UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

“ALGORITMO BASEADO EM COLÔNIA DE ABELHAS ARTIFICIAIS PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE UM *JOB SHOP* FLEXÍVEL MULTIOBJETIVO”

ALUNO: Guilherme Felipe Florêncio

ORIENTADOR: Prof. Dr. Edilson R. R. Kato

São Carlos – SP

Fevereiro/2019

CAIXA POSTAL 676

FONE: (16) 3351-8233

13.565-905 – São Carlos – SP

Brasil

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO BASEADO EM COLÔNIA DE ABELHAS ARTIFICIAIS PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE UM *JOB SHOP* FLEXÍVEL MULTIOBJETIVO

Guilherme Felipe Florêncio

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a Obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência artificial.

Orientador: Dr. Edilson Reis Rodrigues Kato.

São Carlos – SP

Fevereiro/2019

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Guilherme Felipe Florêncio, realizada em 25/02/2019:

Prof. Dr. Edilson Reis Rodrigues Kato
UFSCar

Prof. Dr. Roberto Hideaki Tsunaki
USP

Prof. Dr. Mário César San Felice
UFSCar

Agradecimentos

Agradeço primeiramente a minha esposa Ednéia por todo apoio e incentivo que me deu durante esta árdua jornada, aos meus pais que mesmo indiretamente me apoiaram e incentivaram a continuar sempre buscando o melhor.

Agradeço ao meu Orientador Dr. Edilson R. R. Kato, por todo auxílio e orientação, por todas nossas conversas muito esclarecedoras e por todas as ideias e sugestões.

Agradeço ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) pela flexibilização da minha jornada de trabalho em todos estes meses de estudo, possibilitando as viagens necessárias para cursar as disciplinas e as reuniões de orientação.

Agradeço aos meus colegas de serviço que conviveram comigo e que sempre me ajudaram das mais diversas formas.

Agradeço a todos meus amigos que me incentivaram e me ajudaram a superar todos os obstáculos com os quais me deparei neste período.

Finalmente, agradeço aos colegas que cursaram as disciplinas comigo, que estudaram juntos e desenvolveram trabalhos em grupos, possibilitando um maior rendimento e cumprimento dos prazos das disciplinas, entregando os trabalhos, e estudando para as provas.

“Não conseguirás resultados diferentes fazendo sempre as mesmas coisas.”

Albert Einstein

Resumo

O *Flexible job shop problem* (FJSP), considerado um dos problemas mais complexos da programação da produção, compõe a classe de problemas NP-difícil na área da ciência da computação. Embora este problema da programação da produção seja bastante complexo, pode ficar computacionalmente mais custoso para ser tratado, quando impõe-se restrições ou critérios adicionais ao problema. O FJSP consiste em um conjunto de “j” *jobs* que são constituídos por “o” operações, e estas operações são processadas individualmente por uma máquina “m” que faz parte do conjunto de máquinas do ambiente de produção. Nesse tipo de sistema, cada operação dentro dos *Jobs* pode ser processada em uma máquina diferente. No processo totalmente flexível, todas as máquinas são capazes de processar todas as operações e no parcialmente flexível, ao menos uma não processa pelo menos uma operação. Este problema pode ser dividido em dois subproblemas, o roteamento e o agendamento. O roteamento consiste em definir qual máquina irá processar a operação e o agendamento consiste em definir a ordem que as operações serão processadas. Neste trabalho, objetivou-se minimizar multicritérios de desempenho, como: tempo de conclusão de todas as operações, representado pelo termo *makespan*, carga da máquina mais carregada e carga total de todas as máquinas, buscando alta diversidade de soluções. Para alcançar os objetivos deste trabalho foi implementada a metaheurística Colônia de Abelhas Artificiais, juntamente com o método de Pareto, para ajudar a tratar os multiobjetivos. Os resultados observados foram satisfatórios na maioria das instâncias às quais o algoritmo foi aplicado e foram encontrados os melhores resultados conhecidos para algumas instâncias.

Palavras-Chave: Programação da produção, FJSP, multiobjetivo, Colônia de abelhas artificiais, Pareto.

Abstract

The flexible job shop problem (FJSP), considered one of the most complex problems of work scheduling, composes the class of NP-hard problems in the computer science field. Although this scheduling production problem is quite complex, it can be computationally more costly to be treated when additional restrictions or criteria are imposed to the problem. The FJSP consists in a set of "n" job that are constituted by "i" operations, and these operations are processed individually by a machine 'M' that is part of the production environment machine set. In this type of system, each operation inside the job can be processed at a different machine. On the fully flexible process, all the machines are able to process all the operations, and on the partly flexible, at least one does not process at least one operation. This problem can be divided into two sub-problems, the routing and the scheduling. The routing consists in to define which machine will process the operation, and the scheduling consists in the order that operations will be processed. Through this work, it has been aimed to minimize performance multiobjectives, such as: makespan, more loaded machine load and full load of all machines, seeking high diversity of solutions. To achieve the objectives of this paper it was implemented the metaheuristic Artificial Bee Colony along with the Pareto method to help treating the multiobjectives. The observed results were satisfactory in most instances for which the algorithm was applied, and optimal results were found for some instances.

Key Words: Production Program, FJSP, multiobjective, Artificial Bee Colony, Pareto.

Lista de Figuras

Lista de Tabelas

Lista de siglas

FJSP – *Flexible job shop problem*

JSP – *Job shop problem*

TS – *Tabu search*

SA - *Simulated Annealing*

GA – *Genetic Algorithm*

ACO - *Ant Colony Optimization*

ABC – *Algorithm bee colony*

PCP – Planejamento e controle da produção

LIFO – *Last in first out*

FIFO – *First in first out*

cm - *Makespan*

wt – *Work load total*

wm – *Work load machine*

wwm – *Weigh Work load machine*

wwt - *Weigh Work load total*

wcm – *Weigh makespan*

PSO – *Particle swarm optimization*

IA – Inteligência Artificial

BCO – *Bee Colony optimization*

TP – Tempo de processamento

ut – Unidade de tempo

SBH - *Shifting bottleneck heuristic*

EDD - *Earliest due date*

mTESN5 - *Modified two-enhancement scheme with neighborhood N5 perturbation*

GS - *Greedy selection*

LRS - *Linear ranking selection*

HPABC - *Hybrid Pareto artificial bee colony algorithm*

SPEA2 - *Improved strength Pareto evolutionary algorithm*

NSGAII - *Non-dominated sorting genetic algorithm II*

MO – Matriz orthogonal.

DABC – *Discreet algorithm bee colony*

PABC – *Pareto algorithm bee colony*

ABC-DIS - *algorithm bee colony discreet*

SOMA - *Sequential Operations by Machine Attribution*

BEG-NSGA-II - *Bee evolutionary guiding nondominated sorting genetic algorithm II*

Lista de Equações

Sumário

CAPÍTULO 1

1 INTRODUÇÃO

A fim de atender às necessidades atuais do mercado em constante desenvolvimento, os sistemas produtivos têm se tornado cada vez mais flexíveis (WANG et al., 2010). Com esse avanço do mercado rumo à flexibilização dos processos de manufatura, cada vez mais pesquisadores publicam trabalhos relacionados com FJSP (SANCHES, 2017). O FJSP é considerado uma extensão do sistema JSP (GAO, SUGANTHAN, et al., 2015).

O FJSP tem atraído muita atenção na área da ciência da computação e informação nos últimos anos, e muitos algoritmos heurísticos, como algoritmos genéticos, otimização de enxame de partículas, algoritmos de colônias de abelhas artificiais, e otimização de colônias de formigas foram apresentados para resolvê-lo (ASADZADEH, 2016).

Os trabalhos propostos estão sendo desenvolvidos considerando multiobjetivos, pois a tomada de decisão é um processo de avaliação das alternativas disponíveis, e a seleção apropriada de uma ação entre um conjunto de alternativas se torna mais difícil quando o processo envolve vários critérios em vez de um único. Além disso, a maioria das decisões relacionadas a controle de produção em ambientes industriais, é feita sob múltiplos critérios que algumas vezes são conflitantes para satisfazer várias necessidades ao mesmo tempo.

O FJSP é definido por um conjunto de j *jobs*, onde cada *job* é constituído por “o” operações, e estas operações são processadas cada uma por uma máquina m que faz parte do conjunto de máquinas do ambiente de produção. Nesse tipo de sistema cada operação dentro dos *jobs* pode ser processada em uma máquina diferente, e o que torna o processo flexível é que todas as máquinas são capazes de processar todas as operações.

Este tipo de problema pode ser definido conceitualmente de maneira simples, entretanto, se enquadra no grupo problemas da classe NP - Difícil, ou seja, aqueles problemas da área da ciência da computação que não podem ser solucionados deterministicamente em tempo polinomial (GAREY; JOHNSON; SETHI, 1976).

Os sistemas desenvolvidos para resolução do FJSP são programados de forma que os métodos e heurísticas utilizadas trabalhem em busca de um ou mais objetivos simultaneamente, como minimização do *makespan* (tempo decorrido até a última operação ser finalizada) ou minimização do *idle time* (tempo em que as máquinas ficam ociosas). Na literatura, o critério de minimização do *makespan* é o mais utilizado para a aplicação do FJSP, e com o incremento de mais objetivos/critérios de desempenho a dificuldade em programar e os custos computacionais aumentam, necessitando de mais poder computacional e métodos de otimização. Com isso este tema vem sendo utilizado cada vez mais em trabalhos desta área, e os métodos de solução são cada vez mais capazes de resolver problemas reais da manufatura.

Diante dos custos computacionais e das características do problema de programação em sistemas do tipo FJSP vários autores apontam o uso de abordagens heurísticas e meta-heurísticas como boas ferramentas para tratar os problemas. No decorrer dos últimos anos encontram-se trabalhos utilizando técnicas como: Busca Tabu (TS) (BARNES e CHAMBERS, 1996), *Simulated Annealing* (SA) (YAZDANI, GHOLAMI, et al., 2009), Algoritmos Genéticos (GA) (WANG et al., 2010), Algoritmo de Colônia de Formigas (ACO) (XING et al.,2010), *Artificial Bee Colony* (ABC) (LI et al., 2011), Pareto (GAO, SUGANTHAN e CHUA, 2012), Algoritmos de Enxame de Abelhas (LI; PAN; TASGETIREN, 2014), entre outros. Estas técnicas, vêm apresentando bons resultados considerando a aplicação ao FJSP, especialmente o algoritmo ABC proposto recentemente por diversos autores como: Li, Pan e Xie (2011), Gao et al. (2015), Wang et al. (2011), Li, Pan e Tasgetiren (2014).

Diante dos resultados apresentados por autores que utilizaram o ABC para tratar o FJSP multiobjetivo, é proposta uma abordagem que visa a utilização da metaheuristica ABC utilizando métodos de busca modificados e o método de Pareto para realizar o roteamento e sequenciamento das máquinas e operações do FJSP.

1.1 Motivação/Justificativa

A motivação para este trabalho vem da necessidade em minimizar os custos operacionais de manufaturas em industrias e aumentar a eficiência da produção. A área industrial atualmente é equipada com maquinário de excelente qualidade,

entretanto a eficiência destas máquinas é de certa forma desperdiçada se a programação da produção não for de qualidade.

1.2 Objetivos

Este trabalho tem como objetivo apresentar a metaheuristica ABC para tratar o FJSP multiobjetivo, que é um dos problemas da programação da produção. Para auxiliar a metaheurística principal serão utilizados outros métodos baseados em heurísticas, com o objetivo de tornar mais eficiente a adaptação do algoritmo ao problema FJSP.

São implementadas técnicas de busca local e global, para que o algoritmo alterne entre os modos de busca e encontre os resultados esperados o mais rápido possível e encontrando a maior diversidade de soluções boas.

Entende-se por diversidade, programações diferentes para os mesmos valores dos objetivos que se pretende minimizar, ou seja:

Encontrou-se um resultado 20,32,14, este foi encontrado 10 vezes com programações diferentes. Desta forma, entende-se que os valores encontrados representam uma diversidade de 10 soluções para o resultado 20,32,14.

O trabalho é focado na diversidade considerando a dificuldade de encontrar trabalhos com esta característica na literatura. A maioria dos trabalhos encontrados buscam a qualidade das soluções e não apresentam as diversidades encontradas.

A proposta de algoritmo será direcionada ao tratamento multiobjetivo, onde serão tratados simultaneamente três objetivos e como resultado espera-se que seja obtida uma diversidade de soluções alta para cada resultado apresentado.

Os objetivos explorados são: minimização do tempo de conclusão de todas as operações, minimização da carga atribuída a máquina mais carregada e minimização da carga total de todas as máquinas. A diversidade não será tratada como objetivo, entretanto será considerada para a apresentação dos resultados.

1.3 Organização do Trabalho

O trabalho está organizado em 7 capítulos. O primeiro contém uma breve introdução ao problema, a motivação que levou à escolha do tema e os objetivos

esperados. O segundo contém toda a fundamentação básica e os conceitos teóricos utilizados no trabalho. Neste capítulo são abordadas as formas mais comuns de se tratar o problema que está sendo estudado, definições mais detalhadas do problema e dos métodos de solução e apresentação dos critérios de desempenho que serão utilizados. O terceiro capítulo contém a revisão da literatura. Nele são apresentados alguns trabalhos relevantes levando em consideração os mais importantes trabalhos da área, ou seja, os trabalhos que encontraram os melhores resultados conhecidos utilizando as mesmas técnicas aplicadas neste trabalho, assim como trabalhos contendo a aplicação de outras meta-heurísticas que também apresentaram resultados bons. O quarto capítulo contém a proposta de desenvolvimento, onde são explicados os detalhes de implementação, as estratégias e o modo de funcionamento do trabalho. Este capítulo conta com o desenvolvimento e explicação das técnicas que levaram aos resultados que serão apresentados no capítulo posterior. O quinto capítulo contém a apresentação dos resultados encontrados e discussões direcionadas com os resultados de várias instâncias tratadas. Este capítulo apresenta todos os valores encontrados comparados com resultados de outros autores, assim como os gráficos que facilitam a visualização da qualidade das soluções encontradas. O capítulo 6 consiste na conclusão do trabalho e apresenta uma visão geral da proposta feita com os resultados alcançados. Este capítulo também apresenta algumas sugestões para melhoria do que foi desenvolvido para ser utilizado em trabalhos futuros.

CAPÍTULO 2

2 FUNDAMENTAÇÃO TEÓRICA/CONCEITOS BÁSICOS

Este capítulo apresenta os conceitos básicos necessários para o entendimento deste trabalho, descrevendo princípios e fundamentos importantes bem como introduzindo siglas e termos utilizados na área.

O problema abordado neste trabalho é uma parte importante do sistema de manufatura, conhecido por *Flexible Job Shop Problem* (FJSP), muito estudado por diversos autores, dos quais alguns serão citados no capítulo 3. Este é um problema NP-difícil muito conhecido e o mais importante problema de agendamento de máquinas (ASADZADEH, 2016). Este problema consiste na generalização do clássico *Job Shop Problem* (JSP), em que cada operação pode ser executada apenas por uma máquina. O FJSP, por ser a generalização do JSP, pode ter muitas máquinas que são passíveis de executar uma mesma operação, o que torna o problema muito mais difícil de ser resolvido (GAO, SUGANTHAN, et al., 2015).

2.1 Planejamento e controle da produção

Planejamento e controle da produção (PCP) é a atividade de decidir sobre o melhor emprego dos recursos de produção, e desta forma garantir a execução do que foi previsto. Planejamento e controle dizem respeito ao enlace entre o que o mercado quer e o que as operações podem fornecer. As atividades de planejamento e controle possibilitam os sistemas, procedimentos e decisões que aproximam diferentes características da oferta e da demanda por produtos (SLACK, CHAMBERS e JOHNSTON, 2009).

Planejamento, segundo Slack, Chambers e Johnston (2009), é a formalização do que se pretende que ocorra em um dado momento no futuro, embora o fato de planejar não garante que um determinado evento realmente aconteça.

Controle é o processo de lidar com as variações que possam ocorrer após um planejamento, variações estas que podem atrapalhar de alguma forma o planejamento e exigem que algumas pequenas mudanças sejam feitas em curto prazo para garantir a execução do plano.

Ainda segundo Slack, Chambers e Johnston (2009), em relação a planejamento e controle existem quatro atividades superpostas que são de extrema importância, estas são: carregamento, sequenciamento, programação e controle.

Carregamento é a quantidade de trabalho alocado para um centro de trabalho processar, e para alocar este trabalho deve ser considerado o tempo útil da máquina.

Quando o trabalho chega para ser processado algumas decisões devem ser tomadas sobre a ordem em que as tarefas devem ser executadas. Essa atividade é chamada de sequenciamento. As prioridades para o sequenciamento geralmente são definidas por um conjunto de regras, como: restrições físicas da planta/*layout*, prioridade ao consumidor, data prometida, último a entrar primeiro a sair (LIFO), primeiro a entrar primeiro a sair (FIFO), operação mais longa primeiro, e operação mais curta primeiro.

A programação está relacionada ao sequenciamento, onde as operações necessitam de um cronograma detalhado indicando em que momento os trabalhos devem começar e terminar. Esta é uma atividade considerada muito complexa quando há compartilhamento de recursos (máquinas) e quanto maior a quantidade de recursos e operações para serem programadas maior a complexidade da atividade (SLACK, CHAMBERS e JOHNSTON, 2009).

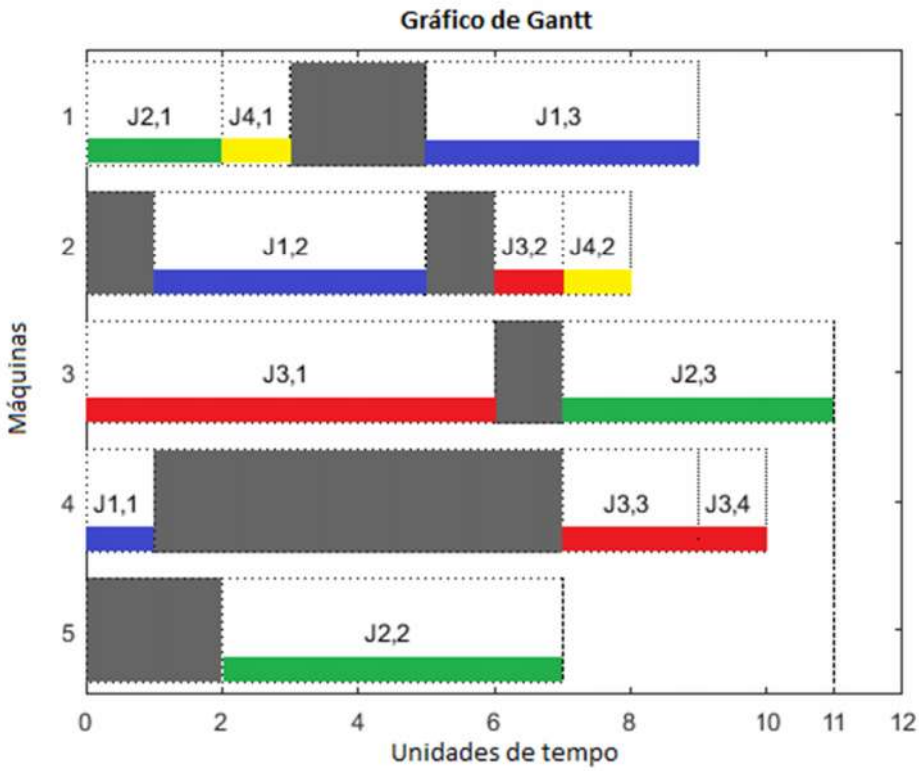
Uma forma muito eficiente e bastante utilizada para a apresentação da programação é o gráfico de Gantt, que não é uma forma de otimização, mas facilita o desenvolvimento e apresentação de programações. Através desse gráfico é possível analisar o tempo necessário (*makespan)* para processar vários *jobs* (produtos) e também pode-se analisar a carga de trabalho total e individual por recurso (máquina), assim como o tempo ocioso de cada máquina (SLACK, CHAMBERS e JOHNSTON, 2009).

O controle é o monitoramento de cada uma das atividades anteriores, que ao menor sinal de desvio pode interferir de diversas formas e retificar a operação, fazendo com que o plano seja seguido conforme planejado. O controle geralmente faz alterações de curto prazo nos planejamentos.

A Figura 1 ilustra um modelo de gráfico de Gantt, e este representa a programação de um problema do tipo *flexible job shop*, composto por 4 *jobs*, um total de 12 operações, sendo a soma de três operações no *job* 1, três operações no *job* 2,

quatro operações no *job* 3 e duas operações no *job* 4. Este problema conta com 5 máquinas disponíveis representados no eixo y e o eixo x representa as unidades de tempo (ut). As barras mais escuras em um tom de cinza representam a quantidade de tempo em que o recurso correspondente permaneceu ocioso. A nomenclatura encontrada nos blocos é composta pela letra J seguida de dois números separados por vírgula. O J vem de *job*, o primeiro número após o J é a indicação do *job* e o número após a vírgula indica a posição da operação dentro do *job*. Exemplo, $J_{1,3}$ representa a terceira operação do *job* um. As barras coloridas representam de forma mais didática todos os trabalhos de um mesmo *job*.

Figura 1 - Ilustração do gráfico de Gantt.



Fonte: Este trabalho.

O planejamento e controle da produção é de suma importância e contribui significativamente para o sucesso de qualquer organização ao utilizar os recursos de forma eficaz para produzir seus bens e serviços de modo a satisfazer seus consumidores. Segundo Slack (2009) uma operação eficaz pode oferecer às

organizações quatro tipos de vantagens: reduzir custos, aumentar a receita, diminuir a necessidade de investimentos, e fornecer base para inovação futura.

2.1.1 Problemas mais comuns da programação da produção

Podem ser encontrados na indústria de manufatura vários problemas ligados a programação da produção, e geralmente estes problemas estão relacionados às regras e restrições do *layout* da fábrica. Estas regras e restrições estão relacionadas com os recursos (máquinas) disponíveis, e com as operações que devem ser processadas para que as ordens de serviço sejam executadas.

Allahverdi, Ng, *et al.* (2008) classificam alguns problemas da programação da produção que têm como característica a programação das operações em diferentes plantas/*layouts*, utilizando desde uma máquina com apenas uma operação até várias máquinas com diversas operações e um problema sem restrição de operação precedente.

Single machine Scheduling: Apenas uma máquina para processar a única operação que o *Job* possui.

Parallel machine Scheduling: Cada *job* possui apenas uma operação, entretanto existe mais de uma máquina para processar o *job*. Cada máquina pode processar um *job* simultaneamente.

Flow shop Scheduling: Esse tipo de problema conta com um conjunto de *m* máquinas e *j jobs*, e as operações de todos os *jobs* são processadas na mesma sequência.

Flexible Flow shop Scheduling: Este tipo de problema é considerado uma extensão do problema anterior, pois conta com o sequenciamento das operações dos *jobs*. O sequenciamento das operações é considerado estágio e dentro de cada estágio podem haver máquinas paralelas processando o mesmo tipo de tarefa.

Job shop Scheduling: O tipo de problema *Job Shop*, diferente do anterior, permite sequências de processamento diferentes em cada máquina.

Flexible Job shop Scheduling: Este tipo de problema é considerado uma extensão do tipo *Job Shop*, ele permite que para cada operação seja escolhida uma máquina qualquer pertencente a um conjunto de recursos. Neste tipo de sistema cada

máquina não precisa ser necessariamente igual, podendo executar cada operação em um tempo diferente.

Open shop Scheduling: Este tipo de problema é bastante diferente dos anteriores, pois não existe relação de precedência, entre as operações, ou seja, cada operação pode ser processada a qualquer momento sem necessitar que outra operação seja processada anteriormente.

O tipo de problema escolhido para ser tratado neste trabalho foi o FJSP No próximo item deste capítulo será apresentada a definição do problema escolhido.

2.1.2 Definição da programação da produção do tipo *Job Shop Flexível*

Como mencionado anteriormente de forma breve, o FJSP é um importante sistema de manufatura bastante estudado no meio acadêmico e com significativa importância no ambiente de produção, programação e gerenciamento.

Como mencionado por Gao, Suganthan, et, al. (2015), o fato do FJSP ser uma generalização do *Job Shop* Problem (JSP), o torna muito mais difícil de ser tratado, pois dependendo do problema é possível que várias máquinas executem a mesma operação.

A formulação do problema de programação do tipo JSP pode ser representado da seguinte forma:

O contador n representa genericamente a quantidade máxima de posições de um determinado conjunto. Este mesmo contador é utilizado ao longo do trabalho para representar genericamente um dado número qualquer.

- Existe um conjunto finito de *j Jobs* (J_k) onde, k = 1, ..., n;
- Existe um conjunto finito de *m* recursos/máquinas (M_p) onde, p = 1, ..., n;
- Cada um dos *Jobs* conta com um número finito de operações $o_{i,j}$ *onde, i = 1, ..., n*;
- Cada operação $o_{j,i}$ do *Job jk* obrigatoriamente é processada por uma máquina *m* do conjunto M de recursos/máquinas, e neste caso *j* corresponde a um determinado *Job*, e *i* corresponde a i-ésima operação do *Job j*;
- *cm* representa o *makespan*;

- *wm* representa a carga em unidade de tempo (ut) da máquina *m*, onde *m* = 1, ..., *n*;
- *wt* representa a carga total em ut de todas as máquinas somadas;

O FJSP pode ser definido da seguinte forma: cada *job* consiste em uma sequência de operações que devem obrigatoriamente ser executadas em ordem, mas não necessariamente em sequência. Deve ser executada a primeira operação de cada *job* antes das demais operações, $O_{1,1}$ ($O_{job,operação}$) antes de $O_{1,2}$ e antes de $O_{1,3}$. Entretanto, entre uma operação e outra de um *job* pode ser executada uma operação de outro *job*, como por exemplo no caso de um problema com 3 *jobs* e 3 operações por *job*: [$O_{1,1}|O_{3,1}|O_{1,2}|O_{2,1}|O_{3,2}|O_{2,2}|O_{2,3}|O_{1,3}|O_{3,3}$]. Esta sequência é perfeitamente aceitável, pois foi respeitada a ordem de execução dentro das operações de cada *job*. Cada operação de um *job* deve ser processada por uma máquina em um determinado tempo, e cada máquina processa apenas uma operação por vez (GAO, SUGANTHAN, *et al.*, 2015).

Algumas outras restrições também são importantes para a definição e tratamento do problema como apresenta Deng et. al. (2017), Li et. al. (2016), Li et. al. (2011), Choo et. al. (2016), entre outros.

- As operações não podem ser interrompidas após iniciadas;
- As máquinas podem processar apenas uma operação por vez;
- Uma operação não pode ser processada por mais de uma máquina simultaneamente;
- O movimento entre as operações e o tempo de configuração das máquinas é considerado insignificante;
- As máquinas são independentes entre si;
- Os *Jobs* são independentes entre si;

Kacem, Hammadi e Borne (2002) definiram que existem dois tipos de problemas do tipo FJSP, o problema parcialmente flexível, no qual pelo menos uma das máquinas não processa pelo menos uma operação, e o problema totalmente flexível, no qual todas as máquinas processam todas as operações. Desta forma considerando todas as regras de definição do FJSP independentemente do tipo de

problema tratado ou seja, parcialmente ou totalmente flexível as possíveis solução devem tem como etapas fundamentais a programação e o agendamento.

Na programação é escolhida a máquina que irá processar determinada operação, e no agendamento é determinado em qual ordem esta operação será executada.

2.1.3 Critérios de Desempenho

Os critérios de desempenho são utilizados para mensurar o desempenho dos processos. Slack, Chambers e Johnston (2009) citam cinco critérios de desempenho utilizados para tal finalidade: Qualidade, Velocidade de produção, Confiabilidade, Flexibilidade e custo. A qualidade está ligada diretamente a produzir produtos ou serviços isentos de erros, possibilitando a seus clientes produtos/serviços adequados a seus propósitos. Velocidade e confiabilidade estão relacionados a entrega dos produtos/serviços no prazo contratado. Flexibilidade se refere a estar preparado para mudar o planejamento para atender um cliente específico, ou contornar circunstâncias inesperadas para satisfazer as demandas de produção. Custo se refere a produzir sempre com o menor custo possível mantendo a qualidade proposta, de forma que o produto/serviço apresente o retorno satisfatório.

Para o FJSP existem os critérios de desempenho que geralmente são utilizados para mensurar a qualidade das programações e estes critérios são variados de acordo com os objetivos que os pesquisadores desejam alcançar. Para avaliação dos resultados obtidos por este trabalho foram escolhidos três critérios de desempenho que comumente utilizam os seguintes termos em inglês: *Makespan* (cm)*, Maximum Workload* (wm)*, e Total Workload* (wt)*.*

Makespan (cm): Relaciona-se com a velocidade de produção, e corresponde ao tempo necessário para a conclusão de processamento de todos os *jobs.* Geralmente relacionado diretamente com o custo da produção.

Maximum Workload (wm): Relaciona-se também com a velocidade de processamento das operações, e corresponde a quantidade de tempo que a máquina mais carregada ficou trabalhando.

Total Workload (wt): Este critério representa a soma de tempo que todas as máquinas que permanecem trabalhando. Pode se considerar que wt também está

diretamente relacionado com o custo de produção, pois quanto menos as máquinas trabalham menos consomem recursos.

Os critérios de desempenho podem ser representados pelas equações a seguir:

Makespan:

Minimizar cm = max(cm) onde, (1 ≤ m ≤ n);.....(1)

Na equação 2, n representa a quantidade máxima de operações do problema.

Maximum Workload:

Minimizar wm = max(wm) onde, (1 ≤ m ≤ n); .....(2)

Na equação 2, n representa a quantidade máxima de máquinas do problema.

Total Workload:

Minimizar wt = $\sum_{k=1}^{m} w_t$; .....(3)

Na equação 3, m representa a quantidade de máquinas disponíveis no conjunto de máquinas.

Baseado na literatura vista até agora os critérios mais utilizados pelos pesquisadores que tratam os problemas do tipo FJSP são estes apresentados acima.

O próximo item abordará um pouco sobre o tratamento dos multiobjetivos.

2.2 A Otimização Multiobjetivo

Uma otimização multiobjetivo consiste em um conjunto de funções objetivo a serem otimizadas, e esta otimização pode ser por minimização ou maximização. Estas funções precisam obrigatoriamente respeitar as regras e restrições impostas a cada função objetivo para que seja considerada factível (DEB, AGRAWAL, *et al.*, 2002).

2.2.1 Técnicas para tratar os multiobjetivos em problemas do tipo FJS

Wang, Zhou, *et al.*, (2011) descrevem em seu trabalho uma forma de representação do problema multiobjetivo do mesmo tipo utilizado neste trabalho sem perder a generalidade:

Minimizar $y = f(X) = (f_1(x), f_2(x), ..., f_q(x))$, onde, $x \epsilon \Omega$, $y \epsilon R^q$..... *(4)*

Onde, X representa o conjunto de variáveis que se pretende minimizar dentro do espaço de soluções Ω, e y é o vetor objetivo, contendo todos os objetivos a serem minimizados, com q > 1 objetivos. A representação $f_1(x)$ representa o objetivo um a ser minimizado.

Os três objetivos a serem minimizados neste trabalho podem ser representados matematicamente pela equação 5, que elenca os objetivos apresentados anteriormente.

Minimizar $y = f(X) = (f_1(Cm), f_2(Wm), f_2(Wt))$, onde, $x \epsilon \Omega$, $y \epsilon R^q$..... *(5)*

Wang, Zhou, *et al.*, (2011) apresentam também no seu trabalho algumas técnicas utilizadas para o tratamento do problema multiobjetivo, como: Dominância de Pareto, *Pareto optimal* ou *nondominated solution*, Frente de Pareto, Conjunto de arquivos, e Seleção não Dominada.

Dominância de Pareto – Uma solução "*a"* domina uma solução "*b"* se e somente se $\forall\, i \epsilon \{1, 2, ..., q\}: f_i(a) \leq f_i(b)$ e $\exists\, i \epsilon \{1, 2, ..., q\}: f_i(a) < f_i(b)$ (WANG, ZHOU, *et al.*, 2011). Em outras palavras, uma solução "a" somente domina uma solução "b" se ao menos um dos atributos de "a" tem valor menor que o mesmo atributo da solução "b" (isto considerando a otimização por minimização), e os demais atributos da solução "a" permanecem no mínimo iguais aos mesmos atributos da solução "b".

