UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO DE ENXAME DE ABELHAS PARA RESOLUÇÃO DO PROBLEMA DA PROGRAMAÇÃO DA PRODUÇÃO *JOB SHOP* FLEXÍVEL MULTIOBJETIVO

RAFAEL FRANCISCO VIANA SANCHES

ORIENTADOR: PROF. DR. EDILSON REIS RODRIGUES KATO

São Carlos – SP

Fevereiro/2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO DE ENXAME DE ABELHAS PARA RESOLUÇÃO DO PROBLEMA DA PROGRAMAÇÃO DA PRODUÇÃO *JOB SHOP* FLEXÍVEL MULTIOBJETIVO

RAFAEL FRANCISCO VIANA SANCHES

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: Prof. Dr. Edilson Reis Rodrigues Kato

São Carlos – SP

Fevereiro/2017

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 Rafael Franscisco Viana Sanches, realizada em 14/02/2017:

Prof. Dr. Edilson Reis Rodrigues Kato
UFSCar

Profa. Dra. Veronica Oliveira de Carvalho
UNESP

Prof. Dr. Roberto Hideaki Tsunaki
USP

Dedico este trabalho a minha mãe Scheila e ao meu pai Odésio.

AGRADECIMENTOS

A minha mãe Scheila e ao meu pai Odésio por todo apoio e incentivo que me deram durante essa jornada da minha vida. Sem eles nada disso seria possível e também quero agradecer as minhas irmãs Elisângela e Elisandra por sempre estarem do meu lado me apoiando e me incentivando.

A minha noiva Cristiane por todo apoio, amor, carinho, motivação e por compreender minha fase de ausência.

Ao meu orientador o professor Dr. Edilson R. R. Kato, pelo auxílio e discussões compartilhando seu conhecimento comigo, dando o suporte necessário para a realização desse trabalho.

Ao professor Dr. Roberto Tsunaki pelas suas sugestões que contribuíram para o desenvolvimento desse trabalho.

Aos meus amigos e amigas do laboratório de pesquisa: Maykon, Marcela, Rogers, Vinícius, Marcos, Gabriel, Denis, Diego S., Bruna, Monique, Flavio R., Flavio M, Cleverson, Thales, Prof. Dr. Orides, Breno e Pedro.

Aos meus amigos e amigas do Departamento de Computação da UFSCar: Odair, Tiago Jesus, Bento, Abade, Cleiton Silva, Diego, Carlos, José Roberto, Roussian, Pablo Bottom, Pablo Bizzi, Fernando, Renato, Durelli, João Moreira, Suzane, Carol, Francielle, Kathiani, Marcelo, Amir, Felipe, Eduardo, Cédrick, Renata G., Elias, Guido, Steve, Alfredo, Augusto e Ivan.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão de bolsa de estudos durante o período de realização dessa dissertação.

Enfim, aos amigos que fiz no mestrado e todos aqueles que estão direta e indiretamente ligados que os nomes não estão aqui presentes.

“Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento."

Albert Einstein

RESUMO

A atividade de programação da produção é considerada como uma das atividades mais complexas no gerenciamento da produção. Essa atividade faz parte da classe de problemas NP-Difícil encontrados na área da ciência da computação, ou seja, aqueles problemas que não podem ser solucionados deterministicamente em tempo polinomial. Além disso, a complexidade dessa atividade pode aumentar de acordo com as restrições impostas a cada sistema/problema de programação. Nesta pesquisa, estuda-se o problema de programação da produção *Job Shop Flexível* (JSF). Esse problema é considerado como uma extensão do problema de programação *Job Shop*. No JSF, deve-se programar um grupo de *jobs* (i.e., produtos, itens, parte de um item) formados por um conjunto de operações e cada operação é processada por um recurso (i.e., máquina) que pertence a um grupo de recursos que possuam mesmas caraterísticas funcionais (e.g., cortar, lixar, pintar). Esse problema é caracterizado em dois sub-problemas, sendo eles, a atividade de roteamento e de sequenciamento. O roteamento implica em definir qual recurso irá processar uma determinada operação. O sequenciamento é a ordem em que cada operação será processada em um recurso. Por meio da programação estabelecida objetiva-se nessa pesquisa, otimizar multicritérios de desempenho, sendo eles: *makespan* (i.e., tempo gasto para produzir um conjunto de *jobs*), tempo de processamento gasto no recurso que trabalhou por mais tempo e tempo total de produção. Para alcançar os objetivos supracitados é proposto nessa pesquisa uma abordagem híbrida de enxame de abelhas. Nessa abordagem, utiliza-se dois métodos auxiliares para tratar os sub-problemas supracitados, sendo eles: operador genético de mutação para realizar a atividade de roteamento e para a atividade de sequenciamento é proposto um método adaptativo de estruturas de vizinhança. Para tratar a multiobjetividade do problema, propõe-se o método dominância de Pareto. Resultados experimentais obtidos por meio de *benchmarks* comumente usados comprovam a eficácia e a superioridade da abordagem proposta quando comparada com outras abordagens também aplicadas ao problema estudado.

Palavras-chave: programação da produção, JSF, multicritério, enxame de abelhas, *makespan*, mutação, estruturas de vizinhança, dominância de Pareto

ABSTRACT

The production scheduling activity is considered as one of the most complex activities in production management. This activity is part of the class of NP-Hard problems found in the area of computer science, that is, those problems that can not be solved deterministically in polynomial time. In addition, the complexity of this activity may increase according to the constraints imposed on each programming system/problem. In this research, the problem of programming of production the Flexible Job Shop (JSF) is studied. This problem is considered an extension of the Job Shop programming problem. In JSF, a group of jobs (i.e., products, items, part of an item) formed by a set of operations and each operation must be programmed by a resource (i.e., machine) that belongs to a group of resources that have the same functional characteristics (e.g., cut, sanding, painting). This problem is characterized in two sub-problems being routing and sequencing activity. Routing involves determining which resource will process a given operation. Sequencing is the order in which each operation will be processed on a resource. Through established programming, the objective of this research is to optimize performance multicriteria: the makespan (i.e., time spent to produce a set of jobs), processing time spent on the resource that worked by more time and total production time. In order to reach the objectives mentioned above, a hybrid swarm approach is proposed in this research. In this approach, two auxiliary methods are used to treat the abovementioned sub-problems: genetic operator of mutation to perform the routing activity and for the sequencing activity, an adaptive method of neighborhood structures is proposed. In order to deal with the multiobjectivity of the problem, we propose the Pareto dominance method. Experimental results obtained through commonly used benchmarks prove the efficacy and superiority of the proposed approach when compared to other approaches also applied to the problem studied.

Keywords: production scheduling, JSF, multicriteria, bee swarm, makespan, mutation, neighborhood structures, Pareto dominance

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Encontrar boas soluções para problemas de programação da produção é algo considerado de grande importância para a indústria, uma vez que a taxa de produção e os custos de uma organização dependam das programações realizadas para controlar o fluxo de trabalho diário do ambiente de produção (WANG et al., 2010).

A fim de aumentar a sua taxa de produção e reduzir custos, as organizações vem tornando seus sistemas de produção cada vez mais flexíveis (WANG et al., 2010). Contudo, encontra-se nesses sistemas o problema de programação da produção *Job Shop* Flexível (JSF). No JSF visa-se a programação de operações em um ambiente no qual ocorre o compartilhamento de recursos (i.e., máquinas) entre as operações (e.g., cortar, lixar, pintar) dos *jobs* (i.e., produto, item, parte de um item) disponíveis no ambiente produtivo. Devido as suas características o JSF tornou-se alvo de diversos pesquisadores pelo motivo da sua importância para indústria (CHAUDHRY; KHAN, 2016).

Para Gao et al. (2015), Kacem, Hammadi e Borne (2002), Brandimarte (1993), o JSF é considerado como uma extensão dos sistemas *Job Shop* (JS). Segundo os autores citados, um sistema JS é caracterizado por possuir um conjunto de *jobs* onde, cada *job* é formado por um conjunto de operações pré-definidas pelo gestor da produção que são processadas cada uma por um recurso que faz parte de um conjunto de recursos no ambiente de produção. Além das características mencionadas as operações dos *jobs* também podem ser processadas em ordens distintas e somente por um recurso.

O problema de programação em sistemas JS pode ser facilmente explanado conceitualmente, mas computacionalmente, esse problema apresenta alto custo computacional e se enquadra nos problemas da classe *Non-Deterministic Polynomial Time* (NP-Difícil), ou seja, aqueles na área da ciência da computação que não podem ser solucionados deterministicamente em

tempo polinomial (GAREY; JOHNSON; SETHI, 1976).

Além das complexidades herdadas do JS, nos sistemas JSF cada operação dos *jobs* pode ser processada por um conjunto de recursos, por isso, explana-se a ideia de utilizar recursos compartilhados com as mesmas características para processar a mesma operação. Nesse âmbito, a programação de operações em sistemas JSF pode ser considerada ainda mais complexa, umas vez que seja necessária a definição de qual recurso utilizar para processar uma operação (i.e., atividade de roteamento); além disso, faz-se necessária a atividade de sequenciar as operações nas máquinas previamente definidas (i.e., atividade de sequenciamento).

Em sistemas JSF a programação realizada visa a otimização de algum índice/critério de desempenho. Isso ocorre com a intenção de minimizar tempo, custos de produção e recursos humanos relacionados ao chão de fábrica (e.g., índice *makespan*, significa o tempo de produção gasto para produzir um grupo de *jobs* nos recursos disponíveis no chão de fábrica).

Além do *makespan* outros índices de desempenho podem ser utilizados para mensurar a programação no chão de fábrica (e.g., *maximum workload* e *total workload* que representam respectivamente o tempo de trabalho máximo isolado da máquina que mais trabalhou e o tempo total de todas as máquinas do chão de fábrica).

Os índices de desempenho utilizados para mensurar a programação da produção podem variar de acordo com cada organização e também serem analisados de forma isolada (i.e., mono-objetivo) ou em conjunto (i.e., multiobjetivo). Quando trabalha-se com um conjuntos de índices a complexidade da programação tende-se a tornar-se ainda mais complexa e cada vez mais próxima da realidade do chão de fábrica de uma organização.

Devido às características do JSF os autores Brandimarte (1993), Chaudhry e Khan (2016), Li, Pan e Liang (2010a), Kacem, Hammadi e Borne (2002), Najid, Dauzere-Peres e Zaidat (2002) sugerem o uso de abordagens heurísticas e meta-heurísticas como boas opções para tratar o JSF. Algoritmos Genéticos (WANG et al., 2010), Algoritmo de Enxame de Partículas (ZHANG et al., 2009), *Simulated Annealing* (NAJID; DAUZERE-PERES; ZAIDAT, 2002), Busca Tabu (BRANDIMARTE, 1993), Algoritmo de Colônia de Formigas (XING et al., 2010) e Algoritmos de Enxame de Abelhas (LI; PAN; TASGETIREN, 2014) (ABC, do inglês *Artificial Bee Colony algorithm*) são algumas das técnicas indicadas para tratar o JSF e que tem alcançado bons resultados.

Entre as técnicas supracitadas às abordagens de enxame assim como a técnica ABC tem sido foco de estudo por diversos pesquisadores. Em trabalhos como os de Li, Pan e Xie (2011), Gao et al. (2015), Wang et al. (2012a), Li, Pan e Tasgetiren (2014) abordagens ABC's propostas foram comparadas a outras técnicas. Nas comparações realizadas pelos autores mencionados,

as suas abordagens ABC apresentaram resultados tão bons quanto os das outras abordagens utilizadas para comparação em seus trabalhos.

Além disso, a meta-heurística ABC é composta por três tipos de abelhas fundamentais para seu funcionamento sendo elas as empregadas, espectadoras e exploradoras. A abordagem ABC também possui poucos parâmetros para serem configurados como o tamanho da população de abelhas (enxame), limite para abandonar uma solução que não melhorou e número máximo de iterações (forrageamento entre as abelhas).

Com objetivo de buscar melhores programações para problemas JSF por meio da abordagem ABC, neste trabalho é proposta uma nova abordagem ABC híbrida. Essa nova abordagem é sugerida com base nos trabalhos de Zhou et al. (2011), Li, Pan e Gao (2011), Li et al. (2011), Wang et al. (2012a, 2012b), Liu et al. (2013), Li, Pan e Tasgetiren (2014) e Gao et al. (2015) visando realizar as atividades de roteamento e sequenciamento de operações mesclando alguns métodos é ideias utilizados em comum nos trabalhos desses autores nas fases das abelhas empregadas, espectadoras e exploradora.

Por meio da abordagem proposta foram alcançados nesta pesquisa resultados bons e em alguns casos até melhores do que os de outras abordagens conhecidas na literatura analisadas por meio de *benchmarks* conhecidos nessa área.

1.1 Justificativa

Neste trabalho o problema escolhido foi definido pelo motivo da importância de encontrar boas programações (soluções) para problemas de programação da produção *Job Shop* Flexível (JSF). Por meio da programação realizada no chão de fábrica é possível reduzir diversos custos para as organizações o que tende a ser algo de grande importância para a indústria e também uma motivação para a realização dessa pesquisa.

Tratar problemas NP-Difícil é considerado algo bastante complexo e tal complexidade ocorre devido a esses problemas terem a característica de não determinismo. O JSF é um problema que se enquadra nessa categoria. Para tratar o JSF alguns autores como Kacem, Hammadi e Borne (2002), Brandimarte (1993), Chaudhry e Khan (2016) indicam o uso de meta-heurísticas devido essas abordagens encontrarem boas soluções em tempo considerável.

Devido a capacidade de abordagens meta-heurísticas para tratar problemas NP-Difícil, nesse trabalho foi definida a abordagem ABC para tratar o problema JSF. Essa abordagem foi adotada devido as características do ABC e as do JSF. O ABC é uma técnica que trata a questão

de realizar busca local e global no espaço de busca do problema estudado. O ABC diferente de outras abordagens oferece a possibilidade de reforçar a busca global que visa tentar encontrar soluções consideradas ótimas globais para o problema de otimização estudado nesse trabalho.

Pelo motivo da possibilidade de reforçar a busca global realizada pelo ABC espera-se que a abordagem proposta nesse trabalho seja capaz de escapar de soluções ótimas locais e consiga encontrar soluções ótimas globais para o problema estudado nesse trabalho.

Para que a abordagem proposta consiga fugir de pontos ótimos locais no espaço de busca do problema, foi adotado nesse trabalho o operador genético de mutação. Esse método tem a função de gerar uma pequena alteração na solução encontrada pelo ABC mas especificamente na parte que se refere a definição da máquina utilizada para processar uma operação. A escolha desse método para realizar a tarefa de roteamento no JSF foi definida com base nos trabalhos de Zhou et al. (2011), Li, Pan e Gao (2011), Wang et al. (2012a, 2012b), Liu et al. (2013) e também devido aos resultados gerados por esses trabalhos.

Outra tarefa importante no JSF é a de sequenciamento das operações dos *jobs* nas máquinas. Para realizar a tarefa de sequenciamento foi definido nesse trabalho um método que junta características dos trabalhos de Li, Pan e Tasgetiren (2014), Gao et al. (2015). A definição de qual a melhor sequencia para despachar as operações nas máquinas no chão de fábrica é uma tarefa bastante difícil (GAREY; JOHNSON; SETHI, 1976). A ideia do método proposto para a atividade de sequenciamento junta diferentes estruturas de vizinhança em um vetor e a escolha dessas estruturas ocorre por meio de uma roleta probabilística. A escolha por utilizar a roleta evita a interferência humana na definição da estrutura de vizinhança e se auto adapta tendendo a estrutura que apresentar melhor qualidade ao realizar o sequenciamento das operações.

Além dos métodos para tratar os sub-problemas de roteamento e sequenciamento, é importante definir o método de seleção utilizado pelo ABC e a forma de comparar/mensurar as soluções obtidas pela abordagem ABC proposta nesse trabalho.

Nesse trabalho é proposto o método de seleção por torneio. Essa definição é feita com base nos trabalhos de Li, Pan e Xie (2011), Li et al. (2011), Li, Pan e Gao (2011), Wang et al. (2012b, 2013), Li, Pan e Tasgetiren (2014), Gao et al. (2015). Um dos pontos fortes desse método está na capacidade de evoluir a população de abelhas de forma mais justa isso, não evoluindo somente as soluções consideradas melhores do mesmo modo que ocorre no método de seleção por roleta. Desse modo a população de abelhas tende a evoluir como um todo e não somente as melhores soluções.

Com relação a forma de avaliar as soluções nesse trabalho, utiliza-se o método dominância

de Pareto. Nesse método diferentemente da técnica de soma pondera, não é necessário definir pesos para os índices de desempenho utilizados. Um dos fatores que levaram a escolha do método de Pareto é que nesse trabalho utiliza-se múltiplos objetivos. Assim como nos trabalhos de Li, Pan e Gao (2011), Li, Pan e Xie (2011), Li et al. (2011), Wang et al. (2012b), Liu et al. (2013), Li, Pan e Tasgetiren (2014) observa-se que o uso desse método fornece também a possibilidade de recuperar mais de uma solução sejam elas locais ou globais o que permite que a decisão da solução utilizada fique a cargo do gestor da produção a definição da programação que harmonize melhor com os objetivos da sua organização.

