DISSERTAÇÃO DE MESTRADO Nº 1117

ANÁLISE DE ESTRUTURAS DE VIZINHANÇA PARA O PROBLEMA DE SEQUENCIAMENTO DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM TEMPOS DE PREPARAÇÃO

Letícia Mayra Pereira

DATA DA DEFESA: 25/02/2019

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

ANÁLISE DE ESTRUTURAS DE VIZINHANÇA PARA O PROBLEMA DE SEQUENCIAMENTO DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM TEMPOS DE PREPARAÇÃO

Letícia Mayra Pereira

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Felipe Campelo França Pinto
Coorientador: Prof. André Luiz Maravilha Silva

Belo Horizonte - MG

Fevereiro de 2019

Pereira, Letícia Mayra.
P436a Análise de estruturas de vizinhança para o problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação [manuscrito] / Letícia Mayra Pereira. – 2019.
xviii, 59 f., enc.: il.

Orientador: Felipe Campelo França Pinto.
Coorientador: André Luiz Maravilha Silva.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 55-59.

1. Engenharia elétrica - Teses. 2. Heurística - Teses. 3. Otimização - Teses. I. Pinto, Felipe Campelo França. II. Silva, André Luiz Maravilha. III. Universidade Federal de Minas Gerais, Escola de Engenharia. IV. Título.

CDU: 621.3(043)

"Análise de Estruturas de Vizinhança Para O Problema de Sequenciamento de Máquinas Paralelas Não Relacionadas Com Tempos de Preparação"

Letícia Mayra Pereira

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 25 de fevereiro de 2019.

Por:

Prof. Dr. Felipe Campelo França Pinto
DEE (UFMG) - Orientador

Prof. Dr. André Luiz Maravilha Silva
DIGD-DV (CEFET-MG - Divinópolis)

Prof. Dr. Eduardo Gontijo Carrano
DEE (UFMG)

Prof. Dr. Lucas de Souza Batista
DEE (UFMG)

Dedicado à minha avó Edith e aos meus amigos do ORCS Lab,
por tudo que aprendi com vocês.

Resumo

O uso de heurísticas baseadas em busca local é bastante comum para otimizar os problemas de sequenciamento de máquinas paralelas, principalmente aqueles que consideram os tempos de preparação da máquina (*setup*). Com isso, as estruturas de vizinhança desempenham um papel essencial na capacidade dessas heurísticas de explorar adequadamente o espaço de soluções. Este trabalho apresenta uma análise exploratória e estatística para caracterização de seis estruturas de vizinhança comumente utilizadas na solução do problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e da máquina, buscando a minimização do *makespan* como objetivo principal. As seis estruturas de vizinhança são exploradas em diferentes estágios de busca de várias instâncias do problema de sequenciamento, com a busca sendo conduzida usando uma implementação própria de um *Simulated Annealing*, que é o estado da arte para solucionar essa classe de problema. Os resultados obtidos são usados para avaliar e modelar a qualidade relativa dessas vizinhanças em termos de melhoria esperada de um único movimento em diferentes pontos ao longo do procedimento de busca. Os resultados dessa análise indicam a superioridade de uma vizinhança em relação às demais. Com os resultados obtidos nessa análise estatística, foi desenvolvido um modelo de regressão linear para predizer a utilidade esperada de cada função de vizinhança e, com isso determinar a probabilidade de escolher cada uma das funções de vizinhança ao longo das iterações do *Simulated Annealing*. Os resultados obtidos com a modificação na escolha das vizinhanças foi superior à versão original. A partir deste experimento, a proposição de novas heurísticas baseadas em busca local pode aproveitar a análise realizada neste trabalho para priorizar a escolha das vizinhanças e quando utilizá-las ao longo do processo de otimização para acelerar a convergência para melhores soluções.

Abstract

Local search heuristics are usually employed in the optimization of parallel machine scheduling problems, particularly those in which setup times are considered. Neighborhood structures represent a central aspect of these heuristics, enabling them to adequately explore the search space. This work presents an exploratory statistical analysis of six neighborhood structures commonly used for the unrelated parallel machine scheduling problem with sequence dependent setup times, in which the main objective is the minimization of the makespan. The neighborhood structures are equipped on a specific implementation of Simulated Annealing that is considered state-of-the-art for this particular problem, and are explored at different stages of the search. The results indicate the superiority of one neighborhood concerning the others. Besides, the results obtained are used to fit a regression model capable of providing quantitative guidelines for the selection of each structure at different stages of the search. The resulting model is used to devise a modified version of the Simulated Annealing in which an adaptive approach for neighborhood selection is employed when solving instances belonging to this particular problem class. The results obtained with the modified Simulated Annealing overcome the original one. From the experiment, the proposition of new heuristics based on local search can take advantage of the analysis performed in this paper to prioritize the choice of neighborhoods and when to use them throughout the optimization process to speed up the convergence to better solutions.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

No setor industrial, a atividade da programação de tarefas é uma das mais complicadas funções do gerenciamento da produção. A definição da melhor sequência para as ordens de produção, dentro de um horizonte de planejamento de qualquer setor produtivo, assegura o cumprimento de prazos de entregas, qualidade dos produtos e reduções dos custos de produção. Além disso, determinar o melhor sequenciamento faz com que a empresa faça uma melhor utilização dos seus recursos produtivos [Allahverdi, 2015].

Os problemas de sequenciamento que consideram tempo de preparação têm um papel importante em processos industriais e serviços que envolvem atribuição e sequenciamento de tarefas a recursos (pessoas, máquinas etc.). Grande parte dos trabalhos de sequenciamento presentes na literatura ignoram a existência do tempo de preparação. De fato, existem situações em que o tempo de preparação pode ser ignorado ou agregado ao tempo de processamento, mas ignorá-los quando isso não ocorre tende a gerar um impacto negativo na qualidade do processo ou serviço, o que pode significar prejuízos e atrasos [Allahverdi et al., 2008; Allahverdi, 2015].

O problema de sequenciamento de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência é uma generalização de diversos problemas clássicos de sequenciamento de máquinas paralelas. Esse problema é bastante relevante em linhas de produção e serviços envolvendo recursos heterogêneos [Santos et al., 2019].

Na área de otimização, essa classe de problema tem sido muito explorada. Devido esse problema pertencer à classe de problemas NP-difícil [Garey & Johnson, 1979], a maioria dos pesquisadores buscam solucionar esse problema utilizando algoritmos heurísticos. Por mais que as heurísticas não forneçam a garantia da qualidade das soluções retornadas, elas são capazes de encontrar boas soluções [Hillier & Lieberman, 2013], principalmente em instâncias de grande porte, onde métodos exatos se tornam inviá-

veis, pois não conseguem obter a solução ótima, ou até mesmo soluções de qualidade razoável em um tempo hábil.

Muitas heurísticas utilizadas para solucionar esse tipo de problema são baseadas em busca local, e uma parte fundamental dessas heurísticas são as estruturas de vizinhança que auxiliam na capacidade de explorar adequadamente o espaço de soluções.

O presente trabalho faz uma análise estatística e exploratória da caracterização de estruturas de vizinhança. Para isso, foram detalhadas e ilustradas seis estruturas de vizinhança comumente utilizadas, que são exploradas em diferentes estágios da busca em várias instâncias do problema de sequenciamento, com a busca sendo conduzida usando o algoritmo *Simulated Annealing*. Além disso, é proposta uma mudança no algoritmo utilizado para se escolher qual a estrutura de vizinhança que será utilizada ao longo do algoritmo, buscando explorar essas observações para possivelmente melhorar o desempenho do algoritmo de busca para problemas de sequenciamento.

1.1 Motivação

Com a necessidade de se resolver problemas cada vez mais complexos e de maior dimensão, os trabalhos desenvolvidos na área de sequenciamento de tarefas (e otimização combinatória de uma forma geral) possibilitaram o desenvolvimento de métodos heurísticos cada vez mais eficientes. Em problemas de sequenciamento de tarefas, as heurísticas baseadas em busca local são frequentemente utilizadas devido à capacidade de obterem soluções de boa qualidade em um tempo viável. Assim, as estruturas de vizinhança têm um papel fundamental na qualidade das soluções retornadas por essas heurísticas. Apesar de grande parte dos trabalhos utilizarem múltiplas estruturas de vizinhança para explorar o espaço de soluções, poucos avaliam a contribuição dessas estruturas no desempenho das heurísticas propostas.

Diante disso, o estudo das estruturas de vizinhança e o comportamento delas ao longo do algoritmo são a motivação para a realização desse trabalho, pois um melhor entendimento desse componente, que é fundamental em heurísticas de busca local, permite o desenvolvimento de heurísticas mais eficientes.

1.2 Objetivos

O objetivo geral deste trabalho é fazer uma análise exploratória para caracterizar estruturas de vizinhança para o problema de sequenciamento de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência, de forma a permi-

tir projetos de algoritmos de busca local mais eficientes. Para isso, são definidos os seguintes objetivos específicos:

- Definição de um protocolo experimental para avaliação de funções de vizinhança.
- Definição de um modelo de regressão linear para predizer a utilidade esperada de cada vizinhança ao longo da execução do algoritmo, considerando a dimensão do problema e o momento em que a busca será realizada.
- Incorporação do modelo de predição em um algoritmo baseado no *Simulated Annealing*, para determinar a probabilidade de escolha das vizinhanças utilizadas por ele.

1.3 Contribuições

Com o desenvolvimento deste trabalho, são alcançadas as seguintes contribuições:

- Definição de um protocolo experimental para avaliação da utilidade esperada de funções de vizinhança em heurísticas de busca local.
- Melhoria no desempenho do algoritmo *Simulated Annealing* (que representa o estado-da-arte) para problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência através da incorporação de modelos de predição para determinar a probabilidade de escolha de cada vizinhança ao longo do algoritmo.

1.4 Estrutura do Texto

Neste capítulo foi feita uma introdução para contextualização do problema de sequenciamento de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência, para destacar a importância de se utilizar estruturas de vizinhança em heurísticas baseadas em busca local para resolver essa classe de problema. Para concluir esse capítulo, são apresentados breves resumos dos capítulos seguintes.

A primeira parte do trabalho engloba o Capítulo 2 e o Capítulo 3, que abrangem a definição do problema em questão e a revisão da literatura. No Capítulo 2, uma descrição detalhada do problema de sequenciamento de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência é apresentada, juntamente com sua formulação matemática e a caracterização da função objetivo. O Capítulo 3

apresenta alguns trabalhos encontrados na literatura que abordam essa classe de problema e, também alguns trabalhos com diferentes funções objetivo, mas que possuem similaridades em relação às restrições.

A segunda parte do trabalho é composta pelos Capítulos 4, 5 e 6, que abordam a análise estatística das estruturas de vizinhança. O Capítulo 4 descreve e ilustra as estruturas de vizinhança utilizadas para o experimento de caracterização, juntamente com as características de cada vizinhança; O Capítulo 5 descreve o protocolo experimental realizado para fazer as análises; No Capítulo 6, a análise gráfica e exploratória dos dados é detalhada.

A terceira parte do trabalho é composta pelo Capítulo 7 que, a partir da análise realizada na segunda parte do trabalho, busca investigar como é possível melhorar o desempenho de um algoritmo baseado no *Simulated Annealing* para problemas de sequenciamento. O algoritmo utilizado é detalhado, bem como a modificação feita nele através de um modelo de regressão linear que determina a probabilidade de escolher cada uma das funções de vizinhança ao longo do algoritmo. A comparação dos resultados obtidos com o algoritmo original e o algoritmo modificado também é detalhada.

Por fim, no Capítulo 8 são feitas as considerações finais, concluindo sobre as vantagens de se escolher as estruturas de vizinhança probabilisticamente ao longo do algoritmo.

Parte I

Definição do problema e revisão da literatura

Capítulo 2

Definição do Problema

O estudo de problemas de sequenciamento de tarefas tem grande importância tanto prática quanto teórica [Pinedo, 2008]. Essa classe de problemas trata da alocação de tarefas à máquinas, buscando minimizar um ou mais objetivos. Portanto, o estudo desse tipo de problema tem despertado bastante interesse em muitos pesquisadores na área de otimização [Zhu & Wilhelm, 2006].

Este trabalho trata especificamente o problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e da máquina (do inglês, *Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times* - UPMSP), que é uma generalização de muitos problemas clássicos de programação de máquinas paralelas. O problema tratado neste trabalho tem alta relevância devido a sua ampla aplicabilidade nas indústrias e a dificuldade existente em sua resolução. Sua aplicabilidade em problemas reais pode ser vista em indústrias têxteis [Gendreau et al., 2001], siderúrgicas [Tang & Wang, 2009], eletrônicas [Kim et al., 2002], na engenharia elétrica [Maravilha et al., 2018], entre outras.