Ótima de Pareto ou não Dominada – Uma Solução a é considerada ótima de Parato se não existe nenhuma solução "b" que domina "a".

Frente de Pareto – É o conjunto de todas as soluções ótimas de Pareto, conjunto das soluções que não foram dominadas.

Conjunto de arquivos – É um conjunto que armazena apenas as soluções ótimas de Pareto, e este arquivo é atualizado a cada iteração, apenas novas soluções ótimas são adicionadas ao arquivo, e quando esta nova solução é adicionada as soluções dominadas são retiradas do conjunto.

Seleção/classificação não nominada – Esta técnica consiste de "n" conjuntos de soluções não dominadas. A ordenação dos conjuntos ocorre de forma que cada um seja imediatamente inferior ao anterior e tenha um valor de classificação de acordo com seu grau de dominância. A primeira frente (ou seja, primeiro conjunto) contém apenas soluções não dominadas e recebe um valor de classificação 1, a segunda

frente contém apenas soluções dominadas pela primeira frente e recebe um valor de classificação 2, a terceira frente contém as soluções dominadas pela segunda frente e, consequentemente, também dominadas pela primeira frente, e assim sucessivamente.

2.2.2 Formas de tratamento de problemas de programação da produção

O tratamento Multiobjetivo está relacionado com a forma que se deseja analisar os resultados dos problemas abordados. Existem algumas possibilidades a serem utilizadas para tratamento do problema do tipo FJSP, que pode ter sua análise comparando individualmente cada objetivo ou de forma multiobjetiva, analisando todos os objetivos de uma só vez.

Goldberg et al. (2016) explana que a cópia dos sistemas naturais, também denominada de Biomimética, tem sido uma fonte inspiradora para algoritmos computacionais desenvolvidos para tratar diversos tipos de problemas. Entre diversas outras formas de se tratar problemas da ciência da computação, Goldberg et al. (2016) apresenta como parte da Computação Natural as seguintes técnicas:

Computação bioinspirada, que engloba computação evolucionária, inteligência coletiva, redes neurais, computação orgânica e sistemas imunológicos.

Computação inspirada na físicoquímica, que engloba *simulated annealing*, busca harmônica, busca gravitacional e busca caótica;

Computação com materiais naturais, que engloba computação com DNA, computação quântica e computação molecular;

E por último, simulação computacional de sistemas naturais, que engloba geometria fractal e vida artificial.

Além dos descritos acima, ainda existem vários outros métodos, e dentro de cada técnica citada existem as hibridizações com outros métodos. Dentro da área da computação bioinspirada, estão as técnicas que utilizam o comportamento animal para modelar um algoritmo. Pode-se citar como exemplos: Algoritmo de colônias de formigas (ACO), Algoritmo de colônia de Abelhas (ABC), Algoritmo de enxame de partículas (PSO), entre outros. Como o escolhido para este trabalho foi o ABC, a seguir será explicado um pouco mais sobre a base biológica desta meta-heurística.

2.3 Base biológica da meta-heurística utilizada

Existem três tipos diferentes de abelhas em uma colônia: rainha, zangões e operárias. A rainha é o indivíduo que tem como função reproduzir colocando ovos que podem ser fertilizados ou não. Os zangões tem como função fertilizar novas rainhas, eles nascem de ovos não fertilizados e portanto agem amplificando o genoma de sua mãe (a rainha). As operárias são o tipo responsável pela manutenção da colônia, atuando em tarefas como: limpeza, alimentação, estocagem de suprimentos. As rainhas e as operárias nascem de ovos fertilizados, que após nascerem enquanto larvas recebem uma alimentação diferenciada (GOLDBARG, GOLDBARG e LUNA, 2016).

Existem algumas tarefas importantes em uma colônia de abelhas como: Acasalamento, encontrar o local de um novo ninho, e busca por alimentos. Esta última é a base biológica do método estudado neste trabalho. Esta busca consiste em uma abelhas empregada encontrar uma flor, e após isso a abelha recolhe, guarda o néctar em seu estômago e retorna a colônia. Após encher as células de depósito de néctar anteriormente vazias com o conteúdo encontrado no seu estômago, a abelha informa às outras abelhas a localização da fonte de alimento. Esta informação consiste basicamente em direção, distância e quantidade de néctar disponível. Para fazer esta indicação cada abelha faz uma espécie de dança em uma área específica da colônia. Cada abelha expectadora escolhe probabilisticamente uma fonte de alimento baseada na dança das abelhas empregadas e volta a explorar esta região até que a fonte seja extinta. Após isso as abelhas expectadoras se transformam em uma abelha exploradora e encontra aleatoriamente uma nova fonte para ser explorada e o ciclo começa novamente (GOLDBARG, GOLDBARG e LUNA, 2016).

2.3.1 Algoritmo de colônia de abelhas artificiais (ABC)

O método selecionado para solucionar o FJSP é o *Algorithm Bee Colony* (ABC), proposto por Karaboga (2005) para resolver problemas contínuos, o algoritmo desenvolvido é baseado em três grupos de abelhas: empregadas, expectadoras e exploradoras. As abelhas empregadas são responsáveis por determinar as fontes de alimento (soluções) aleatoriamente. Essas informações são passadas para as abelhas expectadoras através de uma agitação no voo conhecida como dança citada anteriormente. As abelhas expectadoras memorizam as soluções e buscam nas

proximidades (isto é, na vizinhança) soluções melhores até que estas sejam exauridas. Quando uma solução é exaurida as abelhas abandonam esta solução e a abelha empregada responsável por essa fonte de alimento se transforma em uma abelha exploradora, que se encarrega da obrigação de buscar uma nova fonte de alimento (solução) aleatoriamente.

2.4 Algoritmos de busca local

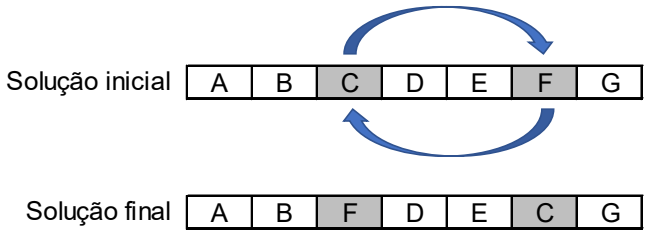
O objetivo de alcançar eficiência e eficácia na solução de problemas de otimização do tipo NP-difícil de grande porte pode ser uma tarefa quase impossível, pois, a garantia de eficiência prejudica a garantia de eficácia, e vice-versa, e para construir um algoritmo de busca com esta finalidade é necessário um casamento perfeito entre exploração e explotação de acordo com Goldbarg, Goldbarg E Luna (2016).

Entende-se por explotação a capacidade que um algoritmo tem de explorar as variáveis do problema esgotando de forma eficiente a busca em regiões restritas do universo de soluções, e entende-se por exploração a capacidade de um algoritmo identificar partes promissoras do universo de soluções sem ter que examinar o espaço todo, examinando apenas as partes selecionadas.

Tendo como objetivo melhorar as buscas locais, foram desenvolvidos com o passar do tempo mecanismos de busca local ou mecanismos de busca na vizinhança para auxiliar as meta-heurísticas no tratamento dos problemas. Alguns mecanismos de busca local serão apresentados a seguir.

O *Swap* é um operador que troca posições aleatórias em um vetor de solução, tem uma definição bastante simples e consiste em escolher aleatoriamente duas posições de um vetor de solução, após isso transfere o conteúdo da posição 1 para a posição 2 e vice-versa, como é ilustrado na Figura 2.

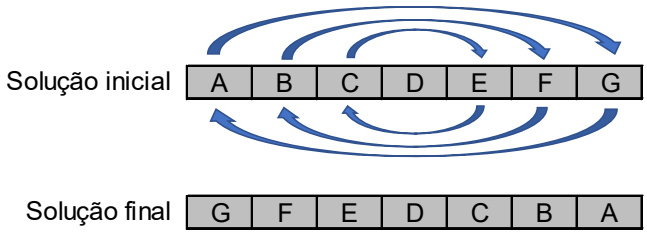
Figura 2 - Exemplo *Swap*.



Fonte: Este trabalho.

O operador *Reverse* consiste em inverter uma sequência de posições em um vetor de solução, esta sequência pode ser o vetor de solução inteiro ou apenas parte do vetor. A Figura 3 ilustra a inversão do vetor de solução inteiro.

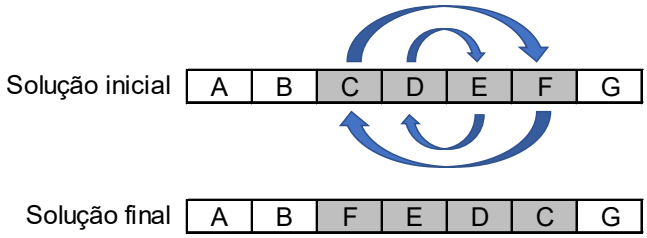
Figura 3 - Exemplo *Reverse* 1.



Fonte: Este trabalho.

Como o operador *reverse* pode ser aplicado de outra forma também é ilustrada esta aplicação na Figura 4, onde são selecionadas aleatoriamente duas posições diferentes do vetor e inverte-se os valores que estão entre testas posições.

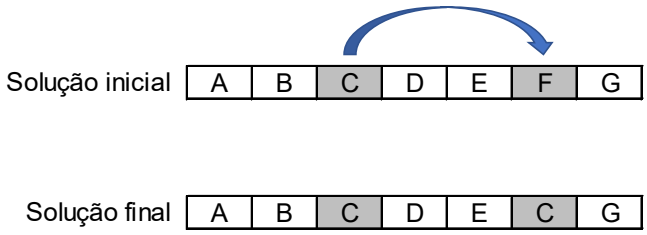
Figura 4 - Exemplo *Reverse* 2.



Fonte: Este trabalho.

A utilização do operador *Insert* consiste em selecionar aleatoriamente duas posições do vetor de solução, a primeira posição fornecerá o valor que será inserido na segunda posição. Considerando que alguns problemas podem ter restrições quanto ao uso duplicado de valores em um mesmo vetor de solução, este operador também tem dois modos de operação que são frequentemente utilizados. O primeiro modo que pode ser aplicado ao problema que será tratado neste trabalho é apresentado na Figura 5 e “executa” basicamente o processo descrito anteriormente.

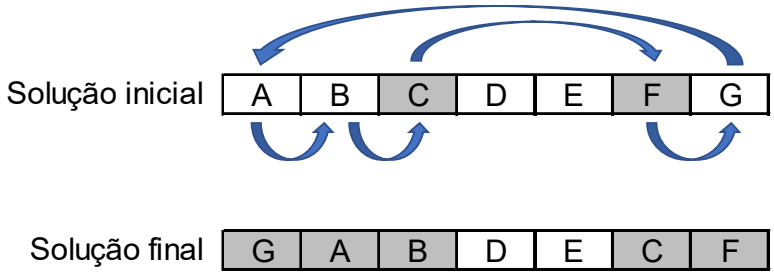
Figura 5 - Exemplo *insert*.



Fonte: Este trabalho.

Para problemas com regras que não permitem a duplicidade de valores no vetor de solução como o problema do caixeiro viajante, o operador *insert* pode funcionar da seguinte forma: São selecionadas aleatoriamente duas posições do vetor de solução, a primeira posição fornecerá o valor que será inserido na segunda posição, entretanto como não podemos perder o valor que está na segunda posição ele é deslocado para a próxima posição e assim por diante, o valor da última posição “pula” pra primeira e assim até preencher a primeira posição escolhida aleatoriamente como é ilustrado na Figura 6.

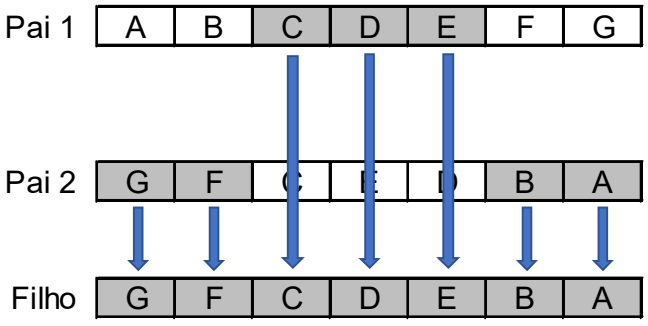
Figura 6 - Exemplo *inverse* 2.



Fonte: Este trabalho.

O operador de Cruzamento Genético é baseado no processo evolucionário de espécies e consiste em cruzar informações contidas em dois vetores denominados vetores pais. Esta estratégia é considerada parte da computação natural e da computação bioinspirada (GOLDBARG, GOLDBARG e LUNA, 2016). Funciona da seguinte maneira, seleciona-se entre a população duas soluções que chamaremos de indivíduos pais. São escolhidas duas posições aleatoriamente e os valores que estiverem entre estas duas posições do pai 1 serão utilizados juntamente com a parte complementar do pai 2 para formar um novo indivíduo como é ilustrado na Figura 7.

Figura 7 - Exemplo de cruzamento genético.



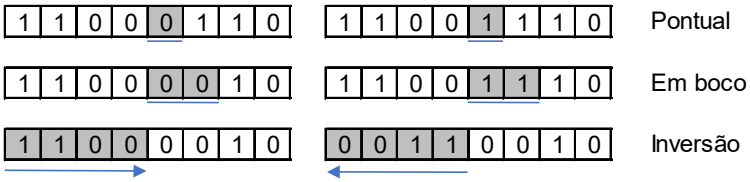
Fonte: Este trabalho.

A Mutação Genética é um processo estocástico que visa alterar o material genético e aumentar a diversidade populacional, e é um processo que de acordo com Goldbarg, Goldbarg E Luna (2016) supostamente imita os mecanismos naturais de

mutação. Funciona de maneira bastante semelhante ao operador *swap* apresentado anteriormente, mas pode ser configurada uma taxa de mutação, que é um percentual de posições que serão alteradas em um indivíduo.

A mutação pode ocorrer de três formas diferentes de acordo com Goldbarg, Goldbarg e Luna (2016), o primeiro modo é o pontual, o segundo é em blocos e o terceiro é por inversão, como podem ser vistos na Figura 8.

Figura 8 - Exemplo de Mutação.



Fonte: Adaptado de Goldbarg e Luna (2016).

Em algumas aplicações estas operações podem complicar um pouco, pois devem obedecer às restrições do problema no qual estão sendo aplicadas. No caso deste trabalho deve-se considerar que se o problema for parcialmente flexível, ou seja, ao menos uma máquina não executa ao menos uma operação nem todas as alterações feitas pelos operadores descritos seriam factíveis.

CAPÍTULO 3

3 REVISÃO BIBLIOGRÁFICA

Para embasamento teórico do trabalho realizado, este capítulo apresenta as análises de alguns estudos encontrados na literatura, cujos temas são semelhantes ao problema ou ao método escolhido para a solução do problema abordado. A maioria destes trabalhos foram publicados nos últimos anos, a fim de apresentar algumas das descobertas mais recentes na área.

Os artigos analisados nesta pesquisa bibliográfica usam métodos diferentes para alcançar seus objetivos, entre metaheurísticas e estratégias de busca local e global. Alguns dos autores optaram por minimizar o *makespan*, outros por minimizar o tempo de espera entre as operações, outros inseriram tarefas de manutenção preventiva nos intervalos em que as máquinas estavam em espera, e outros abordaram a utilização de multiobjetivos utilizando minimização dos critérios: *Makespan*, Carga máxima total e máquina com maior carga. Estas estratégias foram utilizadas para melhorar as soluções, alcançando os objetivos pretendido pelos autores.

Os trabalhos com abordagem de objetivo único foram utilizados como referência levando-se em conta que a abordagem multiobjetivo seria dividida em duas partes, e estas partes seriam tratadas individualmente, sendo elas o roteamento e o agendamento da produção. Outro ponto interessante é que quando aplicamos a equação multiobjetivo transformando os três objetivos em apenas um valor (*fitness*), o problema pode ser parcialmente tratado de forma menos complexa.

A seguir serão apresentadas análises de trabalhos desenvolvidos por autores conceituados na área de pesquisa da proposta e que contribuíram para seu desenvolvimento. As pesquisas estão organizadas em ordem cronológica.

Considerando que as estratégias que tratam multiobjetivos podem dividir o problema em vários subproblemas de um único objetivo, e levando em conta que esta pode ser uma alternativa viável, os trabalhos encontrados serão classificados em dois tipos: trabalhos com abordagens de objetivo único e trabalhos com abordagem multiobjetivo.

3.1 Trabalhos com abordagem de objetivo único

Os trabalhos que serão apresentados a seguir foram feitos com abordagem de apenas um objetivo.

3.1.1 Solução do FJSP com ABC híbrido

Thammano e Phu-Ang (2013) elaboraram uma pesquisa interessante tendo como objetivo minimizar o tempo de execução de todas as operações em todos os *jobs*. Neste trabalho, o FJSP foi decomposto em dois subproblemas: determinar a sequência de operações e selecionar a melhor máquina para cada operação.

Foram utilizados cinco métodos de busca. O algoritmo de busca de harmonia (LEE e GEEM, 2005) foi usado para inicialização da população, a técnica de busca local iterada, (THAMMANO e PHU-ANG, 2013) a técnica de busca de dispersão (THAMMANO e PHU-ANG, 2013) e o algoritmo *filter and fan* (THAMMANO e PHU-ANG, 2013) foram usados para fazer a busca local na vizinhança das soluções e, por fim, o algoritmo *simulated annealing* foi aplicado à solução “presa” em um ótimo local. A união e sincronização destes 5 métodos garantiu que os autores obtivessem um ótimo desempenho do algoritmo comparado com outros três.

Foi apresentado que nas dez instancias nas quais foram aplicadas o algoritmo, em duas, foram alcançados os melhores valores. Nas instâncias em que não alcançou o melhor valor conhecido, se aproximou bastante e na maioria das vezes, se mostrou mais eficiente que os demais algoritmos colocados em comparação.

Embora Thammano e Phu-Ang (2013) tenham neste trabalho pesquisado uma abordagem com apenas um objetivo, a estratégia utilizada de separar o problema em dois subproblemas e tratá-los separadamente trouxe avanço significativo ao desenvolvimento da proposta.

3.1.2 Solução do FJSP comparando várias heurísticas, GA, ACO e ABC

Os autores Zheng, Lian e Mesghouni (2014) abordam na pesquisa três heurísticas diferentes para a resolução do FJSP: *Genetic Algorithm Integrated* (GA integrado), *Ant Colony Optimization* (ACO) e ABC, entretanto, diferente das demais abordagens estudadas, os autores utilizam as heurísticas propondo um algoritmo de

inserção (*Inserting Algorithm*) (IA) para resolver o problema da programação dinâmica da manutenção, utilizando-a para fazer o agendamento de manutenções preventivas nas máquinas nos tempos ociosos que o procedimento de resolução do FJSP gera. Essa estratégia, conhecida como manutenção baseada em condições, é um tipo de manutenção preventiva utilizada para reduzir a indisponibilidade das máquinas.

A utilização das três heurísticas juntamente com algoritmo de inserção (IA) foi sugerida para encontrar uma melhor solução para o FJSP, e com a aplicação do algoritmo de inserção (IA) a incerteza das máquinas é levada em conta, fazendo com que o agendamento da manutenção preventiva diminua a indisponibilidade das máquinas devido à quebra.

De acordo com os autores a quebra da máquina não é o único fator que gera incerteza e induz à indisponibilidade das máquinas. A indisponibilidade de funcionários, erros operacionais, e avaria da máquina também geram atrasos na produção.

Embora existam vários fatores que geram a indisponibilidade das máquinas, os autores sugerem a manutenção baseada em condições para diminuir um dos fatores que causam prejuízo e aumentar a confiabilidade da máquina, aumentando assim a eficiência.

Pelo fato do problema ser bastante inovador e não existir *benchmark* usado para demonstrar a eficácia deste tipo de algoritmo, os autores compararam o algoritmo de inserção com o algoritmo de agendamento simulado usado em Neale e Cameron (1979), apud Zheng, Lian e Mesghouni (2014), para ter um critério de comparação do algoritmo proposto.

O procedimento da solução foi decomposto em duas etapas. A primeira resolve o FJSP em um esquema de pré-programação e a segunda etapa insere as tarefas de manutenção preventiva para a solução do pré-agendamento com o algoritmo de inserção proposto. Com o intuito de comparar os três algoritmos entre si, o algoritmo foi aplicado às 3 heurísticas.

Algoritmo de inserção consiste basicamente em identificar os intervalos de tempo ociosos e agendar as manutenções preventivas nestes intervalos, que sempre existem e são inevitáveis na programação. É necessário encontrar os intervalos na sequência de programação de cada máquina e “encaixar” a manutenção preventiva

desta máquina, mesmo que o intervalo de tempo for menor do que a duração da manutenção, esta é executada no início do intervalo e todas as sequências de operações posteriores são atrasadas.

A comparação do desempenho das três abordagens propostas foi feita com relação aos melhores resultados conhecidos obtidas em (XIA e WU, 2005) e Neale e Cameron (1979) apud Zheng, Lian e Mesghouni (2014), e os resultados estão ilustrados na Tabela 1.

Os problemas aplicados são as instâncias de Kacem, Hammadi e Borne, (2002) e os nomes representam os tamanhos dos problemas, por exemplo: J8M8 é uma instância composta por 8 Jobs e 8 Máquinas.

Tabela 1 - Comparação dos resultados.

	J8M8[1]	**J10M10[1]**	**J15M10[1]**
GA Integrado[2]	**14**	**7**	**12**
ACO[2]	15	7	16
ABC[2]	16	8	15
SA+PSO[3]	15	7	12
HGA[4]	15	7	11

Fonte: Zheng, Lian e Mesghouni, (2014).

As conclusões que Zheng, Lian e Mesghouni (2014) obtiveram nesta pesquisa foi que a qualidade do resultado final do agendamento é bastante dependente do esquema pré-agendado adquirido no estágio de resolução do FJSP normal.

Pode-se notar que entre os cinco algoritmos implementados, o GA integrado obteve resultados iguais ao HGA, e Ambos foram melhores que os demais. O HGA ficou melhor na instância J15M10 e o GA integrado apresentou melhor resultado para a instância J8M8.

[1] (KACEM, HAMMADI e BORNE, 2002)
[2] Zheng, Lian e Mesghouni (2014)
[3] (Xia e Wu, 2005)
[4] (Neale e Cameron, 1979)

Este trabalho está sendo utilizado como referência pois prova a viabilidade da utilização de um algoritmo para inserir manutenção programada em uma programação e obter resultados bem próximos dos melhores resultados conhecidos.

3.1.3 Solução do FJSP com BCO e estratégias de buscas

Choo, Wong e Khader (2016) propuseram o algoritmo *Bee Colony Optimization* (BCO) juntamente com *modified two-enhancement scheme with neighborhood N5* (mTESN5) (BCO + mTESN5) para resolver o FJSP. O algoritmo BCO + TESN5 usa uma estratégia de força bruta, onde se tem um custo muito elevado em tempo de processamento (TP). A busca local TESN5 trabalha com duas fases de busca. A primeira envolve operações de troca e inserção e a segunda é a fase de perturbação menor, com base na estrutura de vizinhança N5. Esta etapa que executa a vizinhança N5 consiste em trocar todas as possíveis operações das máquinas que estão no caminho crítico. Entretanto, esta estratégia de força bruta é muito demorada, o que se torna muito pior quando é aplicada em problemas grandes. Para resolver o problema do custo computacional desta estratégia, o autor utiliza o método *shifting bottleneck heuristic* (SBH) para selecionar uma máquina que participa do caminho crítico (máquina “gargalo”) e assim fazer a perturbação apenas nas operações desta máquina. Este método garante a redução da sobrecarga gerada pelo mecanismo de pesquisa local TESN5.

A busca local vizinhança N5 é definida como um intercâmbio de operações consecutivas ao longo do caminho crítico. Este intercâmbio faz a troca entre as operações sem restrição de antecedência, ou seja, as operações que podem ser trocadas no caminho crítico são trocadas por esta estratégia. Quando é dito que as operações são trocadas significa que operações em sequência podem ser invertidas.

Esta estrutura de vizinhança N5 busca encontrar um ótimo local executando todas as possíveis trocas entre as operações do caminho crítico, entretanto, este método de busca é muito caro e a quantidade de operações possíveis de troca aumenta muito quando o tamanho do problema aumenta. Para evitar isso o autor utiliza esta estratégia, visando encontrar os ótimos locais e selecionar o caminho crítico através de uma heurística de identificação do gargalo. Esta heurística também faz alterações em algumas operações e garante a redução da quantidade de

operações alteradas, fazendo com que o problema não fique tão complexo mesmo em grandes instâncias.

A busca local que analisa apenas a máquina gargalo é nomeada como esquema de dois aprimoramentos modificados com perturbação na vizinhança N5 (mTESN5).

O funcionamento do SBH é basicamente a decomposição dos FJSP em vários problemas de programação de apenas uma máquina. De acordo com os autores, isto garante que a complexidade do problema seja reduzida, pois é mais fácil executar vários problemas simples denominados "*one machines*". A fim de minimizar o atraso máximo os autores utilizam uma regra de despacho denominada *earliest due date* (EDD), que possibilita a redução do caminho crítico.

Foram utilizados para os testes duas estratégias de seleção, sendo elas estratégia gananciosa e estratégia de classificação linear. Desta forma, foram executados dois códigos BCO diferentes com o algoritmo mTESN5. A estratégia gananciosa com o mTESN5 foi chamada pelo autor de "BCO com GS-TESN5" e a utilização da estratégia de classificação linear (*linear ranking selection*) (LRS) nomeada de "BCO + LRS-TESN5".

Fica claro no trabalho que Choo, Wong e Khader (2016) aplicaram aos problemas 3 códigos, sendo eles BCO com GS-TESN5, BCO + LRS-TESN5 e o BCO+TESN5, os dois primeiros com as alterações da proposta e o último o "código puro" BCO com a aplicação da busca local vizinhança N5.

Estes algoritmos propostos foram testados nas 82 instâncias de referências obtidas na OR-Library (BEASLEY, 2017). Esta biblioteca é frequentemente atualizada, conta com problemas de várias categorias e as dimensões destes problemas variam entre 6 *jobs* por 6 máquinas até 50 *jobs* por 10 máquinas, sendo que alguns destes problemas são flexíveis, alguns totalmente e outros parcialmente.

Com os testes foi verificado que a utilização das estratégias propostas se mostrou eficiente, sendo que o BCO com GS-TESN5 resolveu 56% dos 82 problemas com desvios menores ou iguais a 1% do melhor valor conhecido. O BCO com LRS-TESN5 resolveu 54% dos 82 problemas com os desvios ≤ 1% dos melhores valores conhecidos. Utilizando como parâmetro de comparação o tempo de processamento (TP), os resultados apresentados mostram que o BCO com o algoritmo GS-TESN5

reduz o tempo computacional em 10% em comparação com o algoritmo BCO + TESN5 e o algoritmo BCO com LRS-TESN5 reduz o tempo computacional em 25% em comparação com BCO + TESN5 e a conclusão extraída dos resultados obtidos é que ambas estratégias são capazes de diminuir o TP médio para resolver os problemas de *benchmark* abordados pelos autores.

Embora este trabalho tenha utilizado como estratégia a busca de apenas um objetivo, este mostra que algoritmos baseados em colônias de abelhas podem ser muito mais rápidos que os demais algoritmos.

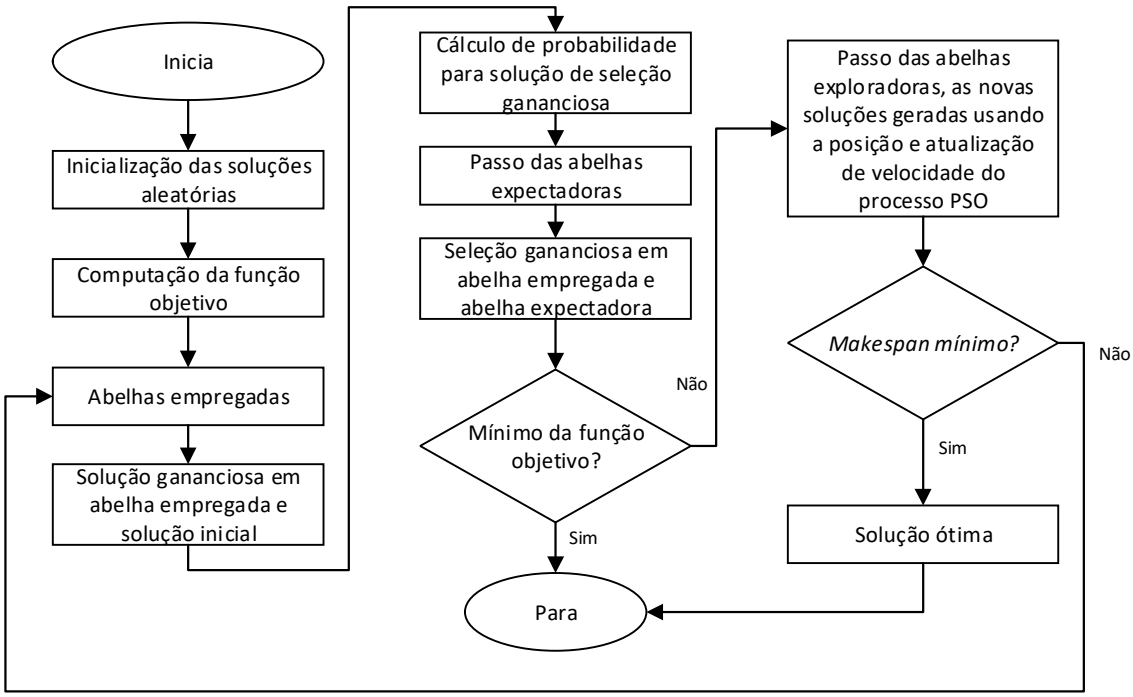
3.1.4 Solução do FJSP com ABC e PSO

No trabalho de Muthiah, Rajkumar e Rajkumar (2016) é proposta uma abordagem híbrida entre ABC e PSO, e com esta abordagem buscam reduzir o *makespan,* superando as desvantagens existentes nos dois algoritmos.

Os autores classificam o ABC como um algoritmo bom em encontrar o ótimo global, entretanto, o algoritmo, assim como os outros, também possui desvantagens.

O algoritmo ABC substitui soluções que não melhoram após uma determinada quantidade de iterações por meio das abelhas exploradoras, da mesma forma que funciona a bioinspiração. Cada solução “x” tem uma variável “X-limite” que é incrementada a cada “n” iterações. Quando a solução “x” não melhora na iteração “n”, a variável X-limite incrementa uma unidade. Assim é possível controlar o máximo de vezes que uma solução poderá ser explorada sem melhorar. Quando X-limite atinge o valor máximo pré-estabelecido, a abelha expectadora que explora a solução se torna uma abelha exploradora. Esta abelha exploradora tem a função de substituir a solução “x” que não melhorou. Na estratégia híbrida a abelha expectadora não se transforma em uma abelha exploradora baseada em uma variável X-limite, e sim baseada nas equações de posição e velocidade das partículas encontradas no PSO. Estas equações pertencentes ao método PSO utilizadas juntamente com as estratégias do ABC tornam o processo híbrido e mais eficiente. O fluxograma da Figura 9 ilustra o funcionamento do código desenvolvido pelos autores.

Figura 9 - Fluxograma representativo do algoritmo.



Fonte: Adaptado de Muthiah, Rajkumar e Rajjumar (2016).

De acordo com os resultados obtidos e apresentados pelos autores, a união do ABC com o PSO como uma proposta híbrida é uma estratégia boa, pois se mostra superior quando comparada com os resultados obtidos por outros métodos.

Este trabalho, embora utilize a minimização de apenas um objetivo, se mostrou importante, pois aborda a hibridização do ABC com outra meta heurística de uma forma que o algoritmo alcance os melhores resultados conhecidos nas instâncias utilizadas para validação, e foi importante para a decisão de hibridizar a proposta que será apresentada no próximo capítulo.