1.2 Objetivos

Esse trabalho tem como objetivo principal a otimização (i.e., minimização) de múltiplos índices/critérios de desempenho (i.e., multiobjetivos) no problema de programação JSF.

1.2.1 Objetivos específicos

- Melhorar o tempo do critério de desempenho *Makespan*;
- Reduzir o tempo de produção gasto por cada máquina de forma individual (i.e., *Maximum Workload*);
- *Total Workload* (Tempo total de produção dos recursos): Diminuir a somatória do tempo de trabalho dos recursos (máquinas) como um todo.

1.3 Organização do Trabalho

No capítulo 2 são apresentados os conceitos e fundamentos básicos referentes a pesquisa realizada. No capítulo 3 é apresentado uma revisão da literatura referente ao uso de abordagens ABC utilizadas para tratar o problema de programação da produção de tipo JSF. No capítulo 4 é apresentado a abordagem proposta para tratar o problema de programação da produção de tipo JSF mencionado anteriormente. No capítulo 5 são ilustradas as comparações, gráficos e análises obtidas por meio desse trabalho. Por fim, no capítulo 6, são apresentadas as considerações finais obtidas sobre a abordagem proposta e também os possíveis trabalhos futuros a partir dessa pesquisa.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, será apresentada a fundamentação teórica necessária para o entendimento do trabalho realizado. Para tal, serão apresentados os seguintes assuntos: planejamento e controle da produção na seção 2.1, otimização multiobjetivo na seção 2.2 e meta-heurística Algoritmo de Enxame de Abelhas (ABC, do inglês: *Artificial Bee Colony Algorithm*) artificiais na seção 2.3.

Na seção 2.1, são apresentados os conceitos e fundamentos relacionados a: atividade de planejamento e controle da produção, os problemas mais comuns de programação estudados na literatura, o problema de programação tratado nesta pesquisa e os critérios de desempenho na manufatura.

Na seção 2.2, são apresentados os conceitos, os fundamentos e os termos relacionados a otimização multiobjetivo. Nesta seção, são descritas as terminologias mais comuns sobre otimização e o uso de múltiplos objetivos sejam eles de maximização ou minimização. Além disso, são apresentados também os métodos mais comumente usados para tratar problemas de otimização multiobjetivo sendo eles a soma ponderada e dominância de Pareto.

Por conseguinte, estuda-se na seção 2.3 a meta-heurística ABC. Nesta seção, é abordada a base biológica do algoritmo com base no comportamento das abelhas na natureza, o que possibilitou a geração dos algoritmos baseados nesses insetos. Além disso, é explanado também o passo a passo desse algoritmo de forma geral, destacando cada etapa para sua construção.

A seguir, serão apresentados os assuntos supracitados acima para compreensão do trabalho.

2.1 Planejamento e Controle da Produção

Em uma organização, geralmente o departamento de Planejamento e Controle da Produção (PCP), atua como o responsável por coordenar e destinar os recursos produtivos das empresas. O PCP é quem determina, o que, quanto, como, onde, quando e quem irá produzir, ou seja, ele é quem administra os recursos produtivos de forma a atender os planos estabelecidos pelas organizações nos níveis: estratégico, tático e operacional (TUBINO, 2009; SLACK; CHAMBERS; JOHNSTON, 2009).

No nível estratégico (longo prazo) o departamento PCP é responsável pela formulação do Planejamento Estratégico da Produção (PEP). Já no nível tático (médio prazo) tal departamento, deve elaborar o Planejamento-Mestre da Produção (PMP) e no nível operacional (curto prazo) o PCP deve preparar a programação da produção (TUBINO, 2009; SLACK; CHAMBERS; JOHNSTON, 2009). Na Figura 2.1 a seguir, são ilustrados esses níveis de planejamento desenvolvidos pelo PCP.

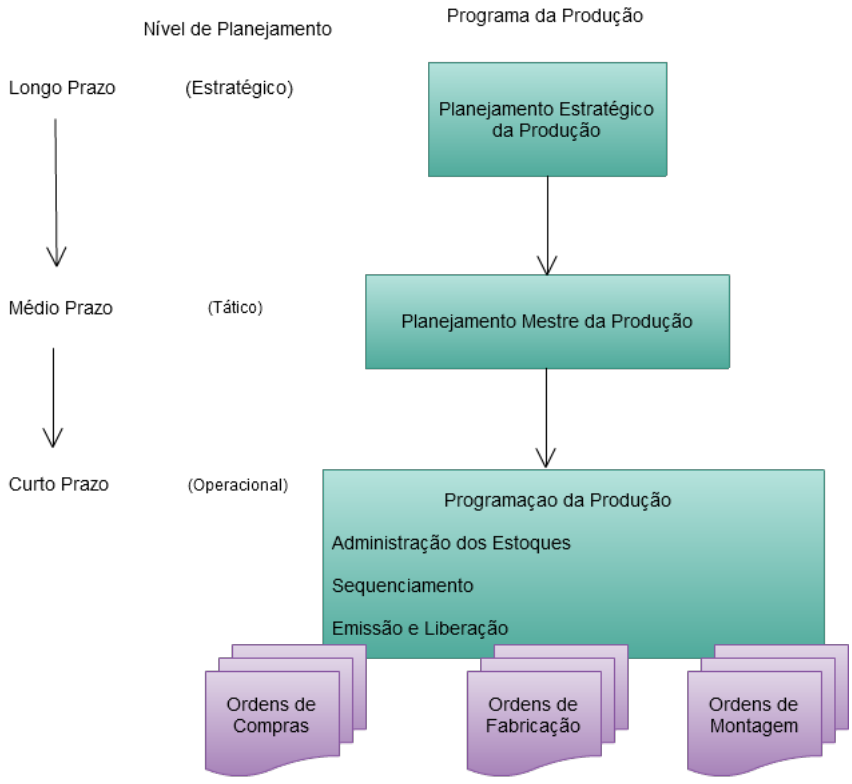

Figura 2.1: Níveis de planejamento desenvolvidos pelo departamento de PCP adaptado de Tubino (2009)

Em nível de PEP, o departamento de PCP busca maximizar os resultados das operações e minimizar os riscos nas tomadas de decisão. O impacto das decisões executadas no PEP, são de longo prazo e afetam a natureza e as características das empresas no sentido de garantir o

atendimento de sua missão. Por meio do PEP, gera-se o plano de produção que segundo Frazier e Gaither (2005), ressaltam ser um mapa daquilo que a função de produção deve fazer se quiser que suas estratégias de negócios sejam realizadas.

Após o planejamento estratégico da produção, o próximo passo no processo de planejamento e controle da produção é realizar o PMP. No PMP, ocorre o desmembramento do plano estratégico de longo prazo em planos específicos de produtos acabados (bens ou serviços) para médio prazo, direcionando as etapas de programação e execução das atividades operacionais (montagem, fabricação e compras), ou seja, faz a conexão entre o planejamento estratégico (plano de produção) e as atividades operacionais, através do plano mestre da produção gerado no PMP (TUBINO, 2009).

O planejamento e controle de curto prazo, consistem no sequenciamento, na programação e no controle da produção. A partir do PMP e com base em registros de estoque, o PCP define quanto e quando comprar, fabricar ou montar cada item necessário à composição dos produtos acabados (TUBINO, 2009). A atividade de programação segundo Arnold (1999), é fazer com que os prazos de entrega da organização sejam cumpridos e que os recursos produtivos sejam utilizados da melhor forma possível durante a execução desse processo.

Para que o processo supracitado ocorra como se deseja, o departamento de PCP deve estabelecer as cargas para os centros de trabalho, garantindo a disponibilidade de materiais, ferramentas, pessoal, informações e programar as datas de início e fim para cada pedido (ARNOLD, 1999). Ainda sobre a atividade de programação, existem dois conceitos muito importantes ligados as atividades de curto prazo no nível operacional. Tais conceitos, são conhecidos como “empurrar a produção" e “puxar a produção".

Para Tubino (2009), o conceito de empurrar a produção, representa atender periodicamente o PMP por meio de um programa de produção completo, de compra de matéria-prima à montagem do produto acabado, e transmiti-lo aos setores responsáveis através da emissão de ordens de compra, fabricação e montagem. Em relação ao conceito de puxar a produção, Tubino (2009) elucida que não se deve produzir até que o cliente (interno ou externo) de seu processo solicite a produção de determinado item.

As atividades de programação, dentro do sistema de empurrar a produção, procuram atender o PMP, por meio da administração de estoques, do sequenciamento, e da emissão e liberação de ordens de produção. Para o controle dessas atividades de programação, no sistema de produção puxada, costuma-se utilizar o sistema *Kanban* (TUBINO, 2009; SLACK; CHAMBERS; JOHNSTON, 2009).

A atividade de administração dos estoques supracitada, é responsável pelo planejamento e controle do estoque das matérias-primas até os produtos acabados entregue aos clientes. Essa atividade é parte do planejamento da produção e devido a isso, ela é considerada em cada nível de planejamento (ARNOLD, 1999).

A atividade de sequenciamento, refere-se à definição das prioridades das ordens de produção nas quais, as atividades devem ocorrer para atingir os seus objetivos e a programação consiste em distribuir no tempo especificado pré-determinadamente essas atividades, seguindo o sequenciamento definido e as restrições impostas à planta/layout de produção no chão de fábrica adotado (e.g., *Flow-Shop* e ou *Job-Shop*) (CORRÊA, 2006).

Para Slack, Chambers e Johnston (2009), a atividade de programação é considerada uma das mais complexas ao administrar a produção. Dentre os motivos que conduzem tal complexidade, Slack, Chambers e Johnston (2009) mencionam o compartilhamento de recursos (e.g., máquinas) para a execução das operações referentes a produção de algum produto. Além disso, quanto mais operações e recursos tiverem a necessidade de ser programados, mais complexa torna-se essa atividade.

A apresentação da programação realizada para o chão de fábrica, geralmente é ilustrada por meio do gráfico de Gantt. O gráfico de Gantt é um diagrama geralmente utilizado para visualizar as programações definidas pelo PCP, nele, são ilustradas as operações definidas para serem processadas nos recursos produtivos com seu tempo de duração (i.e., inicio e fim) (TUBINO, 2009; SLACK; CHAMBERS; JOHNSTON, 2009). Por meio desse gráfico, é possível analisar o tempo necessário (*makespan*) para concluir um grupo de *jobs*(e.g., produtos). Além disso, pode-se ainda por meio deste, analisar a carga de trabalho alocada aos recursos entre outras analises possíveis.

Na Figura 2.2 é ilustrado um exemplo de Gantt. Neste exemplo, ocorre a programação de um sistema de produção do tipo *Job-Shop* que possui 3 *jobs* (i.e, item, *job* ou produto)(J1,J2 e J3), 3 recursos (i.e., máquinas)(M1,M2 e M3) e cada *job* é composto por 3 operações cada. Na Figura 2.2, as operações são representadas pela letra *O* seguida pelo *job* e a operação a ser realizada desse *job*, por exemplo: na máquina M2 a primeira operação realizada nessa máquina, pertence ao *job* J3. O tempo apresentado na Figura 2.2 é considerado em unidades de tempo. O tempo de *makespan* gasto nessa programação foi de 12 unidades de tempo. As barras mais escuras no gráfico, representam que determinada máquina ficou parada por algum determinado tempo.

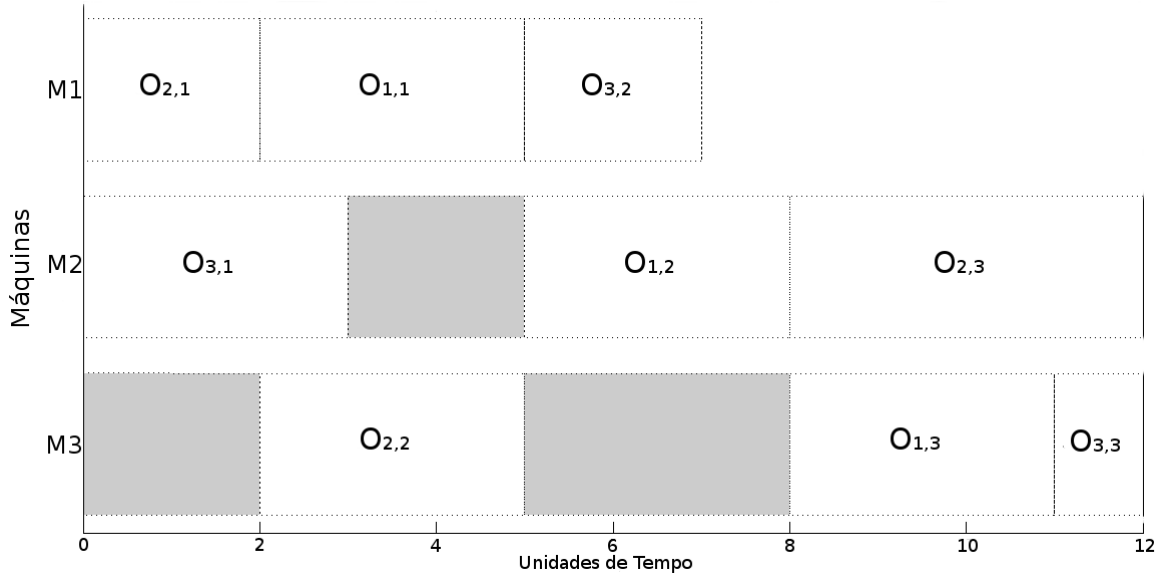

Figura 2.2: Exemplo de uma programação ilustrada por um gráfico de Gantt

Após a conclusão da programação e liberadas as ordens de fabricação, o processo de produção deve ser monitorado, comparando os resultados com o plano formulado. Segundo Frazier e Gaither (2005), as decisões de controle estão relacionadas com às atividades diárias executadas e com a qualidade dos produtos, serviços, custos de produção, gastos gerais e a manutenção dos recursos produtivos.

O controle da produção para Corrêa (2006), representa a coleta e analise de informações utilizadas para monitorar o desempenho efetivo e o esperado pela programação da produção definida. Uma das formais mais comuns de se medir o desempenho da programação definida, ocorre por meio de índices/critérios de desempenho (e.g., *makespan*).

2.1.1 Problemas de programação da produção

Na indústria de manufatura, os problemas de programação da produção geralmente estão relacionados a características e restrições impostas sobre o sistema produtivo utilizado no chão de fábrica. Nesse âmbito, considera-se a quantidade de recursos (i.e., máquinas), ordens de produção, as operações dos *jobs* e o fluxo das operações pelo layout do chão de fábrica (PINEDO, 2002; BRUCKER; BRUCKER, 2007).

Com base nas considerações supracitadas sobre os problemas de programação da produção, Allahverdi et al. (2008) classificam os sistemas produtivos da seguinte maneira:

***Single Machine Scheduling*:** cada *job* possui apenas uma operação para ser executada por um

único recurso (i.e., máquina) produtivo.

***Parallel Machines Scheduling*:** cada *job* continua possuindo apenas uma operação a ser executada, mas existem *m* recursos (i.e., máquinas) idênticos atuando de modo paralelo, sendo que cada *job* pode ser processado por qualquer um dos *m* recursos disponíveis.

***Flow Shop Scheduling*:** para este cenário, há um conjunto de *m* recursos e *n jobs*, no qual as operações de todos os *jobs* apresentam a mesma sequência de produção nas máquinas.

***Flexible Flow Shop Scheduling*:** este cenário é considerado como uma extensão dos sistemas de tipo *flow shop* supracitado. O diferencial nesse tipo de sistema, é que a sequencia de produção das operações dos *jobs* passa a ser considerada como estágios, e dentro de cada estágio (i.e., operação) pode haver recursos paralelos que desempenham o mesmo tipo de tarefa. Neste caso, somente os recursos disponíveis naquele estágio podem executar determinada operação referente aos *jobs* sem alterar a sequencia de estágios pré-estabelecida.

***Job Shop Scheduling*:** diferentemente do problema *Flow Shop*, no cenário de sistemas de manufatura do tipo *job shop* os *jobs* podem possuir sequências de produção diferentes nas máquinas.

***Flexible Job Shop Scheduling*:** este cenário é considerado como uma extensão dos sistemas de tipo *Job Shop* supracitado. O diferencial nesse tipo de sistema, é que para cada operação referente aos *jobs*, um recurso pertencente a um conjunto de recursos pode ser adotado para processar cada operação.

***Open Shop Scheduling*:** este cenário de produção, pode ser interpretado como um problema totalmente diferente dos demais apresentados. Neste cenário as operações dos *jobs* não apresentam restrições referentes a ordenação de suas operações, neste caso, suas operações não possuem relações de precedência pré-estabelecidas podendo ser processadas em qualquer ordem.

