18,1}$	19	8	-	-	-	14	-	13	-	18
	$O_{18,2}$	-	-	-	6	-	16	9	14	9	-
	$O_{18,3}$	-	-	-	11	6	-	-	-	-	-
	$O_{18,4}$	-	8	-	-	5	-	-	13	-	6
	$O_{18,5}$	-	-	14	-	-	-	-	-	-	-
	$O_{18,6}$	-	-	11	-	8	-	11	-	17	-
	$O_{18,7}$	-	-	-	-	-	-	-	-	11	-
	$O_{18,8}$	11	10	-	-	-	-	-	-	-	-
	$O_{18,9}$	-	5	-	-	11	-	13	-	-	10
	$O_{18,10}$	-	-	-	-	-	-	-	14	-	-
	$O_{18,11}$	8	-	-	-	14	12	-	-	-	15
J_{19}	$O_{19,1}$	-	-	14	-	-	-	-	-	-	-
	$O_{19,2}$	11	10	-	-	-	-	-	-	-	-
	$O_{19,3}$	16	-	10	11	-	-	9	-	-	-
	$O_{19,4}$	-	16	-	-	-	-	-	-	-	18
	$O_{19,5}$	-	16	11	-	-	-	-	-	-	-
	$O_{19,6}$	6	-	-	11	-	-	-	16	15	-
	$O_{19,7}$	16	-	11	7	-	-	17	-	-	-
	$O_{19,8}$	-	5	-	-	11	-	13	-	-	10
	$O_{19,9}$	-	8	-	-	5	-	-	13	-	6
	$O_{19,10}$	-	-	-	-	9	-	-	8	-	-
	$O_{19,11}$	-	5	-	-	-	-	-	-	-	-
	$O_{19,12}$	-	8	-	-	-	10	5	15	-	-
	$O_{19,13}$	-	-	-	-	9	-	-	-	-	-
J_{20}	$O_{20,1}$	16	-	11	7	-	-	17	-	-	-
	$O_{20,2}$	-	5	-	-	11	-	13	-	-	10
	$O_{20,3}$	-	8	-	-	-	10	5	15	-	-
	$O_{20,4}$	-	-	14	-	-	-	-	-	-	-
	$O_{20,5}$	-	-	-	11	6	-	-	-	-	-
	$O_{20,6}$	6	-	-	11	-	-	-	16	15	-
	$O_{20,7}$	-	-	-	-	9	-	-	-	-	-
	$O_{20,8}$	10	-	-	-	-	-	-	-	-	-
	$O_{20,9}$	-	-	-	-	-	-	-	17	-	-
	$O_{20,10}$	-	12	-	-	-	-	-	-	6	-
	$O_{20,11}$	19	8	-	-	-	14	-	13	-	18
	$O_{20,12}$	-	16	11	-	-	-	-	-	-	-
	$O_{20,13}$	-	16	-	-	-	-	-	-	-	18

Tabela A.40: Problema Mk10 de [Brandimarte 1993]

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
J_1	$O_{1,1}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{1,2}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{1,3}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{1,4}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{1,5}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{1,6}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{1,7}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{1,8}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{1,9}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{1,10}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{1,11}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{1,12}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
J_2	$O_{2,1}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{2,2}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{2,3}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{2,4}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{2,5}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
	$O_{2,6}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
	$O_{2,7}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{2,8}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{2,9}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{2,10}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{2,11}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{2,12}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{2,13}$	-	11	-	-	11	-	-	-	-	-	-	-	-	-	-
J_3	$O_{3,1}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{3,2}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{3,3}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{3,4}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{3,5}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{3,6}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{3,7}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{3,8}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{3,9}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{3,10}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
	$O_{3,11}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
J_4	$O_{4,1}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
	$O_{4,2}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{4,3}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{4,4}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-

Tabela A.41: Problema Mk10 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{4,5}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{4,6}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{4,7}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{4,8}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{4,9}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{4,10}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{4,11}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
J_5	$O_{5,1}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{5,2}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{5,3}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{5,4}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{5,5}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{5,6}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{5,7}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{5,8}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{5,9}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{5,10}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{5,11}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{5,12}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{5,13}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
	$O_{5,14}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
J_6	$O_{6,1}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{6,2}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{6,3}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{6,4}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{6,5}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{6,6}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{6,7}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{6,8}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{6,9}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{6,10}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{6,11}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
J_7	$O_{7,1}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{7,2}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{7,3}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{7,4}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{7,5}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{7,6}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{7,7}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{7,8}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-

Tabela A.42: Problema Mk10 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{7,9}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{7,10}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{7,11}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{7,12}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{7,13}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{7,14}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
J_8	$O_{8,1}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{8,2}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{8,3}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{8,4}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{8,5}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{8,6}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{8,7}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{8,8}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{8,9}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{8,10}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{8,11}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{8,12}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{8,13}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
J_9	$O_{9,1}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{9,2}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{9,3}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{9,4}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{9,5}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{9,6}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{9,7}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{9,8}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{9,9}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{9,10}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{9,11}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
J_{10}	$O_{10,1}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{10,2}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{10,3}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{10,4}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{10,5}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{10,6}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{10,7}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{10,8}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{10,9}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{10,10}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-