3.1.5 Solução do problema de agendamento de máquinas com sequência dependente utilizando ABC híbrido

Yue et al. (2016) utilizaram a técnica de Pareto como método para hibridizar o ABC e aplicaram ao problema de agendamento de máquinas, com tempo de configuração dependente de sequência. Os autores descreveram como tempo de configuração independente de sequência o problema no qual sua duração depende

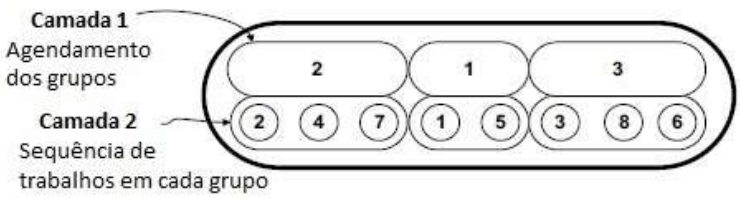
apenas do trabalho atual, e como tempo de configuração dependente da sequência o problema no qual o tempo de configuração é depende do trabalho atual e anterior. O que se entende desta descrição é que, se uma operação depende do término de operações anteriores ela é dependente de sequência, e se ela não depende do término da operação anterior, ela é independente de sequência. O problema de agendamento de máquinas com sequência dependente é conhecido por JSP.

Foram utilizados como objetivos: a minimização do *makespan* e a minimização do atraso total ponderado simultaneamente. O algoritmo foi nomeado como *Hibryd Pareto artificial bee colony algorithm* (HPABC), desenvolvido, testado e comparado com outros três algoritmos também implementados pelos autores. Os algoritmos usados para a comparação foram: algoritmo evolutivo Pareto melhorado (SPEA2) (ZITZLER, LAUMANNS e THIELE, 2001), *nondominated sorting genetic algorithm II* (NSGAII) (DEB, PRATAP, et al., 2002) e algoritmo de otimização de enxame de partículas (PSO) (KENNEDY e EBERHART, 1995).

Yue et al. (2016) utilizaram a mesma quantidade de abelhas empregadas e expectadoras, e trataram o limite de pesquisa em uma fonte de alimento como “ciclos de limite”, no qual as abelhas exploradoras da implementação escolhem aleatoriamente novas fontes de alimento, caso ocorra do contador “ciclos de limite” chegar ao limite pré-estabelecido. As fontes de alimento que são tratadas pelo algoritmo são divididas em 2 camadas: a camada de codificação e permutação e a camada de sequência de trabalhos. A camada de codificação e permutação (camada 1) é responsável por armazenar as máquinas que executarão as tarefas e a camada de sequência de trabalhos (camada 2) é responsável por armazenar a sequência dos trabalhos por grupo. A Figura 10 mostra um exemplo da abordagem dos autores.

A Figura 10 representa nos “balões” de cima as máquinas que executarão as tarefas e, nos “balões” de baixo, a sequência da realização das tarefas. No exemplo da Figura 10, o *Job* de trabalho 2 será executado pela máquina 1, a primeira operação será a primeira a ser executada e a segunda operação desse *Job* será executada na quinta operação.

Figura 10 - Representação das camadas.



Fonte: Traduzido de Yue et al. (2016).

A população inicial das fontes de alimento, conhecidas também por fontes de alimento iniciais são geradas aleatoriamente. A única restrição que existe nesta criação é que cada fonte de alimento seja executável; caso não seja, ela é descartada e novamente gerada aleatoriamente. O número de fontes de alimento geradas é igual ao número de abelhas empregadas, pois cada abelha assumirá uma fonte para pesquisar.

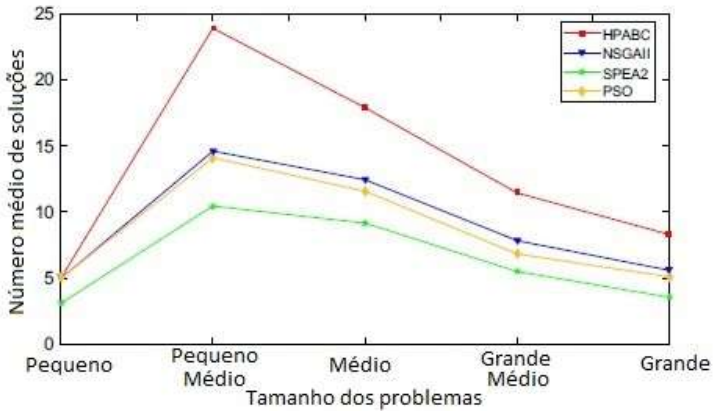
O algoritmo de colônia de abelhas artificiais (ABC), como a maioria dos outros algoritmos de busca, é influenciado principalmente por valores de parâmetros pré-estabelecidos. Yue et al. (2016) aplicaram um método conhecido como Taguchi para configuração dos parâmetros. Este método é usado para projetar um conjunto de experiências na forma de uma matriz ortogonal (MO). De acordo com os autores, este método executa diferentes conjuntos de experiências, cada uma com diferentes níveis de parâmetros e cada parâmetro reflete em vários valores. Desta forma, o Método Taguchi é utilizado para determinar o melhor conjunto de parâmetros para o funcionamento do algoritmo. Na prática, este método é utilizado para sintonizar o algoritmo e assim fazê-lo utilizar os melhores valores para o conjunto de dados que será processado.

O desempenho do algoritmo proposto HPABC foi testado usando o melhor conjunto de parâmetros obtido com o método Taguchi e foi executado 10 vezes, assim como os demais algoritmos implementados utilizados na comparação.

A Figura 11 apresentada pelos autores mostra a quantidade de soluções Pareto para cada algoritmo aplicado a diferentes conjuntos de dados. Estes conjuntos foram classificados em cinco grupos, de acordo com o tamanho dos problemas:

pequenos, pequenos médios, médios, médios grandes e grandes. Ressalta-se que quanto maior o número de soluções Pareto, melhor a eficiência do algoritmo.

Figura 11 - Quantidade de soluções Pareto.



Fonte: Traduzido de Yue et al. (2016).

O gráfico apresentado na Figura 11 representa a quantidade de soluções diferentes encontradas com o algoritmo considerando o tamanho das instâncias dispostos no eixo x do gráfico. É nítido que o algoritmo proposto representado pela linha vermelha apresenta mais diversidade de soluções encontradas que os demais algoritmos utilizados na comparação.

A conclusão obtida com esta pesquisa é que o HPABC proposto por Yue et al. (2016) é mais eficiente que os demais algoritmos implementados. Os algoritmos SPEA2, PSO e NSGAII em determinadas execuções apresentam resultados ligeiramente melhores que o HPABC, mas isso apenas mostra que os três algoritmos não têm boa repetibilidade (não consegue chegar aos mesmos valores quando executado várias vezes), de acordo com os autores, e que seus resultados podem não ser estáveis para encontrar consistentemente as melhores soluções conhecidas para problemas de grande porte. Resumindo, todos os resultados obtidos com o HPABC superam SPEA2, PSO e NSGAII, e o HPABC pode gerar bons resultados para diferentes categorias de problemas.

Este estudo foi colocado nesta seção de trabalhos com objetivos únicos mas, segundo os autores, são analisados dois objetivos simultaneamente. Foi decidido que

na seção de trabalhos com abordagens multiobjetivo seriam apresentados apenas trabalhos com abordagem de mais de dois objetivos.

Este trabalho contribuiu para a verificação da diversidade de soluções encontradas (ou seja, quantidade de soluções diferentes com os mesmos valores para os objetivos), e também contribuiu para a escolha da hibridização entre ABC e Pareto devido à qualidade das soluções apresentadas pelos autores.

3.2 Trabalhos com abordagem multiobjetivo

Os trabalhos que serão apresentados a seguir foram feitos considerando a abordagem multiobjetivo.

3.2.1 Algoritmo híbrido entre ABC, *Tabu Search* (TS) e Pareto aplicado ao FJSP multiobjetivo

Li, Pan e Tasgetiren (2014) publicaram um trabalho no qual abordam o tratamento do FJSP multiobjetivo utilizando um algoritmo de dois estágios (DABC). Os autores utilizaram o ABC juntamente com *tabu search* (TS) e Pareto. Cada uma das heurísticas utilizadas atua em um estágio diferente do algoritmo desenvolvido. O TS atua no primeiro estágio e realiza a busca local. Nesta pesquisa, o tratamento dos três objetivos (cm, wt e wm) é feito utilizando pesos para cada objetivo, e o *fitness* é obtido por: min f = cm*w1 + wt*w2 + wm*w3, onde w1,w2 e w3 representam os pesos e cm, wt e wm representam os objetivos.

A simbologia utilizada pelos autores têm a seguinte representação:

cm = *Makespan;*

wt = *Work Load Total* (Carga máxima de todas as máquinas);

wm = *Work Load Machine* (Máquina mais carregada);

O Pareto atua no segundo estágio do algoritmo reunindo as soluções de Pareto não dominadas em um conjunto de soluções. Este conjunto é utilizado como referência para as abelhas tomarem como ponto de partida e realizarem a busca local nas proximidades de cada solução não dominada. O tamanho do conjunto de soluções não dominadas é dinâmico e, caso for diferente do número de abelhas, cada abelha

escolhe aleatoriamente uma solução pra explorar utilizando as estratégias de busca local do primeiro estágio, ou seja, o TS.

Li, Pan e Tasgetiren (2014) apresentaram a análise dos resultados aplicados à várias instâncias conhecidas, como as de Kacem e Brandimarte e obtiveram resultados satisfatórios, alcançando os melhores resultados conhecidos como pode ser visto na Tabela 2, que mostra os resultados encontrados para as instâncias de Kacem. As colunas com as indicações f1, f2 e f3 representam os objetivos 1, 2 e 3 (Cm, Wt, Wm), respectivamente.

Tabela 2 - Resultados das instâncias de Kacem.

Resultados alcançados por Li, Pan e Tasgetiren (2014)			
Problema	DABC		
n x m	Cm	Wt	Wm
4 x 5	11	32	10
	11	34	9
	12	32	8
	13	33	7
8 x 8	14	77	12
	15	75	12
	16	73	13
	16	77	11
10 x 10	7	42	6
	8	42	5
	7	43	5
	8	41	7
15 x 10	11	91	11
	11	93	10

Fonte: Adaptado de Li, Pan e Tasgetiren (2014).

Os autores também apresentaram vários resultados para cada instância de Brandimarte, os quais podem ser encontrados no artigo citado.

Além da análise de resultados com as instâncias supracitadas, os autores também aplicaram o algoritmo a problemas com agendamento de tarefas de manutenções preventivas, com o intuito de utilizar parte do tempo ocioso das máquinas para efetuar as manutenções preventivas de cada máquina. Os resultados alcançados foram satisfatórios quando comparados com resultados obtidos por outros autores, como mostra Tabela 3.

Tabela 3 - Resultados para as instâncias de Kacem.

Resultados alcançados por Li, Pan e Tasgetiren (2014) comparado com os resultados encontrados por outros autores									
Problema	DABC			FBS-Based Algorithm			hGA		
n x m	Cm	Wt	Wm	Cm	Wt	Wm	Cm	Wt	Wm
8 x 8	17	105	15	18	103	16	17	105	15
	18	103	16						
10 x 10	8	61	7	8	61	7	9	60	8
	9	60	8						
15 x 10	12	107	12	n/a			12	107	12

Fonte: Adaptado de Li, Pan e Tasgetiren (2014).

Apenas para ficar mais claro, a Tabela 3 representa os valores encontrados por três algoritmos diferentes, onde os autores aplicaram o algoritmo à instâncias alteradas. Estas instâncias utilizam como base as instâncias de Kacem, mas com a inserção de manutenções preventivas nas máquinas. Estas manutenções são agendadas nos tempos ociosos das máquinas e por isso os resultados são maiores que os comummente encontrados para as instâncias de Kacem.

3.2.2 Algoritmo ABC híbrido visando apresentação de diversidade de soluções.

Chao, Lin e Lu (2017) apresentaram um algoritmo híbrido para tratar o FJSP que foi nomeado como *Artificial Bee Colony algorithm with Diversity Index Search* (ABC-DIS) e utiliza uma técnica de atribuição sequencial de operações por máquina (do inglês *SOMA Sequential Operations by Machine Attribution).* O foco desta pesquisa foi tratar o problema visando como resultados uma grande diversidade de soluções, mesmo que as soluções apresentassem os mesmos valores para os objetivos. Chao, Lin e Lu (2017) apresentaram uma diversidade grande de programações para cada resultado encontrado, como pode ser visto na coluna 5 da Tabela 4, o que mostra que o algoritmo desenvolvido é capaz de encontrar várias soluções para o problema. As instâncias que os pesquisadores utilizaram para validar a estratégia desenvolvida foram a 10x7 e a 10x10 de Kacem. A última coluna representa a diversidade de soluções encontradas, com a correspondência dos objetivos apresentados na mesma linha. A coluna wt representa a carga total das máquinas. A coluna w_m representa a carga da máquina crítica (ou seja, a máquina mais carregada) e a coluna cm representa o *makespan*.

Tabela 4 - Resultados para duas instâncias de Kacem.

Resultados alcançados por Chao, Lin e Lu (2017)				
Problema	ABC-DIS			
n x m	Wt	Wm	Cm	Qtd Soluções
10 x 7	61	11	15	22
	60	12	16	36
	62	10	15	16
10 x 10	41	7	8	66
	42	5	8	42
	43	5	7	58
	42	6	7	34

Fonte: Adaptado de Chao, Lin e Lu (2017).

A Tabela 4 mostra que o ABC-DIS tem como resultado uma grande diversidade de soluções. Esta diversidade considera os valores dos objetivos iguais para programações e/ou agendamentos diferentes. Isto mostra que o algoritmo tem grande capacidade de encontrar várias formas de chegar ao mesmo lugar por caminhos diferentes.

3.2.3 GA com algoritmo de abelhas aplicado ao FJSP multiobjetivo.

Deng, Gong et al. (2017) apresentaram uma pesquisa aplicada ao FSJP utilizando uma otimização multiobjetivo com o intuito de minimizar o *makespan,* a carga de trabalho da máquina mais carregada e a carga de trabalho total de todas as máquinas. A estratégia utilizada é composta por um mecanismo de otimização de dois estágios. No primeiro estágio é utilizado o *bee evolutionary guiding nondominated sorting genetc algorithm II* (BEG-NSGA-II) para obter a população inicial, no qual as abelhas são colocadas nas soluções iniciais para explorar exaustivamente o espaço de soluções. Cada estágio é composto por alguns passos. O primeiro estágio é composto por nove passos e o segundo estágio por sete passos, todos detalhados no trabalho. A população inicial é gerada aleatoriamente e os cálculos para o *fitness* são feitos individualmente para cada objetivo.

Os autores utilizam vários operadores genéticos como *crossover* e diversos tipos de mutação para fazer a busca no espaço de solução. De acordo com os autores, estes operadores são aplicados pelas abelhas levando sempre em consideração os três objetivos. A passagem do estágio 1 para o estágio 2 ocorre levando em conta a

qualidade das soluções encontradas até o momento. Ao final do primeiro estágio, ocorre uma seleção dos “n” melhores indivíduos e essa seleção é dividida novamente no segundo estágio, no qual passa por um torneio binário. Cada indivíduo que venceu no torneio passa por um tipo de *crossover*, e os que perderam passam por outro tipo. Feitas estas alterações, a população gerada passa por dois tipos de mutação, em seguida os indivíduos são combinados e as soluções são classificadas e avaliadas.

Um detalhe que ganha destaque por não ser comumente encontrado em outros trabalhos é o uso de valores diferentes entre as iterações do estágio 1 e as iterações do estágio 2. Os autores utilizam 100 iterações para o estágio 1 e o dobro disso para o estágio 2.

Com estas estratégias, os autores alcançaram quase todos os melhores resultados encontrados até agora para as instâncias de Kacem, Hammadi e Borne (2002), com exceção do resultado 11, 93, 10 que é considerado um dos melhores resultados para a instância 15x10, e que não foi apresentado na Tabela 5. Mesmo não encontrando este resultado, o algoritmo apresentado se mostra bastante promissor para grandes instâncias com tratamento multiobjetivo.

A Tabela 5 mostra os valores encontrados pelos pesquisadores com o algoritmo proposto aplicado às instâncias de Kacem.

Tabela 5 - Resultados alcançados.

Resultados alcançados por Deng, Gong, et al. (2017)			
Problema	BEG-NSGA-II		
n x m	Wm	Cm	Wt
4 x 5	11	10	32
	11	9	34
	12	8	32
	13	7	33
8 x 8	14	12	77
	15	12	75
	16	11	77
	16	13	73
10 x 10	7	5	43
	7	6	42
	8	5	42
	8	7	41
15 x 10	11	11	91
	12	10	93
	11	**10**	**95**

Fonte: Adaptado de Deng, Gong, et al. (2017)

Este trabalho se mostrou bastante relevante para a pesquisa considerando que os autores aplicaram o algoritmo proposto a várias instâncias que são comumente utilizadas na literatura e encontraram vários dos melhores resultados conhecidos, os quais utilizaremos como referência para fazer a validação e verificar a eficiência do algoritmo desenvolvido.

Embora os autores não tenham utilizado o ABC, foi utilizado um princípio de evolução das abelhas que auxilia o GA na varredura do espaço de soluções. As estratégias apresentadas que permitem que o algoritmo seja capaz de sair dos ótimos locais podem ser implementadas na proposta que será apresentada no próximo capítulo.

3.3 Hibridização com outras meta-heurísticas

Alguns trabalhos utilizados como referência tiveram uma abordagem híbrida, onde os autores utilizaram pelo menos uma segunda meta-heurística juntamente com o ABC.

Estas hibridizações possibilitaram resultados bons na maioria das vezes, e como o uso desta meta-heurística auxiliar trouxe avanços aos pesquisadores, este trabalho também explorará tal estratégia.

Muthiah, Rajkumar e Rajkumar (2016) apresentaram em seu trabalho uma hibridização de ABC com *Particle Swarm Optimization* (PSO) para resolver o FJSP, e em uma abordagem monoobjetivo, utilizando como critério de minimização o *makespan*, obtiveram resultados melhores do que os encontrados apenas com ABC ou PSO separadamente.

Li e Peng *et al.* (2017) apresentaram em seu trabalho uma abordagem híbrida utilizando ABC e *tabu search* (TS) para chegar em resultados iguais aos melhores resultados conhecidos, eles apresentaram uma comparação entre sete algoritmos aplicados a 13 instâncias diferentes, e a abordagem híbrida obteve resultados satisfatórios, não alcançando o melhor resultado conhecido em apenas uma instância.

Li, Pan e Gao (2011) trouxeram uma abordagem híbrida direcionada ao tratamento multiobjetivo utilizando ABC e Pareto para tratar o FJSP. Os autores aplicaram o algoritmo desenvolvido a cinco instâncias e alcançaram os melhores valores conhecidos até o momento, provando que esta abordagem híbrida utilizada é promissora.

Li, Pan e Tasgetiren (2014) apresenteram uma abordagem híbrida entre ABC e TS, e Pareto, utilizando duas meta-heurísticas como auxiliares do ABC. A proposta dos autores foi aplicada a cinco instâncias de Kacem e os resultados encontrados foram comparados com vários outros autores. Os resultados obtidos por Li, Pan e Tasgetiren (2014) foram satisfátórios, alcançando os melhores resultados conhecidos para cada uma das cinco instâncias.

Diversos outros autores têm utilizado atualmente propostas híbridas para tratar as várias instâncias de FJSP existentes. Estes têm alcançado os melhores resultados conhecidos como pode ser visto acima na breve demonstração. Diante disto este trabalho também utilizará uma abordagem hibrida para alcançar os resultados.

3.4 Considerações finais do capítulo

Com base nesta pesquisa bibliográfica direcionada a alguns métodos de solução para o FJSP, levando em consideração todas as outras meta heurísticas, será apresentada uma proposta de trabalho no próximo capítulo para maior aprofundamento no algoritmo ABC híbrido com Pareto, que se mostrou bastante promissor.

Alguns métodos se mostram bastante promissores para serem explorados em busca de melhores resultados. Dentre os métodos interessantes encontrados nos trabalhos utilizados, Pareto, e alguns métodos de busca local são estratégias que podem ser exploradas e possivelmente apresentar resultados satisfatórios para o tratamento do problema abordado. No capítulo 4 será exposta uma proposta de solução para o FJSP utilizando ABC como metaheurística o método de Pareto como método auxiliar.

CAPÍTULO 4

4 PROPOSTA DO TRABALHO

A proposta de pesquisa deste trabalho é desenvolver um método que seja capaz de solucionar o problema *flexible job-shop scheduling* (FJSP) multiobjetivo. Esta proposta consiste em utilizar a meta-heurística *algorithm bee colony* (ABC) adaptada para algoritmos discretos, com algumas alterações que possam trazer melhorias consideráveis ao desempenho do programa, assim como implementar meta-heurísticas auxiliares para melhorar a eficiência e atingir os objetivos propostos de resolução do FJSP multiobjetivo.

O desenvolvimento do algoritmo é avaliado com relação aos objetivos da proposta apresentada e validado com a aplicação de diversas instâncias de benchmarks conhecidos, como Kacem, Hammadi e Borne (2002) e Brandimarte (BR) (1993). Com a aplicação destas instâncias torna-se possível fazer a comparação com os resultados de outros pesquisadores que utilizaram os mesmos benchmarks em outras pesquisas.

Nos ambientes de manufatura do tipo FJSP, onde são encontrados problemas de agendamento e roteamento de operações, segundo a literatura, podem surgir alguns problemas:

1. Necessidade de diminuir os custos computacionais utilizados no tratamento dos problemas de programação da produção;
2. Necessidade de obter várias programações com certo nível de confiabilidade, para garantir que não se está desperdiçando recursos;
3. Necessidade de tratamento dos subproblemas gerados pelas atividades de programação e roteamento, em tempo considerável, devido à complexidade dos problemas;

Considerando estes problemas, foi elaborada uma proposta com o intuito de atender às necessidades existentes atualmente.

Para tratar o problema 1 é proposta a metaheurística ABC, que trabalha com heurísticas e tem custo computacional inferior às técnicas exatas. Esta proposta foi baseada nos trabalhos de Muthiah, Rajkumar e Rajjumar (2016), Zheng, Lian e Mesghouni (2014), Choo, Wong e Khader (2016), Yue et al. (2016), Li, Pan e

Tasgetiren (2014), Chao, Lin e Lu (2017), Deng, Gong et al. (2017), Li, Pan e Gao (2011), Gao et al. (2015). Estes autores mostraram em seus trabalhos que a utilização de metaheurísticas para o tratamento de problemas complexos é muito eficiente do ponto de vista computacional e os resultados obtidos são satisfatórios na maioria das vezes.

Para tratar o problema 2 é utilizada uma estratégia de apresentação dos resultados semelhante à utilizada por Yue, Guan, et al. (2016). Os autores apresentam as quantidades de soluções diferentes com os mesmos resultados para os mesmos valores de objetivos que o algoritmo pode encontrar. Com isso, os gestores de produção podem ter uma grande variedade de programações diferentes com os mesmos objetivos desejados, fazendo uma máquina ou outra trabalhar mais ou menos que outra sem alterar os custos.

Para tratar o problema 3 é proposta a modelagem de separar o FJSP em dois subproblemas, sendo eles roteamento e agendamento. Esta técnica de modelagem é semelhante à utilizada pela maioria dos autores, como: Thammano e Phu-Ang (2013), Zhou et al. (2011), Li, Pan e Gao (2011), Li, Pan e Xie (2011), Gao et al. (2015), Li, Pan e Tasgetiren (2014). Estes autores apresentam os dois subproblemas através de dois vetores distintos, dos quais um armazena o roteamento e o outro armazena o agendamento. Para se obter uma solução, são necessários os dois vetores. A modelagem que é proposta utiliza um vetor que já é criado previamente agendado. Este vetor é formado inicialmente com um agendamento suficiente para representação de uma solução. Esta modelagem proposta utiliza, além deste vetor de roteamento, uma matriz de troca de operações, que tem a funcionalidade de reagendar as operações caso necessário, estratégia que se difere dos trabalhos encontrados na literatura e será detalhada neste capítulo.

Para tratar o problema do roteamento, é proposto o operador de mutação. É previsto que este operador atue em todas as etapas do algoritmo, tais como: etapa de busca das abelhas empregadas, etapa de busca das abelhas expectadoras e etapa de busca das abelhas exploradoras.

Para auxiliar a etapa das abelhas exploradoras é proposta a metaheurística Dominância de Pareto, utilizada por Li, Pan e Tasgetiren (2014). É determinada uma quantidade de soluções dominantes para compor um arquivo de Pareto, no qual apenas as soluções dominantes (melhores soluções) são usadas como referência

para as abelhas exploradoras escolherem. Para se chegar às melhores soluções, é proposta uma função que unifica os valores dos objetivos em apenas um valor, assim, o problema se torna, de certa forma, mono objetivo.

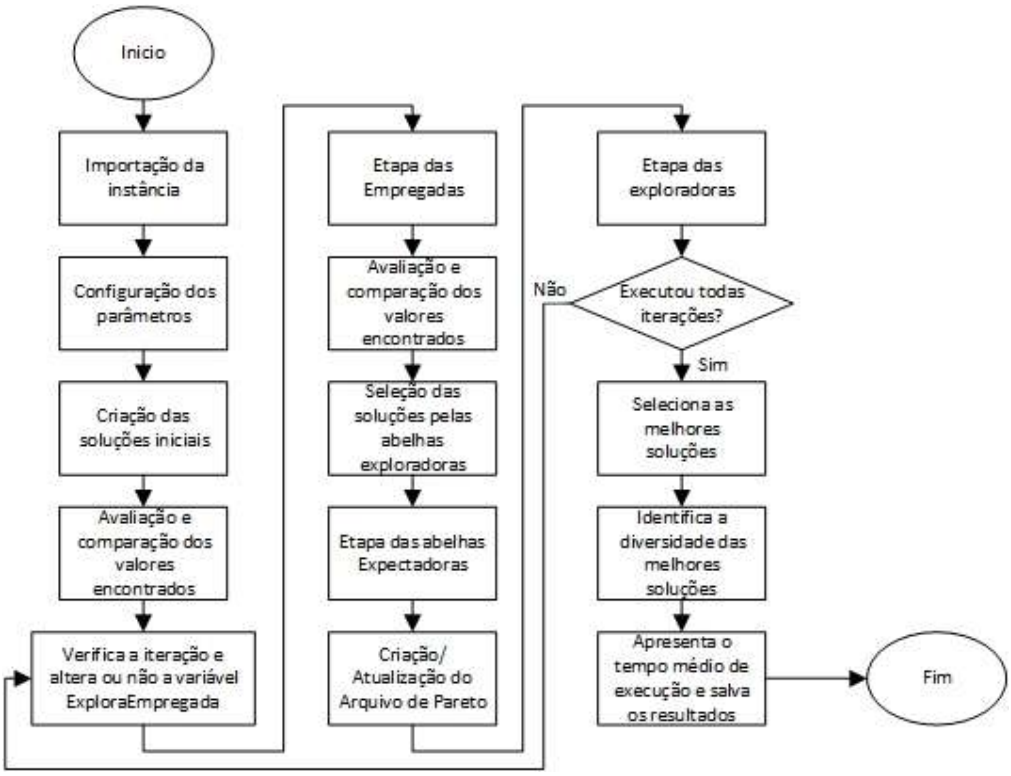
Embora os vetores de roteamento sejam construídos inicialmente agendados, de forma que a solução seja factível, as melhores soluções conhecidas são obtidas com o agendamento final do vetor de roteamento. Este agendamento final ocorre através de uma matriz que faz alterações no agendamento das operações.

A técnica de Fronteira de Pareto utilizada por Yue, Guan, et al. (2016) e Deng et al. (2017) é proposta como forma de apresentação dos resultados. Esta técnica é exposta em forma de gráfico, e apresenta de forma intuitiva a qualidade dos resultados comparados com os existentes na literatura.

Através da presente pesquisa, considerando as técnicas propostas para a abordagem ABC, espera-se que os resultados apresentados sejam de qualidade iguais ou superior aos encontrados na literatura para as instâncias de Kacem, Hammadi e Borne (2002) e Brandimarte (1993).

Para facilitar o entendimento das etapas fundamentais do código descritas no próximo item, a Figura 12 ilustra a rotina principal do código de forma não detalhada. O fluxograma é apresentado apenas com o intuito de orientar o leitor em relação ao local do algoritmo ao qual se refere cada etapa.

Figura 12 - Fluxograma geral do código, representando a função principal.



Fonte: Esta pesquisa.

4.1 Descrição dos métodos e estratégias utilizadas

Nesta seção são detalhadas todas as estratégias utilizadas para a criação do algoritmo.

Inicialmente, é realizada a importação dos dados do *benchmark* que será utilizado para a otimização, as instâncias são carregadas na memória do programa em forma de matrizes e através de uma função. A Tabela 6 ilustra o formato da importação da instância 4x5 (KACEM, HAMMADI e BORNE, 2002) com 4 *jobs* e 5 máquinas.

Tabela 6 - Importação dos tempos da instância 4x5.

Job	Op	M1	M2	M3	M4	M5
Job1	Op1	2	5	4	1	2
	Op2	5	4	5	7	5
	Op3	4	5	5	4	5
Job2	Op1	2	5	4	7	8
	Op2	5	6	9	8	5
	Op3	4	5	4	54	5
Job3	Op1	9	8	6	5	9
	Op2	6	1	2	7	4
	Op3	2	5	4	2	4
	Op4	4	5	2	1	5
Job4	Op1	1	5	2	4	12
	Op2	5	1	2	1	2

Fonte: (KACEM, HAMMADI e BORNE, 2002).

Os títulos das linhas e colunas não são importados e foram inseridos na figura apenas para orientação.

4.1.1 Configurações dos parâmetros de ajuste do código

Nesta etapa do algoritmo são inseridos todos os parâmetros necessários para o funcionamento do algoritmo, no qual as configurações são feitas para que o código seja capaz de tratar várias instâncias com características diferentes.

Serão detalhados aqui os parâmetros de configuração que são essenciais para o funcionamento. Os parâmetros que são apenas visuais, ou seja, modos de saída de dados e apresentação de objetos, estão descritos e brevemente explicados na Tabela 7.

O código foi construído com a capacidade de configuração de quantas vezes o algoritmo será executado. Assim, fica mais fácil de reunir resultados de várias execuções e montar gráficos com estatísticas. Esta configuração não apresenta limite máximo para definição, entretanto, este código foi desenvolvido de forma que o parâmetro é limitado pela quantidade de memória RAM disponível no computador, com algumas alterações esta característica pode ser alterada e o código poderia não ter limite de execução.

O parâmetro t é a quantidade de iterações que o algoritmo executará.

O parâmetro N_Bee é a quantidade de abelhas que serão utilizadas. Geralmente esta quantidade é proporcional ao tamanho do problema que será tratado. Quanto maior o problema maior deve ser a quantidade de abelhas para se obter resultados bons.

O parâmetro Limite configura o limite de “não melhora” para que as abelhas abandonem as possíveis soluções. Este limite define quantas iterações uma solução é explorada sem que apresente melhoria, antes de ser abandonada. Após testes feitos com várias instâncias de características diferentes, o valor desta variável ficou sempre entre 15 e 20.

O parâmetro ExploraEmpregada é uma variável que tem seu valor decrescente com a evolução das iterações. É utilizada para ajustar a característica de busca das abelhas empregadas entre exploração e explotação. Quanto maior o valor desta variável (ExploraEmpregada) maior é a capacidade de explotação da abelha empregada. Esta configuração é feita inicialmente com valores próximos a 20, e no decorrer do avanço das iterações este valor vai diminuindo até chegar a 2. Quando o valor chega a 2, o algoritmo já executou 90% das iterações e a característica de busca local desta abelha já não é mais explotação como era inicialmente. Neste ponto das execuções a explotação é feita pelas abelhas exploradoras, no momento que faz perturbações nas soluções.

O valor da variável ExploraEmpregada determina quantas alterações serão feitas por solução por vez, e esta estratégia será ilustrada na etapa das abelhas empregadas.