Nesse trabalho, foi adotado para estudo o problema de programação da produção *Job Shop* Flexível (JSF) (do inglês: *Flexible Job Shop Scheduling*). Na seção seguinte, o JSF é apresentado com mais detalhes e de forma genérica seguindo as nomenclaturas mais comumente utilizadas na literatura para sua descrição/formulação.

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

O problema de programação da produção de tipo *Job Shop* Flexível (JSF), é considerado como um caso particular do problema de programação de tipo *Job Shop* (JS). O problema de programação JS, por característica, é considerado como sendo de difícil resolução. O JS segundo Garey, Johnson e Sethi (1976), é um problema combinatório discreto que pertence aos problemas da classe NP-Difícil, ou seja, aqueles que apresentam alta complexidade e custo computacional em sua resolução. Logo, deduz-se que, o problema JSF herda essas características e também é NP-Difícil por redução.

O problema de programação de tipo JS, é caracterizado e formulado da seguinte maneira:

- Há um conjunto finito de *n jobs* (J_i) onde, $i = 1, ..., n$;
- Há um conjunto finito de k recursos (i.e., máquinas (M_k)) onde, $k = 1, ..., m$;
- Cada *job* J_i, possui uma sequência de operações o de tamanho S_j onde $j = 1, ..., o$;
- Cada operação $O_{i,j}$ do *job* J_i, deve ser processada em uma máquina do conjunto M de máquinas. Nesse contexto, i representa um determinado *job* e j a j-ésima operação do *job* i;
- $t_{i,j,k}$ representa o tempo de processamento da operação $O_{i,j}$ no recurso M_k;
- C_i representa o tempo de conclusão do *job* i;
- W_k representa a carga de trabalho referente ao recurso M_k.

No JSF, as operações podem ser processadas alternativamente em mais de uma máquina. Cada operação é associada a um subconjunto de máquinas capazes de realizar seu processamento, o que torna o problema flexível. Cada operação deve ser processada isoladamente, sem interrupção, em uma única máquina do sub-conjunto de máquinas a ela associado. Devem ser determinadas as atribuições das operações às máquinas e as sequências de processamentos que otimizem algum critério de desempenho estabelecido (CHAUDHRY; KHAN, 2016).

A flexibilidade do JSF, faz com que um *job* possa ser processado por diferentes roteiros através das máquinas, aumentando muito o número de soluções factíveis do problema. A flexibilidade do problema, possibilita que a resolução seja feita em duas etapas distintas, cada uma delas abordando sub-problemas menos complexos. A primeira etapa é a atribuição de cada

operação a uma máquina habilitada. Essa fase é denominada “roteamento" por determinar os caminhos que os *jobs* percorrerão. A segunda etapa é o “sequenciamento" das operações em cada máquina. Ela equivale à resolução do JSF resultante das atribuições da etapa anterior (CHAUDHRY; KHAN, 2016).

Os autores Kacem, Hammadi e Borne (2002), definem que o JSF pode ser caracterizado em 2 tipos, sendo eles: totalmente flexível e parcialmente flexível.

***Job Shop* Totalmente Flexível (JSTF):** nesse caracterização do problema, considera-se que todas as operações podem ser processadas por todos os recursos (i.e., máquinas) disponíveis no chão de fábrica;

***Job Shop* Parcialmente Flexível (JSPF):** diferentemente do anterior, nessa caracterização feita por Kacem, Hammadi e Borne (2002), no mínimo 1 operação pode não ser realizada por pelo menos um dos recursos disponíveis no chão de fábrica.

2.1.3 Critérios de Desempenho na Manufatura

Os critérios utilizados para expressar e medir o desempenho de um processo são: qualidade, entrega, custo e flexibilidade (FILIPPINI, 1998). Para Krause, Pagell e Curkovic (2001), mesmo estes critérios tendo forte ligação a manufatura, eles não tem seu uso restrito a ela. Portanto, diversos setores de uma organização podem utilizá-los para estabelecer seus objetivos.

Os critérios supracitados são multidimensionais, assim, para obter seus atributos é preciso desmembrá-los em dimensões. As dimensões desses critérios de desempenho são: qualidade (conformidade com as especificações e capacidade de identificar e desdobrar as necessidades dos clientes), entrega (rapidez e confiabilidade), flexibilidade (volume, *mix* de produtos, entrega e desenvolvimento de novos produtos) e custo.

Baseado nas características mencionadas acima, é possível encontrar na literatura diversos indicadores de desempenho utilizados para mensurar a atividade de programação da produção realizada por meio do departamento de PCP de uma empresa.

Para esta pesquisa, os indicadores de desempenho adotados para estudo de mensurar a programação são: *Makespan* (C_m) tempo de processamento gasto para concluir um conjunto de *jobs*, *Maximum Workload* (W_m) tempo de produção máximo gasto na máquina que trabalhou por mais tempo no chão de fábrica e o *Total Workload* (W_t) somatório do tempo de trabalho dos recursos (máquinas) como um todo. A seguir, são descritos os critérios utilizados:

As equações 2.1, 2.2 e 2.3 a seguir, representam os respectivos indicadores de desempenho estudados nesse trabalho.

Makespan:

$$Minimizar\ C_m = max(C_i)\ onde,\ (1 \leq i \leq n); \tag{2.1}$$

Nesse objetivo, procura-se minimizar a carga de trabalho da máquina que trabalhou por mais tempo no chão de fábrica:

$$Minimizar\ W_m = max(W_k)\ onde,\ (1 \leq k \leq m); \tag{2.2}$$

Carga total de trabalho das máquinas:

$$Minimizar\ W_t = \sum_{k=1}^{m} W_k; \tag{2.3}$$

Segundo o trabalho realizado por Chaudhry e Khan (2016), esses são os critérios mais estudados em problemas que tratam multiobjetividade no problema de programação da produção de tipo JSF.

2.2 Otimização Multiobjetivo

Um problema de otimização multiobjetivo, é formado por um conjunto de funções objetivo a serem otimizadas (minimizar ou maximizar). Além de possuir diferentes funções objetivo, é necessário respeitar as restrições impostas sobre cada função para que a solução seja considerada factível ao problema multiobjetivo a ser otimizado (DEB, 2001). Segundo Deb (2001), Ringuest (2012), Collette e Siarry (2013), um problema de otimização multiobjetivo pode ser enunciado da seguinte forma:

- Tem-se $f_i(X)$ onde, (i=1, 2, 3, ..., q) e f representa um conjunto de funções objetivo a serem otimizadas e X significa os parâmetros a serem otimizados dessa função. Além disso, q representa a quantidade de funções objetivo a serem otimizadas;
- Sabe-se que X é formado por $(x1, x2, x3, ..., xD)$ variáveis de decisão onde, D representa a quantidade de variáveis a serem otimizadas;
- Sabe-se que para cada função $f_i(X)$ pode haver várias restrições $g(X)$ impostas para elas delimitando as soluções consideradas factíveis ao problema multiobjetivo;
- $X \in \Omega$ onde, Ω representa o espaço de busca do problema de otimização.

Com base nos trabalhos de Deb (2001), Ringuest (2012), Collette e Siarry (2013), os conceitos mais comuns a qualquer metodologia de otimização serão apresentados a seguir:

Função Objetivo: equação matemática que representa o que se deseja melhorar (minimizar ou maximizar). Tem como sinônimos: critério de otimização, função de custo ou ainda função de mérito (*fitness function*);

Parâmetros: correspondem às variáveis da função objetivo. São ajustados durante o processo de otimização visando obter a(s) solução(ões) ótima(s). Além disso, podem ser chamados de variáveis de otimização;

Espaço de busca: domínio (delimitado ou não) que contém os valores dos parâmetros. Corresponde ao espaço de soluções. A dimensão do espaço de busca é definida pelo número de parâmetros envolvidos nas soluções (por exemplo, se cada solução é formada por três parâmetros, o espaço de busca é tridimensional);

Espaço de objetivos: conjunto imagem do espaço de busca determinado por todos os valores possíveis das funções objetivo;

Restrições: especificações do problema que delimitam os espaços de parâmetros e que não permitem determinada faixa de valores nos objetivos e.g., para determinado problema, pode-se impor que abaixo de certo valor a solução não seja considerada válida;

Solução ótima local: solução encontrada no espaço de busca em uma determinada região na qual, esta é considera a melhor de todas as soluções descobertas naquela região;

Solução ótima global: diferente da solução ótima local, a solução ótima global é aquela na qual nenhuma outra solução consegue ser melhor que ela, ou seja, em todas as regiões do espaço de busca, essa solução é, melhor que qualquer outra independente de onde essa solução esteja no espaço de busca.

2.2.1 Métodos mais utilizados no tratamento de conflitos em problemas de otimização multiobjetivo do tipo *Job Shop* Flexível

Nesta seção serão apresentadas as técnicas mais usadas para tratar problemas de otimização multiobjetivo do tipo *Job Shop* Flexível. São elas: somatório de pesos (soma ponderada) e dominância por Pareto.

O método somatória de pesos, segundo Deb (2001), Ringuest (2012), Collette e Siarry (2013), é formado por uma "função objetivo" somando cada objetivo multiplicado por um peso.

Os pesos são fornecidos como parâmetros. A escolha dos pesos é um problema importante que depende da relevância de cada objetivo. É necessário realizar a normalização de cada função objetivo dado que os diferentes objetivos podem ter diferentes extremidades. Na equação 2.4 é apresentado abaixo um modelo exemplo dessa técnica.

$$minimizar\ F = \sum_{i=1}^{q} f_i(X).w_i\ \therefore\ 1 = \sum w_i \tag{2.4}$$

Na equação 2.4 o elemento w_i representa o peso associado a cada função objetivo f_i e, o total desses pesos deve respeitar a restrição imposta sobre eles apresentada na equação 2.4. A restrição aplicada sobre os pesos, implica que o somatório dos pesos deve ser igual a 1.

Para exemplificar a ideia da soma ponderada, imagine um cenário onde, deseja-se comprar um computador no qual, o mesmo tenha um preço barato e com bom desempenho. Pensando nesse cenário, o preço será chamado de x_1 e o desempenho da máquina de x_2, imagine um segundo resultado encontrado chamados respectivamente de y_1 e y_2. Na soma ponderada é necessário o uso de pesos em cada objetivo. O problema é que os pesos podem tendenciar o resultado da soma de acordo com o valor dado aos pesos.

Imagine que você esteja mais preocupado com a questão do preço da máquina, a soma ponderada, lhe dá a liberdade de tendenciar o resultado, dando mais atenção para esse quesito. Nesse caso, vamos assumir que X = (x_1.0,7 + x_2.0,3) e Y = (y_1.0,7 + y_2.0,3) respeitando a restrição imposta ao método da soma ponderada que implica que a soma dos pesos não pode ultrapassar o valor igual 1. Depois de multiplicar os objetivos pelos respectivos pesos fictícios adotados, é necessário realizar a soma de ambos. O resultado gerado por essa soma, possibilita que os resultados encontrados durante essa pesquisa possam ser mensurados e distinguir por meio dessa soma o que é bom ou ruim, por exemplo: (X $<$ Y) ou (X $>$ Y).

O método dominância de Pareto, tem por característica comparar duas soluções factíveis de um problema de otimização multiobjetivo qualquer. Para exemplificar esta ideia, suponha o mesmo problema de exemplo utilizado no método de soma ponderada apresentado na seção anterior. Imagine agora, os mesmo objetivos contidos em X e Y só que sem os pesos. Então, tem-se X = {x_1,x_2 } e Y = {y_1,y_2 }. Quando se fala em dominância de Pareto, diz-se que uma solução domina outra. Nesse caso, vamos supor que a solução X domina Y.

Para dizer que X domina Y (DEB, 2001; RINGUEST, 2012; COLLETTE; SIARRY, 2013):

1. As variáveis de decisão que compõem a solução X, devem ser no mínimo iguais as variáveis de decisão encontradas em Y;

2. Na solução X, deve haver pelo menos uma variável de decisão com valor superior a variável equivalente presente na solução Y;

3. O terceiro caso que pode ocorrer, X é totalmente superior a Y em todas variáveis de decisão que a compõem;

A ideia de dominância, separa as soluções factíveis do problema de otimização multiobjetivo encontradas em dois grupos: soluções não dominadas (boas) e dominadas (ruins). As soluções não dominadas, são aquelas no espaço de busca que apresentam melhor qualidade em relação as demais. Além disso, a região do espaço de busca que separa as soluções não dominadas das dominadas é conhecida como fronteira de Pareto. Essa fronteira, é como o limite entre as melhores soluções e as piores para um problema de otimização multiobjetivo qualquer. Portanto, uma solução razoável para um problema multiobjetivo é uma solução que não seja dominada por qualquer outra solução. Melhor ainda, é ter como resultado o conjunto das soluções não dominadas encontradas.

2.3 Meta-heurística: Algoritmo de Enxame de Abelhas Artificiais

O comportamento das abelhas na natureza, serviu como inspiração para que diversos pesquisadores pudessem desenvolver algoritmos de otimização. Uma das características mais chamativas nesses insetos, está na sua capacidade de alocar tarefas de forma dinâmica e também por se auto-adaptarem em resposta a mudanças ambientais (KARABOGA; BASTURK, 2008).

Dentre as características mais promissoras das abelhas para o desenvolvimento de algoritmos estão: memória fotográfica, sistemas de sensores e navegação, tomada de decisão em grupo na escolha de novas fontes de alimento e na seleção de novos locais para as colmeias, manutenção da colônia, busca de provisões (alimentos), comunicação e reprodução.

Por motivo das abelhas apresentarem diversas características utilizadas no seu comportamento cotidiano, pesquisadores tem desenvolvido diversos algoritmos baseados nos comportamentos sociais desses insetos. Com base nos exames de abelhas, duas linhas básicas de desenvolvimento tem sido utilizadas para criação dessas abordagens: acasalamento e busca por alimentos.

Na linha de desenvolvimento com foco no acasalamento das abelhas, tem-se como base os trabalhos iniciais desenvolvidos por: Abbass (2001a, 2001b, 2002). No âmbito dos algoritmos com base na busca por alimentos, há diversas versões de enxames de abelhas sugeridas, assim

como: *Artificial Bee Colony* (ABC) algoritmo (KARABOGA; BASTURK, 2007, 2008), Algoritmo de Abelhas Virtuais (YANG, 2005), *Bee Colony Optimization* (BCO) (TEODOROVIĆ; DELL'ORCO, 2005), *BeeHive* (WEDDE; FAROOQ; ZHANG, 2004), Otimização por Nuvem de Abelhas (DRIAS; SADEG; YAHI, 2005) e o Algoritmo de Abelhas (PHAM et al., 2006).

Dentre as abordagens de enxame de abelhas mencionadas, Karaboga et al. (2014) apontam que a abordagem ABC é a que tem ganhado mais a atenção dos pesquisadores nas ultimas décadas. Na Figura 2.3, é ilustrado os resultados do estudo realizado pelos autores no qual, demonstra-se a atenção dada à abordagem ABC com relação as publicações realizadas utilizando essa meta-heurística em diferentes áreas de pesquisa. Na Figura 2.4 é apresentado os números de publicações das abordagens de enxame de abelhas através dos anos realizada por Karaboga et al. (2014) referentes a Figura 2.3.

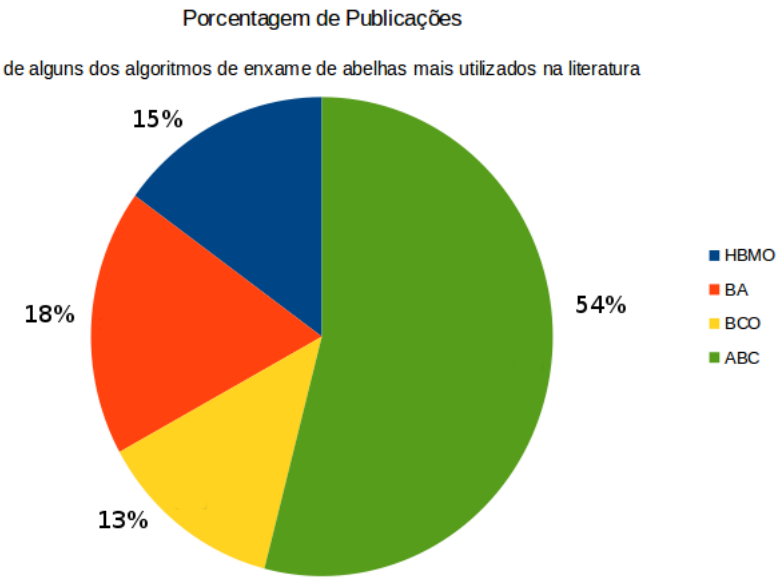

Figura 2.3: Porcentagem de publicações referentes a quantidade de publicações realizadas por meio das abordagens *Honney-Bees Mating Optimization* (HBMO), *Bees Algorithm* (BA), BCO e ABC adaptada de (KARABOGA et al., 2014)

Na Figura 2.3, os 54% representam a quantidade de publicações utilizando o ABC nas seguintes áreas de aplicação: redes neurais, engenharia (industrial, mecânica, elétrica, eletrônica, controle, civil e de software), processamento de imagens, mineração de dados, redes sensoriais e estrutura de proteínas.