O UPMSP pode ser definido como um conjunto de tarefas $N = \{1, \ldots, n\}$ que devem ser processadas por um conjunto de máquinas $M = \{1, \ldots, m\}$, sendo que cada tarefa $j \in N$ é processada por uma única máquina $k \in M$. Além disso, o tempo de processamento de cada tarefa j em uma máquina k é dado pelo parâmetro p_{jk} e o tempo de preparação da máquina k para processar a tarefa j imediatamente após a tarefa i é dado pelo parâmetro s_{ijk}. O objetivo desse problema é atribuir as tarefas às máquinas e definir a sequência em que cada máquina deve processar as tarefas atribuídas a ela, de forma a minimizar o *makespan*, ou seja, minimizar o tempo de conclusão da última tarefa a deixar o sistema. Utilizando a notação de Graham et al. [1979], esse problema pode ser descrito como $R|s_{ijk}|C_{max}$. Nessa descrição, R representa um ambiente com máquinas paralelas não relacionadas, s_{ijk} indica a presença de tempos

de preparação dependentes da sequência e da máquina, e C_{max} indica que o objetivo é a minimização do *makespan*. Esse problema pertence a classe de problemas NP-difícil [Garey & Johnson, 1979; Vallada & Ruiz, 2011].

2.1 Formulação Matemática

Nessa seção é apresentada uma formulação matemática para o problema $R|s_{ijk}|C_{max}$, proposta por Vallada & Ruiz [2011], na qual o problema é formulado como um problema de programação linear inteira mista.

Nesse modelo existe uma tarefa fictícia com tempo de conclusão e tempo de preparação iguais a zero. Essa tarefa fictícia é utilizada para indicar o início da programação de uma máquina, aparecendo no sequenciamento de uma máquina qualquer, antes da primeira tarefa que é de fato processada pela máquina.

A formulação matemática retratada para resolver esse problema tem os seguintes parâmetros:

- N: conjunto de tarefas que serão processadas;
- M: conjunto de máquinas paralelas não relacionadas;
- p_{jk}: tempo de processamento da tarefa j na máquina k;
- s_{ijk}: tempo de preparação da máquina k para processar a tarefa j imediatamente após a tarefa i;

e as seguintes variáveis de decisão:

- x_{ijk}: variável binária que assume o valor 1 se a tarefa i for processada imediatamente antes da tarefa j na máquina k, ou o valor 0, caso contrário;
- C_{jk}: tempo de conclusão da tarefa j na máquina k;
- C_{max}: tempo total de processamento de todas as tarefas, ou seja, o *makespan*.

Desta forma, o modelo proposto por Vallada & Ruiz [2011] é apresentado a seguir:

$$\text{Min} \quad C_{max} \tag{2.1}$$

$$\text{s.a.:} \quad \sum_{k \in M} \sum_{\substack{i \in \{0\} \cup \{N\} \\ i \neq j}} x_{ijk} = 1 \qquad \forall j \in N \tag{2.2}$$

$$\sum_{k \in M} \sum_{\substack{j \in N \\ j \neq i}} x_{ijk} \leq 1 \qquad \forall i \in N \tag{2.3}$$

$$\sum_{i \in N} x_{0ik} \leq 1 \qquad \forall k \in M \tag{2.4}$$

$$\sum_{\substack{h \in \{0\} \cup \{N\} \\ h \neq i, h \neq j}} x_{hik} \geq x_{ijk} \qquad \forall j, i \in N, j \neq i, \forall k \in M \tag{2.5}$$

$$C_{jk} + V(1 - x_{ijk}) \geq C_{ik} + s_{ijk} + p_{jk} \qquad \begin{array}{r} \forall i \in \{0\} \cup \{N\}, \forall j \in N, \\ j \neq i, \forall k \in M \end{array} \tag{2.6}$$

$$C_{0k} = 0 \qquad \forall k \in M \tag{2.7}$$

$$C_{jk} \geq 0 \qquad \forall j \in N, \forall k \in M \tag{2.8}$$

$$C_{max} \geq C_{jk} \qquad \forall i \in N, \forall k \in M \tag{2.9}$$

$$x_{ijk} \in \{0, 1\} \qquad \begin{array}{r} \forall i \in \{0\} \cup \{N\}, \forall j \in N, \\ j \neq i, \forall k \in M \end{array} \tag{2.10}$$

A função objetivo, representada pela Equação (2.1), busca minimizar o *makespan*. As restrições (2.2) asseguram que cada tarefa seja atribuída a somente uma máquina e tenha exatamente uma antecessora. O conjunto de restrições (2.3) estabelece que o número máximo de sucessoras de cada tarefa seja igual a um. O conjunto de restrições (2.4) define um limite igual a um para o número de sucessoras das tarefas fictícias em cada máquina. As restrições representadas em (2.5) garantem que se uma tarefa j é processada imediatamente após uma tarefa i, ambas pela máquina k, então deve existir uma tarefa (real ou fictícia) h precedente à tarefa i na máquina k. O conjunto de restrições (2.6) controla os tempos de conclusão das tarefas nas máquinas. Se uma tarefa j for atribuída a uma máquina k após a tarefa i ($x_{ijk} = 1$), o tempo de conclusão da tarefa j na máquina k deve ser maior ou igual ao tempo de conclusão da tarefa i, mais o tempo de preparação entre i e j, mais o tempo de processamento da tarefa j. No entanto, se $x_{ijk} = 0$, a constante V (uma constante com valor grande), torna a restrição redundante. O conjunto de restrições (2.7) determina o tempo de conclusão como zero para as tarefas fictícias e as restrições (2.8) definem que o tempo de conclusão das tarefas não fictícias seja não-negativo. O conjunto de restrições representadas em (2.9)

define o máximo tempo de conclusão. Por fim, as restrições (2.10) definem o domínio das variáveis.

Outras formulações para esse problema podem ser encontradas em Guinet [1991], Avalos-Rosales et al. [2015], Tran & Beck [2012], Tran et al. [2016] e Fanjul-Peyro et al. [2019]. Como o foco desse trabalho não é em métodos exatos para a resolução do $R|s_{ijk}|C_{max}$, optou-se por apresentar a formulação proposta por Vallada & Ruiz [2011] por ser mais simples.

2.2 Caracterização da Função Objetivo

Seja uma função de vizinhança $\eta : x \in \mathcal{X} \rightarrow \mathcal{N} \in 2^{\mathcal{X}}$, em que $\mathcal{X}$ é o conjunto de todas as soluções factíveis de um problema de otimização e $2^{\mathcal{X}}$ é o conjunto potência de $\mathcal{X}$, ou seja, o conjunto de todos os subconjuntos de $\mathcal{X}$. Assim, uma função de vizinhança mapeia, para cada solução factível, um conjunto de outras soluções viáveis. Uma estrutura de vizinhança é um grafo direcionado $G_\eta = \{\mathcal{X}, \mathcal{E}\}$ induzido por uma função de vizinhança η, em que $\mathcal{E} = \{(x_s, x_t)\ x_t \in \eta(x_s)\}$.

Dada uma solução incumbente, sabe-se que o *makespan* só pode ser melhorado se a máquina *makespan*[1] for alterada (a sequência das tarefas processadas na máquina *makespan* ou a realocação de uma tarefa atualmente atribuída à máquina *makespan* para outra máquina). Mudanças em soluções que não envolvem a máquina *makespan* não são capazes de melhorar o *makespan*. Tais mudanças podem ser neutras (elas não alteram o *makespan*) ou podem piorar o *makespan*. Além disso, o número de soluções diferentes, mas com o mesmo valor de *makespan*, pode ser grande, o que resulta em uma função objetivo com várias regiões planas.

Esse comportamento tende a degradar o desempenho das heurísticas baseadas em busca local, uma vez que tais algoritmos geralmente exploram o espaço de busca movendo-se de uma solução atual para uma solução melhorada, caso contrário, o movimento não é executado. Algumas meta-heurísticas, como *Iterated Local Search* [Baxter, 1981; Lourenço et al., 2010] e *Variable Neighborhood Search* [Mladenović & Hansen, 1997], geralmente empregam um mecanismo de diversificação para escapar de soluções ótimas locais (e de regiões planas). No entanto, o número de regiões planas no $R|s_{ijk}|C_{max}$ é tão grande que até essas estratégias não conseguem resolver esse problema de maneira adequada.

Para evitar esse comportamento, um segundo critério pode ser usado ao comparar soluções. Para o $R|s_{ijk}|C_{max}$, um segundo critério pode ser a soma do tempo de

[1]Neste trabalho é adotado o termo "máquina *makespan*" para referenciar a máquina que executa a última tarefa a sair do sistema.

conclusão de todas as máquinas:

$$\sum_{k \in M} \max_{j \in N} \{C_{jk} : j \text{ é processado em } k\} \tag{2.11}$$

em que C_{jk} é o tempo de conclusão da tarefa $j \in N$ na máquina $k \in M$. Por uma questão de brevidade, este segundo critério será referido como $\sum C^k_{max}$ e para o *makespan* geral será usado C_{max}.

Mesmo se C_{max} não mudar, uma redução em $\sum C^k_{max}$ pode permitir que os próximos movimentos minimizem o valor atual de C_{max}. Uma vez que as outras máquinas, e não a máquina *makespan*, são menos carregadas, as tarefas atualmente processadas na máquina *makespan* podem ser realocadas sem degradar o *makespan* e talvez melhorá-lo. Então, ao comparar uma solução vizinha com a incumbente, ela pode ser classificada em uma das quatro classes descritas na Tabela 2.1.

Tabela 2.1. Comparação classes de vizinhos vs. solução incumbente

Classe	C_{max}	$\sum C^k_{max}$
Melhoria-1	Melhor que incumbente	Melhor que incumbente
	Melhor que incumbente	Igual a incumbente
	Melhor que incumbente	Pior que incumbente
Melhoria-2	Igual a incumbente	Melhor que incumbente
Neutro	Igual a incumbente	Igual a incumbente
Piora	Igual a incumbente	Pior que incumbente
	Pior que incumbente	Melhor que incumbente
	Pior que incumbente	Igual a incumbente
	Pior que incumbente	Pior que incumbente

O uso deste segundo critério pode explicar o bom desempenho do *Simulated Annealing* proposto por Santos et al. [2019] para resolver o $R|s_{ijk}|C_{max}$, uma vez que esse segundo critério é utilizado nos procedimentos de escolha de uma solução vizinha no algoritmo proposto por eles; além da vantagem do *Simulated Annealing* em escapar de regiões planas em relação a outras meta-heurísticas.

Embora o presente trabalho trate especificamente do problema $R|s_{ijk}|C_{max}$, os desafios de se ter uma função objetivo com muitas regiões planas, devido à minimização do *makespan*, em geral, permanecem para outras variantes de problemas de programação de máquinas paralelas que têm C_{max} como função objetivo.

Capítulo 3

Trabalhos Relacionados

Existem diversos trabalhos na literatura que abordam o problema de sequenciamento de máquinas paralelas não relacionadas com tempo de preparação dependente da sequência e da máquina. Esses trabalhos empregam desde métodos exatos a heurísticas para sua solução. No entanto, o uso de métodos exatos é limitado a problemas de pequeno porte devido à sua complexidade. Assim, a maior parte dos trabalhos fazem uso de heurísticas.

Apesar do presente trabalho focar na minimização do C_{max}, nesse capítulo são revisados, também, alguns trabalhos com diferentes funções objetivo, devido às similaridades em relação às restrições.

Nos trabalhos de Guinet [1993] e Gendreau et al. [2001] são propostas uma heurística de atribuição (uma extensão do método Húngaro) e uma heurística de divisão e mesclagem, respectivamente, com o objetivo de minimizar o C_{max}.

Lee & Pinedo [1997] desenvolveram uma heurística de três fases para esse problema, com o objetivo de minimizar a soma do atraso ponderado das tarefas ($\sum w_i T_i$ na notação de Graham et al. [1979]). Na primeira fase, é realizado um pré-processamento para determinar os fatores e melhores valores dos parâmetros que caracterizam a instância. A segunda fase consiste em uma regra de atribuição e sequenciamento controlada pelos parâmetros determinados na primeira fase. Na terceira fase, como um procedimento de melhoria de soluções, uma heurística baseada no *Simulated Annealing* [Kirkpatrick et al., 1983] foi aplicada a partir da solução obtida na segunda fase.