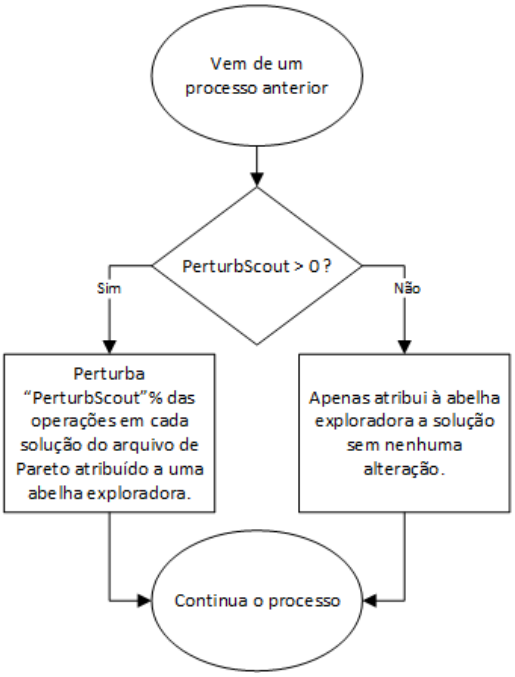
O próximo parâmetro de configuração determina se a variável anterior vai passar de explotação para exploração no decorrer das iterações ou se vai se manter explotando do início ao fim do código. Esta variável é a “ExploracaoEmpregadaVariavel”. Se esta estiver definida como “sim”, a variável ExploraEmpregada é alterada no decorrer das iterações. Mas, se tiver definida com o “não”, o valor de ExploraEmpregada permanece do início ao fim em um valor fixo previamente definido.

LFP é o parâmetro que determina o tamanho do arquivo de Pareto que será criado. Esta variável é configurada com valores próximos de 0,2, que corresponde a

20% da quantidade de abelhas. Este valor é utilizado com base na literatura e, através de testes empíricos, pôde ser concluído que arquivos muito grandes geram uma lentidão desnecessária no código. O arquivo de Pareto é gerado sempre com as melhores soluções encontradas (20% melhores soluções), entretanto, se o arquivo é configurado com valores maiores, mais soluções ruins passam a compô-lo.

Este tipo de situação gera um possível problema em algumas instâncias que têm tendência a estagnação. Por este motivo foi criada a variável "*PerturbScout*". Esta configuração é maior que 0 (zero) quando é desejado que as abelhas exploradoras perturbem a solução de Pareto selecionada. A fonte de soluções utilizadas pelas abelhas exploradoras está no arquivo de Pareto. E estas soluções podem ou não serem perturbadas pelas abelhas exploradoras quando são selecionadas. O valor que fica contido na variável *PerturbScout* corresponde a porcentagem de operações que serão perturbadas pelas abelhas exploradoras, conforme ilustrado na Figura 13.

Figura 13 - Fluxograma de funcionamento da perturbação das soluções



Fonte: Esta pesquisa.

Os próximos parâmetros de configuração correspondem às definições do sistema de busca desenvolvido para as abelhas expectadoras. Este método de busca

é composto por três valores: M1, M2 e M3. A soma dos três valores deve ser igual a 100 (que corresponde a 100% das operações). Estes valores definem qual a porcentagem de chance das abelhas expectadoras fazerem sua busca local em cada modo de busca, entre os três que estão sendo utilizados. A funcionalidade dos modos de operação desta configuração é descrita detalhadamente na etapa das abelhas expectadoras.

Os parâmetros w_{cm}, w_{wt} e w_{wm} são os três pesos de configuração da função *fitness*. Cada um dos pesos corresponde a um objetivo que está sendo tratado. Estes valores são configurados de forma diferente para cada problema, e fazem com que haja uma tendência maior ou menor para cada objetivo de acordo com a configuração e de acordo com a necessidade das características de cada instância.

O algoritmo também conta com algumas configurações de apresentação de resultados, sinalizações sonoras que avisam quando foi alcançado algum resultado procurado e que configuram a forma como será exibida a representação da evolução das iterações. Entretanto, estas não influenciam na qualidade das soluções e as variáveis responsáveis foram omitidas destas apresentações.

Finalizando as configurações iniciais, o algoritmo passa para a etapa de carregamento dos dados da instância a ser tratada e chama a função que cria as soluções iniciais.

Conforme mencionado acima a Tabela 7 representa todas as configurações do algoritmo de forma sucinta.

Tabela 7 - Parâmetros e configurações do código

Lista de configurações do algoritmo		
nº	Nome do Parâmetro	Função
1	instancia	Definir qual a instância que deve ser carregada para tratamento.
2	SearchCm	Variável que armazena o valor do objetivo Cm desejado
3	SearchWt	Variável que armazena o valor do objetivo Wt desejado
4	SearchWm	Variável que armazena o valor do objetivo Wm desejado
5	BuscaComp(1 a n,1:3)	Conjunto de 3 valores variando de 1 a "n" correspondentes a [Cm Wt Wm]. Estes valores serão buscados em todas as soluções encontradas pra contabilização da diversidade encontrada. (Utilizado para valores não ótimos, pois os valores ótimos o algoritmo busca por padrão e apresenta a diversidade)
6	t	Variável que indica quantas vezes o algoritmo vai ser executado, o limitante é a quantidade de memória RAM.
7	max_iteracoes	Quantidade máxima de iterações que serão executadas por execução.
8	N_Bee	Quantidade de abelhas que serão utilizadas, abelhas empregadas e expectadoras.
9	Limite	Variável que determina o limite de busca em uma solução sem melhorar.
10	ExploraEmpregada	Quantidade de alterações que serão feitas nas soluções pelas abelhas empregadas, este valor pode ou não ser reduzido no decorrer das iterações.
11	ExploracaoEmpregadaVariavel	Variável que determina se a variável anterior será descendente ou se será fixa ao longo do código.
12	LFP	Length Fitness Pareto, esta variável determina a porcentagem de soluções que comporão o arquivo de Pareto. O conteúdo desta variável será multiplicado pela quantidade de abelhas para determinar o tamanho do arquivo de Pareto.
13	PerturbScout	Determina se as abelhas batedoras farão perturbações nas soluções do arquivo de Pareto. Esta variável varia de 1 a 100% de operações perturbadas por solução.
14	MinimizationCriterion	Define o objetivo que será minimizado caso encontre a solução desejada, cujos objetivos foram inseridos nas variáveis 2, 3 e 4. Recebe os valores 'Makespan' e 'WorkLoad'.
15	AutomatcMinCriterion	Se for configurado como 'Não', a variável 14 se mantém até o fim do código. Se configurado como 'Sim' a variável 14 alterna entre os dois valores cada vez que um é encontrado.
16	DefineBuscaManual	Esta variável determina se a busca pelos objetivos será automática ou manual, se o valor for 'Sim' os objetivos buscados são os que estão armazenados nas variáveis 2, 3 e 4. Se o valor for 'Não' a selecionados aleatoriamente valores ótimos conhecidos para serem procurados.
17	M1	Armazena a procentagem de chance do modo 1 de busca ser seleconado pela abelha expectadora. Pode variar de 1 a 100. entretanto a somatória de M1, M2 e M3 deve ser 100.
18	M2	Armazena a procentagem de chance do modo 2 de busca ser seleconado pela abelha expectadora. Pode variar de 1 a 100. entretanto a somatória de M1, M2 e M3 deve ser 100.
19	M3	Armazena a procentagem de chance do modo 3 de busca ser seleconado pela abelha expectadora. Pode variar de 1 a 100. entretanto a somatória de M1, M2 e M3 deve ser 100.
20	WCm	Peso do objetivo Cm, utilizado para aumentar ou diminuir a relevância dos valores de Cm na equação fitness
21	WWt	Peso do objetivo Wt, utilizado para aumentar ou diminuir a relevância dos valores de Cm na equação fitness
22	WWm	Peso do objetivo Wm, utilizado para aumentar ou diminuir a relevância dos valores de Cm na equação fitness
23	ApresentaTempoMedio	Define se o tempo médio de execução será apresentado no fim de cada execução. ('Sim' ou 'Não')
24	Apitar	Define se soará ou não um apito quando encontrar os valores desejados. ('Sim' ou 'Não')
25	ArmazenarCadaSolucaoEncontrada	Variável que determina se todas as soluções por abelha por iteração será armazenada, esta variável deve ficar habilitada para a contagem da diversidade final. ('Sim' ou 'Não')
26	ShowIterations	Mostra a iteração atual na janela de comando. Se sim deixa o código mais lento. ('Sim' ou 'Não')
27	WaitBar	Mostra a evolução da execução através de uma barra de carregamento (WaitBar). Se ativa inabilita a variável 26. ('Sim' ou 'Não')
28	ResetaCriteriosCadaExecucao	Utilzado apenas quando utilziar 2 ou mais execuções (configurado pela variável 6). Se habilitada os critérios de busca serão resetados todo início de execução. ('Sim' ou 'Não')
29	PrintFrontier	Utilzada para gerar ou não a fronteira de pareto da execução. ('Sim' ou 'Não')

4.1.2 Criação das soluções iniciais

As soluções iniciais são geradas através de uma função, que consiste em criar uma roleta probabilística e distribuir os tempos das máquinas para cada operação. Esta roleta inicialmente identificava e excluía a máquina mais lenta (pior máquina para executar a operação) e, com isso, esta máquina nunca participava das soluções iniciais. Entretanto, após orientações e testes, ficou claro que não era a melhor solução para aplicar ao problema. A proposta foi alterada e atualmente funciona com a penalização da pior máquina em 10% do tempo desta máquina.

Exemplo: se uma máquina tem tempo de 10 ut e é a máquina mais lenta, apenas para a distribuição na roleta, este tempo passa a ser considerado 11 (10% a mais do seu tempo de 10 ut). Com esta estratégia, a possibilidade desta máquina ser escolhida diminui um pouco, mas não é eliminada, como era feito na versão anterior.

O pseudocódigo para esta etapa é o seguinte:

a. *Seleciona a operação i, encontra a máquina que tem o pior tempo e penaliza ela com 10% do seu valor.*
b. *Encontra as possíveis máquinas com os respectivos tempos para executar essa operação.*
c. *Distribui em uma roleta probabilística os tempos, dando uma maior fatia para o menor tempo e sorteia a máquina que vai executar essa operação.*
d. *Volta ao "a" e faz isso para todas as soluções, e para todas as abelhas.*
e. Fim.

Este valor de penalização pode ser alterado. No entanto, os testes realizados mostraram bons resultados com a penalização de 10%. Os testes foram feitos aplicando de 10% a 60% de penalidade para as piores máquinas nas soluções iniciais.

4.1.3 Função fitness

Esta função é definida para simplificar o tratamento dos objetivos. É executada toda vez que ocorre alguma alteração nas soluções e a primeira execução ocorre após a criação das soluções iniciais. A função utilizada é a somatória do produto dos objetivos normalizados, multiplicados pelos pesos previamente definidos. Esta função é representada pela equação 6.

$F_n = cm_n * w_{cm} + wt_n * w_{wt} + wm_n * w_{wm}$ | { cm, wt e wm } esteja normalizado.....(6)

Onde:

f_n representa o *fitness* da solução n.

cm_n representa o objetivo *makespan* da solução n.

wt_n representa o objetivo carga total das máquinas da solução n.

wm_n representa o objetivo carga da máquina mais carregada da solução n.

w_{cm} representa o peso predefinido para o conjunto de valores do *makespan*.

w_{wt} representa o peso predefinido para o conjunto de valores da caga total das máquinas.

w_{wm} representa o peso predefinido para o conjunto de valores da carga da máquina mais carregada.

{ cm, wt e wm } impõe a condição de que os conjuntos de objetivos estejam normalizados para não haver tendência dos valores maiores em relação aos menores.

Com esta estratégia o problema se torna mais fácil de ser tratado, mesmo que aumente um pouco a dificuldade de sintonização dos pesos.

4.1.4 Agressividade das alterações feitas pelas abelhas empregadas (explotação x exploração)

Esta etapa do código verifica quantas iterações já foram executadas pois, levando em consideração a evolução do algoritmo, as abelhas empregadas passam a diminuir a explotação. Com esta estratégia, as soluções são cada vez menos alteradas com o passar do tempo, fazendo com que as abelhas empregadas passem a explorar a vizinhança e não mais explotar. Isto faz com que o algoritmo alcance valores bons mais rapidamente.

4.1.5 Representação da solução

Uma solução completa pode ser representada por apenas um vetor. Este é composto por todas as máquinas que executarão as operações e posteriormente pela referência da ordem que as máquinas executarão as operações. As operações são

representadas pelo índice das colunas onde estão armazenadas as máquinas. A Tabela 8 apresenta um exemplo de um vetor de solução ilustrativo.

Tabela 8 - Vetor completo de uma solução

Máquinas											Posição da execução										
3	4	1	1	2	2	2	5	4	1	3	4	7	1	4	11	2	8	12	5	3	9	6	10

Fonte: Este trabalho.

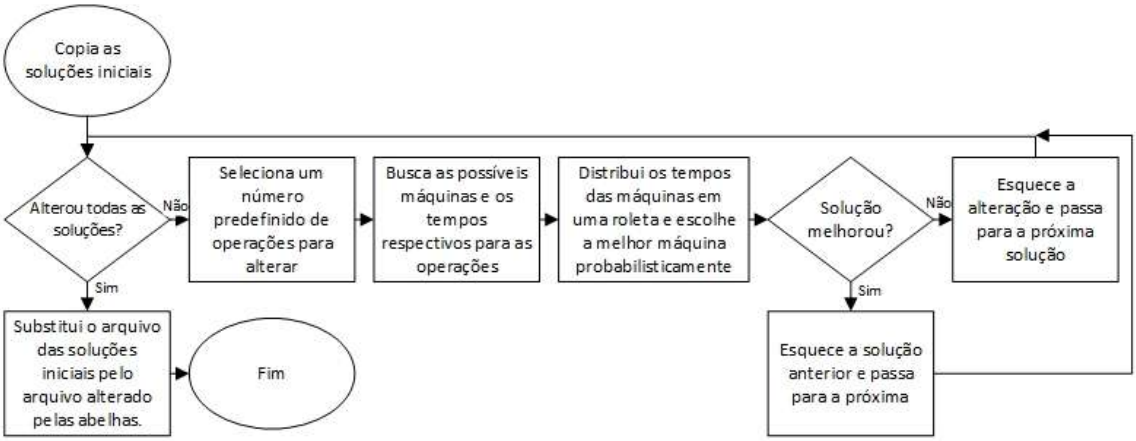
A alteração de uma solução é feita de duas formas: alterando uma das máquinas da primeira parte do vetor, lembrando que no caso de uma instância parcialmente flexível, a máquina substituta deve ser factível; A outra forma de alterar é utilizar a operação *suap* na segunda parte do vetor, fazendo com que a ordem dos agendamentos seja alterada, mas para essa estratégia deve-se levar em consideração as restrições do problema, a ordem das operações dentro de cada *job*.

4.1.6 Etapa das abelhas empregadas

É proposta para esta etapa uma roleta onde são distribuídas as possíveis máquinas para comporem as alterações nas soluções iniciais. A quantidade de abelhas empregadas e soluções iniciais é a mesma, assim cada abelha assume uma solução para alterar. O funcionamento desta roleta é análogo ao das soluções iniciais, a diferença é que este processo não penaliza a pior máquina, permitindo que todas tenham as mesmas chances, proporcionalmente.

A Figura 14 apresenta um fluxograma das operações realizadas por esta etapa.

Figura 14 - Fluxograma de representação da etapa das abelhas empregadas.



Fonte: Este trabalho.

Embora esta etapa seja facilmente explicável, o funcionamento completo é um pouco mais complexo. Esta parte do código leva em consideração as instâncias parcialmente flexíveis e atribui as operações apenas às máquinas que podem executar cada processo (máquinas factíveis). Esta característica faz com que não seja necessário nenhum outro tratamento nas soluções geradas pelas abelhas empregadas, e isto aumenta um pouco a velocidade de execução em instâncias com estas características.

O conjunto de abelhas empregadas também é responsável por verificar se as soluções encontradas são melhores que as soluções anteriores. Caso forem melhores, as soluções são armazenadas e as anteriores são esquecidas. Mas se as soluções encontradas forem piores, elas não são memorizadas e o contador “Limite” é incrementado individualmente por solução. A verificação se a solução melhorou é feita através da função *fitness,* explicada anteriormente.

4.1.7 Etapa das abelhas expectadoras

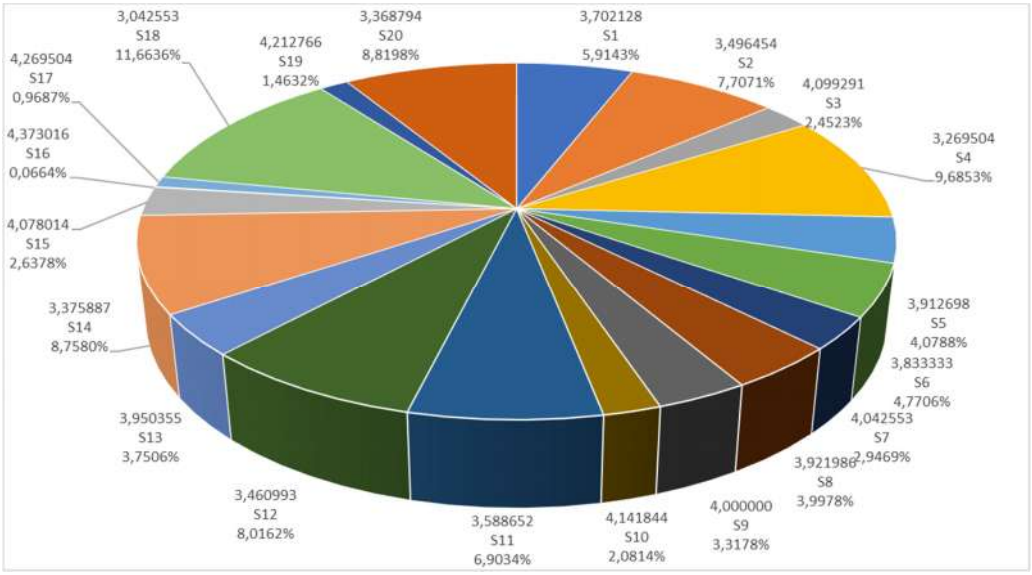
A etapa das abelhas expectadoras ocorre posteriormente à atribuição das soluções encontradas pelas abelhas empregadas. Esta atribuição ocorre baseada em uma roleta que distribui os valores de *fitness* das soluções de forma inversamente proporcional, ou seja, quanto menor o *fitness* maior a fatia da roleta. Neste caso, o

fitness representa a solução respectiva e, desta forma, o problema pode ser tratado mais facilmente do que considerando os 3 objetivos que são abordados.

O *fitness* é gerado da mesma forma em todas as etapas do algoritmo, então, não será abordada a equação toda vez que for citada.

Para facilitar a visualização desta proposta, é ilustrada na Figura 15 a roleta de atribuição das soluções com as porcentagens de vinte soluções. Os valores foram gerados apenas para ilustração, considerando que um problema comum utiliza muito mais que 20 soluções, o que tornaria inviável a apresentação ilustrativa.

Figura 15 - Roleta probabilística ilustrativa.



Fonte: Este trabalho.

Nesta roleta probabilista são distribuídas todas as soluções com fatias de tamanhos proporcionais às qualidades das soluções. Quanto melhores forem as soluções, maior será a probabilidade da abelha expectadora escolher a solução.

Como pode ser observada nesta roleta ilustrativa da Figura 15, a pior solução entre as 20 representadas, é a solução S16 e a melhor é a solução S18. A solução S16 conta com o maior *fitness,* sendo 4,373017 e uma porcentagem de chance de ser escolhida de 0,0664%, já a solução S18 conta com o menor *fitness* de 3,042553 e uma porcentagem e chance de ser escolhido de 11,6636%.

Após a distribuição de todos os *fitness* na roleta, cada abelha expectadora é atribuída a uma solução através de um sorteio, podendo uma mesma solução ser atribuída para várias abelhas. Desta forma, as melhores soluções são atribuídas a várias abelhas, e assim são mais exploradas que as soluções piores.

O sistema de busca para as abelhas expectadoras é diferente do sistema de busca das abelhas empregadas. Consiste em três métodos de busca local que têm probabilidades configuráveis de serem escolhidos. Os métodos são sorteados de forma que a probabilidade de cada um ser atribuído é definido no começo do código. Cada um dos três métodos é configurado de acordo com a necessidade e caraterística da instância que será utilizada.

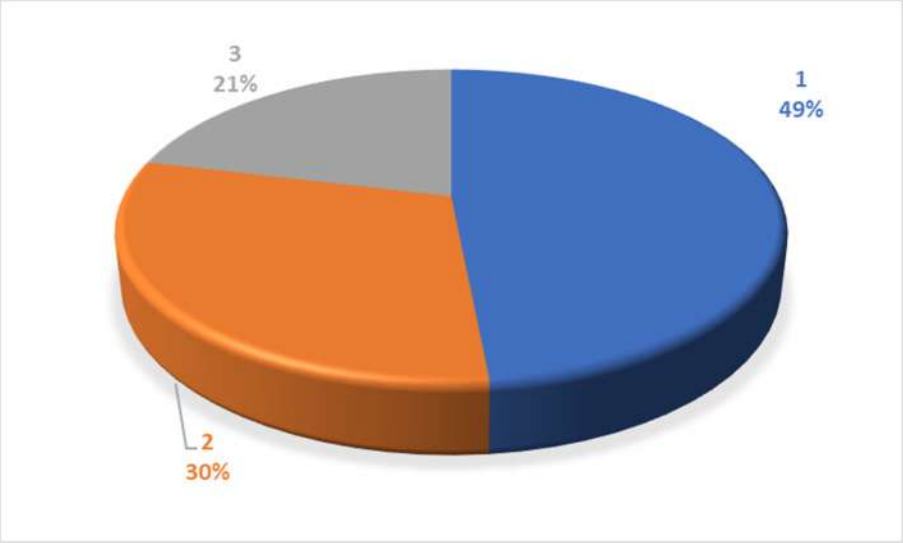
A única restrição desta configuração é que a soma dos três valores totalizem 1, que equivale a 100% das buscas das abelhas expectadoras.

Os três métodos de busca utilizam como prioridades:

Método 1: roleta probabilística entre as três melhores máquinas.

Este método identifica as três melhores máquinas e distribui os tempos proporcionalmente em uma roleta, para escolher uma entre elas. A máquina escolhida é sempre uma das três melhores entre as máquinas factíveis. A roleta desta etapa é ilustrada na Figura 16.

Figura 16 - Roleta ilustrativa do modo de busca M1 da abelha expectadora.



Fonte: Este Trabalho.

Os tempos que foram distribuídos para gerar esta roleta foram de 2, 4 e 5, que correspondem aos índices 1, 2, e 3, respectivamente.

Método 2: melhor máquina em tempo de execução e desempate por máquina mais ociosa.

Este método, como o próprio nome já diz, utiliza a melhor máquina factível em tempo de execução, independentemente da máquina estar muito carregada. Entretanto, quando duas ou mais máquinas são as melhores (Ex.: duas máquinas que consomem 1ut para executar a operação), um outro critério de seleção passa a valer e, neste momento, é considerada a máquina que está menos carregada entre as duas melhores. Esta estratégia tende a aumentar bastante a velocidade de convergência, mas induz um viés em instâncias que contam com uma das máquinas muito mais rápida que as demais.

Método 3: máquina mais ociosa.

Este último método de funcionamento considera a máquina mais ociosa para executar a operação. Este parâmetro não considera nenhuma outra estratégia, e compensa um pouco a tendência do método anterior pois, enquanto o anterior considera a máquina mais rápida, esse considera a mais ociosa, fazendo com que o algoritmo não fique sempre selecionando a mesma máquina para todas as operações.

A escolha do método é feita de forma probabilística pelas abelhas expectadoras. Há três variáveis que determinam as chances de cada método ser escolhido e estas variáveis são configuradas no início do código (M1, M2 e M3). Os valores da configuração variam de instância para instância, pois cada uma tem sua característica e isso é levado em consideração para a configuração.

O conjunto de abelhas expectadoras também é responsável por verificar se as soluções encontradas são melhores que as soluções anteriores. Caso forem melhores, as soluções são armazenadas e as anteriores são esquecidas. Mas se as soluções encontradas forem piores, elas não são memorizadas e o contador “Limite” é incrementado individualmente por solução. O contador “Limite” é vinculado às soluções, então, se a solução não melhorou com a alteração das abelhas empregadas, estas acrescentam 1 ao contador e, se a mesma solução não melhorou com os procedimentos realizados pelas abelhas expectadoras, o contador vai a 2., lembrando que, quando o contador chega ao valor limite pré-definido, uma abelha

exploradora substitui a solução por outra, utilizando procedimentos que serão detalhados mais à frente.

4.1.8 Arquivo de Pareto

O arquivo de Pareto é um conjunto gerado após as abelhas expectadoras executarem as buscas. É um conjunto composto por uma fração das melhores soluções encontradas até o momento. A variável que determina o tamanho deste conjunto é a LFP, predefinida no início do algoritmo. Esta configuração cria um conjunto de tamanho fixo que não varia ao longo da execução.

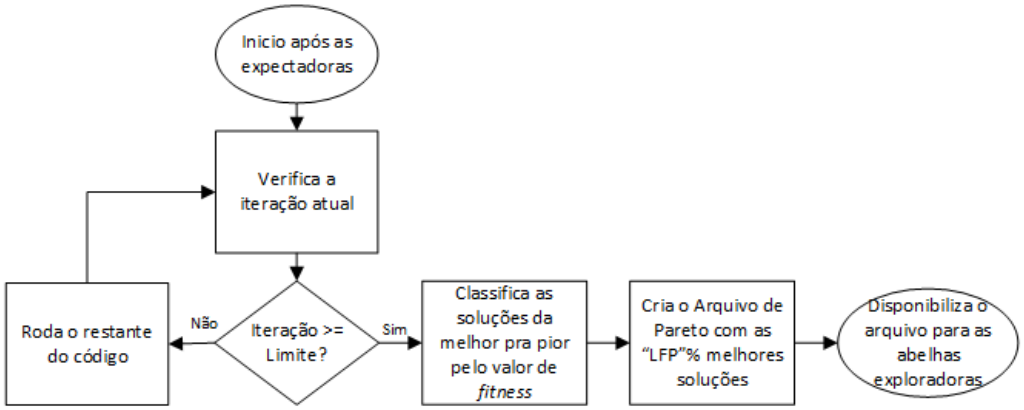
A composição deste conjunto é feita após a classificação das soluções encontradas até o momento pelas abelhas expectadoras. Estas soluções são salvas e utilizadas pelas abelhas exploradoras durante o processo de busca de novas soluções.

Como a proposta trabalha com multiobjetivos, a classificação é feita levando em consideração o *fitness* das soluções. O tamanho do conjunto, durante os testes preliminares, geralmente variou entre 10 e 30% do total de abelhas. Os melhores resultados preliminares foram encontrados com o conjunto tendo 20% do total das abelhas.

Uma das estratégias para deixar o algoritmo mais rápido nas primeiras iterações consiste em inicializar o Arquivo Pareto após algumas iterações. O arquivo só começa a ser utilizado pelas abelhas exploradoras após o decorrer de “n” iterações, onde “n” é o número predefinido de alterações Limite, ou seja, se esta solução não melhorar em “n” iterações, ela será substituída por uma solução encontrada pela abelha exploradora. Isso indica que, se o Arquivo de Pareto for gerado logo nas primeiras iterações, ocorrerão vários procedimentos que serão executados, mas que não serão utilizados, como a classificação das soluções para determinar as melhores, a construção do arquivo, entre outras.

Para ficar mais clara a explicação acima, a Figura 17 ilustra o fluxograma da estratégia descrita.

Figura 17 - Fluxograma da etapa de criação do arquivo de soluções Pareto.



Fonte: Este trabalho.

4.1.9 Etapa das abelhas exploradoras.

Esta etapa consiste basicamente em substituir as soluções que não melhoraram em “n” iterações (onde “n” representa um número arbitrário de iterações e é definido pela variável limite). São propostos dois modos de funcionamento configuráveis para esta etapa.

O primeiro modo consiste em as abelhas exploradoras utilizarem o Arquivo de Pareto como referência e pegarem uma solução deste conjunto, sem fazer nenhuma alteração.

O segundo modo é a junção do primeiro modo com uma perturbação predefinida no início do código, ou seja, as abelhas exploradoras, além de escolherem aleatoriamente uma solução do conjunto de soluções Pareto, fazem uma perturbação em alguns indivíduos desta solução. Esta perturbação é predefinida pela variável “*PerturbScout*”, que representa a porcentagem de operações que serão perturbadas em cada solução. Em testes preliminares esta perturbação foi definida com valores próximos a 1% e foram obtidos resultados satisfatórios que serão apresentados no próximo capítulo.

4.1.10 Estratégia para encontrar vários resultados para os mesmos valores dos objetivos.

Após o tratamento das soluções pelas três abelhas, é feita a avaliação e verificação dos resultados encontrados até o momento. A proposta é que o código procure por um valor predefinido e não busque apenas o melhor valor que puder encontrar (embora possa ser alterado facilmente para funcionar desta forma). Então, no decorrer das etapas deste algoritmo, são feitas verificações para avaliar se o algoritmo já encontrou os valores buscados para cada objetivo. Estas verificações apenas comparam se os valores desejados foram encontrados por alguma das abelhas.

Se, em qualquer que seja a iteração, ou nas verificações parciais entre cada etapa da iteração, forem encontrados os valores buscados, ou menores, para cada objetivo, o algoritmo se desvia do seu fluxo normal para fazer o armazenamento deste resultado. Este armazenamento salva o estado atual de todas as variáveis do código e apresenta o gráfico de Gantt para a solução encontrada. São apresentados também em qual iteração o código chegou ao valor solicitado, qual abelha encontrou, tempo que demorou e os valores encontrados para os três objetivos. Embora o algoritmo apresente o que encontrou e salve tudo para uma análise mais detalhada dos estados das variáveis, ele não para. Reduz-se um dos valores dos objetivos e continuam as buscas para tentar encontrar um valor menor ou outras programações diferentes para os mesmos valores.

Se o algoritmo não for configurado com um valor alvo, ele minimiza o máximo que conseguir e apresenta os melhores resultados encontrados, ou seja, não é necessário um valor alvo para o algoritmo funcionar.

A variável *MinimizationCriterion* pré-configurada indica a preferência de minimização do objetivo a ser buscado. Esta pode receber os valores *Makespan* ou *WorkLoad*. Estes valores indicam a preferência de minimização ao encontrar os valores desejados. Há um trecho que executa o seguinte pseudo código para que isso aconteça.

---

a. *Se encontrou valores menores ou iguais aos buscados para os 3 objetivos, faça:*
b. *Verifica qual objetivo que se deseja minimizar armazenado na var. MinimizationCriterion*.

c. *Se MinimizationCriterion* = Makespan, Faça:
    a. *Makesan* buscado = *Makesan* encontrado – 1.
d. *Se MinimizationCriterion* = *WorkLoad*, Faça:
    a. *WorkLoad* buscado = *WorkLoad* encontrado – 1.
e. Fim

Esta parte do código garante que o algoritmo não seja interrompido em toda solução que encontrar. Sem esta minimização do valor buscado, o algoritmo pararia várias vezes por iteração após encontrar o valor pela primeira vez e mostraria os mesmos valores sempre. Com isto, são mostrados apenas os melhores valores encontrados sempre que eles são superados. Mesmo não mostrando os valores, todos são armazenados para consultas futuras. Este armazenamento de todos os valores é importante para a contagem da quantidade de soluções que foram encontradas.

4.1.11 Avaliação dos resultados

A proposta de avaliação dos resultados compara os resultados buscados com os resultados dos objetivos encontrados no algoritmo. Este procedimento de verificação dos resultados encontrados e comparação com os resultados desejados ocorre três vezes por iteração. Esta estratégia garante que, se uma abelha empregada encontrou um resultado desejado, a solução já é sinalizada nesta etapa e não necessita passar por todas as outras etapas do código para ser apresentada.

4.1.12 Formas de apresentação dos resultados encontrados

Esta proposta sugere a apresentação dos resultados encontrados no formato do gráfico de Gantt, que é uma forma bastante visual e utilizada para a representação das soluções do FJSP.

Uma outra apresentação proposta é o gráfico que representa a fronteira de Pareto. Este gráfico mostra as soluções não dominadas em um gráfico de pontos, onde são representados os melhores resultados conhecidos na área e os resultados encontrados pelo algoritmo proposto. Esta é uma forma visual de apresentação e

comparação da qualidade das soluções encontradas em comparação com as soluções conhecidas até o momento e encontradas na literatura.

4.1.13 Validação dos resultados encontrados

Os valores encontrados pelo algoritmo foram validados de duas formas.

A primeira consiste em aplicar manualmente os princípios de funcionamento do FJSP em uma programação apresentada através do gráfico de Gantt. Este método é bastante trabalhoso e foi realizado em diversas programações em instâncias de tamanhos menores e com menor quantidade de operações. Com o gráfico de Gantt gerado pelo algoritmo, foi feita a contabilização do *makespan* (que é visualmente identificado em um gráfico de Gantt) e a contabilização da carga total e máquina mais carregada, desconsiderando os tempos que as máquinas ficam paradas ao longo das execuções das operações de todos os *jobs*.