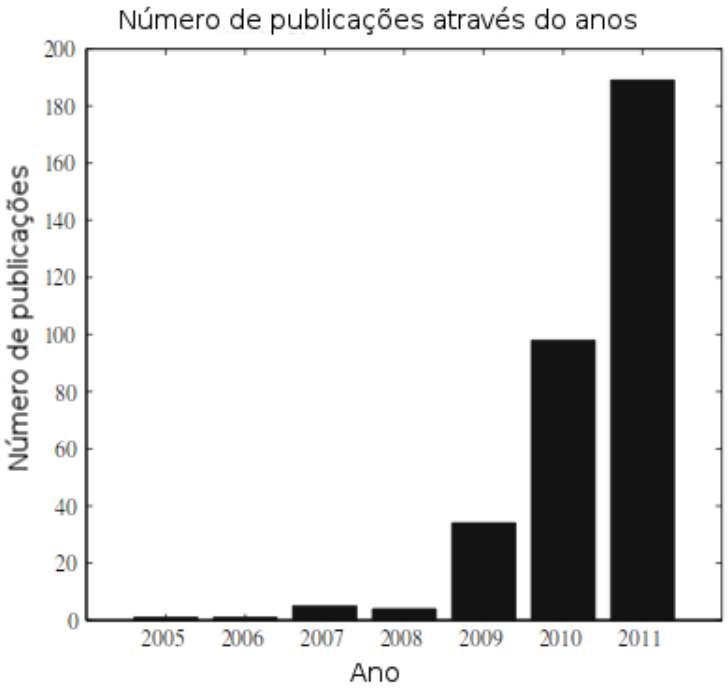

Figura 2.4: Gráfico das publicações das abordagens de enxame de abelhas publicadas através do anos adaptada de (KARABOGA et al., 2014)

2.3.1 Base Biológica

Em uma colônia de abelhas, existem três tipos diferentes desses insetos, são eles: rainha(s), zangões e as operárias. A função da rainha na colônia, consiste na reprodução de ovos que podem ou não ser fertilizados. Os zangões, são os machos da colônia, ou seja, tem a função de fertilizar uma nova rainha. As abelhas macho da colônia, são geradas a partir dos ovos não fertilizados e eles tem como função amplificar o genoma de suas mães e também, eles tem a função de habilitar geneticamente as abelhas fêmeas a agirem como machos (ABBASS, 2002). As abelhas operárias, tem como função dar manutenção na colônia, cuidar da alimentação dos indivíduos da colmeia, da limpeza, da estocagem de suprimentos dentre outras ações como por exemplo cuidar da procura por alimentos ao redor da colmeia. As abelhas rainhas e as operárias, são geradas a partir de ovos fertilizados e recebem alimentações diferentes após tornarem-se larvas.

Baseado no comportamento social dentro e fora da colmeia, duas ações bastante importantes relacionadas ao comportamento das abelhas são: o acasalamento (voo nupcial) e a procura por alimentos.

O voo de acasalamento, ou voo nupcial, é uma dança iniciada pela abelha rainha, que logo depois, torna-se um voo no qual passa a ser seguida pelos zangões. O ato de acasalamento, acon-

tece durante o voo. Geralmente, a rainha tende a acasalar com sete a vinte zangões (ADAMS et al., 1972). Um zangão no processo de acasalamento, acasala com uma rainha probabilisticamente, isso, de acordo com a velocidade da rainha e a adequação entre ele e ela. O esperma obtido do voo, é armazenado em uma espermateca, que é uma especie de repositório onde fica armazenada as informações genéticas das abelhas. Nesse âmbito, uma rainha pode permanecer fertilizada por dois anos ou mais. Toda vez que a rainha põe ovos, ela recupera uma mistura do material da espermateca aleatoriamente para os ovos fertilizados. Quando os espermas armazenados acabam, a rainha passa a por ovos não fertilizados.

Na procura por alimento, uma abelha, ao encontrar uma flor, armazena o néctar no seu estômago e volta para a colmeia. Depois de preencher os favos (células) de mel vazios na colmeia, ela compartilha as informações sobre a sua descoberta com outras abelhas. Para isso, a abelha realizada uma especie de dança na qual, de acordo com seus movimentos ela indica localização (distância e direção) e quantidade de néctar desse novo local.

A tomada de decisão quanto a novos locais para o ninho, acontece de forma conjunta. Nesse âmbito, as abelhas exploradoras, voam a procura de novas fontes de alimentos de forma aleatória ao redor da colmeia. Quando uma nova fonte de alimento é encontrada, se a qualidade dessa fonte for considerada boa, outras abelhas tendem a utilizá-la também. Quando uma fonte de alimento começa a apresentar perda de qualidade ou diminuição da quantidade de néctar, as abelhas tendem a abandonar essa fonte e partem para o princípio da exploração novamente em busca de novas fontes de alimento.

2.3.2 Algoritmo de Colônia de Abelhas Artificiais (ABC)

Inspirado pelo comportamento inteligente de forrageamento (i.e., busca por alimento) dos enxames de abelhas produtoras de mel, o algoritmo de colônia de abelhas (ABC)(do inglês, *Artificial Bee Colony Algorithm*) é uma meta-heurística baseada em população (KARABOGA et al., 2014). O ABC, foi proposto inicialmente por Karaboga (2005) para solucionar problemas de otimização contínua (KARABOGA, 2005; KARABOGA; BASTURK, 2007, 2008; KARABOGA, 2009; KARABOGA; AKAY, 2009; KARABOGA et al., 2014).

O ABC, é composto por três tipos de abelhas, são elas: empregadas, espectadoras e a exploradora. A função da abelha empregada, é a de explorar fontes de alimentos em sua vizinhança. A abelha que fica na colmeia esperando para tomar a decisão de qual fonte de alimento escolher para explorar é conhecida como espectadora. A abelha que sai a procura de novas fontes de alimentos de forma aleatória é a exploradora.

Em um problema de otimização qualquer, uma fonte de alimento referente ao ABC, representa uma possível solução para o problema a ser otimizado. A qualidade da solução ou *fitness* corresponde a quantidade de néctar associado a uma fonte de alimento.

A seguir, é apresentada uma descrição mais detalhada do ABC encorporando as características supracitadas sobre essa meta-heurística com base nos trabalhos de: Karaboga (2005), Karaboga e Basturk (2007, 2008), Karaboga (2009), Karaboga e Akay (2009), Karaboga et al. (2014).

Para o melhor entendimento desse algoritmo será utilizado nessa etapa o problema do Caixeiro Viajante (CV) para a explicação e definições encontradas no ABC.

O CV é um problema clássico de otimização bastante conhecido na área da ciência da computação e também um dos problemas mais complexos de ser solucionado computacionalmente em termos de tempo de execução. Essa complexidade no CV é dada de acordo com as restrições do problema e pela quantidade de cidades envolvidas no mesmo.

O problema do CV representa a ideia de um CV que tem de visitar um conjunto de cidades sem repetir nenhuma delas de modo que ele retorne a cidade inicial por meio da menor distância possível.

Na Tabela 2.1 é ilustrado um problema do CV com 5 cidades (A, B, C, D e E) com suas respectivas distâncias de uma para outra. No CV a Tabela 2.1 representa uma matriz de distâncias referênciada pela letra *D*. Em *D* são armazenadas as distâncias entre essas cidades na forma de uma matriz.

Tabela 2.1: Problema do CV com 5 cidades tabela de distâncias - D

	A	B	C	D	E
A	0	5	2	7	8
B	5	0	5	4	4
C	2	5	0	7	3
D	7	4	7	0	3
E	8	4	3	3	0

Com o problema do CV descrito e ilustrado agora vem a parte referente ao ABC. A primeira coisa a definir no ABC é a modelagem da solução referente ao problema estudado. Nesse caso a solução para o CV seria uma lista com a sequência das cidades visitadas pelo CV de modo que a última cidade visitada seja a mesma que ele iniciou seu trajeto. Na Figura 2.5 é ilustrada a modelagem adotada para esse exemplo e ela também representa uma possível solução utilizada para resolver o CV. Essa sequência de cidades modelada na Figura 2.5 representam no ABC uma fonte de alimento ou seja, uma possível solução para o CV.

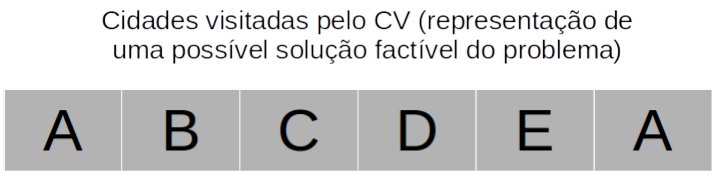

Figura 2.5: Modelagem da solução para o problema do CV

Com o problema e a modelagem da soluções definidos o próximo passo é o entendimento das variáveis de decisão/configuração (i.e., parâmetros) do ABC e os seus respectivos significados.

No ABC controla-se o tamanho da população i.e, a quantidade de fontes de alimento representadas pelas abelhas empregadas e espectadoras ou seja, o Número de Fontes de Alimentos (NFA). Além do tamanho da população existe a variável Limite e cada fonte de alimento (i.e., solução) tem a sua respectiva variável limite relacionada a ela. A variável Limite representa a quantidade de tentativas que uma abelha emprega ou espectadora tem para abandonar essa solução e buscar uma nova. Por último, o ABC tem de controlar a quantidade de iterações realizadas no ABC e esse parâmetro é conhecido como o Número Máximo de Ciclos por Forrageamento (NMCF).

Um detalhe importante sobre o ABC é que a quantidade de abelhas empregadas e espectadoras são iguais a metade do tamanho da população (MP). Essa relação é definida por Karaboga (2005) com base na troca de funções dentro de uma colmeia e também é considerada uma variável de controle no ABC.

O ABC com base nas informações já mencionadas até esse ponto terá então uma população que vai ter tamanho igual a (i = 1 até MP) e cada solução será composta por n cidades onde representa-se cada cidade por (j = 1 até n). Então, temos uma população representada por $X_{i,j}$ (i.e., soluções/fontes de alimentos) e Limite$_i$ relacionado a cada solução X_i (i.e., fonte de alimento).

No Algoritmo 1 ocorre a representação do fluxo de trabalho do ABC (i.e., pseudo-código) com suas respectivas características. Nessa representação na linha 3 ocorre a etapa de inicialização das soluções no caso isso pode ocorrer por meio de alguma regra especifica do problema em questão ou pode ser de modo randômico.

Nas linhas 4 e 5 por padrão os valores de *fitness* (i.e., qualidade da solução) e limite relacionados as soluções são iniciados com valor igual a zero. Na linha 6 a função *DefinirMelhorSolução* tem a função de retornar para o ABC a melhor sequência de cidades (solução) e o valor

de *fitness* dessa solução referentes a população inicial criada.

Na linha 9 é apresentado o laço de repetição referente a quantidade de ciclos de forrageamento executadas no ABC.

Na fase das abelhas empregadas no Algoritmo 1 ocorre uma atualização nas soluções que tem o seu pseudo-código apresentado no Algoritmo 2. Nessa fase cada solução sofre uma atualização por meio de uma estrutura de vizinhança. Essa estrutura pode ser representada por qualquer método que altere a ordem das cidades na solução em questão (e.g., troca *swap, insert, reverse*, cruzamento, mutação ou outro método) mais detalhes desses métodos são detalhados na seção 2.4.

Depois de aplicado o método de atualização na solução corrente, uma nova solução é gerada e com ela um novo *fitness* também é gerado por meio dessa mudança. Com essa nova solução e o novo *fitness* criados uma comparação entre o valor do *fitness* da nova solução e o valor de *fitness* da solução antiga é realizado. Se o novo *fitness* for melhor que o *fitness* da antiga solução, a nova solução e substituída no lugar da antiga e o novo *fitness* e substituído no lugar do antigo *fitness* e essas informações são inseridas na população.

No caso da antiga solução conter melhor valor de *fitness* em relação a nova solução o valor de Limite dessa solução deve ser incrementado de modo que seja representada a não evolução dessa solução.

Ainda sobre as abelhas empregadas a função de fitness utilizada para mensurar o caminho percorrido pelo CV é representada pelo Algoritmo 3. Essa função por meio da Matriz de distâncias apresentada na Tabela 2.1 possibilita o calculo da distância gasta no trajeto realizado pelo CV fornecendo ao ABC um valor referente a cada solução gerada nesse Algoritmo.

Na fase das abelhas espectadoras em cada iteração uma solução da população é selecionada por um método de seleção que nesse exemplo é utilizada uma roleta probabilística que pode ser representada assim como na Equação 2.5. Essa roleta mensura a qualidade do *fitness* de cada solução da população e gera um valor probabilístico para cada solução.

$$P_i = \frac{Fitness_i}{\sum_{i=1}^{MP} Fitness_i} \tag{2.5}$$

Depois que são gerados os valores de probabilidade P_i um valor randômico dentro do intervalos de valores de P_i é gerado e esse valor tende a ir para posição de P_i que tenha melhor valor de *fitness* mas não é garantido que o valor com maior *fitness* seja escolhido. Com base nessa escolha probabilística uma solução da população é escolhida para sofrer atualização na

Algoritmo 1: ALGORITMO ABC ADAPTADO DE KARABOGA (2005)

Entrada: $NFA, limite, NMCF, n, D$
Saída: O melhor caminho de cidades encontrado pelo ABC

```
1  início
2      MP ← (NFA/2);
3      X[MP][n] ← inicializarPopulacao();
4      Fitness[MP] ← inicializarValorDeFitness();
5      Limite[MP] ← inicializarLimiteDasSolucoes();
6      MelhorSolucao[n], MelhorFitness = DefinirMelhorSolucao(X);
7      para cont1 ← 1 até NMCF faça
8          ***Fase das abelhas empregadas***
9          para cont2 ← 1 até MP faça
10             X[cont2], Fitness[cont2], Limite[cont2] ← atualizarSolucao(X[cont2],
                Fitness[cont2], Limite[cont2], D,n);
11         fim
12         ***Fase das abelhas espectadoras***
13         para cont3 ← 1 até MP faça
14             posicaoEscolhida ← aplicarRoletaProbabilistica(Fitness);
15             X[posicaoEscolhida], Fitness[posicaoEscolhida], Limite[posicaoEscolhida]
                ← atualizarSolucao(X[posicaoEscolhida], Fitness[posicaoEscolhida],
                Limite[posicaoEscolhida], D, n);
16         fim
17         MelhorSolucao[n], MelhorFitness = memorizarMelhorSolucao(X);
18         ***Fase da abelha exploradora***
19         posicaoPiorLimite ← indicePiorLimite(Limite);
20         se Limite[posicaoPiorLimite] ≥ limite então
21             X[posicaoPiorLimite] ← gerarNovaSolucao();
22             Fitness[posicaoPiorLimite] ← 0;
23         fim
24     fim
25 fim
26 retorna MelhorSolucao
```

sua vizinhança. Para esse exemplo a atualização realizada pela abelha espectadora é idêntica a que ocorre na fase das abelhas empregadas.

Nesse exemplo não está sendo utilizado nenhum método de busca global para as abelhas espectadoras mas, Karaboga (2005) em seu trabalho ilustra a possibilidade de aplicar na fase das abelhas espectadoras técnicas/métodos de busca em vizinhança que possam ajudar o ABC a escapar de soluções ótimas locais. Um exemplo de técnica com essa característica seria o operador genético de mutação que é um método usado em Algoritmos Genéticos que ajuda esse algoritmo a não ficar estagnado em soluções que não possibilitam a evolução das soluções.

Ainda sobre o Algoritmo 1 depois de realizada a fase das abelhas empregadas e especta-

Algoritmo 2: MÉTODO ATUALIZARSOLUCOES

```
Entrada: X, Fitness, Limite, D,n
Saída: X, Fitness e Limite
1  início
2      novaSolucao[n];
3      novoFitness ← 0;
4      novaSolucao ← aplicarEstruturaDeVizinhanca(X);
5      novoFitness ← calcularFitness(novaSolucao, D, n);
6      se novoFitness > Fitness então
7          X ← novaSolucao;
8          Fitness ← novoFitness;
9      fim
10     senão
11         Limite ← Limite+1;
12     fim
13 fim
14 retorna X,Fitness,Limite
```

Algoritmo 3: FUNÇÃO DE *fitness* CALCULARFITNESS DO ABC PARA O PROBLEMA DO CV

```
Entrada: X,D,n
Saída: A distância gasta pelo CV durante se percorrer o caminho tratado nesse método
1  início
2      para indice ← 1 até n − 1 faça
3          Fitness ← Fitness + D[X[indice]][X[indice+1]];
4      fim
5      Fitness ← Fitness + D[X[n]][X[1]];
6  fim
7  retorna Fitness
```

doras Karaboga (2005) aponta em seu trabalho que é necessário armazenar a melhor solução encontrada até o momento. Essa decisão ocorre devido a próxima etapa do ABC a fase da abelha exploradora.