No trabalho de Kurz & Askin [2001], os autores propuseram quatro heurísticas, dentre elas, um algoritmo genético, para minimização do C_{max}. As outras três heurísticas foram denominadas como: *Slicing*, *Multiple MULTI-FIT* e *Multiple insertion*. Em uma análise realizada pelos autores, foi constatado que a heurística *Multiple insertion* é superior às demais em relação a qualidade das soluções. Embora o algoritmo genético

tenha apresentado melhores resultados para algumas instâncias, na análise realizada pode-se observar que seu desempenho geral, especialmente em função do tempo de execução, é inferior à heurística *Multiple insertion*.

Pfund et al. [2008] apresentam uma heurística para minimizar $\sum w_i T_i$. Nessa heurística é utilizada uma regra de despacho que permite analisar qual tarefa tem maior prioridade para ser processada primeiro. Para definir os parâmetros de escala para a regra de despacho criada, foi desenvolvida uma abordagem, que considera vários valores para esses parâmetros, gerando várias sequências. Por fim, a melhor sequência é escolhida. Os resultados obtidos pela heurística proposta foram comparados com outras heurísticas construtivas propostas por Lee & Pinedo [1997], Mason et al. [2002] e Morton & Pentico [1993]. Para todos os casos considerados, a heurística proposta obteve melhor desempenho.

Com exceção do trabalho de Lee & Pinedo [1997] e do algoritmo genético proposto por Kurz & Askin [2001], os trabalhos discutidos anteriormente se limitam a heurísticas construtivas. Inúmeras heurísticas baseadas em buscas locais também são encontradas na literatura para resolver essa classe de problemas.

França et al. [1996] utilizam uma heurística baseada na busca tabu com o objetivo de minimizar o C_{max}. Os autores desenvolveram uma heurística de três fases, onde na primeira fase é construída uma solução inicial, que em seguida é melhorada na segunda fase por meio de uma heurística baseada em busca tabu. A terceira fase tenta obter uma melhoria adicional na solução retornada na segunda fase, buscando melhorar a sequência obtida na máquina mais sobrecarregada, para assim tentar reduzir o valor do *makespan*. Os autores realizaram experimentos computacionais sobre um conjunto de instâncias geradas aleatoriamente, analisando a heurística em diferentes números de tarefas e máquinas. Além disso, o método proposto foi comparado a uma heurística de construção simples. Porém, os trabalhos existentes até então foram ignorados nesse experimento.

Rabadi et al. [2006] também apresentam uma heurística para a minimização do C_{max}. Os autores propõem uma heurística denominada *Meta-RaPS (Meta-heuristic for Randomized Priority Search)*, que é descrita pelos autores como uma estratégia que utiliza heurísticas de construção e melhoria para gerar soluções de alta qualidade. A heurística proposta foi comparada com uma heurística de três fases, proposta por Al-Salem [2004], denominada *Partitioning Heuristic*. Essa heurística consiste em gerar uma heurística construtiva para atribuir tarefas às máquinas na primeira fase, realizar uma busca local para tentar uma melhoria na solução fornecida pela primeira fase, buscando diminuir o *makespan*, e na terceira fase é aplicada uma heurística que trata cada máquina como um problema do caixeiro viajante e determina a sequência das

tarefas em cada máquina. As duas heurísticas foram comparadas com um método exato na solução de instâncias de pequeno porte, e ambas encontraram soluções ótimas em todos os casos. Nas instâncias maiores, as heurísticas foram comparadas entre si, uma vez que, para esse tipo de problema o método exato se torna inviável. Constatou-se que os resultados obtidos pela heurística *Meta-RaPS* foram superiores aos resultados obtidos pela *Partitioning Heuristic*.

De Paula et al. [2007] propõem um algoritmo baseado na meta-heurística *Variable Neighborhood Search* (VNS) e o compara com três versões do *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo & Resende, 1995], cada uma com uma busca local diferente. Os autores buscaram minimizar C_{max} e os atrasos ponderados. Os resultados obtidos de uma série de experimentos mostraram que o VNS forneceu melhores resultados que as três versões do GRASP, principalmente nas instâncias com maior número de tarefas.

O trabalho de Arnaout et al. [2010] apresenta uma heurística de dois estágios baseada em colônia de formigas (ACO, do inglês *Ant Colony Optimization* [Dorigo & Di Caro, 1999]), buscando minimizar o C_{max}. Para resolver o problema, a heurística proposta foi particionada em dois subproblemas: atribuição e sequenciamento. No estágio de atribuição, as n tarefas são alocadas para m máquinas, já no segundo estágio, é feito o sequenciamento das tarefas em cada máquina. A atribuição e o sequenciamento foram baseados nas quantidades de feromônios, que são definidas e ajustadas com base na qualidade da solução em cada iteração. A heurística proposta foi comparada com os métodos propostos por Rabadi et al. [2006] e Al-Salem [2004], citados anteriormente. Os resultados mostram que a heurística baseada em colônia de formigas superou os outros algoritmos.

Ying et al. [2012] tratam desse problema propondo uma heurística baseada no *Simulated Annealing* que incorpora uma estratégia de busca restrita, buscando minimizar o C_{max}. A heurística proposta pelos autores busca reduzir o esforço da busca local para encontrar a melhor solução vizinha, eliminando movimentos ineficazes das tarefas. Testes realizados mostraram que o *Simulated Annealing* proposto obteve melhores resultados que os algoritmos propostos por Al-Salem [2004], Rabadi et al. [2006] e Arnaout et al. [2010], principalmente para instâncias maiores.

Até a publicação do trabalho de Vallada & Ruiz [2011], não se tinha um conjunto padrão de instâncias para a avaliação de desempenho dos métodos propostos para o problema $R|s_{ijk}|C_{max}$. Nesse trabalho, Vallada & Ruiz [2011] propõem um conjunto composto por 1640 instâncias de testes e 200 para *tuning* [SOA-ITI, 2013]. Além do conjunto de instâncias, os autores propõem um algoritmo genético combinado com uma busca local para a resolução do problema e o compara aos melhores métodos existentes

até então. Os resultados indicaram a superioridade do algoritmo proposto.

Para tratar desse problema, Cota et al. [2014] desenvolveram uma heurística baseada em *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) [Mladenović & Hansen, 1997] e *Path-relinking*, buscando minimizar o C_{max}. A heurística proposta parte de uma solução inicial construída pela regra de despacho *Adaptive Shortest Processing Time*, uma extensão da regra do menor tempo de processamento (*Shortest Processing Time*), que basicamente consiste em, dada uma lista de tarefas e máquinas, avalia-se a inserção de cada tarefa em todas as posições de todas as máquinas. A tarefa é inserida na posição que resulta no menor tempo de conclusão. Em seguida, essa solução é refinada pelo ILS, utilizando o VND como método de busca local. O *Path-relinking* foi aplicado como uma estratégia de intensificação e diversificação. Os resultados mostraram que, tanto em termos de qualidade dos resultados quanto em variabilidade das soluções, o algoritmo proposto superou o algoritmo genético proposto por Vallada & Ruiz [2011] citado anteriormente. Os autores explanam a importância do *Path-relinking*, que contribuiu para reduzir a variabilidade de soluções finais da heurística.

Em seu trabalho, Martins Muller et al. [2014] desenvolvem uma heurística híbrida. A heurística proposta pelos autores busca minimizar o C_{max} e tem as suas estruturas de vizinhança modeladas como problemas de programação linear inteira mista. São modelados três movimentos: (*i*) realocação de uma tarefa na mesma máquina; (*ii*) inserção de uma tarefa j como sucessora da tarefa i, ejetando i de sua posição original; e (*iii*) conexão da antecessora à sucessora de uma tarefa j. A busca local é realizada através da solução de um sub-problema de programação linear inteira mista, que é inserida em uma heurística. Essa abordagem proposta foi aplicada a diferentes conjuntos de instâncias da literatura e os resultados superaram os resultados obtidos pelo algoritmo genético de Vallada & Ruiz [2011].

O trabalho de Santos et al. [2019] aborda esse problema investigando o desempenho de quatro heurísticas baseadas em busca local, sendo elas: *Simulated Annealing (SA), Iterated Local Search, Late Acceptance Hill-climbing* (LAH) [Burke & Bykov, 2008] e *Step Counting Hill-climbing* (SCH) [Bykov & Petrovic, 2013], tendo como objetivo minimizar o C_{max}. Para esses quatro métodos, os autores utilizaram seis estruturas de vizinhança para explorar o espaço de busca. As estruturas de vizinhança utilizadas pelos autores foram: *shift*, *switch*, *task move*, *swap*, *two-shift* e *direct swap*. Para cada uma das estruturas de vizinhança, foram adotados diferentes critérios para determinar a forma de escolha da solução vizinha: (i) participação obrigatória, ou não, da máquina *makespan* no movimento realizado; (ii) escolha aleatória ou avaliação parcial da estrutura de vizinhança (considerando a minimização de $\sum C^k_{max}$).

As análises de Santos et al. [2019] se concentraram em questões de projeto dos algoritmos, esforço computacional e qualidade dos resultados. Os resultados encontrados mostraram uma melhoria significante quando comparados aos trabalhos anteriores de Vallada & Ruiz [2011] e Cota et al. [2014], principalmente nas instâncias de maior dimensão. O algoritmo *Simulated Annealing* teve destaque dentre os outros métodos, pois demonstrou, consistentemente, melhores resultados.

Alguns trabalhos considerando múltiplos objetivos também são encontrados na literatura. Por exemplo, Cota et al. [2018] apresentam uma abordagem multiobjetivo, onde buscam lidar com o uso sustentável do consumo de energia e os efeitos ambientais. Os objetivos dos autores eram minimizar o C_{max} e consumo total de energia (TEC). Eles propuseram um modelo de Programação Linear Inteira Mista e, além disso, propõem uma heurística matemática chamada *smart pool* para encontrar um conjunto de soluções não dominadas. Os resultados obtidos para um conjunto de instâncias criado pelos autores mostraram que a heurística matemática proposta alcançou boa convergência em direção à fronteira Pareto, com menor custo computacional quando comparado a métodos tradicionais.

A grande maioria dos trabalhos encontrados, referente a esse tipo de problema, são trabalhos comparativos, onde os autores propõem algoritmos para serem superiores àqueles já encontrados na literatura. A Tabela 3.1 apresenta uma visão geral dos trabalhos citados anteriormente e os métodos utilizados por eles.

Dos trabalhos citados acima, muitos realizam alguma busca local ao longo do algoritmo. A Tabela 3.2 apresenta as funções de vizinhança utilizadas pelos autores que propõem heurísticas baseadas em busca local. Nesta tabela, as vizinhanças são classificadas pelo movimento descrito nos trabalhos e não pelos nomes dados pelos autores, já que alguns autores dão nomes diferentes para um mesmo movimento. A descrição de cada vizinhança será detalhada no Capítulo 4.

Grande parte dos trabalhos encontrados que utilizam busca local, não realizam nenhuma análise da contribuição de cada estrutura de vizinhança na busca realizada pelas heurísticas. Uma exceção é o trabalho de Santos et al. [2019] que faz uma análise do impacto de utilizar ou não cada uma das funções de vizinhança. Apesar dessa análise, os autores utilizam as seis estruturas de vizinhança com a mesma probabilidade de serem selecionadas. Nos capítulos seguintes, é feito um estudo detalhado de diferentes funções de vizinhança.

Tabela 3.1. Resumo dos métodos propostos pelos autores citados

Trabalhos	**Função objetivo**	**Métodos**
Guinet [1993]	C_{max}	Heurística construtiva baseada no método Húngaro
Gendreau et al. [2001]	C_{max}	Heurística construtiva baseada em divisão e mesclagem
Lee & Pinedo [1997]	$\sum w_i T_i$	Simulated annealing
Kurz & Askin [2001]	C_{max}	Slicing Multiple MULTI-FIT Multiple insertion Algoritmo genético
Pfund et al. [2008]	$\sum w_i T_i$	Heurística construtiva
França et al. [1996]	C_{max}	Busca Tabu
Rabadi et al. [2006]	C_{max}	Meta-RaPS
De Paula et al. [2007]	$C_{max} + \sum w_i T_i$	Variable Neighborhood Search Greedy Randomized Adaptive Search Procedure
Arnaout et al. [2010]	C_{max}	Colônia de Formigas
Ying et al. [2012]	C_{max}	Simulated annealing restrito
Vallada & Ruiz [2011]	C_{max}	Algoritmo genético (com busca local)
Cota et al. [2014]	C_{max}	Iterated Local Search + VND + Path-relinking
Martins Muller et al. [2014]	C_{max}	Multi start + PLIM
Santos et al. [2019]	C_{max}	Simulated annealing Iterated Local Search Late Acceptance Hill-climbing Step Counting Hill-climbing
Cota et al. [2018]	C_{max}, TEC	Método exato Smart Pool

Tabela 3.2. Estruturas de vizinhança utilizadas pelos autores para fazer busca local

Trabalhos	**Vizinhanças**					
	Shift	**Switch**	**Task Move**	**Swap**	**Two-shift**	**Direct swap**
Lee & Pinedo [1997]*		✓				
França et al. [1996]**			✓			
Arnaout et al. [2010]		✓	✓			
Ying et al. [2012]	✓		✓	✓	✓	
Rabadi et al. [2006]	✓		✓	✓		
De Paula et al. [2007]	✓		✓	✓		
Vallada & Ruiz [2011]	✓		✓			
Cota et al. [2014]		✓	✓			✓
Martins Muller et al. [2014]***	✓	✓	✓	✓	✓	✓
Santos et al. [2019]	✓	✓	✓	✓	✓	✓

* Utilizou também uma outra estrutura de vizinhança que troca subsequências de tarefas entre máquinas.
** Realizou uma variação do Task Move: a remoção e inserção das tarefas podem alterar o sequenciamento de outras tarefas da máquina.
*** Os movimentos modelados permitem os movimentos equivalentes aos indicados.