A segunda forma de validação foi verificar os gráficos de Gantt gerados pelo algoritmo aplicado a instâncias pequenas, com os gráficos encontrados na literatura. Este processo é um pouco mais simples que o anterior, mas também foi realizado apenas em instâncias pequenas.

CAPÍTULO 5

5 APRESENTAÇÃO E ANÁLISE DOS EXPERIMENTOS

Este capítulo é utilizado para apresentar os resultados obtidos através dos experimentos realizados com a aplicação das estratégias propostas no capítulo anterior. Serão apresentados os resultados alcançados através do algoritmo desenvolvido que foi aplicado às instâncias obtidas da pesquisa de Kacem, Hammadi e Borne (2002). Embora a base de dados seja antiga, ainda é muito utilizada pelos pesquisadores dessa área.

Cada instância das que serão apresentadas foram executadas pelo menos 30 vezes pelo algoritmo para a apresentação da média dos tempos de execução. O algoritmo proposto foi desenvolvido na plataforma MATLAB 2016a em um computador configurado com um processador I3 – 2100, 3.10GHz com 8GB de memória RAM.

Os *benckmarks* utilizados para os resultados foram apresentados por Kacem, Hammadi e Borne (2002) e Brandimarte (1993). As instâncias que compõe estes *benckmarcks* fornecem cenários para simulações que variam de 5 máquinas por 4 *jobs* com 12 operações em uma das instâncias de Kacem, até 14 máquinas com 20 *jobs* e 240 operações em uma das instâncias de Brandimarte.

Os parâmetros utilizados no código para a realização das experiências foram baseados na bibliografia apresentada no capítulo 3 e no conhecimento adquirido ao longo deste trabalho com as diversas leituras de técnicas e conceitos geralmente utilizados.

As formas utilizadas para realizar a avaliação dos resultados são:

- Comparação dos resultados com os encontrados por outros autores que propuseram soluções utilizando pelo menos uma das técnicas utilizadas.
- Comparação dos resultados com os de outras técnicas.
- Comparação da diversidade de soluções encontradas pelo algoritmo com a diversidade encontrada por algoritmos da literatura.
- Posição dos valores encontrados em relação à fronteira de Pareto impressa em forma de gráfico.

Os tempos computacionais necessários para a execução das instâncias não foram comparados pois a comparação não pode ser considerada justa devido às configurações das máquinas utilizadas por diversos autores. Entretanto, a critério de informação, as tabelas utilizadas para apresentar as configurações para cada instância apresentam o tempo total de 30 execuções.

A seguir são apresentados os resultados das experiências realizadas utilizando as instâncias de Kacem, Hammadi e Borne (2002).

5.1 Instâncias de Kacem, Hammadi e Borne (2002)

As instâncias de Kacem utilizadas são as seguintes: 4x5, 8x8, 10x10 e 15x10. A instância 4x5 é considerada por Kacem pequena e totalmente flexível. Tem esta característica pois todas as máquinas são capazes de processar todas as operações. A instância 8x8 é considerada parcialmente flexível devido ao fato de nem todas as máquinas executarem todas as operações. Ela também é considerada por Kacem uma instância de tamanho médio e tem uma complexidade de solução alta. As instâncias 10x10 e 15x10 são instâncias consideradas grandes e totalmente flexíveis.

Todas as instâncias de Kacem, com exceção da 8x8, são totalmente flexíveis, e no geral os tempos necessários para as máquinas executarem uma operação são diferentes entre uma máquina e outra. A instância 8x8 não é totalmente flexível quando nos referimos às operações parcialmente flexíveis, são poucas as máquinas que não podem ser usadas, ou seja, geralmente uma ou duas máquinas em algumas operações que têm sua utilização proibida. Estas características não a distancia tanto das demais instâncias de Kacem, Hammadi e Borne (2002) e por este motivo configurações parecidas apresentam resultados próximos aos melhores resultados conhecidos.

Na instância 4x5 a quantidade de iterações e quantidade de abelhas são menores que nas demais instâncias de Kacem devido ao tamanho e complexidade da instância, que conta com 12 operações no total. Em todas as execuções os resultados desejados foram encontrados, e as configurações podem ser vistas na Tabela 9.

Tabela 9 - Apresentação dos valores de configuração para as instâncias de Kacem.

Instancia	Total de Iterações	N. Abelhas	Limite	Exploração inicial da empregada	Arquivo Pareto	M1	M2	M3	WCm	WWt	WWm	Tempo médio (s)
4x5	400	200	10	8	40	10	50	40	2,2	1,7	1,5	1548
8x8	600	200	15	20	120	20	70	10	1,8	1,5	1,2	1987,2
10x10	600	200	15	20	120	50	40	10	5	2	1	1925,7
15x10	1000	300	20	25	60	10	50	40	8	1	1	5999,4

Fonte: Este trabalho.

Os valores das variáveis “iterações” e “número de abelhas” foram determinados bem acima do necessário para encontrar os resultados desejados, pois geralmente a solução é encontrada nas primeiras iterações. Entretanto, os valores foram propositalmente elevados para que o algoritmo tenha tempo e iterações suficientes para encontrar a maior diversidade possível de soluções.

Com as configurações da Tabela 9, o algoritmo geralmente encontra o resultado desejado (sendo ele um dos melhores resultados conhecidos ou não) nas primeiras 10 iterações, as outras iterações são apenas para procurar uma diversidade grande de soluções.

Os resultados encontrados para as instâncias de Kacem são apresentados na Tabela 10, que coloca lado a lado aos valores de outros trabalhos encontrados na literatura.

Para processamento da instância 8x8 (parcialmente flexível) a quantidade de iterações e abelhas exigidas para encontrar os melhores resultados conhecidos é maior devido à sua complexidade. Com um total de 27 operações, esta instância dividida em 8 *jobs* e tendo como recursos disponíveis 8 máquinas, tem características semelhantes à 4x5. A semelhança está em ter várias máquinas que executam cada operação e geralmente com tempos diferentes, mesmo nas operações que não contam com alguma máquina, ainda há várias outras que a executam.

O algoritmo foi executado várias vezes a fim de determinar qual a configuração mais eficiente. Os valores buscados foram encontrados com várias

configurações e a Tabela 9 mostra uma das configurações que apresentaram a maior diversidade de soluções.

A definição dos pesos w_{cm}, w_{wt}, e w_{wm}, foi realizada observando que dando prioridade para o w_{cm} na maioria das vezes o w_{wt} também diminui, e o contrário também é válido. Pode ser observado também que quando aumenta-se muito o w_{wm} o funcionamento não é satisfatório, pois o algoritmo começa a dar muita prioridade para a minimização do wm.

A instância 10x10 e a instância 15x10 são parecidas em relação a todas as características. Ambas são consideradas grandes, com grande quantidade de operações, tendo 30 operações a 10x10 e 56 operações a 15x10. As duas instâncias contam com máquinas que executam as operações em tempos muito diferentes, variando de 1 a 23 ut na instância 10x10 e de 1 a 85 ut na instância 15x10. Em relação a esta última característica, quando os tempos são diferentes é um pouco menos complicado escolher as máquinas que executarão cada operação, e na maioria das vezes as máquinas muito demoradas não participam das programações.

As configurações das instâncias 10x10 e 15x10, assim como os tempos médios de execução para estas configurações estão apresentados na Tabela 9.

Após breve descrição das características observadas nas instâncias, serão apresentados a seguir os resultados encontrados.

Como o problema é multiobjetivo e existem diversas formas de considerar as melhores soluções, serão apresentadas as soluções ordenadas do menor *makespan* para o maior e, em seguida, considerando a menor carga total. O *makespan* será representado pela sigla cm e a carga total pela wt. O objetivo wm representa a máquina mais carregada e será o último critério de ordenação das soluções na Tabela 10. A última coluna de cada instância (após wm de nome Sol.) apresenta a quantidade de soluções diferentes encontradas para o conjunto de objetivos na mesma linha. A primeira coluna da Tabela 10 apresenta os algoritmos que encontraram os resultados apresentados e o último algoritmo é o proposto neste trabalho.

Os trabalhos encontrados na literatura que não apresentaram mais de uma solução diferente para cada conjunto de objetivo tiveram a quantidade de soluções encontradas considerada como 1, e cabe salientar que os trabalhos que contém 1 na quantidade de soluções não tiveram necessariamente como foco do trabalho

apresentar alta diversidade de soluções. O intuito desta comparação foi pegar os melhores resultados encontrados e comprar o algoritmo proposto, mostrando que o algoritmo é capaz de encontrar os melhores resultados com alta diversidade de configurações.

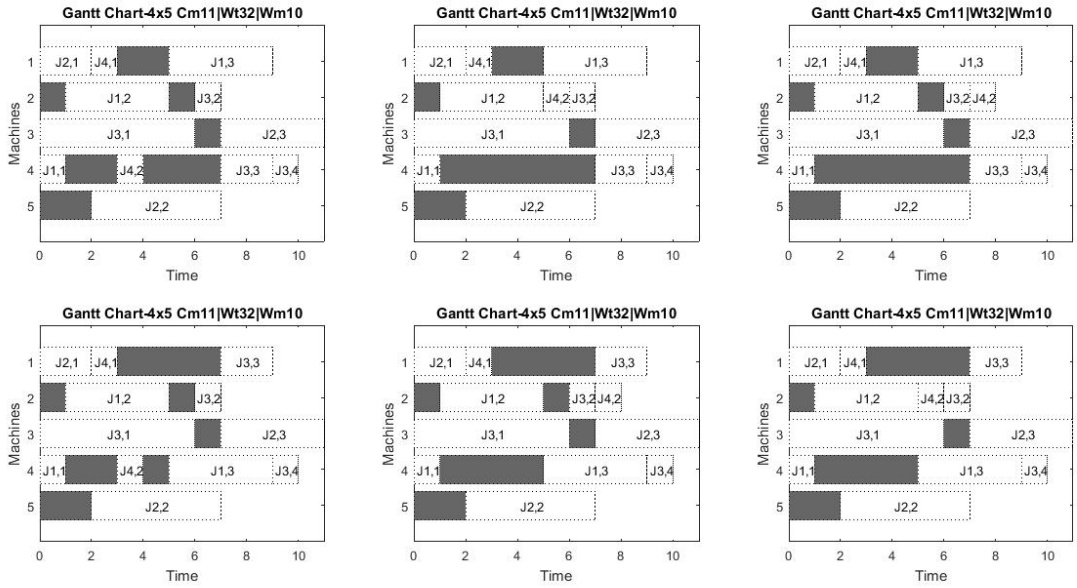
O algoritmo encontrou todos os melhores resultados conhecidos para a instância 4x5, como pode ser visto nas últimas linhas da Tabela 10, e a diversidade de soluções para os valores encontrados estão na coluna “Sol”. A maior diversidade de soluções foi encontrada para a solução 11,32,10 com 6 resultados diferentes que têm os gráficos de Gantt apresentados pela Figura 18.

Tabela 10 - Comparação dos resultados de outros autores com os encontrados.

Comparação entre os resultados obtidos neste trabalho e outros resultados da literatura para o *benchmark* de KACEM.																
nxm	4 X 5				8 X 8				10 X 10				15 X 10			
	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.
DABC[5]	11	32	10	1	14	77	12	1	7	42	6	1	11	91	11	1
Art8	11	34	9	1	15	75	12	1	**7**	**43**	**5**	**1**	11	93	10	1
	12	32	8	1	16	73	13	1	8	41	7	1				
	13	33	7	1	16	77	11	1	8	42	5	1				
BEG-NSGA-II[6]	11	32	10	1	14	77	12	1	7	42	6	1	11	91	11	1
art12	11	34	8	1	15	75	12	1	**7**	**43**	**5**	**1**	11	95	10	1
	12	32	8	1	16	73	13	1	8	41	7	1	12	93	10	1
	13	33	7	1	16	77	11	1	8	42	5	1				
MOPSO-PDS[7]	-	-	-	-	-	-	-	-	**7**	**43**	**5**	**25**	-	-	-	-
Ref 8 do art11	-	-	-	-	-	-	-	-	8	41	7	35	-	-	-	-
	-	-	-	-	-	-	-	-	8	42	5	16	-	-	-	-
ABC-DIS[8]	-	-	-	-	-	-	-	-	7	42	6	34	-	-	-	-
Art 11	-	-	-	-	-	-	-	-	**7**	**43**	**5**	**58**	-	-	-	-
	-	-	-	-	-	-	-	-	8	41	7	66	-	-	-	-
	-	-	-	-	-	-	-	-	8	42	5	42	-	-	-	-
EPABC[9]	11	32	10	1	14	77	12	1	7	42	6	1	11	91	11	1
Art1	11	32	8	1	15	75	12	1	**7**	**43**	**5**	**1**	11	93	10	1
	13	33	7	1	16	73	13	1	8	41	7	1				
	11	34	9	1	16	77	11	1	8	42	5	1				
P-DABC[10]	11	32	10	1	14	77	12	1	**7**	**43**	**5**	**1**	11	93	11	1
Art6	12	32	8	1	15	75	12	1	8	41	7	1	12	91	11	1
	13	33	7	1	16	73	13	1	8	42	5	1				
PSO+RRHC[11]					12	73	13	1	**7**	**43**	**5**	**1**	11	91	11	1
					14	77	12	1	7	42	6	1	14	93	10	1
					15	75	12	1	8	42	5	1				
					16	77	11	1	8	41	7	1				
Pareto-ABC	**11**	**32**	**10**	**6**	**14**	**77**	**12**	**4**	7	42	6	**51**	12	91	12	1
	11	**34**	**9**	**1**	**15**	**75**	**12**	**24**	**8**	**41**	**7**	**3360**	12	93	11	28
	12	**32**	**8**	**4**	**16**	**73**	**13**	**53**	**8**	**42**	**5**	**144**	13	92	11	140
	13	**33**	**7**	**3**	**16**	**77**	**11**	**1**								

Fonte: [5] (LI, at al. ,2014) [6] (DENG, at. al. 2017) [7] (LU at. al. 2013) [8] (CHAO at. al. 2017) [9] (WANG at. al. 2011) [10] (LI at. al. 2011) [11] (KATO at. al. 2018).

Figura 18 - Todas os resultados encontrados para o resultado 11,32,10 da instância 4x5.

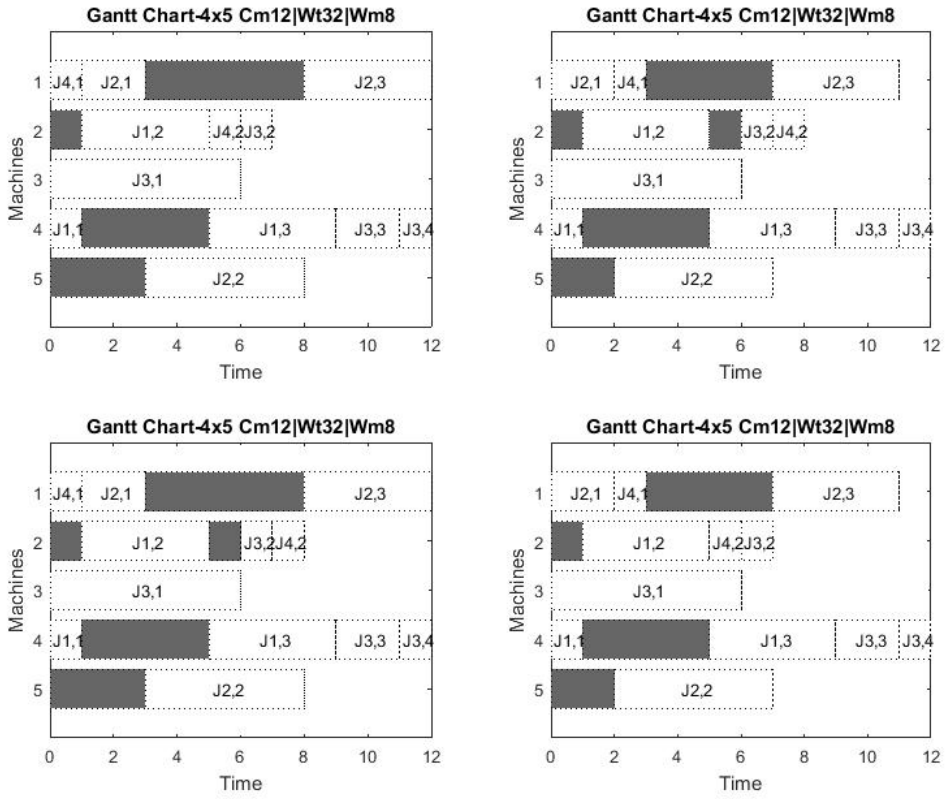


Fonte: Este trabalho.

Pode ser observado na Figura 18 que os gráficos são diferentes, mas apresentam os mesmos valores para os três objetivos (cm, wt e wm).

Os gráficos gerados com os quatro resultados para a solução 12,32,8 estão apresentados na Figura 19, e como pode ser visto também são diferentes entre si e apresentam os mesmos valores para os três objetivos.

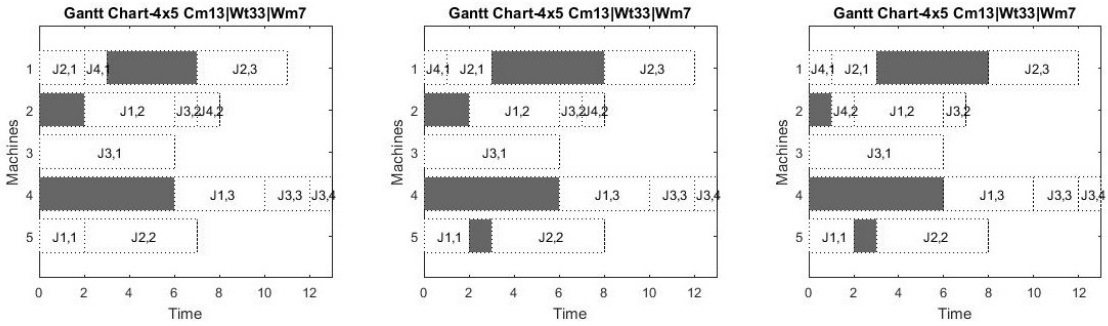
Figura 19 - Todos os resultados encontrados para o valor 12,32,8 da instância 4x5.



Fonte: Este trabalho.

Os gráficos diferentes gerados para o resultado 13,33,7, cujo valor apresenta o menor wm dos melhores resultados conhecidos, são apresentados na Figura 20, onde pode ser visto que os resultados são iguais, entretanto, os gráficos gerados são diferentes.

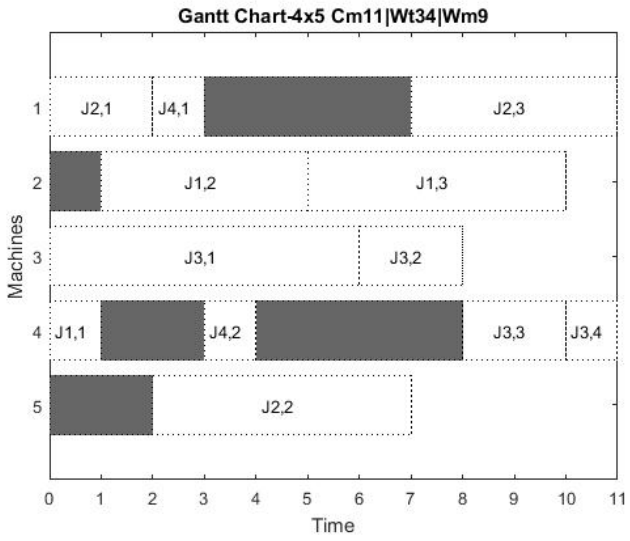
Figura 20 - Todos os resultados encontrados para o valor 13,33,7 da instância 4x5.



Fonte: Este trabalho.

E, por último, a instância 11,34,9, que teve apenas um resultado, é apresentada na Figura 21.

Figura 21 - Resultado encontrado para o valor 11,34,9 da instância 4x5.

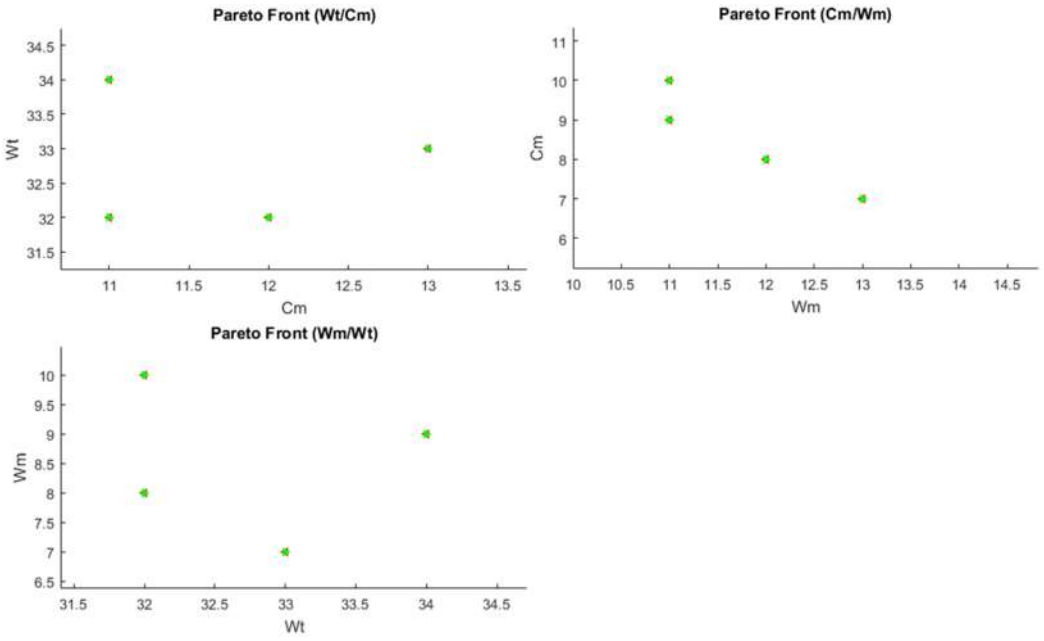


Fonte: Este trabalho.

Para a instância 4x5, se considerarmos todos os resultados encontrados para os quatro melhores resultados conhecidos da literatura, foi encontrado um total de 14 resultados diferentes.

Para uma melhor visualização dos resultados obtidos em contraste com os melhores resultados conhecidos da literatura, a Figura 22 apresenta a fronteira de Pareto para a instância 4x5 para todos os objetivos. Em verde são representados os melhores resultados conhecidos da literatura, e em vermelho são representados os resultados não dominados que foram encontrados. Cada um dos três gráficos apresentados na Figura 22 está identificado com o nome no topo.

Figura 22 - Fronteira de Pareto para a instância 4x5.



Fonte: Este trabalho.

Como pode ser visto os pontos verdes e vermelhos estão sobrepostos, mostrando que os melhores valores conhecidos foram encontrados.

Para a instância 8x8, o resultado com maior diversidade foi o 16,73,13 que tem o menor wt conhecido para os resultados 16,13 (cm,wm respectivamente), contando com 53 resultados diferentes, e o que apresentou a menor diversidade foi o valor 16,77,11 que tem o menor wm conhecido para os valores 16,77 (cm,wt respectivamente), e este resultado encontrou apenas 1 solução.

A Figura 23 apresenta a quantidade de soluções diferentes encontradas para a instância 8x8. Pode ser visto no eixo “x” o resultado multiobjetivo e no eixo “y” a quantidade de soluções diferentes.

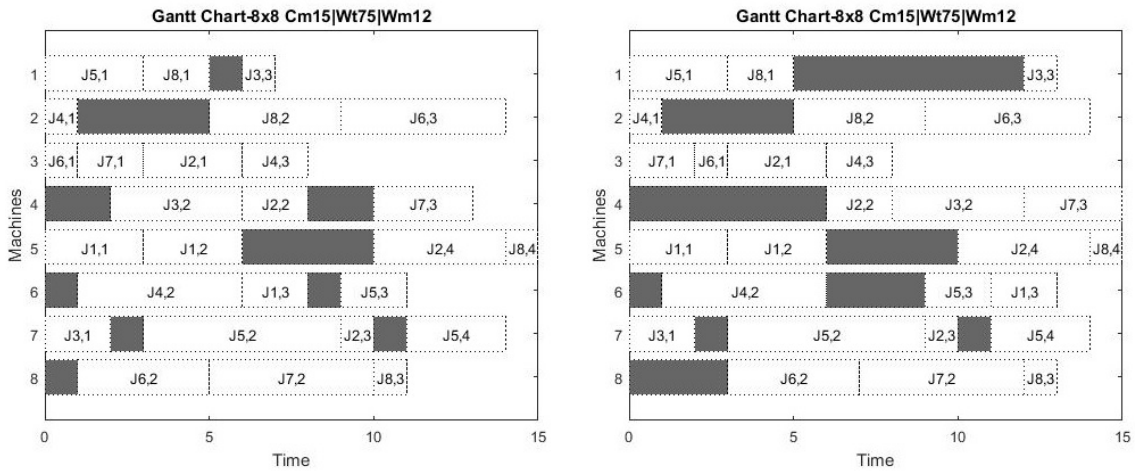
Figura 23 - Quantidade de soluções encontradas por resultado.

Fonte: Este trabalho.

Como a diversidade dos resultados desta instância é alta o suficiente para tornar inviável a apresentação de todos os gráficos, então, a diversidade não será apresentada completamente.

A Figura 24 apresenta 2 dos 24 resultados encontrados para o resultado 15,75,12.

Figura 24 - Dois resultados encontrados para a instância 8x8.



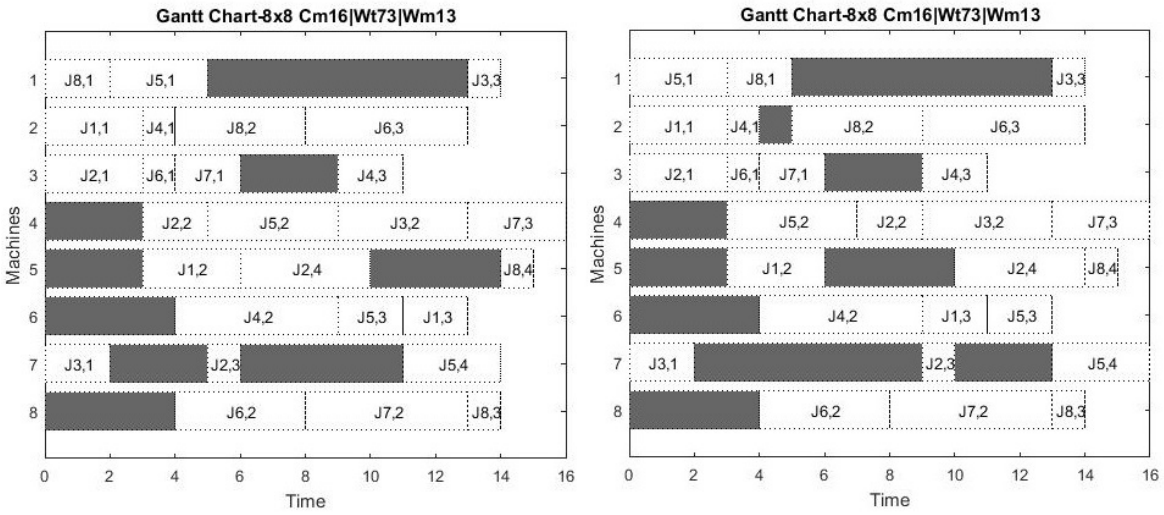
Fonte: Este trabalho.

Nota-se que os gráficos apresentados na Figura 24 são diferentes, mas com os resultados iguais. E neste caso o resultado da esquerda apresenta um agendamento com menor tempo de espera, ou seja, o tempo em que as máquinas ficam paradas entre uma operação e outra, e apenas uma das máquinas termina a execução no tempo máximo.

Já no gráfico da direita as máquinas 4 e 5 terminam as operações no tempo máximo. As diversidades dos agendamentos são várias, e não devem ser classificadas como melhor ou pior, apenas uma é mais adequada que a outra dependendo da necessidade.

A Figura 25 apresenta 2 dos 53 resultados encontrados para o resultado 16,73,13.

Figura 25 - Resultados encontrados para a instância 8x8.

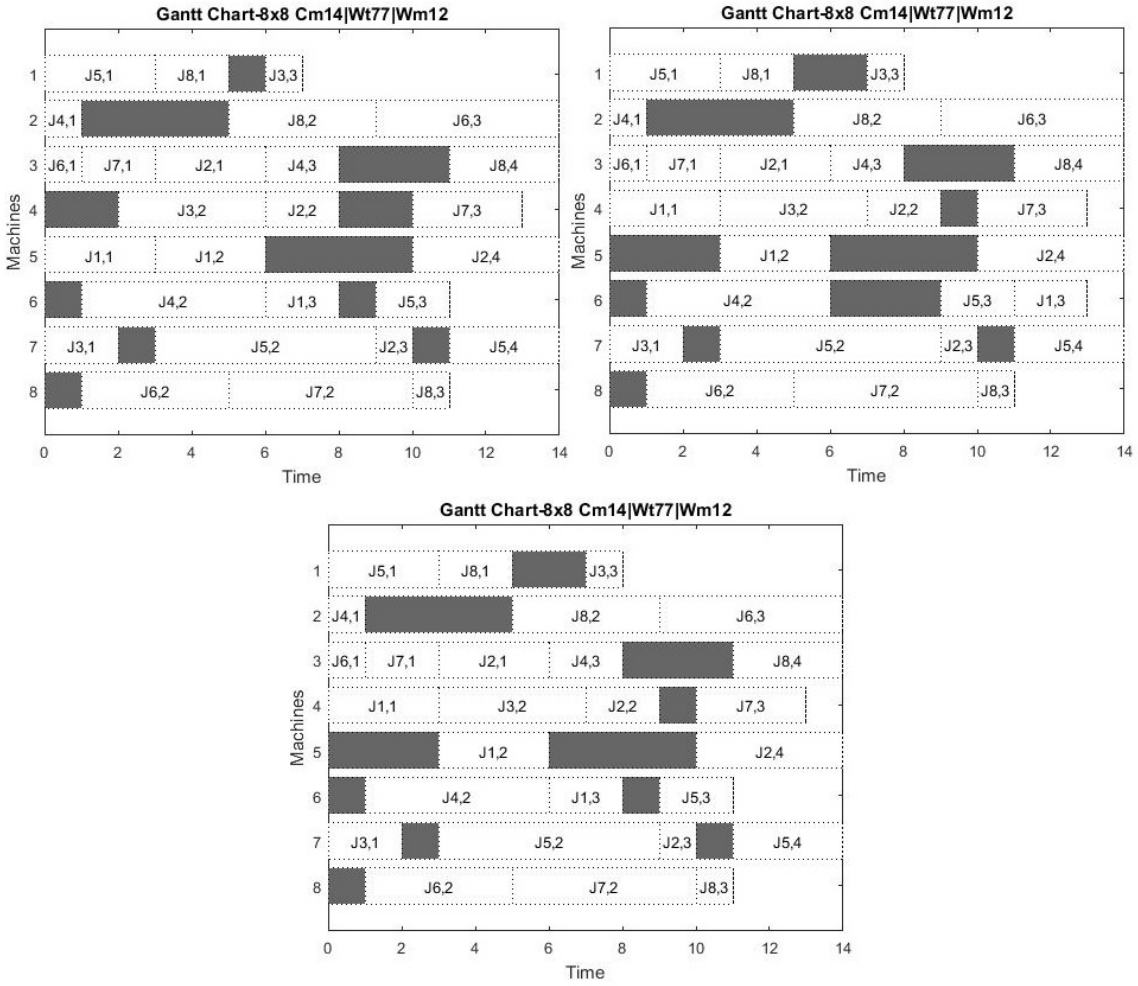


Fonte: Este Trabalho.

Nota-se que os gráficos apresentados na Figura 25 são diferentes, mas com os resultados iguais. A diversidade deste resultado foi alta, com 53 resultados distintos, e pode ser observado que não há nitidamente muita discrepância entre eles. A quantidade de tempo de espera varia muito pouco, e a quantidade de máquinas terminando no tempo máximo fica sempre em uma ou duas.

A Figura 26 apresenta os 3 dos 4 resultados encontrados para o resultado 14,77,12.