Na fase da abelha exploradora os valores de Limite_i são analisados e o Limite_i que tiver alcançado o valor limite de tentativas de uma solução não evoluir será escolhido e a solução referente ao Limite_i deverá ser abandonada e uma nova solução será gerada e alocada em seu lugar na população de abelhas.

Na Figura 2.6 a seguir, é ilustrado o fluxograma do pseudo-código do ABC representado pelo Algoritmo 1 supracitado. Por meio dele, é possível observar mais facilmente a sequência de processos realizada pelo ABC durante a sua execução.

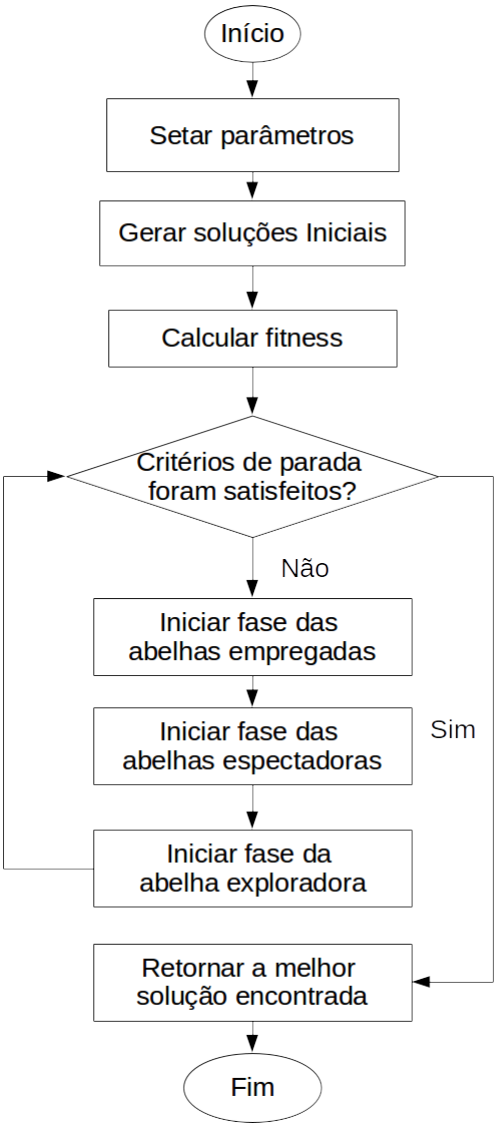

Figura 2.6: Fluxograma do algoritmo ABC adaptado de Karaboga (2005)

2.4 Estruturas de Vizinhança

Estruturas de vizinhança são métodos que visam alterar a vizinhança (i.e., variáveis de decisão de um problema de otimização) de uma solução corrente na tentativa de melhorá-la. Esses métodos geralmente são utilizados em técnicas heurísticas e meta-heurísticas que por natureza são métodos inspirados em algum fenômeno ou acontecimento do mundo real guiadas com foco em solucionar algum problema de otimização.

Na literatura alguns dos métodos de estruturas de vizinhança mais utilizados são troca *swap*, *insert* e *reverse*. Além desses métodos existem outros mais robustos que fazem a utilização de regras (i.e., restrições) como Busca Tabu (considerada como uma heurística), operadores de cruzamento e mutação.

Para entender melhor esses métodos a seguir será apresentado um exemplo dessas estruturas de vizinhança aplicadas no problema do CV utilizado na seção anterior por meio da modelagem utilizada no ABC apresentado.

No método *swap* duas posições do vetor solução são definidas de modo randômico. Com as duas posições do vetor definidas, a próxima ação nesse método é a de trocar o conteúdo da primeira posição definida com o conteúdo da segunda posição. Para facilitar o entendimento desse processo na Figura 2.7 há um exemplo desse método aplicado ao problema do CV.

Na Figura 2.7 a posição 2 e a 4 do verto solução foram escolhidas randomicamente e com isso o conteúdo da posição 2 e da 4 são trocados. Essa troca de conteúdos das posições implica na solução final apresentada na Figura 2.7 abaixo da solução inicial onde, é representado por setas a ideias da troca *swap* realizada.

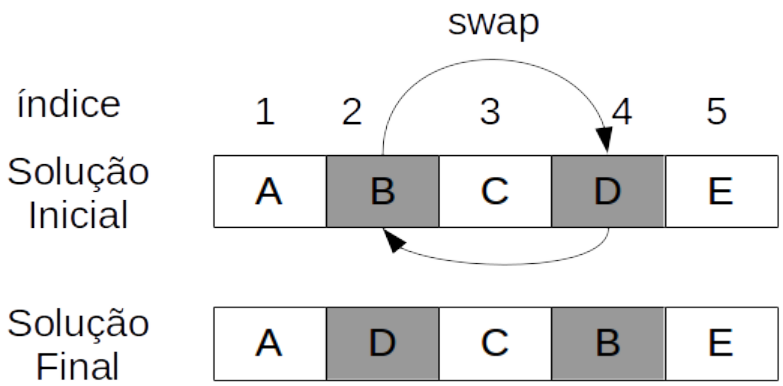

Figura 2.7: Exemplo de aplicação da estrutura de vizinhança *SWAP*

No método *insert* duas posições são escolhidas de forma randômica. Na Figura 2.8 os índices 2 e 4 são escolhidos para este exemplo. Nesse método o conteúdo do índice menor é deslocado para a posição do (índice maior + 1) ou seja, o conteúdo do índice menor é inserido na posição seguinte a do índice maior. Com a aplicação do método *insert*, na Figura 2.8 é possível observar na solução final o deslocamento supracitado e com uma cor mais escura as cidades que tiveram que ser deslocadas para que a regra do *insert* fosse realizada do modo correto.

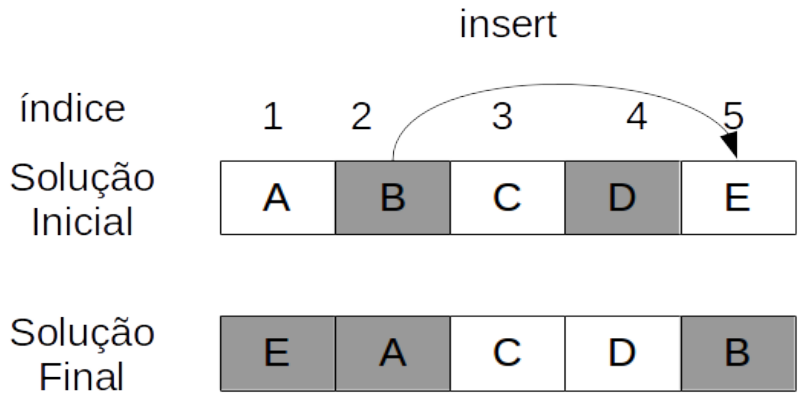

Figura 2.8: Exemplo de aplicação da estrutura de vizinhança *INSERT*

O método *reverse* é mais simples de ser explicado em relação aos outros dois métodos supracitados. Nesse método o vetor com as cidades do CV é invertido na ordem contrária ou seja, a ordem das cidades é invertida. Um exemplo de aplicação desse método é ilustrado na Figura 2.9. Na Figura 2.9 é possível observar que os conteúdos das cidades foi invertido do índice maior para o menor e com isso na solução final demonstra-se o resultado gerado nessa inversão de conteúdo dos índices referentes as posições das cidades no vetor solução.

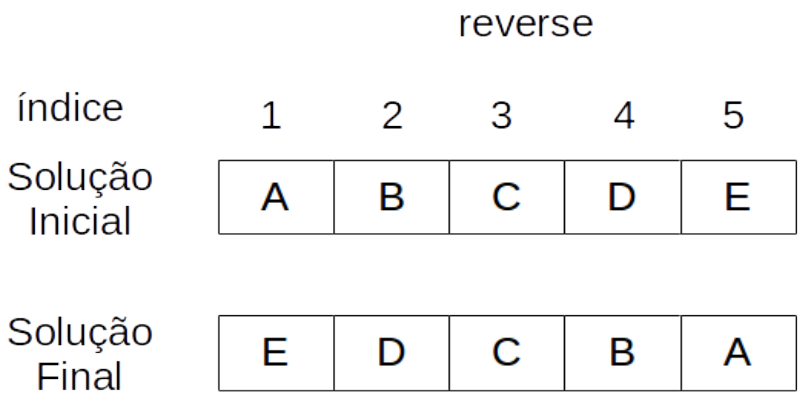

Figura 2.9: Exemplo de aplicação da estrutura de vizinhança *REVERSE*

A busca tabu como já foi supracitado é considerada como uma heurística/meta-heurística então por esse motivo não será descrita aqui mas, para maiores detalhes sobre esse método consulte (GLOVER; KOCHENBERGER, 2006).

Em relação aos operadores genéticos é preciso descrever inicialmente a ideia desses métodos e a sua função em um algoritmo genético e o que esses operadores (estruturas de vizinhança) causam nas soluções de um problema de otimização.

Para falar sobres os operadores genéticos é primeiro importante entender o que é um al-

goritmo genético. Um Algoritmo Genético é uma meta-heurística baseada na ideia da Teoria da Evolução Natural desenvolvida por Darwin (HOLLAND, 1992). Os algoritmos genéticos são considerados como pertencentes a classe de algoritmos baseados na computação evolutiva que é um ramo da computação natural na qual aplica-se conceitos com base na natureza para soluções de problemas computacionais complexos (e.g., problemas de otimização).

Em um algoritmo genético o método de cruzamento funciona de maneira similar a reprodução dos seres humanos. Nesse processo características de dois progenitores (i.e., duas soluções) são cruzadas ou mescladas com alguma restrição de modo que um novo individuo seja criado (i.e., uma nova solução a partir dos progenitores).

Nos algoritmos genéticos as restrições de cruzamento mais comuns encontradas na literatura são as de ponto único, duplo e de cruzamento de pontos aleatórios. Na Figura 2.10 é apresentado um exemplo da aplicação do operador de cruzamento com a restrição de ponto único.

Na Figura 2.10 é possível observar que entre os índices 3 e 4 dos progenitores ocorre o ponto de corte único. Nesse exemplo as características do progenitor 1 do índice 1 ao 3 são recuperadas e inseridas no filho 1 e as características do progenitor 2 do índice 4 e 5 são passadas para o filho 1 também ou seja, um novo individuo/solução é criado a partir das características dos progenitores.

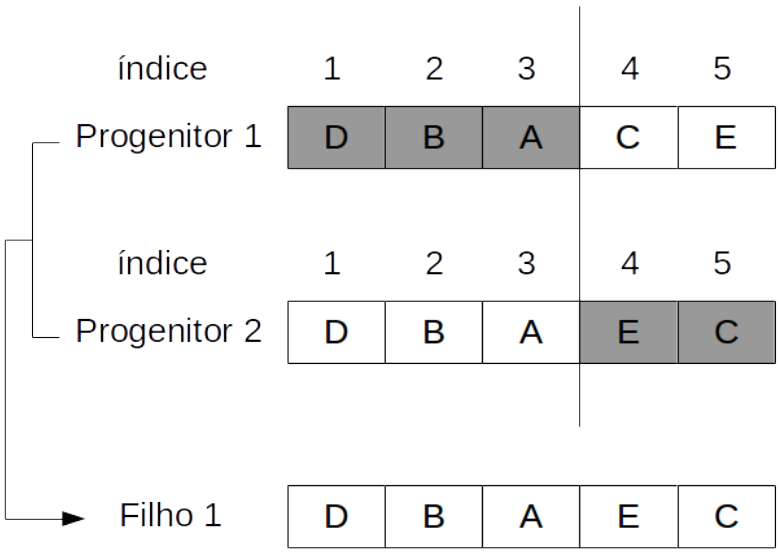

Figura 2.10: Exemplo de aplicação da estrutura de vizinhança operador genético de Cruzamento com restrição de ponto único

Vale ressaltar que os progenitores são escolhidos por meio de algum método de seleção antes de passarem pelo método de cruzamento. Esses métodos de seleção geralmente selecionam

as melhores soluções (e.g., roleta probabilística) de uma população então, a ideia do cruzamento é melhorar uma solução baseada em duas soluções já consideradas boas na população de um algoritmo genético.

O método de cruzamento por afetar uma parte maior do individuo (i.e., uma solução em algoritmos genéticos) geralmente pode fazer com que a solução corrente fique presa em pontos ótimos locais do problema com relação ao espaço de busca do problema. Para evitar que as soluções fiquem presas nesses pontos ótimos locais, em algoritmos genéticos logo após o uso do operador de cruzamento aplica-se o operador de mutação na solução corrente.

O operador de mutação tem por objetivo realizar modificações em determinadas propriedades genéticas de uma população de forma randômica. Devido a essas modificações a mutação torna-se importante uma vez que possibilita a população corrente recuperar características genéticas que não existiam ou eram encontradas em baixa porcentagem devido a alta modificação realizada pelo método de cruzamento.

Com tais características supracitadas é visto que o operador de mutação mostra-se indispensável em um algoritmo genético. Vale ressaltar que assim como na natureza a taxa de mutação deve acometer uma pequena parcela da população, pois caso a porcentagem seja elevada, os indivíduos gerados pouco se assemelharão aos seus pais; caso contrário, a diversidade da população estará comprometida.

Existem várias técnicas de mutação, dentre elas as mais comum são à aleatória i.e., dentro de um alfabeto válido (i.e., variáveis de decisão do problema), um valor é sorteado para substituir o que sofrerá a mutação e outra técnica é a de mutação por troca *n* onde, *n* pares de genes (i.e., posição do vetor/solução que matem alguma característica genética) são sorteados e logo após o sorteio os pares trocam valores entre si (e.g., similar a técnica *swap* supracitada).

Outra técnica de mutação utilizada na literatura é a troca em blocos. Nesse método uma parcela da solução é selecionada e as características genéticas dessa região são sorteadas de modo randômico. Na Figura 2.11 há um exemplo da aplicação desse tipo de mutação.

Na Figura 2.11 na solução inicial uma porcentagem pequena dessa solução é selecionada de forma randômica. Após selecionar as informações a sofrerem mutação, essas informações serão trocadas de forma randômica para garantir a diversidade genética dessa solução na população. Na solução final da Figura 2.11 fica claro a troca realizada nas posições definidas para sofrer a mutação e vale ressaltar que isso deve ocorrer em uma parcela pequena da solução assim como já foi supracitado com relação a essas mesmas características na natureza.

Outro detalhe importante sobre a mutação é que na troca de conteúdo dos genes, não precisa

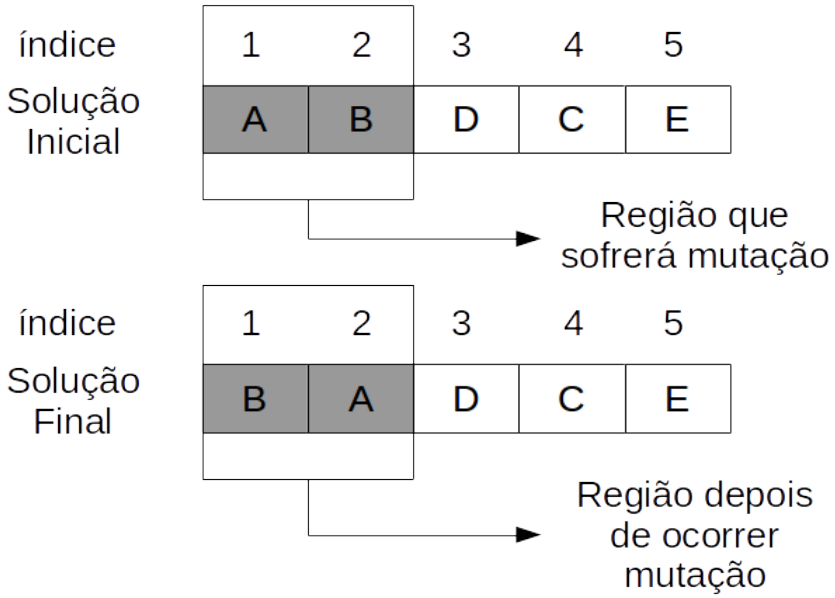

Figura 2.11: Exemplo de aplicação da estrutura de vizinhança operador genético de Mutação

ser necessariamente a informação genética de outro gene mas, pode ser alguma característica do alfabeto desse problema que ainda não se encontra na solução ou população corrente.

2.5 Considerações Finais

Por meio desse capítulo foram apresentados assuntos relacionados ao entendimento dessa pesquisa como a atividade de programação da produção de um modo geral onde englobou-se o ambiente produtivo e as características do chão de fábrica de acordo com o sistema de produção definido com base nas informações relacionadas a quantidade de *jobs*, máquinas, operações e fluxo das operações nas máquinas (i.e., roteiro de produção dos *jobs*) no chão de fábrica.

Ainda com relação a programação foi apresentada uma descrição mais detalhada do problema JSF estudado nesse trabalho e uma contextualização genérica sobre os índices de desempenho relacionados a forma de mensurar a qualidade da programação realizada no chão de fábrica em uma organização.