Parte II

Análise das estruturas de vizinhança

Capítulo 4

Estruturas de Vizinhança

A utilização de heurísticas com base em busca local é bastante comum para otimizar os problemas de programação de máquinas paralelas, principalmente aqueles que consideram os tempos de preparação. Nesse contexto, as funções de vizinhança desempenham um papel essencial em tais heurísticas.

Visando fazer uma análise exploratória e estatística visando caracterizar vizinhanças para o problema de programação de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e minimização do C_{max}, foram consideradas seis estruturas de vizinhança usadas frequentemente em heurística de busca local para este tipo de problema.

As funções de vizinhança consideradas são descritas e ilustradas a seguir.

4.1 Shift

Uma solução vizinha na estrutura de vizinhança *Shift* é gerada pela realocação de uma tarefa para outra posição da mesma máquina em que está atualmente atribuída. A Figura 4.1 ilustra um movimento realizado com a função *Shift*, na qual a tarefa J_1 é deslocada três posições para a direita na máquina m_k.

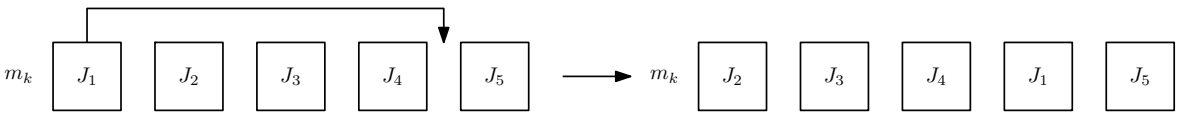

Figura 4.1. Estrutura de Vizinhança *Shift*.

4.2 Switch

Uma solução vizinha na estrutura de vizinhança *Switch* é caracterizada por trocar a posição de duas tarefas processadas em uma mesma máquina. A Figura 4.2 mostra um exemplo, na qual as posições das tarefas J_1 e J_4 são invertidas.

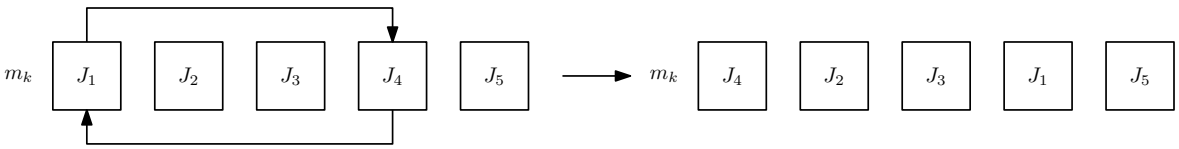

Figura 4.2. Estrutura de Vizinhança *Switch*.

4.3 Task move

Uma solução vizinha na vizinhança *Task move* é gerada pela movimentação de uma tarefa de uma máquina de origem para uma máquina de destino. A inserção dessa tarefa pode ser em qualquer posição na máquina de destino. A Figura 4.3 ilustra um movimento realizado com a função *Task move*, na qual a tarefa J_3 atualmente atribuída à máquina m_k, é deslocada para a máquina m_l.

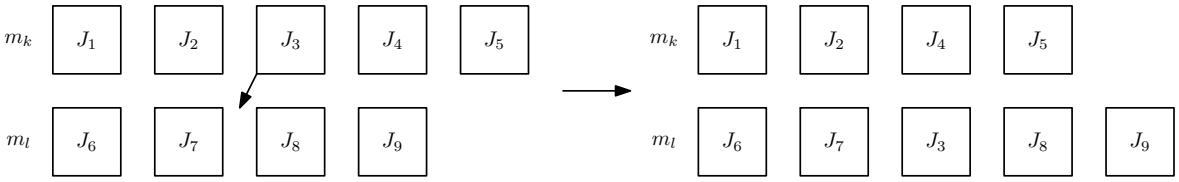

Figura 4.3. Estrutura de Vizinhança *Task move*.

4.4 Swap

Uma solução vizinha na vizinhança *Swap* é gerada trocando duas tarefas entre duas máquinas. Uma tarefa J_i, atualmente atribuída à uma máquina m_k, é realocada para uma máquina m_l; e uma tarefa J_j, atualmente atribuída à uma máquina m_l, é realocada para a máquina m_k. A inserção dessas tarefas podem ser em qualquer posição nas máquinas de destino. A Figura 4.4 mostra o movimento executado com a função *Swap*, na qual a tarefa J_1 é reatribuída da máquina m_k para a máquina m_l, e a tarefa J_9 é reatribuída da máquina m_l para a máquina m_k.

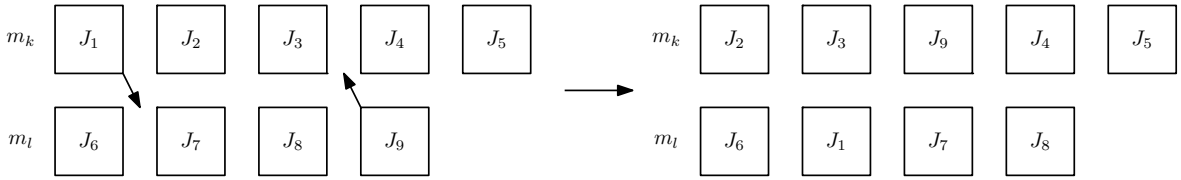

Figura 4.4. Estrutura de Vizinhança *Swap*.

4.5 Two-shift

Uma solução vizinha na vizinhança de *Two-shift* é gerada deslocando a posição de duas tarefas processados em uma mesma máquina. É equivalente a executar dois movimentos independentes da vizinhança *Shift*, mas assegurando que as duas tarefas movidas em cada movimento *Shift* não se repitam. A Figura 4.5 ilustra o movimento realizado com a função *Two-shift*, na qual a tarefa J_1 é deslocada duas posições para a direita e a tarefa J_5 é deslocada duas posições para a esquerda.

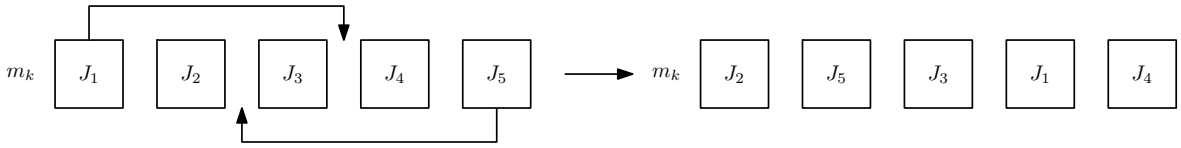

Figura 4.5. Estrutura de Vizinhança *Two-shift*.

4.6 Direct swap

Uma solução vizinha na vizinhança de *Direct swap* é gerada trocando duas tarefas entre duas máquinas, mantendo as posições anteriores nessas máquinas. A Figura 4.6 mostra um movimento realizado com a função *Direct swap*, na qual as tarefas J_1 e J_8 têm suas posições alteradas.

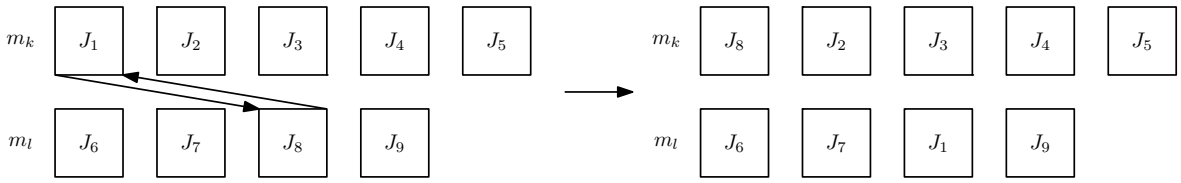

Figura 4.6. Estrutura de Vizinhança *Direct swap*.

4.7 Características das vizinhanças

Algumas características das vizinhanças estão resumidas na Tabela 4.1. A primeira coluna (*Vizinhança*) apresenta qual a vizinhança analisada, a segunda coluna (*Máquinas*) contém o número de máquinas envolvidas no movimento, a terceira coluna (*Cardinalidade*) mostra o número de soluções vizinhas dependendo do número de tarefas (n) e máquinas (m), a quarta coluna (*Complexidade*) mostra a complexidade do crescimento da vizinhança e a quinta coluna (*Altera Makespan*) é a proporção de soluções na vizinhança que alteram a máquina *makespan*. Os valores apresentados nas colunas *Cardinalidade*, *Complexidade* e *Altera Makespan* consideram uma vizinhança de uma solução com tarefas igualmente distribuídas entre as máquinas.

Tabela 4.1. Resumo de algumas características das funções de vizinhança

Vizinhança	Máquinas	Cardinalidade*	Complexidade*	Altera *Makespan*
Shift	1	$\frac{n^2 - nm}{m}$	$O(n^2)$	$\frac{1}{m}$
Switch	1	$\frac{-n^2}{2m} - \frac{n}{2}$	$O(n^2)$	$\frac{1}{m}$
Task Move	2	$\frac{-n^2}{m} + n^2 + nm - n$	$O(n^2)$	$\frac{2}{m}$
Swap	2	$\frac{n^4(m-1)}{2m^3}$	$O(n^4)$	$\frac{2m-2}{m^2-m}$
Direct Swap	2	$\frac{n^2(m-1)}{2m}$	$O(n^2)$	$\frac{2m-2}{m^2-m}$
Two-Shift	1	$\frac{n^4 - 2n^3m + n^2m^2}{2m^3}$	$O(n^4)$	$\frac{1}{m}$

* n é o número de tarefas e m é o número de máquinas disponíveis.

Como mencionado anteriormente, está claro na Tabela 4.1 que para a classe de problema $R|s_{ijk}|C_{max}$ a proporção de soluções que resultam em qualquer mudança na máquina *makespan* (e, consequentemente, que pode ter qualquer expectativa de gerar uma melhoria para a solução corrente) é baixa. Para investigar como isso pode afetar o desempenho de meta-heurísticas usadas para a solução dessa classe de problema em particular, foi realizada uma investigação experimental sobre a utilidade esperada de um único movimento usando cada uma das seis estruturas de vizinhança mencionadas acima, como uma função não apenas do tamanho do problema (número de máquinas e tarefas), mas também em função do momento em que o movimento é realizado. Os

resultados desta investigação foram então usados para propor uma estratégia baseada na utilidade esperada para adaptar as probabilidades de usar cada estrutura de vizinhança ao longo da busca realizada pelo *Simulated Annealing* de [Santos et al., 2019], que é o estado-da-arte para este problema. Esta estratégia é descrita e avaliada nos Capítulos 5, 6 e 7.

Capítulo 5

Protocolo Experimental

Este capítulo apresenta o protocolo experimental realizado para que fossem feitas as análises exploratórias e estatísticas para o experimento de caracterização das estruturas de vizinhança.

Neste experimento, seis estruturas de vizinhança são exploradas em diferentes estágios de busca de várias instâncias do problema de sequenciamento, com a busca sendo conduzida usando uma implementação própria do *Simulated Annealing*, baseado no código de Santos et al. [2019].[1] As estruturas de vizinhança exploradas neste experimento foram as seis estruturas detalhadas no Capítulo 4: *Direct Swap, Shift, Swap, Switch, Task Move* e *Two-Shift*.

O algoritmo foi executado nas 200 instâncias de *tuning* [SOA-ITI, 2013] propostas por Vallada & Ruiz [2011], com dimensões dadas por todas as combinações de:

- $m \in \{10, 15, 20, 25, 30\}$ máquinas e $n \in \{50, 100, 150, 200, 250\}$ tarefas.