Tabela A.43: Problema Mk10 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{10,11}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
	$O_{10,12}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
J_{11}	$O_{11,1}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{11,2}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{11,3}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{11,4}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{11,5}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{11,6}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{11,7}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{11,8}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{11,9}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{11,10}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
J_{12}	$O_{12,1}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{12,2}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{12,3}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{12,4}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{12,5}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{12,6}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{12,7}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{12,8}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{12,9}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{12,10}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{12,11}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
J_{13}	$O_{13,1}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{13,2}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{13,3}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{13,4}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{13,5}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{13,6}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{13,7}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{13,8}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{13,9}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{13,10}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{13,11}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
J_{14}	$O_{14,1}$	-	11	-	-	11	-	-	-	-	-	-	-	-	-	-
	$O_{14,2}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{14,3}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{14,4}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{14,5}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{14,6}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-

Tabela A.44: Problema Mk10 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{14,7}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{14,8}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{14,9}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{14,10}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
J_{15}	$O_{15,1}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{15,2}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{15,3}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{15,4}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{15,5}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{15,6}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{15,7}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{15,8}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{15,9}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{15,10}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
	$O_{15,11}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{15,12}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
J_{16}	$O_{16,1}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{16,2}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{16,3}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{16,4}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{16,5}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{16,6}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{16,7}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{16,8}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{16,9}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
	$O_{16,10}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{16,11}$	-	-	-	-	-	-	-	-	10	5	-	-	-	-	-
	$O_{16,12}$	-	11	-	-	11	-	-	-	-	-	-	-	-	-	-
	$O_{16,13}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
	$O_{16,14}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
J_{17}	$O_{17,1}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{17,2}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{17,3}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{17,4}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{17,5}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{17,6}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{17,7}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{17,8}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{17,9}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{17,10}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-

Tabela A.45: Problema Mk10 de [Brandimarte 1993] - continuação

Job	O_{ji}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
	$O_{17,11}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{17,12}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{17,13}$	19	16	8	-	-	-	-	-	13	19	-	-	-	-	-
J_{18}	$O_{18,1}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{18,2}$	6	-	-	-	11	13	-	-	-	7	-	-	-	-	-
	$O_{18,3}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{18,4}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{18,5}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{18,6}$	-	11	-	-	11	-	-	-	-	-	-	-	-	-	-
	$O_{18,7}$	-	16	12	11	15	-	-	-	-	9	-	-	-	-	-
	$O_{18,8}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{18,9}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{18,10}$	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{18,11}$	-	-	-	11	10	-	15	-	-	14	-	-	-	-	-
J_{19}	$O_{19,1}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{19,2}$	-	5	-	-	-	5	-	-	-	-	-	-	-	-	-
	$O_{19,3}$	-	6	-	13	-	-	-	8	-	-	-	-	-	-	-
	$O_{19,4}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-
	$O_{19,5}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
	$O_{19,6}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{19,7}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{19,8}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{19,9}$	-	-	-	-	-	11	-	-	-	13	-	-	-	-	-
	$O_{19,10}$	9	11	-	-	-	-	-	-	-	-	-	-	-	-	-
	$O_{19,11}$	-	8	-	7	-	-	5	-	-	-	-	-	-	-	-
	$O_{19,12}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{19,13}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
J_{20}	$O_{20,1}$	16	-	6	-	11	-	-	-	-	17	-	-	-	-	-
	$O_{20,2}$	15	19	-	-	-	-	-	-	9	-	-	-	-	-	-
	$O_{20,3}$	12	-	8	-	5	-	-	-	-	-	-	-	11	-	-
	$O_{20,4}$	-	5	-	-	-	-	-	-	19	-	-	-	-	-	-
	$O_{20,5}$	-	16	-	-	9	-	-	16	-	9	-	-	-	-	-
	$O_{20,6}$	-	-	-	-	15	-	-	-	-	-	-	-	-	-	-
	$O_{20,7}$	-	16	-	-	11	-	8	-	-	11	-	-	-	-	-
	$O_{20,8}$	-	-	18	10	-	-	-	10	7	-	-	-	-	-	-
	$O_{20,9}$	-	-	-	-	-	11	11	-	-	-	-	-	-	-	-
	$O_{20,10}$	8	-	-	7	-	-	14	-	12	-	-	-	-	-	-
	$O_{20,11}$	-	-	-	8	-	18	13	-	-	19	-	-	-	-	-
	$O_{20,12}$	-	-	12	-	-	-	15	-	-	-	-	-	-	-	-
	$O_{20,13}$	7	-	-	-	-	-	9	-	-	-	-	-	-	-	-