Depois de introduzido os termos e peculiaridades referentes a atividade de programação e também sobre o problema JSF, foram apresentados ainda nesse capítulo, uma ideia geral sobre a otimização de múltiplos objetivos e também dos métodos mais comumente usados na literatura para tratar decisões sobre conjuntos de objetivos conflitantes.

Com o fechamento dos assuntos supracitados o próximo assunto relevante para essa pesquisa foi o de compreender e introduzir o algoritmo de enxame de abelhas. Na seção onde ele é

explicado para facilitar a interpretação do mesmo é utilizado um problema do caixeiro viajante de modo que seja possível entender o passo a passo desse algoritmo aplicando as etapas das abelhas empregadas, espectadoras e a exploradora na resolução desse problema de exemplo.

O último assunto tratado nesse capítulo foi referente as estruturas de vizinhança que são métodos aplicados a vizinhança das soluções utilizadas no ABC na fase das abelhas empregadas e espectadoras. Nessa seção são apresentados as técnicas mais comuns referentes a estruturas de vizinhança para que o leitor possa compreender melhor qual a ideia desses métodos e o que eles geram quando aplicados nas soluções utilizadas por meta-heurísticas assim como o ABC.

Para concluir o capítulo, os assuntos tratados foram de grande importância para a compreensão dessa pesquisa devido a característica interdisciplinar desse trabalho no qual junta características da área de ciência da computação (i.e., inteligência artificial) e produção (i.e., pesquisa operacional).

Capítulo 3

REVISÃO DA LITERATURA

Os trabalhos dissertados na literatura com ênfase em ABC para tratar o problema de programação da produção em sistemas de manufatura do tipo JSF serão estudados e analisados de modo que seja possível entender os pontos fortes e fracos ligados a cada abordagem apresentada e os resultados gerados por cada uma.

Para a realização dessa pesquisa bibliográfica, as seguintes máquinas de busca foram utilizadas:

- *IEEE Xplore Digital Library*;
- *ACM Digital Library*;
- *ScienceDirec - Elservier*;
- *Springer Link*;
- *Scopus*;
- *Engineering Village (Ei Compendex)*;
- *Google Scholar*;

Para a execução das buscas nas bases de dados apresentadas anteriormente, os seguintes termos foram utilizados: *"Artificial Bee Colony"*, *"ABC"*, *"Flexible Job Shop Scheduling 'OR' Problem"*, *"FJSSP"* e *"FJSP"*. A seguir são apresentados os trabalhos correlatos a está pesquisa no período de 2011 até 2016.

3.1 Trabalhos dissertados no período de 2011

3.1.1 *An Effective Artificial Bee Colony Algorithm for Multi-objective Flexible Job-Shop Scheduling Problem*

Neste trabalho, Zhou et al. (2011), propuseram uma abordagem ABC para tratar o problema de programação da produção do tipo JSF multiobjetivo, com a intenção de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm), o tempo total de produção de todas as máquinas (Wt).

Na intenção de atingir seus objetivos, Zhou et al. (2011) utilizam em sua abordagem os seguintes processos e características:

- A modelagem para as soluções apresentadas neste trabalho, são construídas na forma de dois vetores no qual, um representa uma sequência de operações (i.e., problema do sequenciamento das operações), e o outro, as máquinas associadas a essas operações (i.e., problema do roteamento ou seja a alocação das máquinas nas operações).

- Para trabalhar com essa modelagem referente as soluções, é utilizado um esquema de decodificação chamado *left-shift*, esse esquema, realiza um deslocamento para a esquerda nas operações do vetor de operações tornando essas operações ativas ou aptas a serem processadas nas máquinas.

- Para calcular a aptidão (i.e., *fitness*) das soluções, é utilizada o método de soma ponderada. Esse método visa transformar as três funções de aptidão Cm,Wm e Wt em uma única função (F), transformando o problema em monoobjetivo facilitando a forma de mensurar e comparar as soluções geradas nesse trabalho.

- A geração das soluções iniciais neste trabalho, ocorre por meio de uma estratégia híbrida que utiliza uma mistura de regras de despacho. Para mais detalhes sobre essas regras consulte Zhou et al. (2011).

- Nesta abordagem, as abelhas empregadas usam o operador genético de cruzamento que é aplicado sobre os vetores de máquinas e de operações usados nesse trabalho como forma de representar uma possível solução factível do problema abordado. Depois que o operador de cruzamento é aplicado, a solução utilizada por essa abelha pode sofrer mutação no vetor de máquinas ou pode ir para a fase da abelha espectadora dependendo de uma escolha randômica. Para detalhes mais específicos sobre os operadores genéticos utilizados consulte (ZHOU et al., 2011).

- Na fase da abelha espectadora, é utilizado o método de seleção por roleta para selecionar as fontes de alimentos das abelhas empregadas. Depois de selecionar as fontes de alimentos da abelha empregada escolhida por meio da roleta, um método de busca local baseado na ideia do método de caminho crítico (ZHOU et al., 2011) é aplicado sobre o vetor de operações relacionado a abelha empregada escolhida para sofrer uma atualização em sua solução.

- Nesta abordagem, pode-se ter mais que uma abelha exploradora e a abelha que apresentar pior valor de aptidão na população (i.e., enxame de abelhas) durante um período definido de tentativas para evoluir, será escolhida para ser abandonada. Desse modo, a estratégia híbrida para gerar as soluções iniciais é utilizada para gerar está nova solução que irá ser adotada na população no lugar da solução que foi abandonada.

Para avaliar a abordagem ABC proposta, os autores utilizaram dois *benchmarks* conhecidos, um desenvolvido por Kacem, Hammadi e Borne (2002) e outro por Brandimarte (1993). A eficácia da abordagem ABC dos autores pode ser observada nas Tabelas 3.1 e 3.2 onde, para cada instância o primeiro valor corresponde a quantidade de *jobs* e o segundo a quantidade de máquinas desse problema. Na Tabela 3.1, Zhou et al. (2011) comparam sua abordagem com *Approach by Localization* (AL) com *Controlled Genetic Algorithm* (CGA), *Particle Swarm Optimization* (PSO) com *Tabu Search* (TS), *Hybrid Thin-Slot Algorithm* (HTSA) e a abordagem de Xing (XING; CHEN; YANG, 2009a). Na Tabela 3.2 a abordagem ABC foi comparada com HTSA e a abordagem de Xing. Na Tabela 3.2 a letra F significa o valor gerado pela soma ponderada aplicada para mensurar as soluções encontradas nesse trabalho.

Com relação a Tabela 3.1 na instância 4x5, a abordagem proposta pelos autores apresentou melhores resultados que as abordagens AL+CGA, PSO+TS e Xing. Na instância 8x8 o ABC de Zhou et al. (2011) obteve melhor resultado apenas em relação a abordagem AL+CGA. Na instância 10x7 o ABC de Zhou et al. (2011) foi comparado somente com Xing e HTSA devido as outras abordagens não apresentarem nenhum resultado em seus trabalhos para essa instância. Nessa instância os resultados foram semelhantes então não houve nenhuma mudança relevante entre os resultados nessa instância.

Na instância 10x10 o ABC proposto por Zhou et al. (2011) obteve melhores resultados que as abordagens AL+CGA, PSO+TS e Xing. Na instância 15x10 fica evidente também que a abordagem ABC de Zhou et al. (2011) obteve resultados melhores que os das abordagens AL+CGA, PSO+TS e Xing.

Na Tabela 3.2 a abordagem ABC de Zhou et al. (2011) teve melhores resultados que Xing

Tabela 3.1: Resultados das 5 instâncias de Kacem referentes a abordagem ABC de Zhou et al. (2011)

Instâncias	Objetivos	AL+CGA		PSO+TS		Xing		HTSA			ABC			
4x5	Cm	16		11		12		11	12		11	12		
	Wt	34		32		32		32	32		32	32		
	Wm	10		10		8		10	8		10	8		
8x8	Cm	15	16	14	15	14	15	14	15		14	15		
	Wt	79	75	77	76	77	76	77	75		77	75		
	Wm	13	13	12	12	12	12	12	12		12	12		
10x7	Cm					11	11	11	11		11	11		
	Wt					61	62	61	62		61	62		
	Wm					11	10	11	10		11	10		
10x10	Cm	7		7		7	8	7	7	8	7	7	8	8
	Wt	45		43		42	42	43	42	42	43	42	42	41
	Wm	5		6		6	5	5	6	5	5	6	5	7
15x10	Cm	23	24	11		11	11	11	11		11	11		
	Wt	95	91	93		91	93	91	93		91	93		
	Wm	11	11	11		11	11	11	10		11	10		

em todas as instâncias menos na instância br08. Em relação a abordagem HTSA o ABC dos autores teve melhores resultados nas instâncias br03, br04, br06, br07, br09 e br010.

Tabela 3.2: Resultados das 10 instâncias de BRdata referentes a abordagem ABC de Zhou et al. (2011)

	Xing				HTSA				ABC			
Instâncias	F	Cm	Wt	Wm	F	Cm	Wt	Wm	F	Cm	Wt	Wm
br01 (10x6)	48	42	162	42	45.75	40	167	36	45.75	40	167	36
br02 (10x6)	34.35	28	155	28	32.25	26	151	26	32.25	26	151	26
br03 (15x8)	236.4	204	852	204	236.4	204	852	204	236	204	850	204
br04 (15x8)	82.05	68	352	67	76.25	61	366	61	76.1	60	382	60
br05 (15x4)	203.25	177	702	177	197.75	172	687	172	197.75	172	687	172
br06 (10x15)	91.6	75	431	67	81.2	65	398	62	79.85	61	444	59
br07 (20x5)	178.35	150	717	150	167.75	140	695	140	166.7	139	693	139
br08 (20x10)	623.05	523	2524	523	623.05	523	2524	523	623.05	523	2524	523
br09 (20x10)	412.35	311	2374	299	407.85	310	2294	301	407.1	309	2301	299
br10 (20x15)	314.2	227	1989	221	305.35	214	2053	210	301.35	209	2065	206

Por meio dos resultados obtidos no trabalho dos autores Zhou et al. (2011) fica evidente a superioridade de sua abordagem ABC e também vale ressaltar que na maioria das instâncias o ABC proveu melhores resultados em uma quantidade maior de instâncias se comparado com as demais abordagens utilizadas nesse trabalho.

3.1.2 *Pareto-based discrete artificial bee colony algorithm for multiobjective flexible job shop scheduling problems*

Neste trabalho, Li, Pan e Gao (2011), desenvolveram a abordagem conhecida como *Pareto Discrete Artificial Bee Colony Algorithm* (P-DABC) para tratar o problema de programação da produção do tipo JSF multiobjetivo, com a intenção de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm) e o tempo total de produção de todas as máquinas (Wt).

Na intenção de atingir seus objetivos, Li, Pan e Gao (2011) utilizam em sua abordagem P-DABC os seguintes processos e características:

- A modelagem para as soluções neste trabalho, são apresentadas na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações.
- Para mensurar a aptidão das soluções encontradas no P-DABC, utiliza-se o método chamado fronteira de Pareto. Esse método, tem a função de comparar duas soluções de modo que uma deles tenha seus objetivos superiores ao outro.
- As soluções iniciais neste trabalho, são geradas de modo randômico para garantir diversidade na população gerada inicialmente.
- As melhores soluções do P-DABC são armazenadas em um arquivo chamado arquivo de Pareto. Esse arquivo, tem a finalidade de memorizar um número definido de soluções consideradas como melhores soluções encontradas durante a execução da abordagem dos autores.
- Para está abordagem, as abelhas empregadas aplicam nas suas soluções o operador genético de cruzamento para rotear as máquinas nas operações a serem realizadas.
- A abelha espectadora nesta abordagem, tem a função de selecionar uma solução das abelhas empregadas utilizando o método de seleção por torneio. No método por torneio deste trabalho, são escolhidas 3 soluções de modo randômico e entre estas soluções, a que conter melhor aptidão é selecionada para sofrer uma atualização. Neste caso, a solução selecionada irá passar por dois processos. No primeiro processo, uma posição do vetor de máquinas da solução selecionada é escolhido de forma randômica. Na posição escolhida, será sorteado uma das possíveis máquinas habilitadas para processar a operação escolhida desde que, essa nova máquina seja diferente da atual. No segundo processo,

o vetor de operações sofrerá uma atualização por meio dos métodos de inserção e *swap* entre as operações da solução.

- No P-DABC, a abelha exploradora não gera uma solução randômica como na versão original do ABC. Nesta abordagem, a abelha exploradora escolhe de forma randômica uma solução do arquivo de Pareto e aplica os métodos de inserção e *swap* diversas vezes sobre essa solução a fim de gerar uma nova solução com certo nível de qualidade.

Para avaliar a abordagem P-DABC, Li, Pan e Gao (2011) comparam sua abordagem com: AL+CGA (KACEM; HAMMADI; BORNE, 2002), PSO+TS, *Particle Swarm Optimization and Simulated Annealing* (PSO+SA)(XIA; WU, 2005) e *Artificial Immune Algorithm* (AIA) (BAGHERI et al., 2010). Na Tabela 3.3, ilustra-se os resultados gerados para comparação da abordagem P-DABC.

Na instância 4x5 a abordagem P-DABC obteve melhores resultados que as abordagens AL+CGA e PSO+TS, as outras abordagens nessa instância não apresentaram resultados. Na instância 8x8 o P-DABC obteve melhores resultados que as abordagens AL+CGA e PSO+SA. Em relação as demais abordagens os resultados foram semelhantes. Na instância 10x7 somente o P-DABC apresentou resultados então, não ouve comparação.

Tabela 3.3: Resultados das 5 instâncias de Kacem referentes a abordagem P-DABC de Li, Pan e Gao (2011)

Instâncias	Objetivos	AIA	AL+CGA		PSO+SA		PSO+TS		P-DABC		
4x5	Cm		16				11		11	12	13
	Wt		34				32		32	32	33
	Wm		10				10		10	8	7
8X8	Cm	14	15	16	15	16	14	15	14	15	16
	Wt	77	79	75	75	73	77	75	77	75	73
	Wm	12	13	13	12	13	12	12	12	12	13
10X7	Cm								12	11	12
	Wt								61	63	60
	Wm								11	11	12
10X10	Cm	7	7		7		7		8	7	8
	Wt	43	45		44		43		41	43	42
	Wm	5	5		6		6		7	5	5
15X10	Cm	11	23	24	12		11		12	11	
	Wt	93	95	91	91		93		91	93	
	Wm	11	11	11	11		11		11	11	

Na instância 10x10 é possível constatar que o P-DABC obteve melhores resultados que as demais abordagens nessa instância. Na instância 15x10 a abordagem P-DABC obteve melhores

resultados que as abordagens AL+CGA e PSO+SA. Em comparativo as demais abordagens usadas os resultados foram bastante semelhantes.

Neste trabalho, os autores Li, Pan e Gao (2011) concluem que sua abordagem é competitiva em relação as outras abordagens utilizadas para comparação. Por meio dos resultados apresentados, nota-se que as outras abordagens utilizadas para comparação alcançaram resultados muito próximos ou até mesmo iguais aos da abordagem sugerida pelos autores.

3.1.3 *Flexible Job Shop Scheduling Problems By A Hybrid Artificial Bee Colony Algorithm*

Neste trabalho, Li, Pan e Xie (2011), desenvolveram a abordagem conhecida como *Artificial Bee Colony Algorithm and Tabu Search* (ABC+TS) para tratar o problema de programação da produção do tipo JSF multiobjetivo, com a intenção de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm), o tempo total de produção de todas as máquinas (Wt) e o critério total *flow time* (Tf)(tempo total gasto por cada *job*).

Na intenção de atingir seus objetivos, Li, Pan e Xie (2011) utilizam em sua abordagem ABC+TS os seguintes processos e características:

- A modelagem para as soluções utilizada neste trabalho, é apresentada na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações.
- Para mensurar a aptidão das soluções encontradas no ABC+TS, é utilizado o método chamado fronteira de Pareto.
- Grande parte das soluções iniciais deste trabalho, são geradas por meio de uma estratégia híbrida que utiliza uma mistura de regras de despacho e as soluções restantes são geradas de modo randômico para garantir diversidade na população. Para mais detalhes sobre as regras utilizadas consulte Li, Pan e Liang (2010b).
- As melhores soluções do ABC+TS são armazenadas em um arquivo chamado arquivo de Pareto. Esse arquivo tem a mesma finalidade do arquivo de Pareto apresentado no trabalho anterior.
- No ABC+TS, as abelhas empregadas, aplicam nas suas soluções a técnica de busca local TS nos seus respectivos vetores que representam uma solução factível, i.e, a técnica TS é aplicada no vetor de máquinas e no vetor de operações.

- A abelha espectadora nesta abordagem, tem a função de selecionar uma solução das abelhas empregadas utilizando o método de seleção por torneio. No método por torneio deste trabalho, são escolhidas 2 soluções de modo randômico e entre estas soluções, a solução com melhor valor de aptidão é selecionada para sofrer uma atualização. Neste caso, a solução selecionada irá ser submetida ao mesmo processo de busca local utilizado na fase das abelhas empregadas por meio da técnica TS.