Para cada tamanho de instância (definido por um par (m, n)), havia oito instâncias distintas disponíveis. O algoritmo foi executado uma vez em cada instância, resultando em oito observações independentes para cada par (m,n).

Em cada execução, o seguinte protocolo foi usado para coletar dados: a todo momento que a solução incumbente fosse alterada, uma enumeração exaustiva era realizada para cada estrutura de vizinhança. Os seguintes dados foram coletados:

- O tempo em que o ponto de dados específico foi coletado.
- Iteração na qual o ponto de dados específico foi coletado.
- Valor do C_{max} da solução incumbente.

[1] Os códigos dos experimentos realizados estão disponíveis on-line: `https://github.com/andremaravilha/mestrado-leticia`.

- O valor de $\sum C^k_{max}$ para a solução incumbente.
- Tamanho da vizinhança (cardinalidade).
- Contagem de vizinhos que pertencem a cada uma das oito classes de comparação listadas na Tabela 2.1.
- Estatísticas resumidas para cada classe na Tabela 2.1:
  - Melhor, média e pior alterações em C_{max}.
  - Melhor, média e pior alterações em $\sum C^k_{max}$.

As alterações foram medidas como $\Delta = f(x) - f(x^*)$, onde x é uma solução vizinha, x^* é a solução incumbente e $f(.)$ podendo ser C_{max} ou $\sum C^k_{max}$. Por esse motivo, valores negativos no quadro de dados representam melhorias reais, enquanto os valores positivos representam a degradação do desempenho.

Este protocolo foi desenvolvido para se permitir investigar algumas questões sobre a qualidade relativa das estruturas de vizinhança ao longo do processo de busca, em termos de:

- Proporção de soluções vizinhas que proporcionam melhorias na solução.
- Magnitude esperada de melhorias observadas

5.1 Proporção de soluções vizinhas que proporcionam melhorias na solução

A relação de soluções em uma determinada vizinhança que representam melhorias em relação à solução incumbente, é dada em dois critérios:

- Melhoria primária: menor valor de C_{max} (classe *Melhoria-1* na Tabela 2.1).
- Melhoria secundária: valor igual de C_{max} e menor valor de $\sum C^k_{max}$ (classe *Melhoria-2* na Tabela 2.1).

Estas proporções representam a probabilidade de observar uma melhoria após um único movimento, para cada estrutura de vizinhança. Seja $H = \{\eta_i\}$ o conjunto de funções de vizinhança, $\mathcal{N}_i = \{x : x \in \eta_i(x^*)\}$ o conjunto de vizinhos de uma determinada solução incumbente x^*, de acordo com a i-ésima estrutura de vizinhança η_i, e $\mathcal{N}^*_i \subset \mathcal{N}_i$ representando o subconjunto de vizinhos que melhoram a solução incumbente. Então:

$$Pr[melhoria_i] = \frac{|\mathcal{N}^*_i|}{|\mathcal{N}_i|} \tag{5.1}$$

onde $melhoria_i$ é uma variável binomial definida como:

$$melhoria_i(x) = \begin{cases} 1, & \text{se} f(x) < f(x^*) \\ 0, & \text{caso contrário} \end{cases} \tag{5.2}$$

e $f(\cdot)$ indica C_{max} ou $\sum C^k_{max}$ para uma dada solução candidata.

5.2 Magnitude esperada de melhorias observadas

Para os vizinhos que representam uma melhoria, a Equação (5.3) mede o quanto a variável em questão é melhorada. Por exemplo, se for utilizada melhorias percentuais como nossa medida:

$$\begin{aligned} E\left[\Delta_{\%} | melhoria_i\right] &= E\left[\frac{f(x) - f(x^*)}{f(x^*)} \Big| f(x) < f(x^*)\right] \\ &= \frac{1}{|\mathcal{N}_i^*|} \sum_{x \in \mathcal{N}_i^*} \left[\frac{f(x) - f(x^*)}{f(x^*)}\right] \end{aligned} \tag{5.3}$$

5.3 Utilidade esperada das melhorias

Dada a probabilidade estimada de melhoria e a magnitude observada de melhorias, a utilidade esperada de uma determinada vizinhança, ou seja, a melhoria esperada após uma única perturbação de acordo com aquela vizinhança é dada pela Equação (5.4):

$$\begin{aligned} E\left[u_i\right] &= E\left[\Delta_{\%} | melhoria_i\right] \times Pr\left[melhoria_i\right] \\ &= \frac{1}{|\mathcal{N}_i|} \sum_{x \in \mathcal{N}_i^*} \left[\frac{f(x) - f(x^*)}{f(x^*)}\right] \end{aligned} \tag{5.4}$$

Essas questões foram investigadas para o grupo de instâncias de *tuning*. Além de investigar a qualidade relativa das vizinhanças e suas mudanças à medida que a busca avança, outra questão de interesse era investigar a relação do número de máquinas e do número de tarefas na proporção e na magnitude das melhorias.

Os resultados obtidos com a realização deste protocolo experimental e a análise gráfica são apresentados e discutidos no Capítulo 6.

Capítulo 6

Análise Exploratória e Discussão dos Resultados

Neste capítulo é realizada a análise gráfica exploratória dos dados, de acordo com os protocolos e questões de interesse descritos no Capítulo 5. As questões de interesse serão seguidas de acordo com a sequência apresentada no Capítulo 5.

6.1 Análise exploratória referente ao C_{max}

Primeiramente, foi investigado o comportamento em relação à probabilidade de melhorias, $Pr[melhoria_i]$, em relação ao C_{max} (Classe Melhoria-1 na Tabela 2.1), para cada estrutura de vizinhança, em função do tempo (normalizado entre 0 e 1), para instâncias de teste. A Figura 6.1 mostra esse comportamento. A Figura 6.2 mostra gráficos semelhantes que foram gerados para a melhoria esperada, $E[\Delta_{\%} \mid melhoria_i]$.

A partir desses dois gráficos, pode-se começar a perceber um padrão: instâncias menores ($n = 50$) tendem a não apresentar nenhuma característica interessante, ou seja, nenhuma diferença sistemática óbvia entre estruturas de vizinhança, e nenhuma tendência aparente à medida que o tempo avança.

As instâncias maiores ($n > 50$), por outro lado, parecem sugerir alguns padrões interessantes: a estrutura de vizinhança *Task Move* parece ser claramente melhor que as demais, independentemente do momento em que a busca é realizada. Além disso, o número de máquinas e o número de tarefas parecem ter pouco efeito tanto na probabilidade quanto na magnitude esperada das melhorias.

A Figura 6.3 apresenta os gráficos das utilidades esperadas de cada vizinhança, que realmente é a questão de interesse.

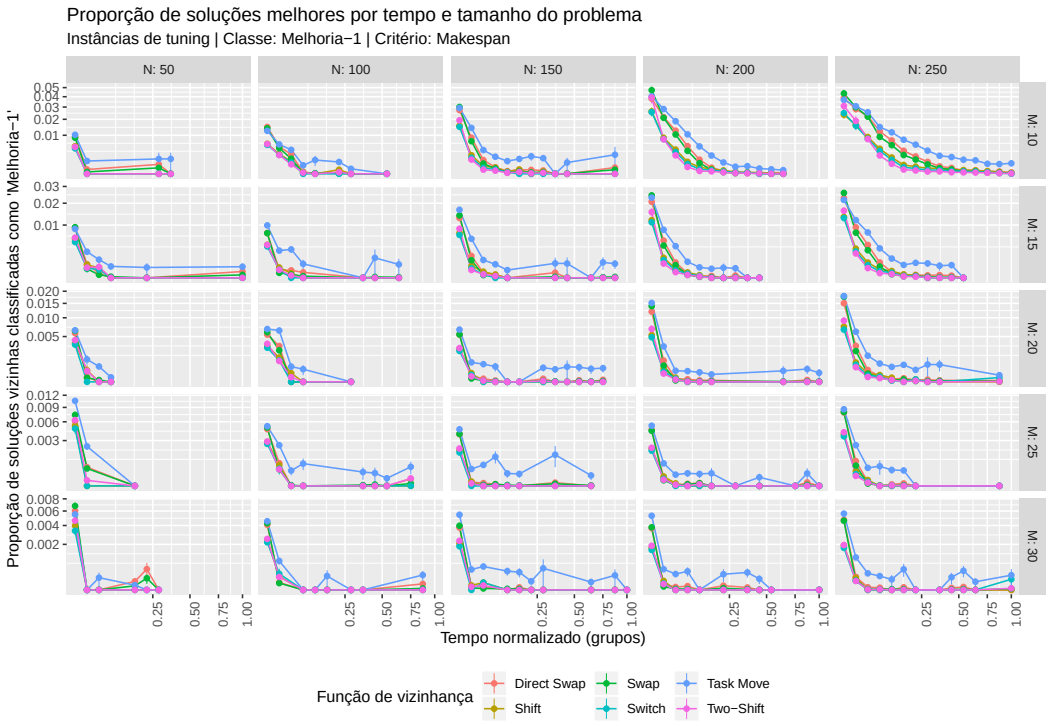

Figura 6.1. Proporção de soluções classificadas por Melhoria-1 por tempo e tamanho do problema

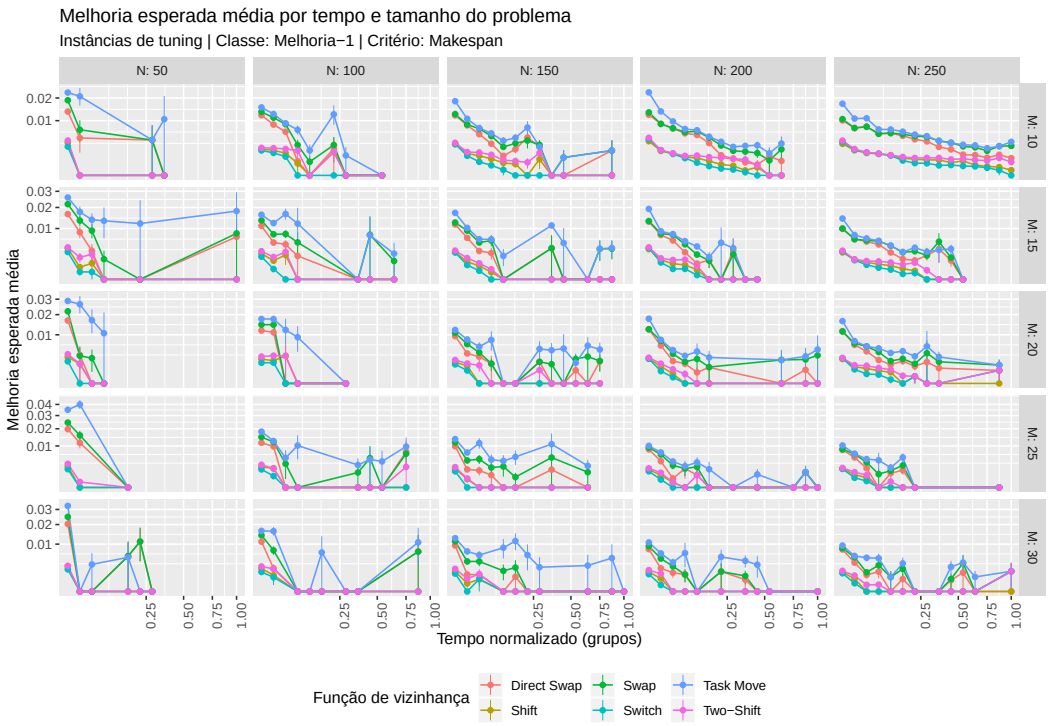

Figura 6.2. Melhoria esperada média classificada por Melhoria-1 por tempo e tamanho do problema