Figura 26 - Resultados encontrados para a instância 8x8.



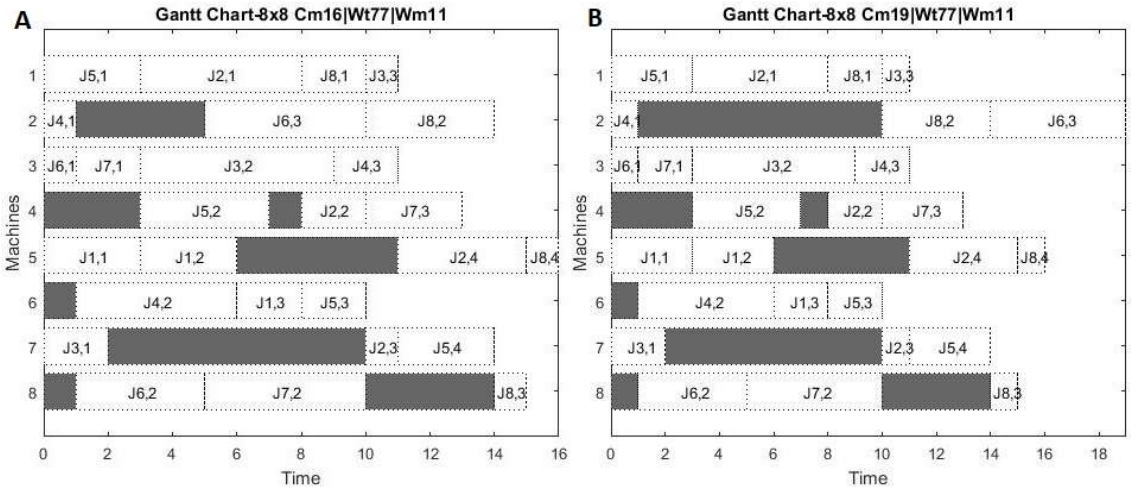
Fonte: Este trabalho.

Nota-se que os três gráficos encontrados são parecidos, com poucas modificações. Os gráficos 2 e 3 são diferentes apenas no agendamento da máquina 6, e os gráficos 1 e 3 são diferentes apenas nas máquinas 1 e 4.

A Figura 27 apresenta o resultado encontrado com o valor 16,77,11, encontrado apenas uma vez. Observando todas as execuções feitas, este resultado foi o mais difícil de encontrar. Entretanto, uma programação semelhante foi encontrada várias vezes, (19,77,11 apresentada também na Figura 27) e, comparando as duas programações, pode ser visto que o algoritmo tem baixa capacidade de fazer vários agendamentos diferentes para a programação encontrada. Esta conclusão foi obtida considerando que, se o gráfico da direita foi encontrado

várias vezes e o da esquerda não, e a única diferença entre eles é a operação “J6,3” (executada pela máquina 2), uma melhoria na etapa de diversidade de agendamentos das operações faria com que a qualidade das soluções aumentasse.

Figura 27 - Resultados parecidos encontrados para a instância 8x8.

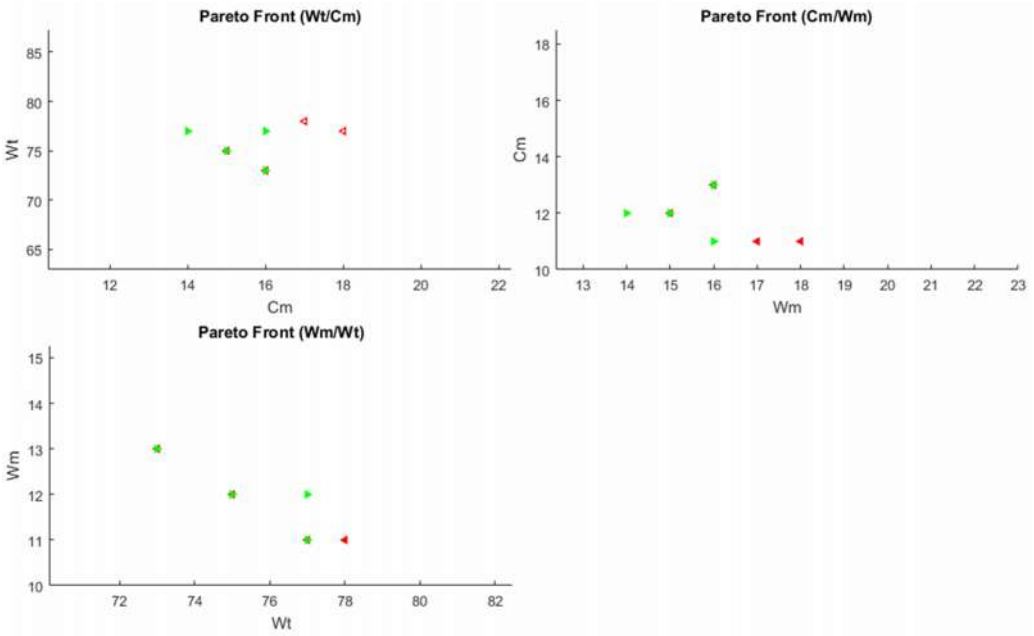


Fonte: Este trabalho.

O total de programações diferentes para esta instância, considerando apenas os melhores resultados conhecidos da literatura, contabilizam 81 gráficos diferentes.

Para uma melhor visualização dos resultados obtidos em contraste com os melhores resultados conhecidos da literatura, a Figura 28 apresenta a fronteira de Pareto para a instância 8x8 para todos os objetivos. Em verde são representados os melhores resultados conhecidos da literatura, e em vermelho são representados os resultados não dominados que foram encontrados. Cada um dos três gráficos apresentados na Figura 28 está identificado com o nome no topo.

Figura 28 - Fronteira de Pareto para a instância 8x8.



Fonte: Este trabalho.

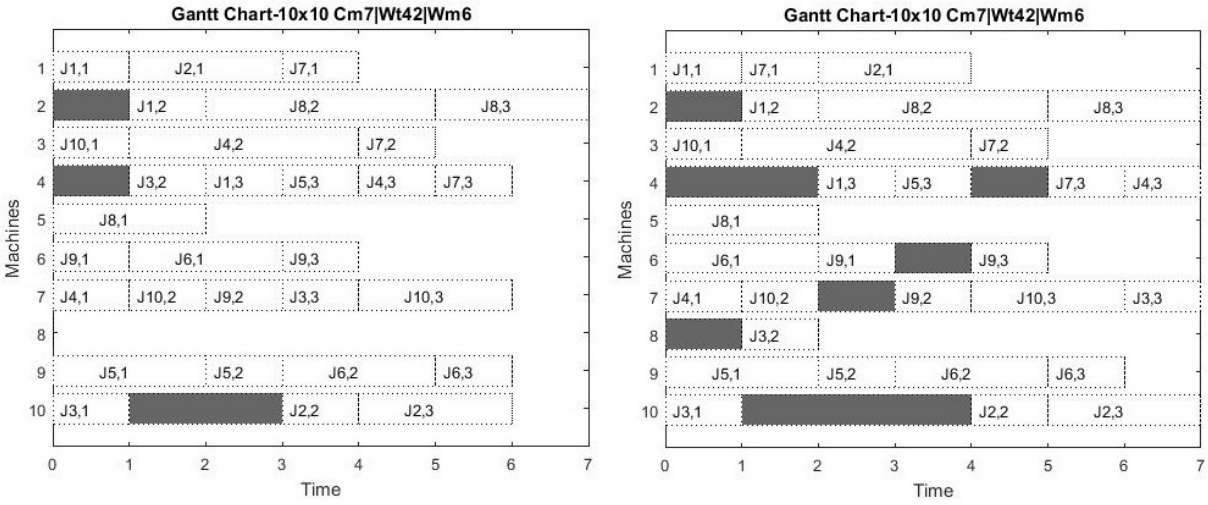
Para a instância 10x10, o resultado com maior diversidade foi o 8,41,7 que tem o menor wt conhecido para os resultados 8,7 (cm,,wm respectivamente), contando com 3.360 resultados diferentes, e o que apresentou a menor diversidade foi o valor 8,43,5 que tem o menor wm conhecido para os valores 8,43 (cm,wt respectivamente), totalizando 7 resultados diferentes.

Dos melhores resultados conhecidos, o único resultado que o algoritmo não encontrou foi o 7,43,5, destacado em negrito na Tabela 10, entretanto, chegou bem próximo com o valor 7,43,6 com uma diversidade de 431 resultados diferentes. A instância exige alta complexidade do algoritmo para alcançar este resultado, já que neste (7,43,5) estão os dois menores valores individuais para a instância, que são o 7 para o Cm e o 5 para o wm.

Para esta instância (10x10) serão apresentados apenas dois gráficos para cada melhor resultado conhecido encontrado nos casos em que se houver algo notado entre os dois, e nos demais casos, apenas um gráfico. Isso é necessário devido à

quantidade de operações e à quantidade de gráficos encontrados. A Figura 29 mostra dois dos 70 resultados encontrados com o valor 7,42,6.

Figura 29 - Resultados encontrados para a instância 10x10.

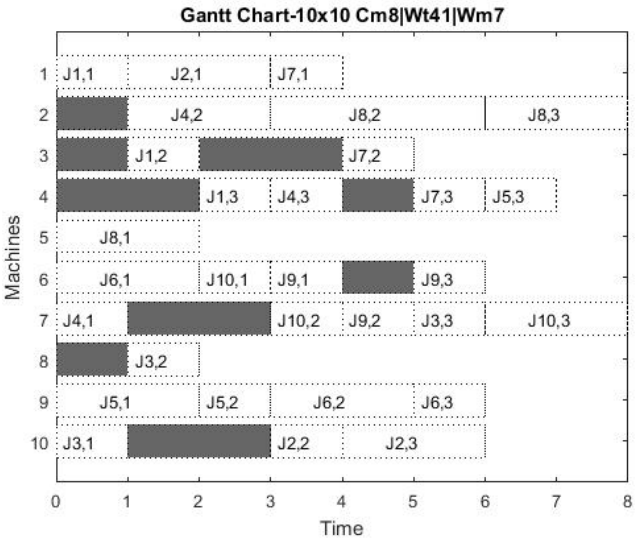


Fonte: Este trabalho.

Como pode ser visto na Figura 29, uma das programações não utiliza a máquina 8 para realizar nenhuma operação e a máquina 5 fica a maior parte do tempo parada. Uma característica positiva para esta configuração é a possibilidade de uma manutenção nas máquinas 5 e 8 com o processo rodando, sem prejuízo de nenhum dos objetivos avaliados. Outra característica interessante é o tempo de espera das duas programações, a primeira tem apenas 4 ut de espera e a segunda tem 10 ut. A última característica observada nestas duas programações que cabe ressaltar é que no primeiro *Gantt* apenas uma máquina termina no tempo 7 e, no segundo, 4 das 10 máquinas terminam juntas no tempo 7.

A Figura 30 mostra um dos 3.360 resultados encontrados com o valor 8,41,7.

Figura 30 - Resultado encontrado para a instância 10x10.



Fonte: Este trabalho.

Já nesta configuração apresentada na Figura 30, a máquina 8 é utilizada, mas está disponível do tempo 2 em diante, assim como a máquina 5. Como são muitas programações, não foi considerada viável a análise e comparação das programações. Entretanto, pôde-se observar que os tempos de espera das máquinas não são discrepantes entre as programações, não foram encontradas programações que deixam de utilizar uma das máquinas e as máquinas que terminam as operações no tempo máximo variam de uma máquina até 4.

Para ficarem mais evidentes as variações encontradas para cada valor apresentado para a instância 10x10, a Figura 31 mostra as diversidades encontradas.

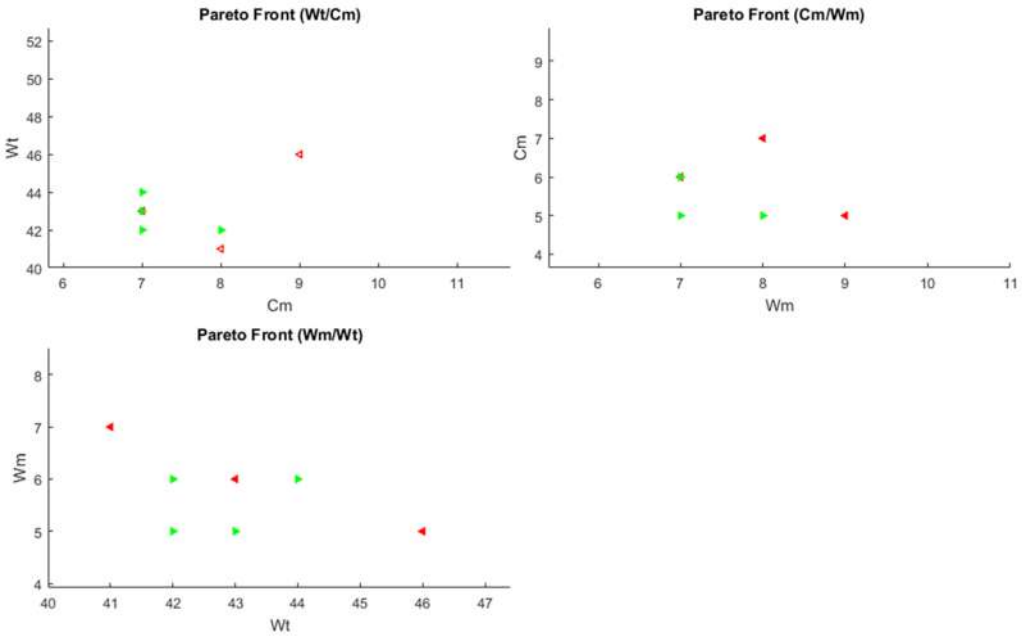
Figura 31 - Diversidade dos valores encontrados para a instância 10x10.

Fonte: Este trabalho.

Como pode ser visualizado na Figura 31, o valor que mais teve variações na programação foi o 8,41,7 e com menor diversidade foi o 7,43,7.

Para uma melhor visualização dos resultados obtidos em contraste com os melhores resultados conhecidos encontrados na literatura, a Figura 32 apresenta a fronteira de Pareto para a instância 10x10 para todos os objetivos. Em verde são representados os melhores resultados conhecidos encontrados na literatura, e em vermelho são representados os resultados não dominados que foram encontrados. Cada um dos três gráficos apresentados na Figura 32 está identificado com o nome no topo.

Figura 32 - Fronteira de Pareto para a instância 10x10.

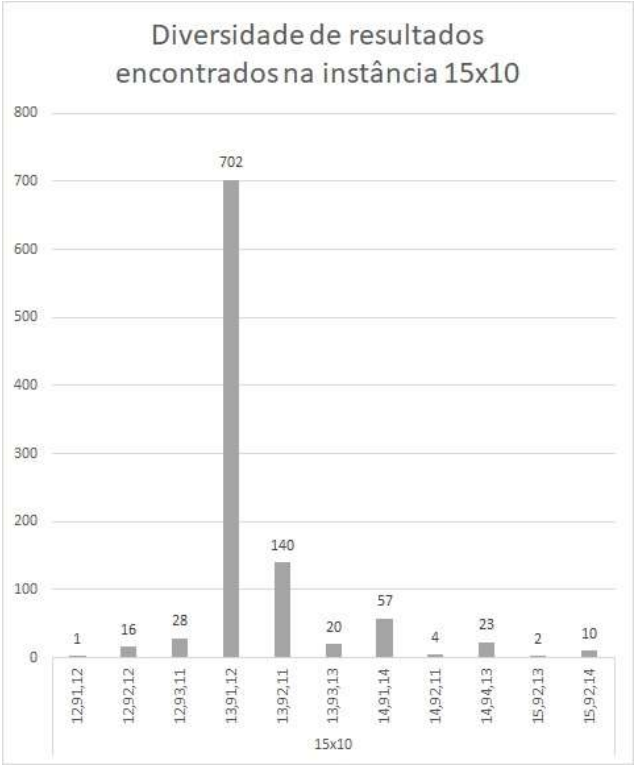


Fonte: Este trabalho.

Como pode ser visto na Figura 32, os gráficos representam que apenas um dos melhores resultados conhecidos não foi encontrado.

Para a instância 15x10 que é a maior instância de Kacem, o resultado com maior diversidade foi o 13,91,12, que tem o menor wt conhecido para os resultados 11,11 (cm,,wm respectivamente). Neste caso especificamente o algoritmo não encontrou o menor valor multiobjetivo desta instância (11,91,11), mas encontrou diversos valores com alta diversidade, como pode ser visto na Figura 33. Para esta mesma instância o resultado que apresentou a menor diversidade, com apenas uma programação/agendamento, foi o valor 12,91,12, que não deixa de ser um resultado bom, apesar de não ser um dos melhores resultados conhecidos, e ter apenas uma solução.

Figura 33 - Diversidade encontrada para a instância 15x10



Fonte: Este trabalho.

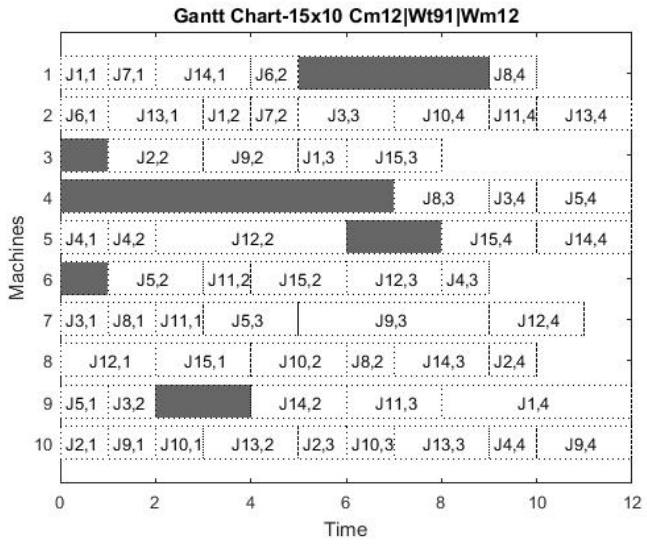
Como pode ser visto na Figura 33, a maior diversidade encontrada para a instância 15x10 foi de 702 para o resultado 13,91,12. E a menor para o valor 12,91,12 com apenas um resultado.

Esta instância, por ser de tamanho grande com muitas operações, apresenta uma dificuldade alta em alcançar os melhores resultados conhecidos. Em todas as execuções realizadas o algoritmo não alcançou os valores desejados. Entretanto, considerando apenas o objetivo wt, individualmente, o algoritmo alcançou o mínimo, sendo ele 91. O cm mínimo conhecido (11) e o wm (10) não foram encontrados.

Como não foram encontrados os melhores resultados conhecidos multiobjetivo, serão apresentados apenas os melhores resultados encontrados através de gráfico de *Gantt*. A Figura 34 mostra um dos melhores resultados encontrados para esta instância, onde pode ser visto que nesta programação duas das máquinas (M2 e M10) permanecem ocupadas por todo o tempo. Dito isso, uma

estratégia que poderia contribuir para alcançar um dos melhores resultados para esta instância (11,91,11), seria atuar no caminho crítico, entretanto, o algoritmo não foi desenvolvido abordando este princípio.

Figura 34 - Resultado encontrado para a instância 15x10

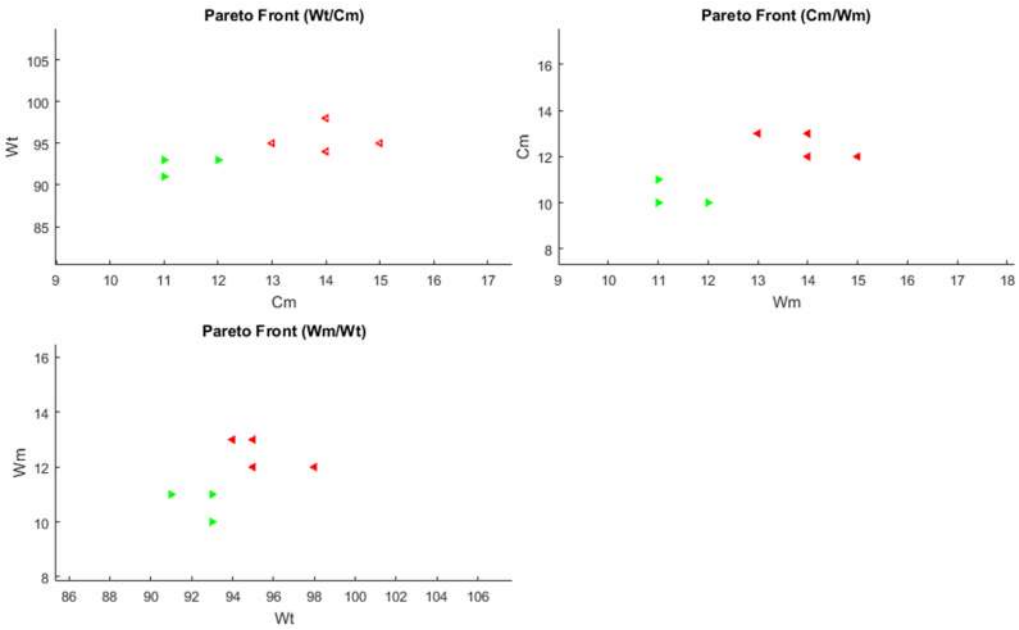


Fonte: Este trabalho.

O que pode ser considerado com os resultados apresentados é que, nas três primeiras instâncias, o algoritmo encontrou todos os melhores resultados, com exceção de um resultado da instância 10x10. Para a instância 15x10 não foram encontrados os melhores resultados conhecidos, mas a diversidade de soluções encontradas foi alta, e para que o algoritmo encontre as soluções da instância 15x10 são necessárias alterações nos mecanismos de busca ou uma sintonização das configurações de forma que fiquem mais alinhadas às características da instância.

Para uma melhor visualização dos resultados obtidos em contraste com os melhores resultados conhecidos da literatura, a Figura 35 apresenta a fronteira de Pareto para a instância 15x10 para todos os objetivos. Em verde são representados os melhores resultados conhecidos encontrados na literatura, e em vermelho são representados os resultados não dominados que foram encontrados. Cada um dos três gráficos apresentados na Figura 35 está identificado com o nome no topo.

Figura 35 - Fronteira de Pareto para a instância 15x10.



Fonte: Este trabalho.

Como pode ser visto na Figura 35, os valores encontrados chegaram perto dos melhores resultados conhecidos, mas não os alcançaram.

Para finalizar a apresentação dos resultados para este *benchmark* cabe salientar que várias programações que não foram apresentadas neste capítulo serão colocadas no apêndice I.

A seguir serão apresentados os resultados das experiências realizadas utilizando as instâncias de Brandimarte (1993).

5.2 Instâncias de Brandimarte (1993)

As instâncias de Brandimarte (1993) utilizada neste trabalho foram: mk01, mk02, mk03, mk04, mk05, mk07, mk08 e mk09. Estas instâncias diferem-se bastante das de Kacem, pois são todas parcialmente flexíveis, e no geral as máquinas executam as operações com os mesmos tempos (na mesma operação). Ao analisar os resultados encontrados através das instâncias com poucas máquinas disponíveis para executar cada operação, observou-se menor diversidade de soluções

encontradas. Esta constatação foi feita devido à quantidade de combinações possíveis que o algoritmo consegue fazer trocando as operações de máquinas. Como o algoritmo foi desenvolvido utilizando como base instâncias que disponibilizam todas ou quase todas as máquinas para executar cada operação, com tempos geralmente bem diferentes, as características opostas (instâncias de Brandimarte) de certa forma suprimem parte da eficiência do algoritmo.

Os mecanismos de busca desenvolvidos atuam diretamente na seleção das máquinas e não no agendamento das operações. A etapa de agendamento, que determina a ordem que as operações serão executadas, mostrou-se como o gargalo do algoritmo em relação às instâncias de Brandimarte (1993).

Estas instâncias são consideradas grandes, variando de 10x6 a 20x10 e operações variando de 55 a 240. Em comparação com as instâncias de Kacem, que variam de 12 a 56, a menor instância de Brandimarte é quase do tamanho da maior de Kacem.

Embora o algoritmo não tenha sido adaptado inteiramente para o *banchmark* de Brandimarte, os resultados obtidos foram satisfatórios em algumas instâncias.

Para facilitar a visualização dos resultados serão apresentadas tabelas e figuras com comparações.

A Tabela 11 mostra as configurações que foram utilizadas para alcançar os resultados apresentados a seguir.

Tabela 11 - Configurações das instâncias de Brandimarte.

Instância	Total de Iterações	N. Abelhas	Limite	Exploração inicial da empregada	Arquivo Pareto	M1	M2	M3	WCm	WWt	WWm	Tempo médio (s)
MK01	600	400	15	15	80	7	80	13	1,8	1	1,2	1857,6
MK02	500	600	15	15	120	7	80	13	1,8	1	1,2	4265,7
MK03	900	400	15	15	80	7	80	13	1,8	1	1,2	5841
MK04	700	400	15	15	80	7	80	13	2	2,5	1	5099,4
MK05	700	400	15	15	80	7	80	13	1,8	1	1,2	5274
MK06	1000	200	15	15	80	2	30	68	3	1	1	7751,7
MK07	700	400	15	15	80	7	80	13	1,8	1	1,2	4380,6
MK08	1000	200	15	15	60	10	20	70	3	1	1	7669,2
MK09	900	200	15	15	60	25	60	15	15	3	1	8204,1

Fonte: Este trabalho.

Como mencionado anteriormente, o algoritmo foi desenvolvido de forma genérica com o intuito de ser capaz de encontrar resultados em qualquer instância. Entretanto, apesar de funcionar como previsto, não é tão eficiente nas instâncias de Brandimarte como é nas instâncias de Kacem. Então, não sendo tão eficiente para estas instâncias, nota-se na Tabela 11 que a quantidade de execuções e a quantidade de abelhas foram extrapoladas para que o algoritmo encontrasse resultados bons.

Serão apresentados a seguir os melhores resultados encontrados e comparados com os melhores resultados conhecidos da literatura.

Considerando todas as instâncias de uma forma ampla, os melhores resultados foram obtidos para as instâncias mk02 e mk05. A instância mk05 foi a que teve maior diversidade nas soluções, entretanto, os melhores valores conhecidos não foram alcançados para nenhuma das instâncias. Serão apresentados os gráficos de fronteira de Pareto para as instâncias para mostrar visualmente a qualidade das melhores soluções encontradas.

Tabela 12 - Comparação dos resultados encontrados com os da literatura para as instâncias mk01,mk02,mk03,e mk04 de Brandimarte.

Comparação entre os resultados obtidos neste trabalho e outros resultados da literatura para o *benchmark* de BRANDIMARTE.																
nxm	MK01 (10x6)				MK02 (10x6)				MK03 (15x8)				MK04 (15x8)			
	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.
DABC[12]	40	167	36	1	28	155	28	1	204	850	204	1	60	374	60	1
	40	162	38	1	26	151	26	1	210	848	210	1	61	368	60	1
	40	164	37	1					813	844	213	1	61	365	61	1
BEG-NSGAII[13]	40	169	36	1	26	151	26	1	204	855	199	1	60	390	59	1
	42	158	39	1	27	146	27	1	204	871	144	1	63	371	61	1
	43	155	40	1	29	145	27	1	204	882	135	1	65	362	63	1
hDPSO[14]	40	167	36	1	26	150	26	1	204	850	204	1	60	372	60	1
	41	163	37	1	27	145	27	1	210	848	210	1	61	366	60	1
	42	165	36	1	28	144	28	1	213	844	213	1	62	363	60	1
Pareto-ABC	**42**	167	42	255	27	151	27	35	204	865	204	1	75	356	75	1
	43	163	43	1	28	146	27	1	204	874	204	20	75	357	75	6
	43	165	42	1	28	146	28	4	204	882	204	21	76	349	75	1
	43	166	42	98	28	147	28	160	209	859	204	1	76	350	75	6
	44	162	43	6	28	148	27	71	213	897	213	18	76	353	75	26
	44	164	42	5	29	144	29	3	231	858	213	1	78	349	75	13
	45	159	44	4	29	146	29	23	236	891	204	54	79	346	75	8
	45	160	43	23	29	146	29	14					81	345	73	2
	45	160	43	16	29	147	29	75					81	347	67	4
	45	162	42	8	29	148	29	221					89	348	67	10
	45	164	43	34	29	150	28	456					90	353	62	26
	46	158	44	12	29	152	26	92					96	353	63	11
	46	160	43	3	30	143	31	1					130	326	130	1
	46	168	**36**	20	30	146	27	742					137	326	137	3
	47	156	45	4	30	147	30	23					145	326	145	6
	47	159	43	4	30	148	27	70					150	324	146	1
	47	167	**36**	3649	30	152	26	2								
	49	164	37	28	31	145	27	54								
	50	153	46	24	31	147	27	11								
	50	165	**36**	1	31	147	27	124								
	53	167	**36**	1	31	151	27	447								
	54	162	39	8	32	143	30	1								
	56	153	42	1	33	142	33	25								
	57	153	42	9	33	145	27	1								
	57	159	39	1	34	142	31	2								
	57	**161**	38	1	35	141	33	1								
	57	166	**36**	1	36	143	29	2								
	58	153	42	6	54	142	25	1								
	59	153	42	39	55	142	26	10								

Fonte: [12] Li, et al. (2012) [13] Deng et. Al. (2017) [14] SHAO et. Al. (2013)

A comparação feita na Tabela 12 e Tabela 13 tem o intuito de mostrar quais os resultados que vêm sendo encontrados por outros pesquisadores nos últimos anos, em comparação com os resultados encontrados pelo algoritmo proposto. A classificação dos valores foi feita considerando o cm depois o wt e por último o wm.

Tabela 13 - Comparação dos resultados encontrados com os da literatura para as instâncias mk05,mk07,mk08,e mk09 de Brandimarte

Comparação entre os resultados obtidos neste trabalho e outros resultados da literatura para o *benchmark* de BRANDIMARTE.

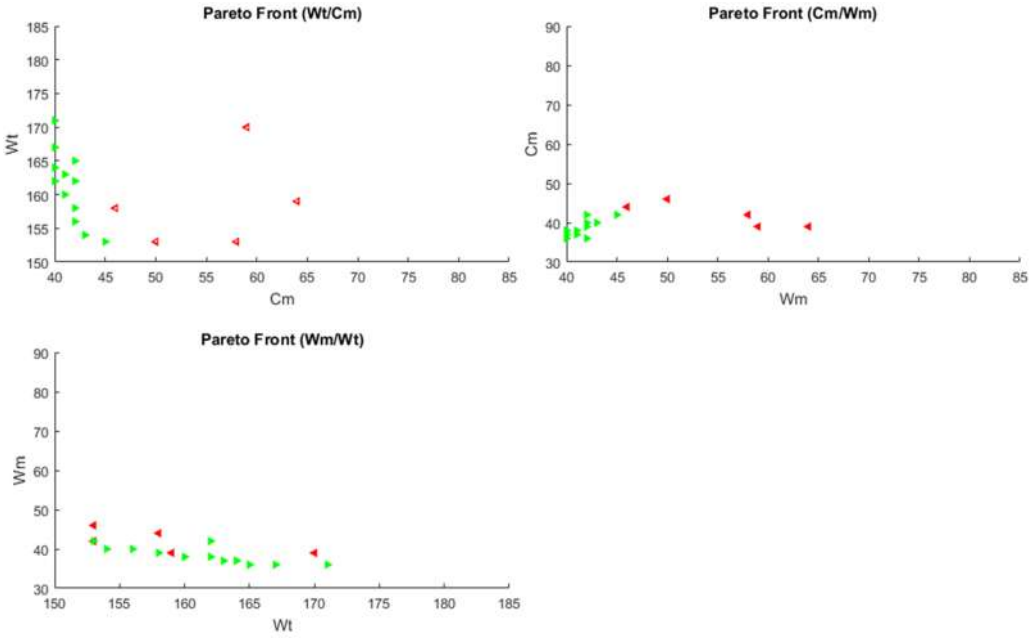
nxm	MK05 (15x4)				MK07 (20x5)				MK08 (20x10)				MK09 (20x10)			
	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.	C_m	W_t	W_m	Sol.
DABC[15]	172	687	172	1	139	693	139	1	523	2524	523	1	-	-	-	-
	173	683	173	1	140	685	140	1	524	2519	524	1	-	-	-	-
	175	682	175	1	143	683	143	1	533	2514	533	1	-	-	-	-
BEG-NSGAII[16]	173	683	173	1	139	693	139	1	523	2524	515	1				
	175	682	175	1	140	686	138	1	523	2534	497	1				
	183	677	183	1	144	673	144	1	524	2519	524	1				
hDPSO[17]	172	687	172	1	139	693	139	1	523	2524	523	1	307	2281	299	1
	173	683	173	1	141	692	141	1	524	2519	524	1	309	2273	299	1
	175	682	175	1	144	673	144	1	533	2514	533	1	311	2260	310	1
Pareto-ABC	173	685	173	5334	143	695	143	3	562	2514	551	1	357	2365	339	1
	173	686	173	7168	144	676	144	5	563	2503	560	1	358	2347	323	1
	174	687	174	2531	144	677	144	12	567	2522	533	1	359	2340	335	1
	175	683	175	465	144	692	144	170	568	2512	551	1	361	2328	388	1
	175	684	175	8855	145	696	141	1	568	2524	533	1	363	2321	347	1
	175	686	175	840	146	688	146	16	570	2522	533	1	369	2308	349	1
	175	687	175	12854	148	693	148	86	570	2534	524	1	369	2313	341	2
	176	682	176	652	156	692	141	1	571	2519	524	1	416	2306	373	1
	176	683	176	18	161	671	161	1	572	2522	533	1	427	2297	410	1
	176	685	173	1	165	669	156	1	572	2527	524	1	435	2292	405	1
	177	686	173	3958	165	690	141	1	573	2498	569	1	440	2288	388	1
	178	685	173	3159	170	665	167	1	573	2517	542	1				
	209	672	209	325	175	664	168	1	573	2522	533	4				
					178	662	166	1	591	2484	587	1				
					184	660	184	1	591	2532	523	1				
					195	667	163	1	592	2538	523	1				
					196	661	181	1	593	2528	523	1				
					211	658	211	1	595	2484	587	1				
									595	2532	523	2				
									596	2484	587	1				

Fonte: [15] Li, et al. (2012) [16] Deng et. Al. (2017) [17] SHAO et. Al. (2013)

Para esta instância, devido ao tamanho dos problemas, serão apresentados apenas os gráficos das fronteiras de Pareto para cada instância.