- Nesta abordagem, a abelha exploradora escolhe de forma randômica uma solução do arquivo de Pareto e aplica a busca local TS diversas vezes sobre a solução selecionada do arquivo de Pareto a fim de gerar uma nova solução com certo nível de qualidade.

Na Tabela 3.4 abaixo, a abordagem ABC+TS É comparada com as abordagens: AL+CGA, PSO+SA e PSO+TS e na Tabela 3.5 apresenta-se os resultados gerados pela abordagem ABC+TS em comparação a abordagem ABC dos autores sem o uso do método Busca Tabu. Ainda sobre a Tabela 3.5, ilustra-se por meio dela os resultados referentes ao objetivo Tf, que é um objetivo ainda pouco explorado na literatura se comparado a outros como o *makespan* por exemplo. Nas Tabelas 3.4 e 3.5 as ausências de valores para determinadas instâncias significa que não foram encontrados resultados para as mesmas.

Na Tabela 3.4 na instância 4x5 o ABC+TS apresentou melhores resultados em comparação a abordagem AL+CGA e PSO+TS. Na instância 8x8 o ABC+TS apresentou melhores resultados que as abordagens AL+CGA, PSO+SA. Na instância 10x7 não houve comparação. Na instância 10x10 o ABC+TS obteve melhores resultados que as abordagens PSO+SA e PSO+TS. Na instância 15x10 o resultado do ABC+TS foi melhor que as demais abordagens no objetivo Wm e em comparação a abordagem AL+CGA ele foi melhor em ambos objetivos Cm e Wm.

Tabela 3.4: Comparação de resultados da abordagem ABC+TS nas 5 instâncias de Kacem, Hammadi e Borne (2002)

Instâncias	Objetivos	AL+CGA		PSO+SA		PSO+TS	ABC+TS		
4x5	Cm	16				11	11	12	13
	Wm	10				10	9	8	7
8x8	Cm	15	16	15	167.75	14	14	15	16
	Wm	13	13	12	13	12	12	12	13
10x7	Cm						11	12	
	Wm						10	12	
10x10	Cm	7		7		7	7		
	Wm	5		6		6	5		
15x10	Cm	23	24	12		11	11		
	Wm	11	11	11		11	10		

Na Tabela 3.5 fica evidente a superioridade da abordagem ABC+TS em relação a abordagem ABC. Somente na instância 4x5 os resultados são todos semelhantes.

Tabela 3.5: Comparação de resultados usando o objetivo Tf nas instâncias de Kacem, Hammadi e Borne (2002)

	Instâncias									
	4x5		8x8		10x7		10x10		15x10	
Abordagens	Cm	Tf	Cm	Tf	Cm	Tf	Cm	Tf	Cm	Tf
ABC	11	37	14	96	11	66	7	45	12	102
	12	36	15	89	12	64	9	44	13	98
	14	35								
ABC+TS	11	37	14	96	11	66	7	45	11	96
	12	36	15	89	12	62	8	44	12	95
	14	35	19	87						

Neste trabalho, os autores Li, Pan e Liang (2010b) concluem que os resultados experimentais gerados por sua abordagem são superiores aos dos outros algoritmos utilizados para comparação, tanto na qualidade de pesquisa como na complexidade computacional gerada por eles.

3.1.4 *A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problems*

Neste trabalho, Li et al. (2011), desenvolveram a abordagem conhecida como *Hybrid Pareto Based Artificial Bee Colony Algorithm* (HPABC) para tratar o problema de programação da produção do tipo JSF multiobjetivo, com a intenção de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm) e o tempo total de produção de todas as máquinas (Wt).

Na intenção de atingir seus objetivos, Li et al. (2011) utilizam em sua abordagem HPABC os seguintes processos e características:

- A modelagem para as soluções neste trabalho são apresentadas na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações.
- Para mensurar a aptidão das soluções encontradas no HPABC, é utilizado o método chamado fronteira de Pareto.
- As soluções iniciais deste trabalho, são geradas de modo randômico para garantir diversidade na população.

- As melhores soluções do HPABC são armazenadas em um arquivo chamado arquivo de Pareto. Esse arquivo tem a mesma finalidade do arquivo de Pareto apresentado nos trabalhos anteriores que já utilizaram este método.

- No HPABC, as abelhas empregadas aplicam nas suas soluções dois operadores de busca local. O primeiro operador é aplicado no vetor de máquinas associadas as operações dos *jobs*. Neste primeiro operador, uma posição do vetor de máquinas é escolhida de modo randômico (i.e., *swap* troca de posições) ou por meio de alguma regra de despacho qualquer como por exemplo *Shortest Processing Time*(SPT) (FERNANDES; FILHO, 2010), deste modo, uma máquina diferente é escolhida para substituir a máquina atual que for selecionada pelo operador. O segundo operador é aplicado no vetor de operações no qual implica a ordem das operações nas máquinas disponíveis para processá-las. Neste segundo operador, utiliza-se os métodos de troca *insert* e *swap* para causar uma pequena alteração no vetor de operações. Para mais detalhes sobre os métodos de troca *insert* e *swap* consulte (LI et al., 2011).

- A abelha espectadora nesta abordagem, tem a função de selecionar uma solução das abelhas empregadas utilizando o método de seleção por torneio. No método por torneio deste trabalho, são escolhidas 3 soluções de modo randômico e entre estas soluções, a solução com melhor valor de aptidão é selecionada para sofrer uma atualização. Neste caso, a solução selecionada irá ser submetida aos mesmos processos de busca local utilizados na fase das abelhas empregadas.

- Na fase da abelha exploradora, não gera-se uma solução randômica como na versão original do ABC. Nesta abordagem, uma quantidade de 5% a 10% das soluções que não melhoraram poderão ser substituídas. Neste caso, metade dessa quantidade definida de soluções será substituída por soluções do arquivo de Pareto que serão escolhidas de forma randômica e então, será aplicado a elas os métodos de troca *insert* e *swap* nos vetores de operações associados a estas soluções que serão substituídas pelas soluções que não conseguiram melhorar. A outra metade das soluções escolhidas será gerada de forma randômica.

Para avaliar a abordagem proposta, os autores usaram as abordagens: AL+CGA, PSO+TS e PSO+SA para comparar sua abordagem HABC proposta. Na Tabela 3.6 é apresentado os resultados da comparação realizada entre as abordagens mencionadas e abordagem HABC.

Na Tabela 3.6 na instância 4x5 os resultados da abordagem HABC foram melhores que os da abordagem AL+CGA em relação a abordagem PSO+TS é possivel constatar que os resulta-

dos são similares mas o HABC apresenta mais opções se comparado a abordagem PSO+TS. Na instância 8x8 a abordagem HABC obteve melhores resultados somente em relação a abordagem AL+CGA e em relação as demais abordagens os resultados foram similares.

Na instância 10x7 não houve comparação mas na instância 10x10 os resultados foram bastante similares. Na instância 15x10 os resultados obtidos pela abordagem HABC foram melhores que os da abordagem AL+CGA.

Tabela 3.6: Resultados e comparação da abordagem HABC nas 5 instâncias de Kacem (KACEM; HAMMADI; BORNE, 2002) multiobjetivo

Instâncias	Objetivos	AL+CGA		PSO+TS		PSO+SA		HABC		
4x5	Cm	16		11				11	12	13
	Wt	34		32				32	32	33
	Wm	10		10				10	8	7
8x8	Cm	15	16	14	15	15	16	14	15	16
	Wt	79	75	77	76	75	73	77	75	73
	Wm	13	13	12	12	12	13	12	12	13
10x7	Cm							11	12	
	Wt							61	60	
	Wm							11	12	
10x10	Cm	7		7		7		8	7	8
	Wt	45		43		44		41	42	43
	Wm	5		6		6		7	6	5
15x10	Cm	23	24	11		12		12	11	
	Wt	95	91	93		91		91	93	
	Wm	11	11	11		11		11	11	

Neste trabalho, os autores Li et al. (2011) concluem que sua abordagem é competitiva em relação as utilizadas para comparação. Por meio dos resultados obtidos pela abordagem dos autores, nota-se que os resultados alcançados pelas outras abordagens são muito próximos ou até mesmo iguais.

3.2 Trabalhos dissertados no período de 2012

3.2.1 *An effective artificial bee colony algorithm for the flexible job-shop scheduling problem*

Neste trabalho, Wang et al. (2012a), desenvolveram a abordagem conhecida como *Effective Artificial Bee Colony Algorithm* (EABC) para tratar o problema de programação da produção do tipo JSF, com a intenção de minimizar o critério de desempenho *makespan*.

Na intenção de atingir seus objetivos, Wang et al. (2012a) utilizam em sua abordagem

EABC os seguintes processos e características:

- A modelagem para as soluções usadas neste trabalho são apresentadas na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações.

- As soluções iniciais deste trabalho, são geradas por meio de diferentes regras de despacho para a construção das soluções iniciais. Para mais detalhes sobre as regras usadas consulte Wang et al. (2012a).

- No EABC, as abelhas empregadas atuam inicialmente sobre o vetor de máquinas. Primeiro aplica-se 2 operadores genéticos de cruzamento (i.e, cruzamento de 2 pontos e cruzamento uniforme) que tem a função de modificar as máquinas atualmente definidas para as operações. A segunda atividade executada nesta fase é a aplicação do operador genético de cruzamento proposto pelos autores para atuar sobre o vetor de operações das soluções. O operador proposto pelos autores é chamado de *modified precedence operation crossover* (MPOX). Com a finalização das atividades que envolvem o operador de cruzamento, dada uma probabilidade de 50%, cada solução pode ser submetida a um processo de mutação definido pelos autores. Para maiores detalhes sobre esse método consulte (WANG et al., 2012a)

- A abelha espectadora nesta abordagem, tem a função de escolher as fontes de alimento das abelhas empregadas para sofrer atualização por meio do método seleção por roleta. A abelha espectadora utiliza um método de busca local baseado na teoria do método de caminho critico para atualizar a solução selecionada.

- Na fase da abelha exploradora, gera-se uma solução randômica como na versão original do ABC.

Para mensurar a abordagem EABC proposta pelos autores, duas comparações foram realizadas. Nessas comparações, foram utilizados os *benchmarks* fornecidos por Kacem, Hammadi e Borne (2002) e Brandimarte (1993). Em comparação a abordagem EABC foram usadas as abordagens: GENACE, PSO+TS, *Parallel Variable Nneighborhood Search* (PVNS), *Knowledge-Based Ant Colony Optimization Algorithm* (KABCO), *Tabu Search Algorithm With a Fast Public Critical Block Neighborhood Structure* (TSPCB), TS, *Genetic Algorithm* (GA), AL+CGA e *Learnable Genetic Architecture* (LEGA). As comparações realizadas são apresentadas nas Tabelas 3.7 e 3.8.

Na Tabela 3.7 a abordagem EABC na instância 4x5 apresenta melhor resultado que a abordagem AL+CGA. Na instância 8x8 o EABC é melhor que a abordagem AL+CGA e PSO+TS. Na instância 10x7 o EABC é melhor que as abordagens AL+CGA e GENACE. Na instância 10x10 todas abordagens utilizadas para comparação apresentam mesmo valor de *makespan*. Na instância 15x10 a abordagem EABC é melhor que as abordagens AL+CGA, GENACE, PSO+TS e PVNS.

Tabela 3.7: Comparação de resultados usando as 5 instâncias de Kacem, Hammadi e Borne (2002)

Instâncias	AL+CGA	GENACE	PSO+TS	PVNS	KABCO	TSPCB	EABC
4x5	16	11			11	11	11
8x8	15		15	14	14	14	14
10x7	15	12			11	11	11
10x10	7	7	7	7	7	7	7
15x10	23	12	12	12	11	11	11

Na Tabela 3.8 na instância mk01 a abordagem EABC é melhor que a abordagem TS mas é inferior a abordagem PVNS. Na instância mk02 a abordagem EABC é melhor que as abordagens TS, LEGA e KBACO. Na instância mk03 a abordagem EABC é melhor somente em relação a abordagem TS. Na instância mk04 a abordagem EABC é melhor que as instâncias TS, LEGA, KBACO e TSPCB. Na instância mk05 a abordagem EABC é melhor que todas as abordagens menos a TSPCB. Na instância mk06 a abordagem EABC é melhor que todas as abordagens menos a abordagem PVNS. Na instância mk07 a abordagem EABC é melhor que todas as abordagens menos em relação a abordagem GA. Na instância mk08 todas as abordagens obtiveram o mesmo resultado. Na instância mk09 o EABC foi melhor que todas as outras abordagens menos em relação a abordagem PVNS. Na instância mk10 o EABC também foi melhor que todas as outras abordagens menos em relação a abordagem PVNS.

Tabela 3.8: Comparação de resultados usando as 10 instâncias de Brandimarte (1993)

Instâncias	TS	GA	LEGA	PVNS	KBACO	TSPCB	EABC
mk01	42	40	40	40	39	40	40
mk02	32	26	29	26	29	26	26
mk03	211	204		204	204	204	204
mk04	81	60	67	60	65	62	60
mk05	186	173	176	173	173	172	172
mk06	86	63	67	60	67	65	60
mk07	157	139	147	141	144	140	139
mk08	523	523	523	523	523	523	523
mk09	369	311	320	307	311	310	307
mk10	296	212	229	208	229	214	208

Neste trabalho, os autores Wang et al. (2012a) concluem que sua abordagem foi eficiente,

eficaz e robusta ao tratar as instâncias dos *benchmarks* utilizadas para comparar os resultados de sua abordagem.

3.2.2 *An enhanced Pareto-based artificial bee colony algorithm for the multiobjective flexible job-shop scheduling*

Neste trabalho, Wang et al. (2012b), desenvolveram a abordagem conhecida como *Enhanced Pareto-based Artificial Bee Colony Algorithm* (EPABC) para tratar o problema de programação da produção do tipo JSF multiobjetivo, com a intenção de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm) e o tempo total de produção de todas as máquinas (Wt).

Na intenção de atingir seus objetivos, Wang et al. (2012b) utilizam em sua abordagem EPABC os seguintes processos e características:

- A modelagem das soluções é representada na forma de 2 vetores no qual, um representa as operações a serem programadas e o outro as respectivas máquinas definidas para processarem as operações a serem programadas.

- As soluções iniciais neste trabalho, são geradas por meio de diferentes regras para garantir soluções com certo nível de qualidade. As regras utilizadas podem ser encontradas em (WANG et al., 2012a).

- Na fase das abelhas empregadas, são tratados os problemas de roteamento das máquinas e a programação das operações. Para realizar estas tarefas distintamente a abelha empregada trata primeiramente o roteamento das máquinas utilizando um método onde, dado uma operação escolhida de modo randômico a máquina que é utilizada atualmente para processar essa operação deve ser trocada por outra máquina disponível. Para tal, essa escolha deve ser feita de forma randômica. A segunda atividade executada pela abelha empregada como mencionado anteriormente atua sobre o problema de programação das operações sobre as máquinas utilizadas para o seu processamento. Neste processo, dois *jobs* são selecionadas de modo randômico em um solução. Se a operação do primeiro *job* selecionado gastar menos tempo de processamento que a operação do segundo *job*, essas operações dos *jobs* mudam de posição no sentido de deslocar-se para a direita da solução a operação do *job* com a operação de maior tempo de processamento. Para mais detalhes desse processo consulte Wang et al. (2012b).

- Nesta fase, as abelhas espectadoras utilizam o método de seleção por torneio de tama-

nho 3 para selecionar as fontes de alimentos das abelhas empregadas para sofrerem uma atualização. Depois de selecionada a solução de uma abelha empregada, os processos descritos na fase da abelha empregada são executadas nessa solução por meio da abelha espectadora. Depois de selecionada a abelha espectadora, um valor randômico dentro do intervalo de 0 e 1 e gerado e com base nesse resultado, as seguintes ações podem ser executadas(1) uma nova solução da população pode ser escolhida utilizando o método de seleção por torneio ou (2) uma solução do arquivo de Pareto utilizado para armazenar as melhores soluções encontradas é escolhida de forma randômica. Depois de definida a solução que será utilizada para continuar a atualização da mesma, serão executados sobre essa solução, métodos de cruzamento aplicados sobre o vetor de máquinas: *swap*, MPOX e também o método de caminho crítico assim como em Wang et al. (2012a).

- Na fase da abelha exploradora, uma solução do arquivo de Pareto é selecionada de forma randômica para substituir a solução que não teve melhoras e atingiu a sua cota de tentativas para ser abandonada. Com a escolha da nova solução, é aplicado sobre ela o método baseado em caminho crítico mencionada na fase da abelha espectadora. A nova solução é inserida na população. No arquivo de Pareto é verificado se essa nova solução pode ser melhor que alguma solução já existente podendo substitui-lá também e atualizando o arquivo de Pareto.

Para mensurar a capacidade da abordagem EPABC Wang et al. (2012b), utilizaram instâncias de problemas de *benchmarks* conhecidos como de Kacem, Hammadi e Borne (2002). Para comparar os resultados gerados pela abordagem EPABC foram usadas as abordagens: AIA, AL+CGA, PSO+SA, PSO+TS e PDABC. Os resultados gerados pela abordagem proposta pelos autores pode ser visto na Tabela 3.9.