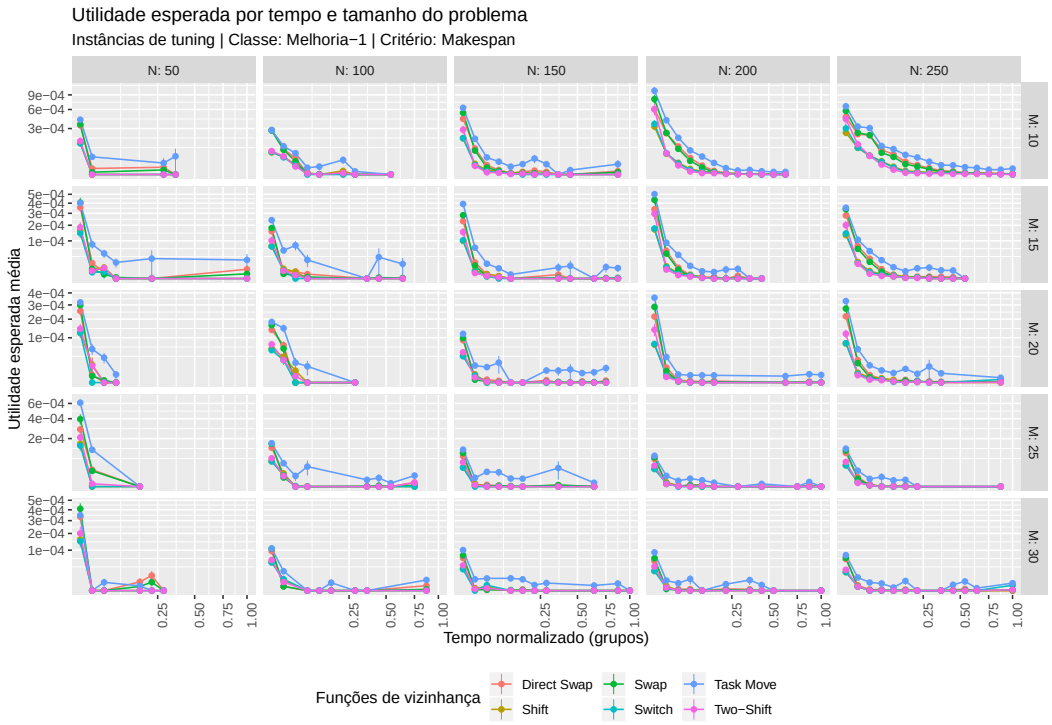

Figura 6.3. Utilidade esperada por tempo e tamanho do problema considerando o critério *makespan*

Como foi sugerido pelos gráficos anteriores, a utilidade esperada dos movimentos, de acordo com a estrutura da vizinhança *Task Move* é nitidamente superior às outras vizinhanças. As utilidades esperadas também são maiores nos estágios iniciais do processo de otimização e diminuem à medida que o tempo avança, o que concorda com a ideia intuitiva de que há mais possibilidades de melhoria da solução incumbente nas primeiras iterações do algoritmo, uma vez que a qualidade dessas soluções ainda pode ser muito melhorada.

6.2 Análise exploratória referente a $\sum C^k_{max}$

Para realizar a análise exploratória do critério de melhoria secundária, basta replicar os métodos utilizados anteriormente, alterando a classe de melhoria de "Melhoria-1" para "Melhoria-2" e os dados relacionados ao C_{max} para aqueles relacionados a $\sum C^k_{max}$.

Primeiro, a Figura 6.4 mostra os gráficos em relação à probabilidade de melhorias, $Pr[melhoria_i]$. Para a melhoria esperada, $E[\Delta_{\%} \mid melhoria_i]$, a Figura 6.5 apresenta os gráficos obtidos. E finalmente, para a utilidade esperada do critério da soma dos tempos de conclusão das máquinas, a Figura 6.6 apresenta os resultados.

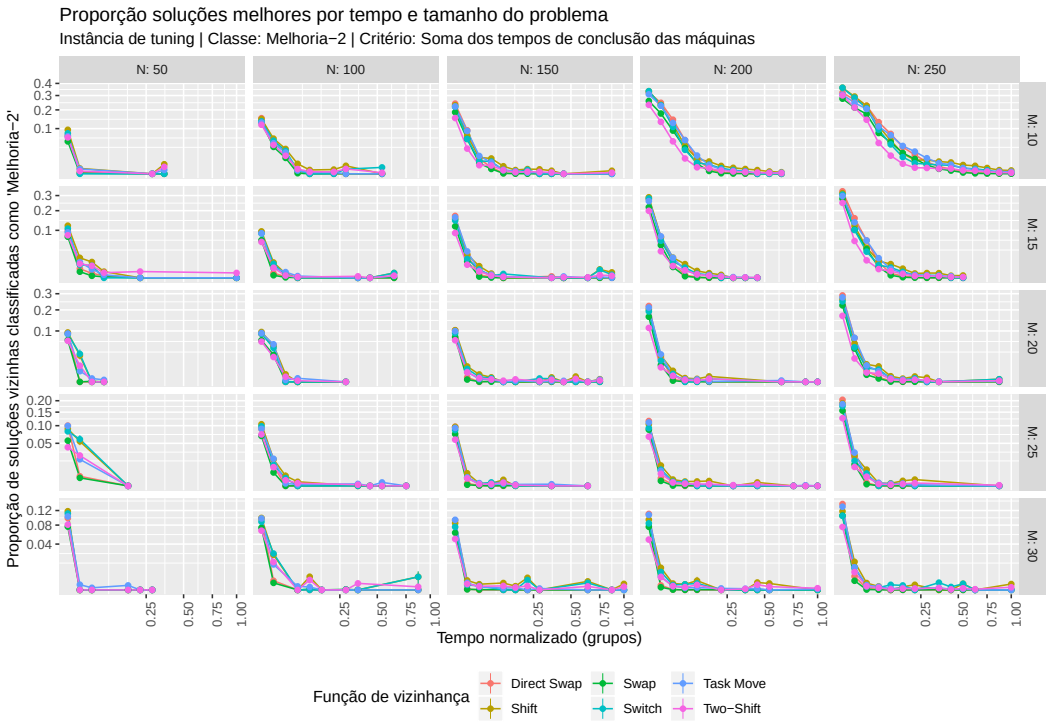

Figura 6.4. Proporção de soluções classificadas por Melhoria-2 por tempo e tamanho do problema

Figura 6.5. Melhoria esperada média classificada por Melhoria-2 por tempo e tamanho do problema

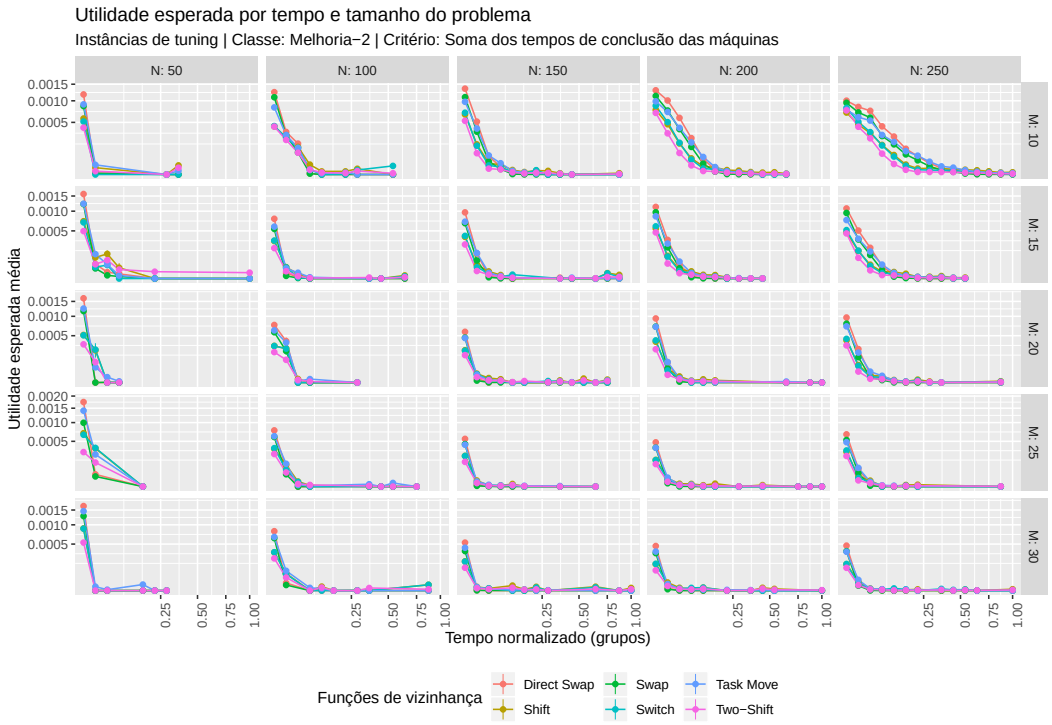

Figura 6.6. Utilidade esperada por tempo e tamanho do problema considerando o critério da soma do tempo de conclusão

Como no primeiro critério, as utilidades esperadas são mais altas nos estágios iniciais da busca e tendem a diminuir rapidamente à medida que a otimização avança. No entanto, ao contrário da situação observada para o C_{max}, parece haver pouco ou nenhum efeito da estrutura da vizinhança sobre as utilidades esperadas em relação ao critério secundário.

Dada essa análise, uma questão interessante, que será investigada no próximo capítulo é como explorar essas observações para tentar melhorar o desempenho de um *Simulated Annealing* para problemas de sequenciamento.

Para isso, a partir dos resultados obtidos nessa análise exploratória, foi desenvolvido um modelo de regressão linear para predizer a utilidade esperada de cada função de vizinhança e, com isso, determinar a probabilidade de escolher cada uma dessas funções ao longo do *Simulated Annealing*. Portanto, foi implementado uma versão modificada do *Simulated Annealing* proposto por Santos et al. [2019], no qual a probabilidade de se escolher uma dada vizinhança é determinada de acordo com suas utilidades relativas, e não totalmente aleatório como era feito. O Capítulo 7 detalha as mudanças feitas e os resultados dessa investigação.

Parte III

Algoritmo proposto

Capítulo 7

Algoritmos utilizados para solução

7.1 Simulated Annealing

O algoritmo Simulated Annealing (SA), proposto por Kirkpatrick et al. [1983], é uma meta-heurística de busca local probabilística. O SA baseia-se em uma analogia com a termodinâmica ao simular o resfriamento de um conjunto de átomos aquecidos, operação que é conhecida como recozimento.

O SA consiste em um processo iterativo que gera, a cada iteração, uma solução vizinha $x \in \eta_i(x')$ a partir da solução corrente x'. Esta técnica começa sua busca a partir de uma solução inicial qualquer x^0. Seja $\Delta = f(x) - f(x')$ a variação na função objetivo obtida a partir do movimento realizado por uma função de vizinhança η_i. Se $\Delta \leq 0$, o método aceita imediatamente a solução candidata. Caso $\Delta > 0$, a solução vizinha candidata pode ser aceita com uma probabilidade $e^{-\Delta/\tau}$, onde τ é um parâmetro chamado de temperatura, que regula a probabilidade de se aceitar soluções de pior custo.

A temperatura é inicialmente definida para um valor τ_0 e, após um número fixo de iterações, a temperatura é gradualmente reduzida por uma taxa de resfriamento α.

Utilizando este procedimento, há uma maior chance de evitar que o algoritmo fique estagnado em ótimos locais nas iterações iniciais e, à medida que τ se aproxima de zero, o algoritmo reduz a probabilidade de aceitação de soluções de piora. Para evitar a estagnação, o reaquecimento pode ser realizado quando a temperatura atingir um limite mínimo ϵ.

O Algoritmo 7.1 apresenta o pseudocódigo do SA implementado por Santos et al. [2019] para o problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência, onde $f_{MS}(\cdot)$ é C_{max}. O SA requer os seguintes parâmetros: x^0 é uma solução inicial; τ_0 é a temperatura inicial; α é a taxa

de resfriamento; sa_{max} é o número de iterações processadas em cada temperatura; e $H = \{\eta_i\}$ é o conjunto de funções de vizinhança.