A Figura 36 apresenta os melhores resultados da literatura para a instância mk01 em pontos verdes e os melhores resultados encontrados pelo algoritmo proposto em pontos vermelhos.

Figura 36 - Fronteira de Pareto para a instância mk01 de Brandimarte.

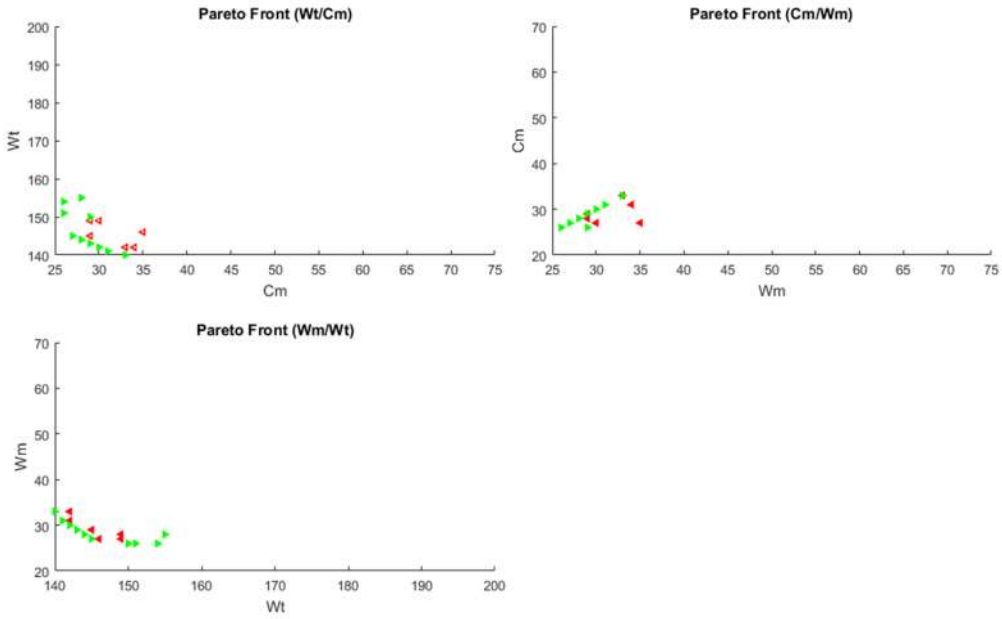


Fonte: Este trabalho.

Como pode ser visto, os melhores resultados conhecidos se aglomeram formando uma barreira, e os resultados encontrados se aproximam da barreira.

A Figura 37 representa a fronteira de Pareto para a instância mk02, na qual pode ser visto que os resultados encontrados se misturam com os melhores resultados conhecidos, não existindo uma separação nítida.

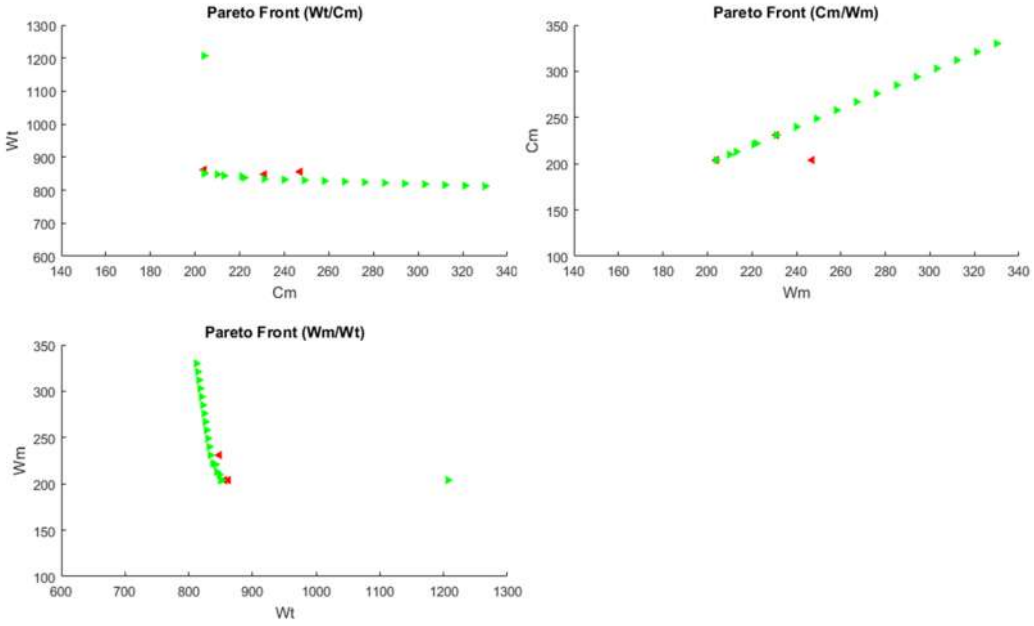
Figura 37 - Fronteira de Pareto para a instância mk02 de Brandimarte.



Fonte: Este trabalho.

A Figura 38 representa a fronteira de Pareto para a instância mk03, na qual pode ser visto que os valores encontrados ficaram distantes da fronteira formada pelos melhores resultados conhecidos. O gráfico wt/cm mostra isso de forma clara. No mesmo gráfico pode ser visto que os valores encontrados para o cm ficam bem próximos aos melhores resultados conhecidos se comparados individualmente. Já no gráfico cm/wm pode ser visto que os melhores valores para o objetivo wm foram encontrados, se analisarmos os objetivos individualmente.

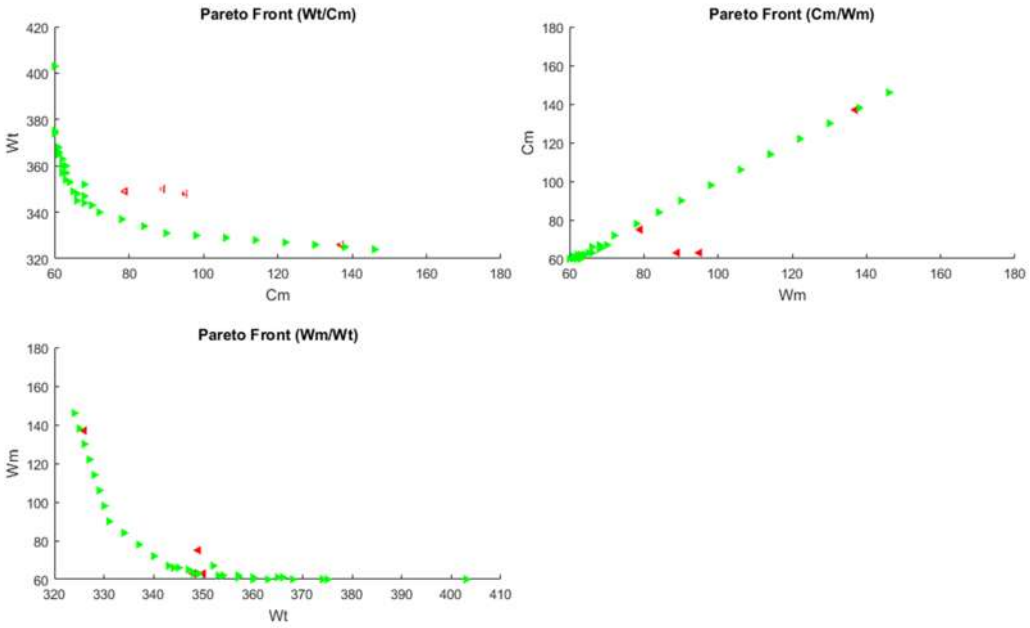
Figura 38 - Fronteira de Pareto para a instância mk03 de Brandimarte.



Fonte: Este trabalho.

A Figura 39 representa a fronteira de Pareto para a instância mk04, podendo ser observado que, na comparação dos resultados wt/cm, houve uma tendência dos resultados formarem uma segunda fronteira, entretanto, dominada pela fronteira de Pareto das melhores soluções. Esta é a fronteira das soluções encontradas pelo algoritmo que, para ser melhorada, necessitaria de alterações nos métodos de agendamento do código proposto.

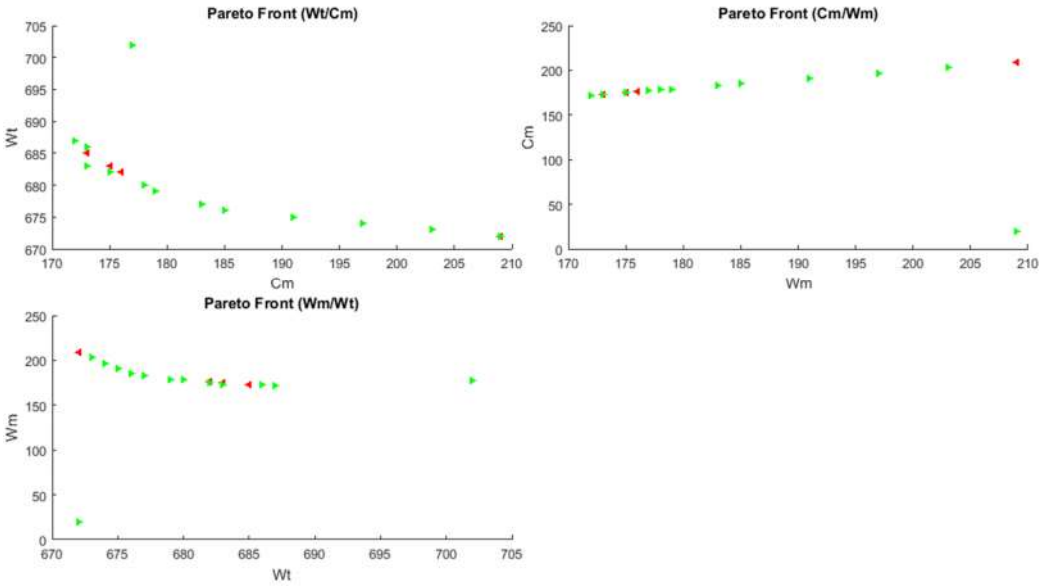
Figura 39 - Fronteira de Pareto para a instância mk04 de Brandimarte.



Fonte: Este trabalho.

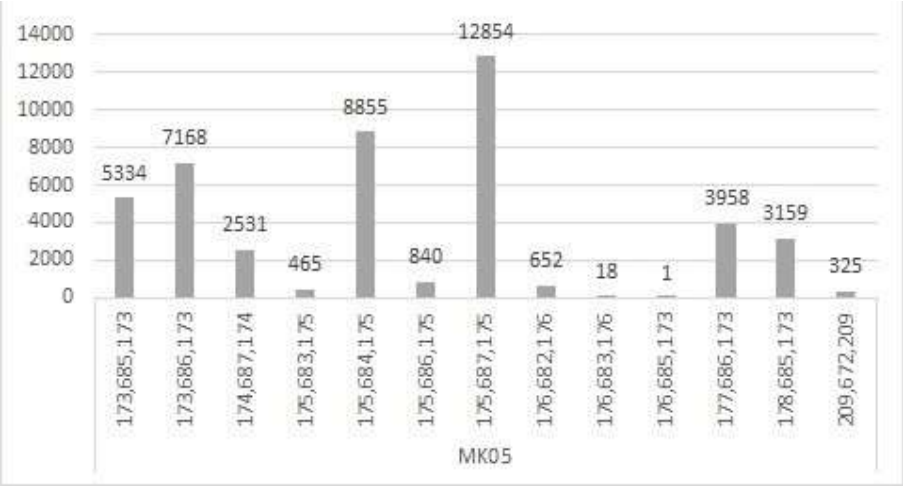
A Figura 40 representa a fronteira de Pareto para a instância mk05, na qual pode ser visto que os valores encontrados estão juntos com os melhores resultados conhecidos. Foi nessa instância que o algoritmo encontrou maior diversidade, totalizando para 13 resultados mais de 46 mil programações diferentes, como pode ser visto na Figura 41.

Figura 40 - Fronteira de Pareto para a instância mk05 de Brandimarte.



Fonte: Este trabalho.

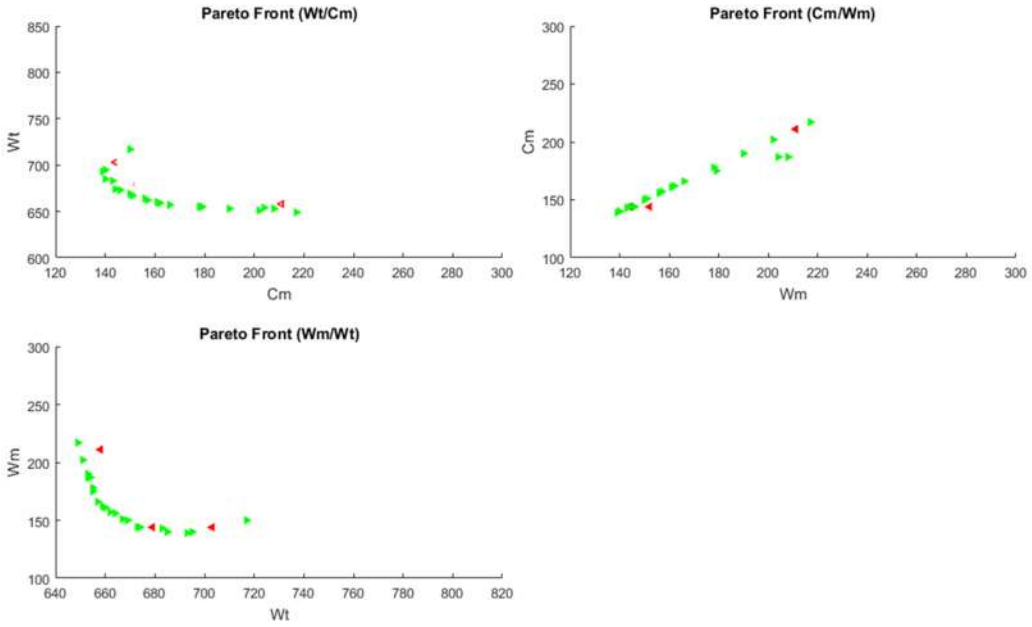
Figura 41 - Diversidade para a instância mk05 de Brandimarte.



Fonte: Este trabalho.

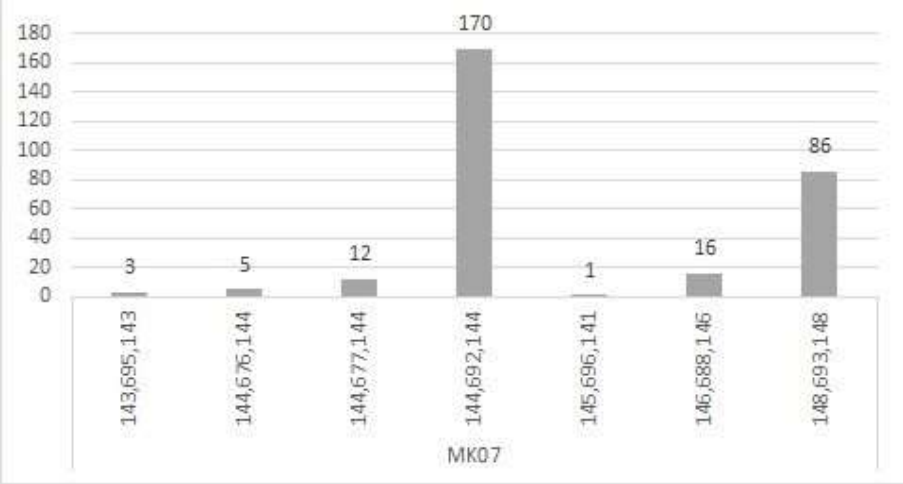
A Figura 42 representa a fronteira de Pareto para a instância mk07. Nesta figura pode ser visto que para a instância mk07 o algoritmo chegou próximo aos melhores resultados conhecidos. Nos gráficos wt/cm e wm/wt as marcações em vermelho se assemelham ao formato das marcações verdes, muito próximas umas das outras. Alguns resultados para esta instância tiveram uma diversidade considerável. Como pode ser visto na Figura 43, a maior quantidade de soluções diferentes foi encontrada para o resultado 144,692,144, com 170 resultados diferentes.

Figura 42 - Fronteira de Pareto para a instância mk07 de Brandimarte.



Fonte: Este trabalho.

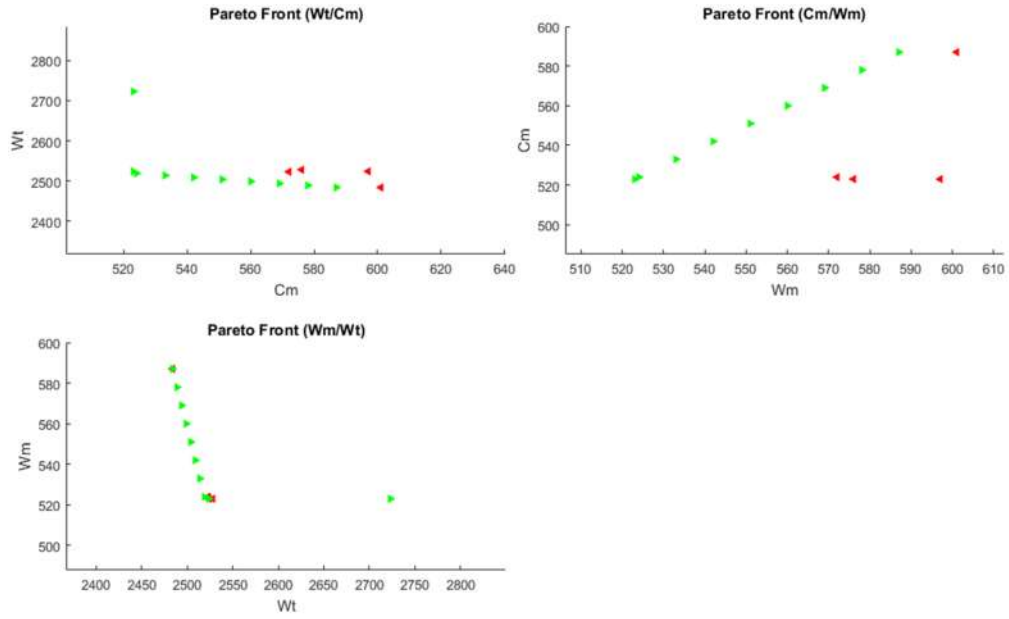
Figura 43 - Diversidade para a instância mk07 de Brandimarte.



Fonte: Este trabalho.

A Figura 44 representa a fronteira de Pareto para a instância mk08, na qual pode ser visto que para esta instância os melhores valores não foram alcançados se analisados de forma multiobjetivo, entretanto, se analisarmos individualmente os objetivos, quase todos os melhores resultados conhecidos para wt e wm foram alcançados. A diversidade de soluções foi baixa para esta instância, apenas um dos valores encontrados teve 4 configurações diferentes, e um com 2 configurações diferentes, como pode ser visto na Tabela 13.

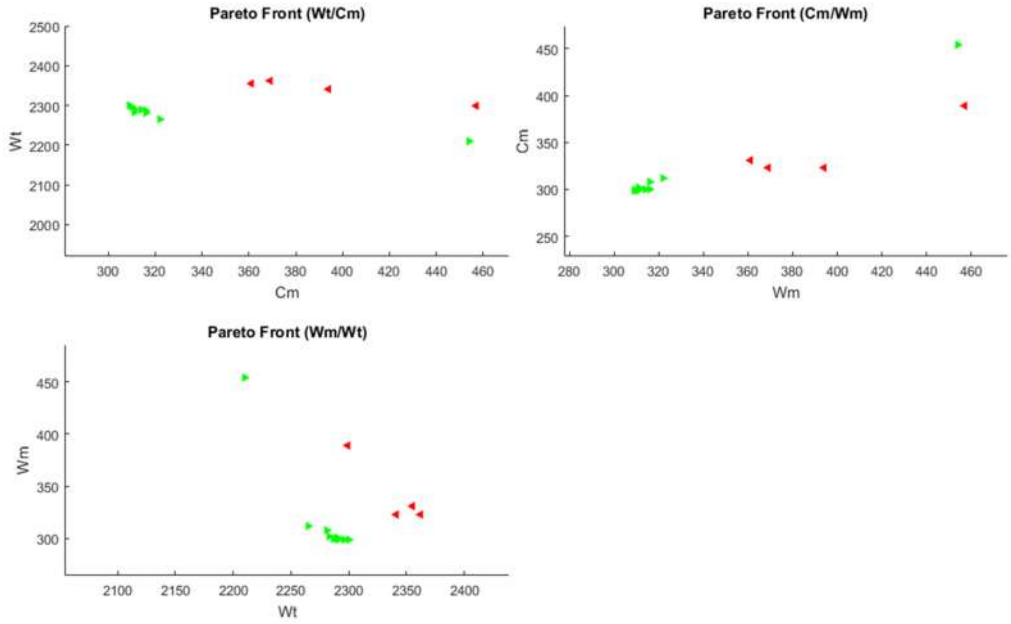
Figura 44 - Fronteira de Pareto para a instância mk08.



Fonte: Este trabalho.

A Figura 45 representa a fronteira de Pareto para a instância mk09, onde pode ser visto que esta instância também não alcançou os melhores resultados conhecidos nem analisando os valores como multiobjetivo e nem analisando individualmente. A diversidade destes resultados também foi baixa, encontrando na maioria das vezes apenas um resultado, com apenas uma exceção para o valor 369,2313,341, que obteve dois resultados diferentes.

Figura 45 - Fronteira de Pareto para a instância mk09.



Fonte: Este trabalho.

Embora os resultados para as instâncias de Brandimarte em sua maioria não tenham alcançado os melhores resultados conhecidos como o esperado, uma diversidade alta de valores bons foi encontrada para a maioria das instâncias.

CAPÍTULO 6

6 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto um algoritmo para tratar o problema da programação da produção FJSP utilizando um método de inteligência coletiva da computação bioinspirada conhecido como ABC. Para contribuir com este método, foi proposta a utilização de dominância de Pareto visando auxiliar o tratamento multiobjetivo e, para melhorar a apresentação dos resultados, foi utilizada uma representação dos valores encontrados em forma de fronteira de Pareto. Os métodos de busca local foram desenvolvidos e propostos de três formas distintas, para que o algoritmo tivesse a capacidade de tratar e apresentar resultados satisfatórios em diversas instâncias.

Os resultados apresentados no capítulo anterior mostraram que o algoritmo proposto tem grande capacidade de resolver os problemas FJSP considerando multiobjetivos e apresentar alta diversidade de soluções. Para algumas instâncias o algoritmo não alcançou os melhores resultados conhecidos devido à construção da etapa de agendamento das operações, mas para a maioria dos problemas que não encontrou, ficou muito próximo desses resultados. Para as instâncias que não tiveram os melhores resultados conhecidos multiobjetivos encontrados, quando a análise foi feita de forma mono objetivo, pôde-se verificar que a maioria dos objetivos foram encontrados, com exceção dos cm.

Embora o algoritmo não tenha encontrado todas as melhores soluções para as instâncias as quais foi submetido, a diversidade de soluções foi alta. Esta diversidade é importante para a programação da produção, pois, em um cenário real, o processo pode ser executado considerando os mesmos custos para todos os objetivos, entretanto utilizando uma programação diferente. Conforme descrito no capítulo anterior, algumas programações com os melhores resultados conhecidos permitem que uma máquina seja minimamente carregada, um exemplo é a Figura 29, que mostra uma programação contendo um dos melhores resultados conhecidos onde a máquina 8 não é utilizada. Com esta máquina menos carregada é possível efetuar manutenções com o processo funcionando sem prejuízos, também é possível até utilizar essa máquina pra outras operações ou remover a máquina, já que não é necessária, considerando que existe uma programação com os melhores resultados

conhecidos que não necessita da utilização dela. Uma outra utilidade para diversas programações, surge com a necessidade de escolher uma programação com menor ou maior *idle time*, dependendo da necessidade dos gestores. Com alta diversidade as programações tendem a apresentar a mais variada quantidade de *idle time* das máquinas.

Durante a análise dos resultados, pôde-se perceber que o algoritmo tem dificuldade em realizar os procedimentos de agendamento das operações. Este procedimento é responsável por fazer as modificações na ordem em que as operações são executadas. O algoritmo foi desenvolvido para fazer os agendamentos, mas isso é não configurável e é limitado considerando grandes soluções. Para soluções que têm muitas operações por *job,* o algoritmo se torna ineficiente, dada essa limitação. Considerando estas conclusões pode-se dizer que o algoritmo não é eficiente para grandes problemas entretanto pode ser alterado para funcionar também para problemas maiores.

Para concluir, os resultados apresentados para as três primeiras instâncias de Kacem foram excelentes se analisarmos o caráter multiobjetivo, pois foram encontradas diversidades altas para os melhores valores conhecidos. Em nenhum dos trabalhos utilizados como referência são apresentadas diversidades altas para todos os valores na maioria das melhores instâncias de um *benchmark*. Se forem considerados apenas os melhores resultados conhecidos para as três primeiras instâncias de Kacem, foram encontrados 3.651 resultados diferentes, sendo eles 14 para a instância 4x5, 82 para a instância 8x8 e 3.555 para a instância 10x10.

Considerando os resultados e análises apresentadas, pôde-se concluir que a recomendação de utilização para este código sem mais alterações é aplicação em cenários totalmente flexíveis, ou parcialmente flexíveis em problemas que disponibilizam várias máquinas e com tempos diferentes entre si nas mesmas operações. Características semelhantes às instâncias de Brandimarte onde se tem poucas máquinas disponíveis por operação e geralmente com o mesmo tempo não são recomentadas para aplicação devido ao gargalo de agendamento do código.

6.1 Trabalhos futuros

Diante dos resultados obtidos nas instâncias de Brandimarte apresentadas no capítulo anterior, e dos resultados analisados como um todo, sugere-se algumas melhorias que podem trazer grandes avanços para o algoritmo proposto. Uma sugestão de melhoria significativa seria uma função específica para realizar uma sequência de agendamentos alternativos, buscando resultados que atualmente o algoritmo tem muita dificuldade de alcançar. Outra sugestão seria a paralelização de algumas etapas do código para aumentar a velocidade de execução. Considerando que algumas das instâncias são muito grandes, o algoritmo tem um tempo computacional elevado. Juntamente com a estratégia de reagendar as operações, sugere-se que seja implementado o método de análise do caminho crítico ou SBH propostos por Choo, Wong e Khader (2016) para facilitar a minimização do *makespan* em algumas instâncias que apresentam maior dificuldade devido à alta complexidade. Um método para sintonização dos parâmetros é sugerido para que a configuração seja feita de forma mais eficiente. O método de Taguchi (TAGUCHI, ELSAYED e HSIANG, 1988) foi avaliado inicialmente para executar esta tarefa, mas devido à complexidade dos problemas, foi deixado como tarefa secundária e não foi implementado.

7 REFERÊNCIAS

ALLAHVERDI, A.; NG, C. T.; CHENG, T. C. E.; KOVALYOV, M. Y. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research, volume 187, edição 3**, 16 junho 2008. 985-1032.

ASADZADEH, L. A parallel artificial bee colony algorithm for the job shop scheduling. **Computers & Industrial Engineering 102**, Teerã, 2016. 359 - 367.

BARNES, J. W.; CHAMBERS, J. B. Flexible Job Shop Scheduling by Tabu Search", Graduate Program in Operations Research and Industrial Engineering. **The University of Texas at Austin, Technical Report Series**, 1996. Disponivel em: <http://www.cs.utexas.edu/users/jbc/>.

BEASLEY, J. E. OR-Library. **People**, 2017. Disponivel em: <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>. Acesso em: 21 outubro 2017.

CHAO, L.-P.; LIN, T.-C.; LU, C.-L. FLEXIBLE JOB-SHOP SCHEDULING BASED ON HYBRID ARTIFICIAL BEE COLONY ALGORITHM WITH DIVERSITY INDEX SEARCH FOR MULTIPLE DECISIONS MAKING IN MANUFACTURING SYSTEM. **International Journal of Advanced Engineering and Management Research Vol. 2, Issue 5**, Keelung, 2017. 1403 - 1417.

CHOO, W. M.; WONG, L.-P.; KHADER, A. T. A Modified Bee Colony Optimization with Local search Aproach for job shop Scheduling problems Relevant to Bottleneck Machines. **Int. J. Advance Soft Compu. Appl, Vol. 8**, Pulau Pinang - Malaysia, 2 Julho 2016. 52 - 78.

DEB, K.; AGRAWAL, S.; PRATAP, A.; MEYARIVAN, T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. **International Conference on Parallel Problem Solving from Nature**, Agosto 2002. 1 - 11.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. **IEEE Trans Evol Comput 6(2)**, 2002. 182-197.

DENG, Q.; GONG, G.; GONG, X.; ZHANG, L.; LIU, W.; REN, Q. A Bee Evolutionary Guiding Nondominated Sorting Genetic Algorithm II for Multiobjective Flexible Job-Shop Scheduling. **Computational Intelligence and Neuroscience Volume 2017**, Changsha, 28 mar. 2017. 20. Disponivel em: <https://doi.org/10.1155/2017/5232518>. Acesso em: 19 jul. 2018.

GAO, K. Z.; SUGANTHAN, P. N.; CHUA, T. J. Pareto-based discrete harmony search algorithm for flexible job shop scheduling. **Proceedings of 12th International Conference on Intelligent Systems Design and Applications**, 2012. 953 - 956.

GAO, K. Z.; SUGANTHAN, P. N.; PAN, Q. K.; CHUA, T. J.; CHONG, C. S.; CAI, T. X. An improve d artificial b ee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. **Expert Systems With Applications**, p. 52 - 67, Outubro 2015.

GAREY, M. R.; JOHNSON, D. S.; SETHI, R. The complexity of flowshop and jobshop scheduling. **Mathematics of operations research, INFORMS, v. 1, n. 2**, 1976. 117–129.

GOLDBARG, M. C.; GOLDBARG, E. G.; LUNA, H. P. L. **Otimização Combinatória e Meta-Heurísticas:** Algoritmos e Aplicações. 1. ed. Rio de Janeiro: Elsevier Editora ltda, v. 1, 2016. 392 p.

KACEM, I.; HAMMADI, S.; BORNE, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, fevereiro 2002. 1 - 13.

KARABOGA, D. **An idea based on honey bee swarm for numerical optimization**. Erciyes University, Engineering Faculty, Computer Engineering Department. Turquia, p. 1 - 10. 2005.