Algoritmo 7.1: Simulated Annealing (SA) proposto por Santos et al. [2019]

Entrada: $x^0,\ \tau_0,\ \alpha,\ sa_{max},\ H$
Saída : x^*

$x^* \leftarrow x^0$;
$x' \leftarrow x^0$;
$\tau \leftarrow \tau_0$;
enquanto *o tempo limite não é atingido* **faça**
  para $i \leftarrow 1$ **até** sa_{max} **faça**
    Escolha $\eta_i \in H$ aleatoriamente;
    $x \leftarrow \eta_i(x')$;
    $\Delta \leftarrow f_{MS}(x) - f_{MS}(x')$;
    se $\Delta \leq 0$ **então**
      $x' \leftarrow x$;
      se $f_{MS}(x) < f_{MS}(x^*)$ **então** $x^* \leftarrow x$;
    senão
      Escolha $r \in U(0,1)$ aleatório;
      se $r < e^{-\Delta/\tau}$ **então** $x' \leftarrow x$;
  $\tau \leftarrow \alpha \times \tau$;
  se $\tau < \epsilon$ **então** $\tau \leftarrow \tau_0$;

7.2 Simulated Annealing modificado

Com base nos resultados apresentados no Capítulo 6 foi construído um modelo de regressão linear múltipla (Eq. 7.1) para predizer a utilidade esperada de cada vizinhança η_i ao longo do SA na solução de uma instância de m máquinas com n tarefas no momento t, sendo t o tempo normalizado entre 0 e 1:

$$y_{i,t} = \beta_0 + \beta_{1,i} + \beta_2 z + \beta_3 n + \beta_4 m + \beta_5 z^2 + \beta_6 z^3 + \beta_7 z^4 + \beta_8 zn + \beta_9 zm + \beta_{10} nm + \epsilon, \quad (7.1)$$

onde $y_{i,t}$ é a estimativa de $\hat{u_i}^{1/4}$ no momento t. A variável z é uma transformação da variável t dada por $z = t^{1/4}$ e ϵ é a variabilidade não explicada pelo modelo. Através da análise gráfica da utilidade esperada u_i das funções de vizinhança, foi constatado que u_i apresenta um decaimento exponencial, e não linear, em função do tempo t (Figura 6.3). Assim, através de testes preliminares para determinação do modelo de regressão

Coeficiente	Estimativa
β_0	4.368E-01
β_2	-2.041E+00
β_3	6.379E-04
β_4	-3.196E-03
β_5	3.897E+00
β_6	-3.452E+00
β_7	1.157E+00
β_8	-4.513E-04
β_9	6.495E-03
β_{10}	-1.426E-05
$\beta_{1,DirectSwap}$	-3.002E-02
$\beta_{1,Shift}$	-4.587E-02
$\beta_{1,Swap}$	-3.259E-02
$\beta_{1,Switch}$	-4.931E-02
$\beta_{1,TwoShift}$	-4.635E-02

linear, as transformações de t em $z = t^{1/4}$ e $\hat{u}_i$ em $y_{i,t} = \hat{u}_i$ possibilitaram ao modelo explicar melhor a variabilidade dos dados.

Os coeficientes do modelo foram ajustados a partir dos mesmos dados obtidos para a análise exploratória, utilizando a função `lm` da linguagem R (versão 3.4.4). Os valores retornados para os coeficientes são apresentados na Tabela 7.2.

Note que a Tabela 7.2 não apresenta o coeficiente $\beta_{1,TaskMove}$, pois o efeito da vizinhança *Task Move* sobre o valor predito pelo modelo está incorporado em β_0. Com isso, os coeficientes das demais vizinhanças representam a diferença entre elas e o *Task Move*. O *Task Move* foi utilizado como base por representar a vizinhança com melhor desempenho na análise exploratória realizada e, a partir dos coeficientes $\beta_{1,i}$ pode-se observar a diferença estimada no desempenho de cada uma das vizinhanças em relação ao *Task Move*.

O valor do coeficiente de determinação ajustado foi de $R^2_{ajustado} = 0.83$, o que indica que o modelo ajustado consegue explicar cerca de 83% da variabilidade dos dados.

Utilizando esse modelo de regressão, o algoritmo SA proposto por Santos et al. [2019] foi modificado para que as probabilidades de escolha de cada vizinhança pudessem ser determinada ao longo da sua execução. Em análises preliminares, observou-se que o algoritmo apresentava uma convergência prematura, ficando estagnado em soluções ótimas locais. Isso pode ser explicado pelo fato do *Task Move* assumir uma probabilidade alta (cerca de 98%), dificultando a capacidade do SA de sair de regiões de mínimos locais utilizando a mesma estrutura de vizinhança que levou até esse

ponto. Esse comportamento será apresentado na análise dos resultados dos algoritmos, na Seção 7.3.

Para evitar esse problema, optou-se por estabelecer um valor máximo de probabilidade de escolha que uma vizinhança poderia assumir. Assim, a probabilidade máxima é dada por $Pr_{max} \in \left[\frac{1}{|H|},\ 1\right]$. Com isso, a probabilidade de escolha de cada vizinhança η_i é dada por:

$$Pr_i = \frac{1 - Pr_{max} + (|H|Pr_{max} - 1)w_i}{|H| - 1}, \quad \text{com } w_i = \frac{\hat{u_i}}{\sum_j \hat{u_j}} \tag{7.2}$$

O Algoritmo 7.2 apresenta a estrutura geral do SA modificado, onde $f_{ST}(.)$ é $\sum C^k_{max}$ e $f_{MS}(.)$ é C_{max}. Observe que, nessa versão modificada a escolha da vizinhança é realizada de acordo com as probabilidades calculadas pela Eq. (7.2).

Além disso, o SA modificado utiliza o critério secundário $\sum C^k_{max}$ para atualização da solução corrente x', evitando que o algoritmo fique estagnado em uma mesma solução corrente por muitas iterações devido as grande número de regiões planas em C_{max} (soluções vizinhas com mesmo valor de C_{max}). Devido a isso, a atualização da solução incumbente x^* é realizada antes de atualizar a solução corrente (linha 8), evitando que soluções com melhor valor de C_{max} seja descartado erroneamente, já que podem podem existir soluções que melhoram o C_{max} apesar de piorarem o $\sum C^k_{max}$.

7.3 Resultados obtidos

Para validar as alterações propostas, foram realizados novos experimentos considerando as 1640 diferentes instâncias de testes [SOA-ITI, 2013], propostas por Vallada & Ruiz [2011], divididas em dois grupos:

- Instâncias pequenas: $m \in \{2, 3, 4, 5\}$ e $n \in \{6, 8, 10, 12\}$
- Instâncias grandes: $m \in \{10, 15, 20, 25, 30\}$ e $n \in \{50, 100, 150, 200, 250\}$

Os testes foram realizados em uma máquina executando o sistema operacional Linux Ubuntu 18.04 (64 bits), com dois processadores Intel(R) Xeon(R) Silver 4116 e 156GiB de memória principal. Todos os algoritmos foram implementados utilizando a linguagem de programação Java (versão 8) e todas as análises foram feitas utilizando o R (versão 3.4.4).

Como no algoritmo modificado foi introduzido o parâmetro Pr_{max}, que limita a probabilidade máxima para a escolha das vizinhanças, foram considerados quatro diferentes configurações, sendo $Pr_{max} \in \{0,50;\ 0,75;\ 0,85;\ 1,00\}$, sendo que o valor de

Algoritmo 7.2: Simulated Annealing (SA) modificado

Entrada: $x^0,\ \tau_0,\ \alpha,\ sa_{max},\ H,\ Pr_{max}$
Saída : x^*

```
1  x* ← x^0;
2  x' ← x^0;
3  τ ← τ_0;
4  enquanto o tempo limite não é atingido faça
5      para i ← 1 até sa_max faça
6          Escolha η_i ∈ H com probabilidade dada por Eq. (7.2);
7          x ← η_i(x');
8          se f_MS(x) < f_MS(x*) então x* ← x;
9          Δ ← f_ST(x) − f_ST(x');
10         se Δ ≤ 0 então
11             x' ← x;
12         senão
13             Escolha r ∈ U(0,1) aleatório;
14             se r < e^{−Δ/τ} então x' ← x;
15     τ ← α × τ;
16     se τ < ϵ então τ ← τ_0;
```

0,85 foi obtido através do *irace* [López-Ibáñez et al., 2016] para ajuste desse parâmetro. Os demais parâmetros do SA ($\tau_0 = 1,0;\ \alpha = 0,85;\ sa_{max} = 1.776.628$) foram mantidos iguais aos definidos no trabalho de Santos et al. [2019]. Além do algoritmo modificado, foi considerada também uma implementação própria do SA original de Santos et al. [2019] para fins de comparação.

Para cada um dos cinco algoritmos, cada uma das 1640 instâncias de testes foram executadas 10 vezes, com diferentes valores de sementes para o gerador de números aleatórios. Em cada execução foram coletados o valor de C_{max} da melhor solução encontrada.

Para as instâncias pequenas não foi encontrada nenhuma diferença significativa entre os algoritmos. Por outro lado, ao analisar as instâncias grandes, pode-se observar uma diferença no desempenho dos algoritmos testados, conforme apresentado na Figura 7.1. Note que, a configuração que não limita a probabilidade máxima das vizinhanças (Adapt(1.00), $Pr_{max} = 1$) apresenta um desempenho pior em relação ao algoritmo original, como discutido anteriormente. Em relação ao Adapt(0.50), uma sutil melhora para problemas com $n \geq 150$ é observada, porém nenhum padrão é observado em instâncias menores. Já as configurações Adapt(0.75) e Adapt(0.85) apresentam um desempenho consistentemente superior ao algoritmo original.

Além da análise gráfica, foi realizado um teste de hipótese para verificar a existência de diferenças significativas no valor de *makespan* retornado pelos algoritmos, sendo os diferentes algoritmos os fatores de interesse e a dimensão do problema os fatores de blocagem. A hipótese nula H_0 é a ausência de diferenças entre os algoritmos e a hipótese alternativa H_1 é a existência de pelo menos uma diferença (teste bilateral). O teste foi realizado através da ANOVA [Montgomery & Runger, 2013] considerando um nível de confiança de 95%, tendo um valor-p de 2×10^{-16}.

Dada a existência de diferenças significativas no desempenho dos algoritmos, foram calculados os intervalos de confiança para as diferenças entre cada configuração da modificação do SA contra a versão original. Os valores dos intervalos de confiança (de 95%) são apresentados na Figura 7.2. Nela pode-se constatar que a configuração Adapt(1.00) é pior que o SA original, apresentando uma perda de desempenho de 1,25%. Já para a configuração Adapt(0.50) não foi constatado diferença significativa em relação ao SA original. Em relação as configurações Adapt(0.75) e Adapt(0.85) observa-se uma melhoria no valor do *makespan* de 1,25%, sendo essas estatisticamente significativas.

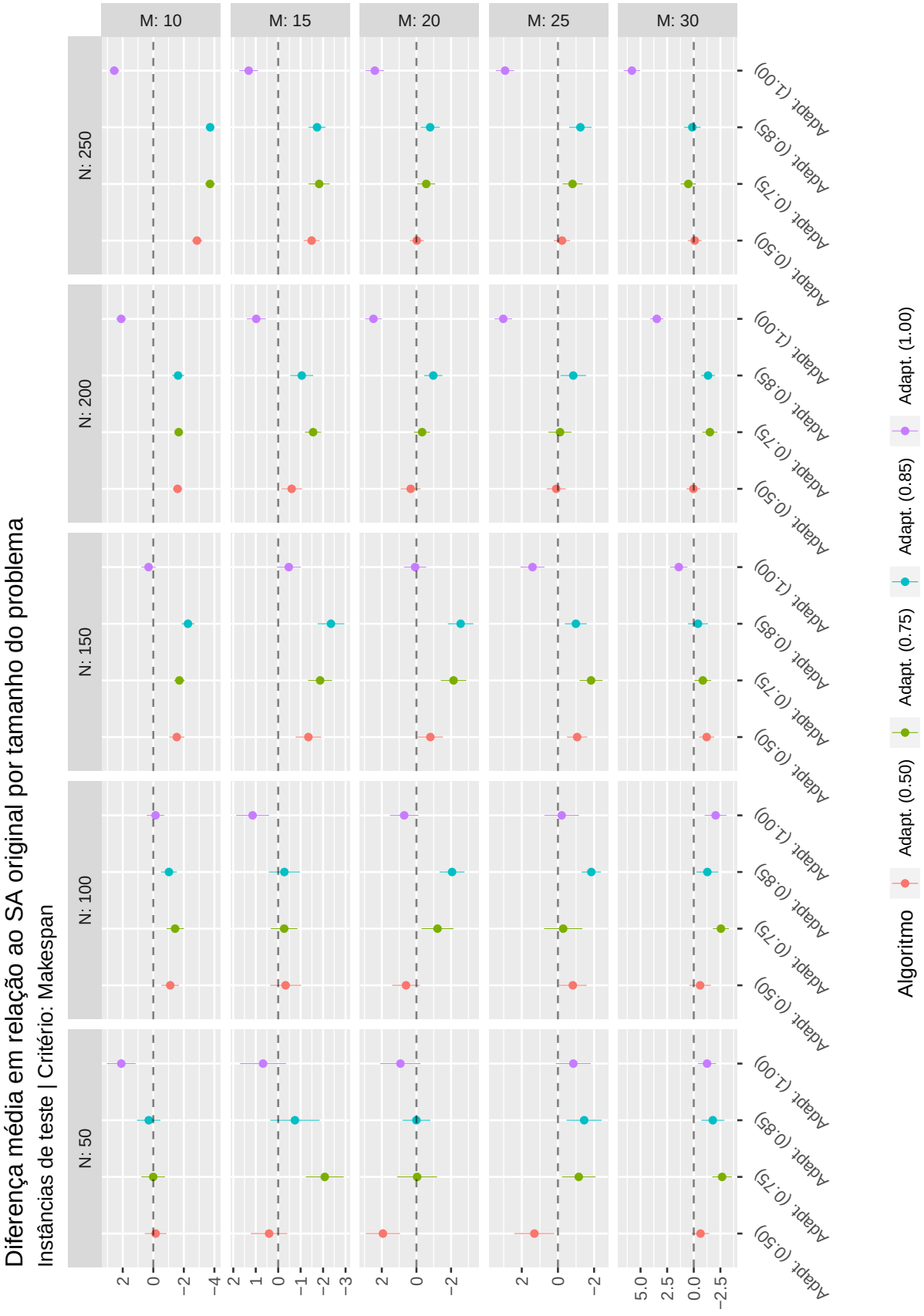

Figura 7.1. Diferença média(%) do *makespan* das soluções retornadas pelas diferentes configurações do SA modificado, em relação ao SA original. Os resultados estão estratificados por número de máquinas e tarefas.

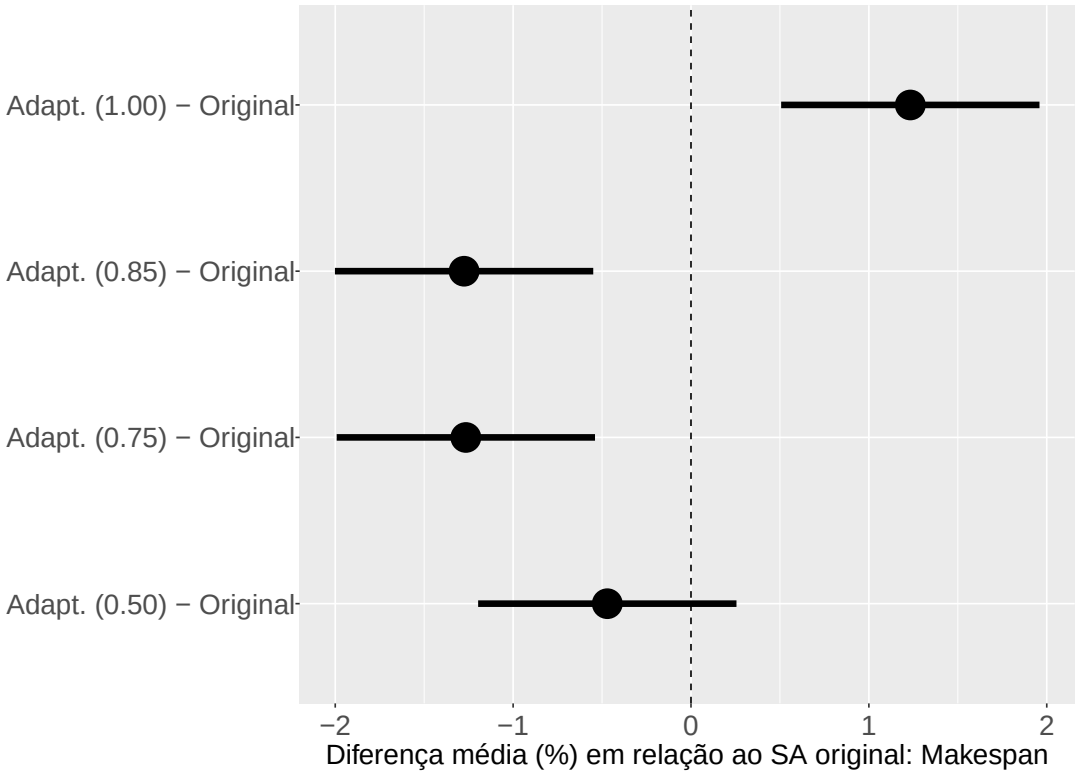

Figura 7.2. Intervalos de confiança de 95% das diferenças médias (%) do *makespan* em relação ao SA original.

Parte IV

Considerações finais e proposta de continuidade

Capítulo 8

Considerações Finais

Neste capítulo são feitas as considerações finais da dissertação, ressaltando os benefícios da análise exploratória e estatística que foi realizada para caracterizar seis estruturas de vizinhança. Também são apresentadas sugestões para continuidade do trabalho realizado.

8.1 Conclusões

Na resolução de problemas de sequenciamento de máquinas paralelas, a utilização de heurísticas baseadas em busca local é comumente aplicada para buscar boas soluções. Nesse contexto, as estruturas de vizinhança exercem uma função relevante na capacidade de explorar adequadamente o espaço dessas soluções. A partir disso, esse trabalho apresentou uma análise de estruturas de vizinhança para o problema de sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependente da sequência, buscando a minimização do *makespan* como objetivo principal.

Para fazer essa análise, foram utilizadas seis estruturas de vizinhança que são frequentemente utilizadas para tratar esse tipo de problema, as quais foram detalhadas e ilustradas. As seis estruturas de vizinhança foram exploradas em diferentes estágios de busca de várias instâncias de *tuning* do problema de sequenciamento, com a busca sendo conduzida usando uma implementação própria do *Simulated Annealing*. A implementação foi baseada num código que é o estado da arte para solucionar essa classe de problema. Foi investigado o comportamento em relação à probabilidade de melhoria, melhoria esperada e utilidades esperadas para cada estrutura de vizinhança em relação a dois critérios: *makespan* e a soma do tempo de conclusão de todas as máquinas. Ao analisar o critério *makespan*, as instâncias menores não apresentaram nenhuma característica interessante. Para as instâncias maiores, a estrutura de vizinhança *Task Move*

se mostrou notoriamente melhor que as demais analisadas, constatando que o número de máquinas e o número de tarefas aparentam ter pouco efeito tanto na probabilidade quanto na magnitude esperada das melhorias. Para o segundo critério, houve pouco efeito da estrutura da vizinhança sobre as utilidades esperadas.

Um ponto importante que foi investigado a partir dessas análises, foi como explorar essas investigações para tentar melhorar o desempenho do SA para essa classe de problemas. A partir dos experimentos, foi desenvolvido um modelo de regressão linear para predizer a utilidade esperada de cada uma dessas funções de vizinhança utilizadas, para assim definir a probabilidade de escolher cada uma dessa funções ao longo do algoritmo. Com isso, foi implementada uma versão modificada do algoritmo original, onde a probabilidade de se escolher uma dada vizinhança é determinada de acordo com suas utilidades relativas e não aleatoriamente como era feito.

Os testes foram realizados em 1640 instâncias de teste, sendo executadas nos cinco algoritmos (quatro configurações de limite do parâmetro Pr_{max} juntamente com o modelo de regressão, e o algoritmo original). Os resultados obtidos com a modificação na escolha das vizinhanças foi superior à versão original do algoritmo em duas configurações testadas, uma das probabilidades definidas foi inferior ao original e a outra não mostrou diferença significativa quando comparadas. As duas configurações que se mostraram superior ao original, obtiveram uma melhoria de 1,25% aproximadamente, no valor do *makespan*.

Com base nesse trabalho, os autores que forem trabalhar com heurísticas baseadas em busca local poderão aproveitar os resultados das análises realizadas. Com isso, poderão priorizar a escolha das vizinhanças e determinar quando utilizá-las ao longo do processo de otimização para acelerar a convergência para soluções melhores, economizando assim tempo e testes desnecessários.

8.2 Trabalhos Futuros

Os trabalhos futuros a partir desse trabalho incluem algumas possibilidades. Uma delas é o caso de testar as probabilidades definidas com o modelo de regressão em outros algoritmos diferentes do *Simulated Annealing*.

Outra possibilidade é considerar a busca local que gerou a solução corrente como um parâmetro na construção do modelo de regressão, uma vez que podem existir sobreposições das estruturas de vizinhança.

Além das questões apresentadas, outra possibilidade é expandir para outros tipos de problemas de sequenciamento e utilizar o mesmo protocolo experimental em outras

classes de problemas diferentes de sequenciamento.

Referências Bibliográficas

Al-Salem, A. (2004). Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times. *Engineering Journal of the University of Qatar*, 17(1):177--187.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345--378.

Allahverdi, A.; Ng, C.; Cheng, T. E. & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985--1032.

Arnaout, J.-P.; Rabadi, G. & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, 21(6):693--701.

Avalos-Rosales, O.; Angel-Bello, F. & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 76(9-12):1705--1718.

Baxter, J. (1981). Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32(9):815--819.

Burke, E. K. & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. Em *PATAT 2008 Conference, Montreal, Canada.*

Bykov, Y. & Petrovic, S. (2013). An initial study of a novel step counting hill climbing heuristic applied to timetabling problems. Em *Proceedings of 6th Multidisciplinary International Scheduling Conference (MISTA 2013).*

Cota, L. P.; Coelho, V. N.; Guimarães, F. G. & Souza, M. J. (2018). Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research.*

Cota, L. P. a.; Haddad, M. N.; Souza, M. J. F. & Coelho, V. N. (2014). AIRP: A heuristic algorithm for solving the unrelated parallel machine scheduling problem. Em *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 1855--1862. IEEE.

De Paula, M. R.; Ravetti, M. G.; Mateus, G. R. & Pardalos, P. M. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. *IMA Journal of Management Mathematics*, 18(2):101--115.

Dorigo, M. & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. Em *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pp. 1470--1477. IEEE.

Fanjul-Peyro, L.; Ruiz, R. & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, 101:173--182.

Feo, T. A. & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109--133.

França, P. M.; Gendreau, M.; Laporte, G. & Müller, F. M. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43(2-3):79--89.

Garey, M. R. & Johnson, D. S. (1979). Computers and intractability. a guide to the theory of np-completeness. a series of books in the mathematical sciences.

Gendreau, M.; Laporte, G. & Guimarães, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133(1):183--189.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K. & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287--326.

Guinet, A. (1991). Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society*, 42(8):655--671.

Guinet, A. (1993). Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *The International Journal of Production Research*, 31(7):1579--1594.

Hillier, F. S. & Lieberman, G. J. (2013). *Introdução à pesquisa operacional*. McGraw Hill Brasil.

Kim, D.-W.; Kim, K.-H.; Jang, W. & Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18(3-4):223--231.

Kirkpatrick, S.; Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671--680.

Kurz, M. & Askin, R. (2001). Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research*, 39(16):3747--3769.

Lee, Y. H. & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464--474.

López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L. P.; Birattari, M. & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43--58.

Lourenço, H. R.; Martin, O. C. & Stützle, T. (2010). Iterated local search: Framework and applications. Em *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, volume 146, capítulo 12, pp. 363--397. Springer.

Maravilha, A. L.; Goulart, F.; Carrano, E. G. & Campelo, F. (2018). Scheduling maneuvers for the restoration of electric power distribution networks: Formulation and heuristics. *Electric Power Systems Research*, 163:301--309.

Martins Muller, F.; Bassi Araújo, O.; Stefanello, F. & Zanetti, M. (2014). Mip–based neighborhood search for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *Revista de Administração da Universidade Federal de Santa Maria*, 7(3).

Mason, S. J.; Fowler, J. W. & Matthew Carlyle, W. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3):247--262.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097--1100.

Montgomery, D. C. & Runger, G. C. (2013). *Estatística Aplicada e Probabilidade Para Engenheiros*. LTC.

Morton, T. & Pentico, D. W. (1993). *Heuristic scheduling systems: with applications to production systems and project management*, volume 3. John Wiley & Sons.

Pfund, M.; Fowler, J. W.; Gadkari, A. & Chen, Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers & Industrial Engineering*, 54(4):764--782.

Pinedo, M. L. (2008). *Scheduling: theory, algorithms, and systems*. Springer.

Rabadi, G.; Moraga, R. J. & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1):85--97.

Santos, H. G.; Toffolo, T. A.; Silva, C. L. & Vanden Berghe, G. (2019). Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, 26(2):707--724.

SOA-ITI, . (2013). Grupo de investigación soa: Sistemas de optimización aplicada. a web site that includes benchmark problem data sets and solutions for scheduling problems. Em *Available at http://soa.iti.es/problem-instances.*

Tang, L. & Wang, X. (2009). Simultaneously scheduling multiple turns for steel color-coating production. *European Journal of Operational Research*, 198(3):715--725.

Tran, T. T.; Araujo, A. & Beck, J. C. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, 28(1):83--95.

Tran, T. T. & Beck, J. C. (2012). Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. Em *ECAI*, pp. 774--779.

Vallada, E. & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612--622.

Ying, K.-C.; Lee, Z.-J. & Lin, S.-W. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, 23(5):1795--1803.

Zhu, X. & Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE transactions*, 38(11):987--1007.