KATO, E. R. R.; ARANHA, G. D. D. A.; TSUNAKI, R. H. A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimizationk and Random-Restart Hill Climbing. **Computers & Industrial Engineering**, São Carlos, SP, Brasil, n. 125 (2018), 23 Agosto 2018. 178 - 189. Disponivel em: <https://doi.org/10.1016/j.cie.2018.08.022>.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. **Proceedings of the IEEE international conference on neural networks, 1995, vol 4. IEEE**, 1995. 1942–1948.

LEE, K. S.; GEEM, Z. W. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. **Computer Methods in Applied Mechanics and Engineering, 194**, 2005. 3902-3933.

LI, J.-Q.; PAN, Q.-K.; GAO, K.-Z. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. **Int J Adv Manuf Technol (2011)**, Londres, 12 janeiro 2011. 1159 - 1169.

LI, J.-Q.; PAN, Q.-K.; TASGETIREN, M. F. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. **Applied Mathematical Modelling, Elsevier, v. 38, n. 3**, 2014. p. 1111–1132. Disponivel em: <http://dx.doi.org/10.1016/j.apm.2013.07.038>. Acesso em: 30 junho 2018.

LI, J.-Q.; PAN, Q.-K.; XIE, S.-X.; WANG, S. A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problems. **International Journal of Computers Communications & Control**, 6, 2 junho 2011. 286-296.

LI, X.; PENG, Z.; DU, B.; GUO, J.; XU, W.; ZHUANG, K. Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems. **Computers & Industrial Engineering 113**, Wuhan, China, 8 setembro 2017. 10 - 26.

LIU, Y.; OUYANG, D.; GU, W.; WANG, L. A discrete artificial bee colony algorithm for permutation flow shop scheduling. **9th International Symposium on Computational Intelligence and Design**, ChangChun, 2016. 161 - 164.

LU, C. L.; CHENG, H.-C.; CHIU, S.-Y.; YEN, S.-J. Solving the Flexible Job-Shop Scheduling Problem Based on Multi-Objective PSO with Pareto Diversity Search. **International Journal of Intelligent Information Processing, Vol. 4**, 2013. 70-81.

MUTHIAH, A.; RAJKUMAR, R.; RAJKUMAR, A. Hybridization of Artificial Bee Colony Algorithm with Particle Swarm Optimization Algorithm for flexible Job Shop Scheduling. **©2016 IEEE**, Savakasi, 2016. 896 - 903.

NEALE, M.; CAMERON, J. R. Committee for Tero technology: A guide to the condition monitoring of machinery. **HMSO Publications**, Londres, 1979. 89-96.

SANCHES, R. F. V. Algoritmo de enxame de abelhas para resolução do problema da programação da produção job shop flexível multiobjetivo. **Dissertassão de Mestrado. Programa de pós-graduação em ciência da computação, Universidade Federal de São Carlos**, São Carlos, fevereiro 2017. p. 100.

SHAO, X.; LIU, W.; LIU, Q.; ZHANG, C. Hybrid discrete particle swarm optimization formulti-objective flexible job-shop scheduling problem. **International Journal of Advanced Manufacturing Technology**, agosto 2013. 2886 - 2902.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da Produção**. São Paulo: Atlas S.A., 2009.

TAGUCHI, G.; ELSAYED, A. E.; HSIANG, C. T. **Quality Engineering in Production Systems**. [S.l.]: McGraw-Hill College, 1988.

THAMMANO, A.; PHU-ANG, A. A Hybrid Artificial Bee Colony Algorithm with Local Search for Flexible Job-Shop Scheduling Problem. **ScienceDirect**, Banguecoque, 2013. 96 - 101.

THAMMANO, A.; PHU-ANG, A. Procedia Computer Science. **ScienceDirect**, Bangkok, 20, 2013. 96 - 101.

WANG, L.; ZHOU, G.; XU, Y.; LIU, M. An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. **Int J Adv Manuf Technol (2012) 60**, Londres, 2 outubro 2011. 1111–1123.

WANG, S.; YU, J. An effective heuristic for flexible job-shop scheduling problem with maintenance activities. **Computers & Industrial Engineering, Elsevier, v. 59, n. 3**, 2010. p. 436–447.

XIA, W.; WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. **Computers & Industrial Engineering, Volume 48, edição 2**, março 2005. 409-425.

XING, L. -N.; CHEN, Y. -W.; WANG, P.; ZHAO, Q. -S.; XIONG, J. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. **Applied Soft Computing V. 10, Issue 3**, Changsha, Junho 2010. p. 888-896.

YAZDANI, M.; GHOLAMI, M.; ZANDIEH, M.; MOUSAKHANI, M. A Simulated Annealing Algorithm for Flexible Job-Shop Scheduling Problem. **Journal of Applied Sciences, 9**, Qazvin - Irã, 2009. 662-670. Disponivel em: <http://scialert.net/abstract/?doi=jas.2009.662.670>. Acesso em: 08 out. 2017.

YUE, L.; GUAN, Z.; SAIF, U.; ZHANG, F.; WANG, H. Hybrid Pareto artificial bee colony algorithm for multi-objective single machine group scheduling problem with

sequence-dependent setup times and learning effects. **SpringerPlus (2016)**, Taxila, 2016. 1 - 31.

ZHENG, Y.; LIAN, L.; MESGHOUNI, K. Comparative Study of Heuristics Algorithms in Solving Flexible Job Shop Scheduling Problem with Condition Based Maintenance. **Journal of Industrial Engineering and Management**, 2014. 518-531. Disponivel em: <http://dx.doi.org/10.3926/jiem.1038>. Acesso em: 10 jun. 2017.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. SPEA2: improving the performance of the strength Pareto evolutionary algorithm. **TIK-Report 103**, maio 2001.

8 Apêndice I

Os resultados abaixo são resultados encontrados pelo algoritmo desenvolvido. Estão separados por instância contendo várias soluções para cada uma delas.

Apenas programações para os melhores resultados conhecidos serão apresentadas.

Os resultados apresentados na Tabela 14 são alguns dos resultados apresentados pelo algoritmo para a instância 4x5 de Kacem. Nem todos os resultados serão apresentados devido às quantidades e dimensões dos valores encontrados.

As tabelas apresentadas a seguir seguem a seguinte formatação: a primeira coluna contém na primeira linha o nome da instância, e nas demais linhas uma numeração sequencial. As colunas cm, wt e wm representam os valores dos objetivos. A coluna máquinas representa a primeira parte do vetor de solução, onde são armazenadas as máquinas que executam cada operação. A segunda parte do vetor é presentada na coluna Ordem de execução, onde os valores representam a ordem que a operação deve ser ordenada no gráfico de Gantt.

Tabela 14 – Vetores de soluções para a instância 4x5

4x5	cm	wt	wm	Máquinas												Ordem de execução											
Sol1	11	32	10	4	1	1	3	2	4	2	5	4	3	1	4	1	4	11	7	2	12	8	5	9	6	3	10
Sol2	11	32	10	3	1	4	1	2	2	5	2	4	3	1	4	7	4	1	11	2	12	5	8	9	6	3	10
Sol3	11	32	10	1	3	4	1	2	2	5	2	3	4	1	4	4	7	1	11	2	8	5	12	6	9	3	10
Sol4	11	32	10	1	1	3	4	2	4	5	2	4	3	1	4	4	11	7	1	2	12	5	8	3	6	9	10
Sol5	11	32	10	4	1	3	1	2	5	2	2	1	4	3	4	1	4	7	11	2	5	8	12	9	3	6	10
Sol6	11	32	10	4	3	1	1	2	2	2	5	3	4	1	4	1	7	4	11	2	12	8	5	6	3	9	10
Sol7	11	34	9	4	1	3	1	4	5	3	2	2	1	4	4	1	4	7	11	12	5	8	2	3	6	9	10
Sol8	12	32	8	1	1	4	3	2	2	2	5	4	1	4	4	11	4	1	7	2	12	8	5	3	6	9	10
Sol9	12	32	8	1	3	4	1	5	2	2	2	4	4	1	4	4	7	1	11	5	2	8	12	3	9	6	10
Sol10	12	32	8	4	1	1	3	2	5	2	2	1	4	4	4	1	11	4	7	2	5	8	12	6	3	9	10
Sol11	12	32	8	4	3	1	1	5	2	2	2	4	4	1	4	1	7	4	11	5	2	12	8	3	9	6	10
Sol12	13	33	7	3	1	1	5	2	2	5	2	4	1	4	4	7	4	11	1	2	8	5	12	3	6	9	10
Sol13	13	33	7	5	1	1	3	2	5	2	2	4	1	4	4	1	11	4	7	2	5	8	12	3	6	9	10
Sol14	13	33	7	1	5	3	1	2	2	2	5	1	4	4	4	11	1	7	4	12	2	8	5	6	3	9	10

Os resultados apresentados na Tabela 15 são alguns dos resultados apresentados pelo algoritmo para a instância 8x8 de Kacem. Nem todos os resultados serão apresentados devido às quantidades e dimensões dos valores encontrados.

Tabela 15 - Vetores de soluções para a instância 8x8

8x8	cm	wt	wm	Máquinas																											Ordem de execução																										
Sol1	14	77	12	5	3	7	2	3	3	1	1	5	2	4	6	4	8	8	7	6	8	1	6	3	2	4	7	3	7	5	1	18	8	11	21	4	14	24	2	25	9	12	5	19	22	15	3	26	10	16	13	20	23	6	27	17	7
Sol2	14	77	12	4	3	7	2	3	3	1	1	5	6	4	4	7	8	8	2	6	7	2	3	4	1	6	8	3	5	7	1	18	8	11	21	4	14	24	2	12	9	5	15	19	22	25	3	6	20	13	23	10	16	26	27	7	17
Sol3	14	77	12	3	3	7	2	1	4	3	1	5	6	4	8	7	4	8	2	1	7	2	3	6	6	4	8	7	3	5	18	21	8	11	14	1	4	24	2	12	9	19	15	5	22	25	10	6	20	13	16	3	23	26	17	27	7
Sol4	15	75	12	5	3	7	2	1	1	3	3	5	6	4	7	4	8	8	2	3	7	1	6	2	6	4	8	5	7	5	1	18	8	11	14	24	21	4	2	12	9	15	5	19	22	25	13	6	10	3	20	16	23	26	7	17	27
Sol5	15	75	12	3	5	7	2	1	3	3	1	5	4	8	7	6	4	8	2	6	1	7	3	6	2	8	4	5	7	5	18	1	8	11	14	4	21	24	2	5	19	15	12	9	22	25	3	10	6	13	16	20	26	23	7	17	27
Sol6	15	75	12	5	3	1	2	7	3	3	1	6	4	4	5	7	8	8	2	6	7	1	6	3	2	4	8	5	7	5	1	18	14	11	8	4	21	24	12	9	5	2	15	19	22	25	3	6	10	16	13	20	23	26	7	17	27
Sol7	15	75	12	7	3	5	2	1	3	3	1	4	6	5	4	7	8	8	2	6	7	6	3	1	8	4	2	7	5	5	8	21	1	11	14	18	4	24	9	12	2	5	15	19	22	25	3	6	16	13	10	26	23	20	17	7	27
Sol8	15	75	12	5	3	7	2	3	3	1	1	5	6	4	8	7	4	8	2	2	7	1	3	6	6	8	4	5	5	7	1	18	8	11	21	4	14	24	2	12	9	19	15	5	22	25	20	6	10	13	16	3	26	23	7	27	17
Sol9	15	75	12	5	3	7	2	1	1	3	3	5	4	4	2	8	7	8	6	6	2	6	3	1	7	4	8	5	7	5	1	18	8	11	14	24	21	4	2	5	9	25	19	15	22	12	3	20	16	13	10	6	23	26	7	17	27
Sol10	15	75	12	5	3	7	1	2	3	3	1	6	4	4	7	5	8	8	2	6	7	8	1	6	2	4	3	7	5	5	1	21	8	14	11	18	4	24	12	5	9	15	2	19	22	25	3	6	26	10	16	20	23	13	17	7	27
Sol11	15	75	12	5	3	7	2	1	3	3	1	5	8	4	6	7	8	4	2	1	7	2	3	6	6	4	8	7	5	5	1	18	8	11	14	4	21	24	2	19	9	12	15	22	5	25	10	6	20	13	16	3	23	26	17	7	27
Sol12	15	75	12	5	1	7	2	1	3	3	3	2	4	8	6	7	4	8	5	4	7	8	3	6	2	6	1	5	5	7	1	14	8	11	24	18	21	4	25	5	19	12	15	9	22	2	23	6	26	13	16	20	3	10	7	27	17
Sol13	15	75	12	5	3	7	2	1	3	3	1	7	4	8	6	5	4	8	2	6	7	1	3	8	2	4	6	5	7	5	1	21	8	11	14	18	4	24	15	5	19	12	2	9	22	25	16	6	10	13	26	20	23	3	7	17	27
Sol14	15	75	12	1	1	7	2	4	3	3	3	5	8	4	6	7	4	8	2	6	6	1	3	7	2	4	8	5	7	5	14	24	8	11	1	18	21	4	2	19	9	12	15	5	22	25	16	3	10	13	6	20	23	26	7	17	27
Sol15	15	75	12	4	3	7	2	3	1	3	1	5	4	4	6	7	8	8	2	6	7	1	4	6	2	3	8	5	7	5	1	18	8	11	4	14	21	24	2	5	9	12	15	19	22	25	3	6	10	23	16	20	13	26	7	17	27
Sol16	15	75	12	4	3	3	2	1	3	7	1	5	4	7	8	4	6	8	2	4	6	1	3	6	2	7	8	7	5	5	1	18	21	11	14	4	8	24	2	5	15	19	9	12	22	25	23	3	10	13	16	20	6	26	17	7	27
Sol17	15	75	12	4	3	7	2	1	3	1	3	4	4	5	8	7	6	8	2	6	2	1	7	6	3	4	8	5	5	7	1	21	8	11	14	18	24	4	9	5	2	19	15	12	22	25	3	20	10	6	16	13	23	26	7	27	17
Sol18	15	75	12	3	3	7	2	1	1	3	5	5	4	7	6	4	8	8	2	6	8	1	3	6	2	4	7	7	5	5	18	4	8	11	14	24	21	1	2	5	15	12	9	19	22	25	16	26	10	13	3	20	23	6	17	7	27
Sol19	15	75	12	3	4	7	2	1	3	3	1	4	4	8	6	7	5	8	2	6	3	7	1	6	2	4	8	7	5	5	18	1	8	11	14	4	21	24	9	5	19	12	15	2	22	25	3	13	6	10	16	20	23	26	17	7	27
Sol20	15	75	12	4	3	7	2	1	3	3	1	5	4	4	6	7	2	8	8	6	2	8	3	6	7	4	1	7	5	5	1	21	8	11	14	18	4	24	2	5	9	12	15	25	19	22	3	20	26	13	16	6	23	10	17	7	27
Sol21	15	75	12	3	3	7	1	2	4	3	1	4	5	4	6	7	8	8	2	6	7	1	8	6	2	4	3	5	7	5	18	4	8	14	11	1	21	24	5	2	9	12	15	19	22	25	16	6	10	26	3	20	23	13	7	17	27
Sol22	15	75	12	3	4	7	2	1	3	3	1	5	4	4	7	2	8	8	6	8	7	1	3	4	2	6	6	5	7	5	21	1	8	11	14	18	4	24	2	5	9	15	25	19	22	12	26	6	10	13	23	20	16	3	7	17	27
Sol23	15	75	12	4	3	7	2	1	3	3	1	5	8	4	6	7	4	8	2	6	7	1	3	6	4	2	8	5	7	5	1	18	8	11	14	4	21	24	2	19	9	12	15	5	22	25	16	6	10	13	3	23	20	26	7	17	27
Sol24	15	75	12	4	2	7	3	1	3	3	1	4	4	6	5	7	8	8	2	6	7	6	3	1	2	4	8	5	7	5	1	11	8	21	14	18	4	24	9	5	12	2	15	19	22	25	16	6	3	13	10	20	23	26	7	17	27
Sol25	15	75	12	5	3	3	2	1	7	3	1	4	4	5	6	7	8	8	2	4	7	1	3	6	8	6	2	7	5	5	1	21	18	11	14	8	4	24	9	5	2	12	15	19	22	25	23	6	10	13	16	26	3	20	17	7	27
Sol26	15	75	12	4	3	3	2	1	3	7	1	5	4	2	6	7	8	8	4	6	7	1	3	6	2	4	8	5	5	7	1	18	21	11	14	4	8	24	2	9	25	12	15	19	22	5	3	6	10	13	16	20	23	26	7	27	17
Sol27	15	75	12	4	3	3	2	1	3	7	1	5	4	4	2	7	8	8	6	8	2	1	3	6	7	4	6	5	5	7	1	18	21	11	14	4	8	24	2	5	9	25	15	19	22	12	26	20	10	13	16	6	23	3	7	27	17
Sol28	16	73	13	2	3	7	3	1	2	3	1	5	8	4	6	4	4	8	2	6	7	1	3	6	2	4	8	5	7	5	1	4	8	18	14	11	21	24	2	19	5	12	15	9	22	25	3	6	10	13	16	20	23	26	7	17	27
Sol29	16	73	13	7	3	1	2	2	3	3	1	4	4	6	4	5	8	8	2	6	7	2	3	6	1	4	8	7	5	5	8	4	14	11	1	18	21	24	15	5	12	9	2	19	22	25	3	6	20	13	16	10	23	26	17	7	27
Sol30	16	73	13	3	3	7	2	1	2	3	1	4	4	4	5	6	8	8	2	6	7	8	3	6	1	4	2	5	7	5	18	4	8	11	14	1	21	24	15	5	9	2	12	19	22	25	3	6	26	13	16	10	23	20	7	17	27
Sol31	16	73	13	2	3	7	2	1	3	3	1	4	4	4	6	5	8	8	2	7	1	6	3	6	2	4	8	5	7	5	1	4	8	11	14	18	21	24	15	5	9	12	2	19	22	25	6	10	3	13	16	20	23	26	7	17	27
Sol32	16	73	13	2	3	1	2	7	3	3	1	5	4	8	6	4	8	4	2	6	7	1	2	8	3	4	6	7	5	5	11	4	14	1	8	18	21	24	2	5	19	12	15	22	9	25	3	6	10	20	26	13	23	16	17	7	27
Sol33	16	73	13	5	1	7	2	3	3	3	1	8	4	4	6	4	5	8	2	7	6	1	3	6	2	4	8	5	7	5	1	14	8	11	4	18	21	24	19	5	15	12	9	2	22	25	6	3	10	13	16	20	23	26	7	17	27
Sol34	16	73	13	2	3	3	3	1	7	2	1	2	4	8	6	4	4	8	5	2	7	1	3	6	6	4	8	5	7	5	1	4	18	21	14	8	11	24	25	5	19	12	15	9	22	2	20	6	10	13	16	3	23	26	7	17	27

Sol35	16	73	13	2	3	3	2	1	3	7	1	4	4	6	4	5	8	8	2	6	7	8	6	3	2	4	1	7	5	5	1	18	21	11	14	4	8	24	15	5	12	9	2	19	22	25	3	6	26	16	13	20	23	10	17	7	27
Sol36	16	73	13	2	3	7	1	3	1	3	2	8	4	2	6	4	5	8	4	1	2	6	3	6	7	4	8	5	5	7	1	4	8	24	18	14	21	11	19	5	25	12	15	2	22	9	10	20	3	13	16	6	23	26	7	27	17
Sol37	16	73	13	2	7	3	2	1	3	3	1	4	4	4	6	8	8	5	2	6	7	6	3	1	2	8	4	7	5	5	1	8	21	11	14	18	4	24	15	5	9	12	22	19	2	25	3	6	16	13	10	20	26	23	17	7	27
Sol38	16	73	13	3	3	7	2	1	2	3	1	5	4	8	6	4	4	8	2	8	7	1	3	6	6	4	2	7	5	5	4	18	8	11	14	1	21	24	2	5	19	12	15	9	22	25	26	6	10	13	16	3	23	20	17	7	27
Sol39	16	73	13	1	7	3	2	1	3	3	2	2	4	8	6	4	4	8	5	1	7	2	3	6	6	4	8	5	7	5	24	8	4	11	14	18	21	1	25	5	19	12	15	9	22	2	10	6	20	13	16	3	23	26	7	17	27
Sol40	16	73	13	2	1	7	2	3	3	3	1	4	4	6	4	5	8	8	2	3	7	1	6	6	2	4	8	7	5	5	1	14	8	11	21	18	4	24	15	5	12	9	2	19	22	25	13	6	10	3	16	20	23	26	17	7	27
Sol41	16	73	13	2	1	3	2	3	7	3	1	5	6	4	4	4	8	8	2	6	1	3	7	6	2	4	8	5	7	5	1	14	18	11	4	8	21	24	2	12	15	5	9	19	22	25	3	10	13	6	16	20	23	26	7	17	27
Sol42	16	73	13	7	2	2	3	1	3	3	1	4	4	4	6	5	8	8	2	4	7	1	3	6	8	6	2	5	5	7	8	11	1	4	14	18	21	24	15	5	9	12	2	19	22	25	23	6	10	13	16	26	3	20	7	27	17
Sol43	16	73	13	3	3	1	2	7	2	3	1	4	4	4	6	5	8	8	2	4	7	1	3	2	6	6	8	7	5	5	18	4	14	11	8	1	21	24	15	5	9	12	2	19	22	25	23	6	10	13	20	16	3	26	17	7	27
Sol44	16	73	13	2	3	7	2	1	3	3	1	5	4	8	6	4	4	8	2	3	7	6	6	1	2	4	8	5	5	7	1	4	8	11	14	18	21	24	2	15	19	12	5	9	22	25	13	6	16	3	10	20	23	26	7	27	17
Sol45	16	73	13	2	3	7	2	3	1	3	1	5	4	8	4	6	4	8	2	7	4	1	3	6	2	6	8	5	5	7	1	4	8	11	18	24	21	14	2	5	19	15	12	9	22	25	6	23	10	13	16	20	3	26	7	27	17
Sol46	16	73	13	3	3	7	2	1	2	3	1	5	4	8	6	4	4	8	2	6	7	1	3	6	2	4	8	7	5	5	18	21	8	11	14	1	4	24	2	15	19	12	5	9	22	25	3	6	10	13	16	20	23	26	17	7	27
Sol47	16	73	13	5	3	1	2	1	3	3	7	5	4	8	2	4	4	8	6	1	7	8	3	6	2	4	6	5	7	5	1	4	24	11	14	18	21	8	2	5	19	25	15	9	22	12	10	6	26	13	16	20	23	3	7	17	27
Sol48	16	73	13	1	3	7	2	1	3	3	2	5	4	8	6	4	8	4	2	6	3	1	7	6	2	4	8	5	5	7	24	4	8	11	14	18	21	1	2	5	19	12	15	22	9	25	3	13	10	6	16	20	23	26	7	27	17
Sol49	16	73	13	2	3	7	2	1	3	3	1	4	4	4	6	5	8	8	2	7	8	1	3	6	2	4	6	5	7	5	1	18	8	11	14	21	4	24	15	5	9	12	2	19	22	25	6	26	10	13	16	20	23	3	27	17	7
Sol50	16	73	13	3	2	7	3	1	3	2	1	5	4	8	6	4	8	4	2	6	7	1	3	6	2	8	4	5	7	5	21	11	8	4	14	18	1	24	2	15	22	12	5	19	9	25	3	6	10	13	16	20	26	23	7	17	27
Sol51	16	73	13	2	3	3	2	1	3	7	1	4	4	4	5	6	8	8	2	6	3	1	6	7	2	4	8	5	7	5	1	18	21	11	14	4	8	24	15	5	9	2	12	19	22	25	3	13	10	16	6	20	23	26	27	17	7
Sol52	16	73	13	3	3	7	2	1	3	2	1	5	4	8	6	4	8	4	2	6	7	2	1	6	3	4	8	5	7	5	21	18	8	11	14	4	1	24	2	15	22	12	5	19	9	25	3	6	20	10	16	13	23	26	7	17	27
Sol53	16	73	13	2	3	7	2	3	3	1	1	4	4	4	6	5	2	8	8	6	7	1	8	6	2	4	3	5	7	5	1	21	8	11	4	18	14	24	15	5	9	12	2	25	22	19	3	6	10	26	16	20	23	13	7	17	27
Sol54	16	73	13	5	3	7	2	3	1	3	1	5	4	4	6	2	8	8	4	2	7	1	3	4	6	6	8	5	7	5	1	4	8	11	18	24	21	14	2	5	15	12	25	19	22	9	20	6	10	13	23	3	16	26	7	17	27
Sol55	16	73	13	2	3	7	2	1	3	1	3	4	4	4	6	8	5	8	2	4	7	1	3	6	6	2	8	5	7	5	1	21	8	11	14	18	24	4	15	5	9	12	19	2	22	25	23	6	10	13	16	3	20	26	7	17	27
Sol56	16	73	13	5	3	2	7	1	3	1	3	4	4	5	6	4	8	8	2	6	7	3	1	6	2	4	8	7	5	5	1	4	11	8	14	18	24	21	15	5	2	12	9	19	22	25	3	6	13	10	16	20	23	26	17	7	27
Sol57	16	73	13	2	3	7	3	1	2	3	1	5	6	4	4	4	8	8	2	7	8	1	3	6	2	4	6	5	7	5	1	18	8	4	14	11	21	24	2	12	15	5	9	19	22	25	6	26	10	13	16	20	23	3	7	17	27
Sol58	16	73	13	5	3	7	1	2	3	3	1	5	6	4	4	4	8	8	2	6	4	6	3	1	2	7	8	5	7	5	1	21	8	14	11	18	4	24	2	12	15	5	9	19	22	25	3	23	16	13	10	20	6	26	7	17	27
Sol59	16	73	13	5	7	3	1	2	3	3	1	4	4	4	5	6	8	8	2	8	2	1	3	6	7	4	6	5	5	7	1	8	4	14	11	18	21	24	15	5	9	2	12	19	22	25	26	20	10	13	16	6	23	3	7	27	17
Sol60	16	73	13	5	3	2	7	1	3	3	1	5	4	4	6	4	8	2	8	7	8	1	3	6	2	4	6	5	7	5	1	4	11	8	14	18	21	24	2	5	15	12	9	19	25	22	6	26	10	13	16	20	23	3	7	17	27
Sol61	16	73	13	5	3	7	2	1	3	3	1	4	4	6	4	5	8	8	2	7	6	1	3	6	2	4	8	5	7	5	1	18	8	11	14	4	21	24	15	5	12	9	2	19	22	25	6	3	10	13	16	20	23	26	7	17	27
Sol62	16	73	13	5	7	3	2	1	3	3	1	4	4	4	6	8	8	5	2	6	7	1	3	8	2	4	6	5	7	5	1	8	21	11	14	18	4	24	15	5	9	12	22	19	2	25	3	6	10	13	26	20	23	16	7	17	27
Sol63	16	73	13	3	3	7	2	1	3	5	1	4	4	8	6	5	4	8	2	6	7	1	3	6	8	4	2	7	5	5	21	18	8	11	14	4	1	24	15	5	19	12	2	9	22	25	16	6	10	13	3	26	23	20	17	7	27
Sol64	16	73	13	3	3	7	2	1	3	5	1	5	4	8	6	4	8	4	2	6	4	8	3	6	2	7	1	5	7	5	21	4	8	11	14	18	1	24	2	15	22	12	5	19	9	25	3	23	26	13	16	20	6	10	7	17	27
Sol65	16	73	13	3	2	7	2	1	3	3	1	4	4	4	6	8	8	5	2	8	6	1	3	7	2	4	6	7	5	5	21	1	8	11	14	18	4	24	15	5	9	12	22	19	2	25	26	16	10	13	6	20	23	3	17	7	27
Sol66	16	73	13	3	3	2	7	1	5	3	1	4	4	2	6	5	8	8	4	2	7	6	3	1	6	4	8	5	7	5	18	4	11	8	14	1	21	24	15	5	25	12	2	19	22	9	20	6	16	13	10	3	23	26	7	17	27
Sol67	16	73	13	1	1	7	2	5	3	3	3	4	4	4	8	5	6	8	2	6	7	1	4	6	2	8	3	5	5	7	14	24	8	11	1	18	21	4	15	5	9	19	2	12	22	25	3	6	10	23	16	20	26	13	7	27	17
Sol68	16	73	13	5	3	7	2	1	3	3	1	4	4	4	6	8	5	8	2	3	7	6	6	1	2	4	8	5	7	5	1	18	8	11	14	21	4	24	15	5	9	12	19	2	22	25	13	6	16	3	10	20	23	26	7	17	27
Sol69	16	73	13	5	3	7	2	3	3	1	1	4	4	8	6	5	4	8	2	6	6	4	3	7	2	1	8	5	5	7	1	18	8	11	21	4	14	24	15	5	19	12	2	9	22	25	3	16	23	13	6	20	10	26	27	7	17
Sol70	16	73	13	5	3	7	2	3	3	1	1	4	4	4	6	5	8	8	2	2	7	1	3	6	6	4	8	5	5	7	1	18	8	11	21	4	14	24	15	5	9	12	2	19	22	25	20	6	10	13	16	3	23	26	27	7	17
Sol71	16	73	13	7	3	5	2	1	3	3	1	4	4	4	6	5	8	8	2	1	7	4	3	6	2	6	8	5	5	7	8	21	1	11	14	18	4	24	15	5	9	12	2	22	19	25	10	6	23	13	16	20	3	26	7	27	17
Sol72	16	73	13	3	3	3	2	1	7	5	1	4	4	4	6	8	8	5	2	4	7	1	3	6	2	8	6	7	5	5	21	4	18	11	14	8	1	24	15	5	9	12	22	19	2	25	23	6	10	13	16	20	26	3	17	7	27
Sol73	16	73	13	3	3	2	2	1	3	7	1	4	4	4	6	5	2	8	8	4	7	6	3	1	2	6	8	7	5	5	21	4	1	11	14	18	8	24	15	5	9	12	2	25	22	19	23	6	16	13	10	20	3	26	17	7	27
Sol74	16	73	13	1	3	7	2	2	3	3	1	4	4	4	6	8	5	8	2	4	7	1	3	6	8	6	2	7	5	5	14	21	8	11	1	18	4	24	15	5	9	12	19	2	22	25	23	6	10	13	16	26	3	20	17	7	27
Sol75	16	73	13	1	1	7	2	2	3	3	3	4	4	4	6	5	8	8	2	6	6	1	3	4	2	7	8	5	5	7	14	24	8	11	1	18	21	4	15	5	9	12	2	19	22	25	3	16	10	13	23	20	6	26	27	7	17
Sol76	16	73	13	2	3	7	2	1	3	3	1	4	4	4	6	8	5	8	2	6	7	2	3	6	1	4	8	5	7	5	11	21	8	1	14	18	4	24	15	5	9	12	19	2	22	25	3	6	20	13	16	10	23	26	7	17	27
Sol77	16	73	13	2	3	7	2	1	3	3	1	4	4	4	6	5	8	8	2	2	7	1	3	6	8	4	6	7	5	5	1	18	8	11	14	21	4	24	15	5	9	12	2	19	22	25	20	6	10	13	16	26	23	3	17	7	27
Sol78	16	73	13	3	3	7	2	2	3	1	1	4	4	4	8	5	8	6	2	8	6	1	3	7	2	4	6	5	7	5	21	4	8	11	1	18	14	24	15	5	9	22	2	19	12	25	26	16	10	13	6	20	23	3	7	17	27

Sol79	16	73	13	1	1	7	2	2	3	3	3	5	4	4	6	4	8	8	2	3	7	6	6	1	2	4	8	5	7	5	14	24	8	11	1	18	21	4	2	15	5	12	9	19	22	25	13	6	16	3	10	20	23	26	7	17	27
Sol80	16	73	13	2	3	7	2	1	3	3	1	5	4	8	6	4	8	4	2	8	7	1	3	6	6	4	2	7	5	5	11	21	8	1	14	18	4	24	2	15	22	12	5	19	9	25	26	6	10	13	16	3	23	20	17	7	27
Sol81	16	73	13	1	1	7	2	2	3	3	3	4	4	6	4	5	8	8	2	8	7	1	3	6	2	4	6	7	5	5	14	24	8	11	1	18	21	4	15	5	12	9	2	19	22	25	26	6	10	13	16	20	23	3	17	27	7

Os resultados da instância 10x10 não serão apresentados devido às dimensões das soluções. E para as outras instâncias testadas não foram encontrados resultados ótimos.