Na Tabela 3.9 na instância 4x5 o EPABC obteve melhores resultados que a abordagem AL+CGA e em relação as abordagens PSO+TS e PDABC é possível constatar que o EPABC possui uma gama maior de resultados e opções em relação aos objetivos estudados. Na instância 8x8 a abordagem EPABC possuir melhores resultados em relação as abordagens AL+CGA e PSO+SA. Na instância 10x7 o EPABC é melhor que a abordagem PDABC. Na instância 10x10 o EPABC mostra melhores resultados em relação as abordagens AL+CGA, PSO+SA e PSO+TS. Na instância 15x10 o EPABC tem melhores resultados se comparado as demais abordagens utilizadas na comparação e vale ressaltar que a instância 15x10 é a mais complexa se comparada as outras instâncias estudadas nesse trabalho.

Os autores Wang et al. (2012b), concluem em seu trabalho que sua abordagem apresentou melhores resultados que as abordagens utilizadas na comparação.

Tabela 3.9: Comparação de resultados da abordagem EPABC usando 5 instâncias de Kacem, Hammadi e Borne (2002)

Instâncias	Objetivos	AIA	AL+CGA		PSO+SA		PSO+TS		PDABC			EPABC			
4x5	Cm		16				11		11	12	13	11	12	13	11
	Wt		34				32		32	32	33	32	32	33	34
	Wm		10				10		10	8	7	10	8	7	9
8x8	Cm	14	15	16	15	16	14	15	14	15	16	14	15	16	16
	Wt	77	79	75	75	73	77	12	77	75	73	77	75	73	77
	Wm	12	13	13	12	13	12	75	12	12	13	12	12	13	11
10x7	Cm								12	11	12	11	12	11	
	Wt								61	63	60	61	60	62	
	Wm								11	11	12	11	12	10	
10x10	Cm	7	7		7		7		8	7	8	8	7	8	7
	Wt	43	45		44		43		41	43	42	41	43	42	42
	Wm	5	5		6		6		7	5	5	7	5	5	6
15x10	Cm	11	23	24	12		11		12	11		11	11		
	Wt	93	95	91	91		93		91	93		91	93		
	Wm	11	11	11	11		11		11	11		11	10		

3.3 Trabalhos dissertados no período de 2013

3.3.1 *Solving Multi-objective Flexible Job Shop Scheduling with Transportation Constraints using a Micro Artificial Bee Colony Algorithm*

Neste trabalho, Liu et al. (2013), propuseram a abordagem conhecida como *multiobjective micro artificial bee colony algorithm* (MMABC) para tratar o problema de programação da produção do tipo JSF multiobjetivo. A intenção por meio desse trabalho, é a de minimizar os critérios de desempenho: *makespan* (Cm) e o tempo total de produção de todas as máquinas (Wt) em um ambiente onde, além das características de um sistema JSF, é necessário lidar com a questão do tempo de transporte gasto por veículos auto-guiados (AGV's, do inglês: Automated Guided Vehicle) utilizados para deslocar as operações dos *jobs* de uma máquina para outra.

Na intenção de atingir seus objetivos, Liu et al. (2013) utilizam em sua abordagem os seguintes processos e características:

- A modelagem para as soluções usadas neste trabalho são apresentadas na forma de um vetor onde, cada operação, máquina utilizada por essa operação e o AGV que realiza o processo de transporte dessa operação são representados juntos em cada posição (lacuna) do vetor solução, utilizando somente um vetor para representar uma possível solução factível referente ao problema abordado neste trabalho.

- Para calcular a distância das máquinas para cada operação em questão, é utilizada uma matriz com os tempos gastos pelos AGVs de acordo com a máquina corrente onde ele se encontra atualmente e a próxima máquina que ele deve visitar.

- Para calcular a aptidão das soluções, é utilizado o método chamado fronteira de Pareto e as melhores soluções encontradas por esse método são armazenadas em um arquivo externo chamado arquivo de Pareto.

- A geração das soluções iniciais neste trabalho, é realizada de forma randômica de modo que haja diversidade na população inicial.

- Nesta abordagem, as abelhas empregadas usam o mesmo método de cruzamento abordado em Li, Pan e Gao (2011).

- Na fase da abelha espectadora, é utilizado o método de seleção por roleta para selecionar as fontes de alimentos das abelhas empregadas. Depois da seleção das fontes de alimento da abelha emprega, é utilizado o mesmo operador de cruzamento mencionado na fase das abelhas empregadas.

- Na fase da abelha exploradora, é utilizado o operador de mutação para explorar soluções ainda não visitadas no espaço de busca referente ao problema abordado neste trabalho. Para a realização da mutação é utilizada uma solução escolhida de forma randômica do arquivo de Pareto no qual se encontram as melhores soluções visitadas por essa abordagem. Sobre essa nova solução adotada, aplica-se a ela o operador genético de mutação atualizar as máquinas referentes ao processamento das operações.

Para avaliar a abordagem proposta, os autores utilizaram a instância 10x10 do *benchmark* desenvolvido por Kacem, Hammadi e Borne (2002) adaptado para o uso de AGVs. Na Tabela 3.10 é possível observar a comparação realizada pelos autores. Por meio dessa comparação é possível notar a superioridade da abordagem MMABC em relação a abordagem *Multiobjective Micro Genetic Algorithm* (MMGA) (PULIDO; COELLO, 2003).

Na Tabela 3.10 tanto no objetivo Cm quanto no Wt a abordagem de abelhas elaborada pelos autores demonstra melhor resultado quando comparada com o MMGA.

Tabela 3.10: Comparação de resultados da abordagem MMABC com a abordagem MMGA usando a instância 10x10 de Kacem, Hammadi e Borne (2002)

Abordagens	Cm	Wt
MMABC	20	91
MMGA	21	94

Neste trabalho os autores Liu et al. (2013), concluíram que sua abordagem MMABC foi superior à abordagem MMGA em ambos critérios utilizados para comparação dos resultados.

3.3.2 *A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem*

Neste trabalho, Wang et al. (2013) desenvolveram uma abordagem ABC híbrida baseada no trabalho dos autores Banharnsakun, Sirinaovakul e Achalakul (2012), para tratar o problema de programação da produção do tipo JSF com tempos de processamento *Fuzzy* (i.e., tempo: minimo, médio e máximo) em cada operação de um *job*, com a intenção de minimizar o critério de desempenho: *makespan* (Cm).

Na intenção de atingir seus objetivos, Wang et al. (2013) utilizam em sua abordagem os seguintes processos e características:

- Nesse trabalho, o tempo de processamento de uma operação nas máquinas é representado na forma de uma número triangular *fuzzy*. Devido a utilização de números triangulares *fuzzy*, os autores apontam que determinados operadores podem ser utilizados para construção de uma solução. Os operadores utilizados pelos autores para construção das soluções são: adição, maximização de dois números difusos e um operador de classificação (i.e., ranking) de números *fuzzy*. O operador de adição, é usado para calcular o tempo de conclusão do *job*. O operador de maximização é utilizado para definir o tempo de início difuso da operação e o operador de classificação, é utilizado para comparar o tempo máximo de conclusão *fuzzy* (*makespan*).

- A modelagem para as suas soluções é apresentada na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações.

- Para trabalhar com essa modelagem referente as soluções, é utilizado um esquema de decodificação chamado *left-shift* que, busca deslocar as operações para a sua esquerda no vetor solução da forma mais eficaz possível, tornando as operações em ativas/aptas para serem processadas nas máquinas.

- A geração das soluções iniciais neste trabalho, é realizada por meio de uma estratégia híbrida que utiliza diferentes regras de despacho, sendo elas baseadas nas regras geradas nos trabalhos de Li et al. (2011) e Pezzella, Morganti e Ciaschetti (2008).

- Nesta abordagem, as abelhas empregadas usam o operador genético de cruzamento com os vetores de máquinas e de operações. Com relação ao roteamento das máquinas Wang et al. (2013), usam os operadores de cruzamento de dois pontos e o operador de cruzamento uniforme para definir as máquinas nas operações. Depois que novas máquinas

são roteadas nas operações, vem a etapa de sequenciar as operações nas máquinas. Para esse processo de sequenciamento das operações, Wang et al. (2013) usam o operador de cruzamento MPOX (ZHANG et al., 2009).

- Na fase da abelha espectadora, é utilizado o método de seleção por torneio para selecionar as fontes de alimentos das abelhas empregadas. Depois de selecionada as fontes de alimento das abelhas empregadas, a abelha espectadora aplica os mesmos processos descritos na fase da abelha empregada para gerar uma nova solução e realizar a avaliação dessa solução em comparação a solução sem alterações.

- Na fase da abelha exploradora, a geração de uma nova solução ocorre de maneira randômica. Está nova solução substituirá a solução escolhida para ser abandonada da população de abelhas (enxame).

- Depois de realizada a fase da abelha exploradora, utiliza-se um método de busca local conhecido como *Variable Neighbourhood Search* (VNS) para tentar minimizar o critério de desempenho *makespan*.

Para avaliar a abordagem proposta, Wang et al. (2013) utilizam 4 instâncias desenvolvidas por Lei (2010). Wang et al. (2013), comparam sua abordagem com: *Decomposition-Integration Genetic Algorithm* (DIGA) (LEI, 2010), Pezzella's Genetic Algorithm (PEGA) (PEZZELLA; MORGANTI; CIASCHETTI, 2008), PSO+SA (XIA; WU, 2005). Na Tabela 3.11 é apresenta a comparação de resultados gerada por esse trabalho. Ainda na Tabela 3.11, é possível observar a superioridade da abordagem proposta pelos autores em relação as demais utilizadas.

Na Tabela 3.11 na instância L1 o HABC apresenta melhores resultados que as demais abordagens utilizadas na comparação. Na instância L2 o HABC é melhor que as abordagens PEGA e PSO+SA. Na instância L3 o HABC demonstra melhores resultados se comparado as demais abordagens usadas na comparação e na instância L4 ocorre o mesmo em relação a superioridade nos resultados obtidos.

Os autores Wang et al. (2013), concluíram em seu trabalho que a sua abordagem foi mais eficaz que as demais comparadas em termos de resultados e também que a abordagem proposta com o método VNS é mais eficaz do que a mesma sem o VNS.

Tabela 3.11: Comparação de Resultados realizada por Wang et al. (2013)

Instâncias	Abordagens	Cm(Fuzzy)Melhores Resultados
L1	HABC	(19,30,43)
	DIGA	(20,31,40)
	PEGA	(22,33,42)
	PSO+SA	(25,32,40)
L2	HABC	(33,46,58)
	DIGA	(33,48,57)
	PEGA	(38,49,61)
	PSO+SA	(38,49,61)
L3	HABC	(33,47,64)
	DIGA	(37,49,64)
	PEGA	(36,51,68)
	PSO+SA	(38,51,65)
L4	HABC	(23,38,53)
	DIGA	(29,41,56)
	PEGA	(34,48,61)
	PSO+SA	(33,48,64)

3.4 Trabalhos dissertados no período de 2014

3.4.1 *A discrete artificial bee colony algorithm for the multiobjective flexible job-shop scheduling problem with maintenance activities*

Neste trabalho, Li, Pan e Tasgetiren (2014), desenvolveram a abordagem conhecida como *Discrete Artificial Bee Colony Algorithm* (DABC) para tratar o problema de programação da produção do tipo JSF multiobjetivo com atividades de manutenção. A intenção por meio dessa abordagem é a de minimizar os critérios de desempenho: *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm) e o tempo total de produção de todas as máquinas (Wt).

Na intenção de atingir seus objetivos, Li, Pan e Tasgetiren (2014) utilizam em sua abordagem DABC os seguintes processos e características:

- A modelagem para as soluções neste trabalho, são apresentadas na forma de dois vetores no qual, um representa uma sequência de operações, e o outro, as máquinas associadas a essas operações. E a solução é a concatenação desses vetores na forma de um único vetor.
- Para mensurar a aptidão das soluções encontradas no DABC, é utilizado o método chamado fronteira de Pareto já mencionado em outros trabalhos mencionados anteriormente.
- As soluções iniciais neste trabalho, são geradas por meio de regras de despacho para garantir uma solução inicial com certo nível de qualidade. Para mais detalhes sobre as

regras adotadas nesse trabalho consulte Li, Pan e Tasgetiren (2014).

- As melhores soluções do DABC são armazenadas em um arquivo chamado arquivo de Pareto. Esse arquivo tem a mesma finalidade dos arquivos de Pareto apresentados em trabalhos anteriores já discutidos nesse capítulo.

- Na fase das abelhas empregadas, utiliza-se um método TS adaptativo para realizar o roteamento e o sequenciamento das operações nas máquinas. Nesse método TS, as estruturas de vizinhanças adotadas são construídas por meio dos métodos: *swap*(troca de posições), 2-*swap*(aplica a troca duas vezes consecutivas), *insert*(insere uma operação na frente de uma segunda operação escolhida de forma randômica), 2-*insert*(aplica o método *insert* duas vezes consecutivas) e *reverse*(inverte a ordem da solução). A cada iteração realizada pela técnica TS, dois desses métodos são escolhidos e definidos para serem utilizados um para o roteamento das máquinas nas operações e o outro para o sequenciamento das operações nas máquinas.

- As abelhas espectadoras nesta abordagem, selecionam as fontes de alimento das abelhas empregadas por meio da técnica de seleção por torneio onde, 3 soluções são escolhidas de modo randômico e a solução entre essas que obtiver melhor valor de aptidão será selecionada para sofrer uma atualização. O processo de atualização realizado nas fontes de alimentos selecionadas é o mesmo ocorrido utilizado na fase das abelhas empregadas.

- Na fase da abelha exploradora são utilizadas duas abelhas exploradoras. Nesta etapa, as duas piores soluções da população são abandonadas (excluídas) da população e duas novas soluções são adotadas nos seus respectivos lugares na população. A geração dessas novas soluções é realizada pela escolha randômica de duas soluções do arquivo de Pareto utilizado nessa abordagem. Depois que as soluções são escolhidas, o método TS é aplicado a essas soluções.

Neste trabalho, utiliza-se os *benchmarks* de Kacem, Hammadi e Borne (2002) e Brandimarte (1993) para avaliar a abordagem DABC. Nas Tabelas 3.12 e 3.13 são apresentados os resultados gerados pela abordagem DABC. Os resultados gerados pela abordagem DABC são comparados com: PSO+SA, PSO+TS, *Xing and Local Search Algorithm* X-LS (XING; CHEN; YANG, 2009b), HTSA, P-DABC, AIA, *Hybrid Genetic Algorithm* (hGA) (GAO et al., 2007) e *Multiobjective Particle Swarm Optimization and Local Search Algorithm* (MOPSO+LS) (MOSLEHI; MAHNAM, 2011).

Na Tabela 3.12 na instância 4x5 os resultados estão bastante similares mas em relação a dispor mais possibilidades de resultados o DABC é melhor em relação as demais abordagens

utilizadas nessa instância. Na instância 8x8 ocorre a mesma ideia que na instância 4x5 porem, a abordagem MOPSO+LS também apresenta possibilidades para o gestor da produção assim como a abordagem DABC nessa instância. Na instância 10x7 o DABC apresenta melhores resultados em relação a abordagem P-DABC. Na instância 10x10 a abordagem DABC apresenta melhores resultados quando comparada as abordagens PSO+SA e PSO+TS. Na instância 15x10 a abordagem DABC obteve melhores resultados que as abordagens PSO+SA, PSO+TS, P-DABC e MOPSO+LS.

Tabela 3.12: Resultados da Abordagem DABC usando as 5 Instâncias de Kacem, Hammadi e Borne (2002)

	Kacem 4x5			Kacem 8x8			Kacem 10X7			Kacem 10x10			Kacem 15x10		
Abordagens	Cm	Wt	Wm	Cm	Wt	Wm	Cm	Wt	Wm	Cm	Wt	Wm	Cm	Wt	Wm
PSO+SA				15	75	12				7	44	6	12	91	11
				16	73	13									
PSO+TS	11	32	10	14	77	12				7	43	6	11	93	11
				15	75	12									
X-LS	12	32	8	14	77	12	11	61	11	7	42	6	11	91	11
				15	76	12	11	62	10	8	42	5	11	93	10
HTSA	11	32	10	14	77	12	11	61	11	7	42	6	11	91	11
	12	32	8	15	75	12	11	62	10	8	42	5	11	93	10
										7	43	5			
P-DABC	11	32	10	15	75	12	12	61	11	7	43	5	12	91	11
	12	32	8	14	77	12	11	63	11	8	41	7	11	93	11
	13	33	7	16	73	13	12	60	12	8	42	5			
MOPSO+LS				15	75	12				7	42	6	11	91	11
				14	77	12				8	42	5	12	93	10
				16	73	13				7	43	5			
				16	78	11				8	41	7			
				17	77	11									
DABC	11	32	10	15	75	12	11	61	11	7	42	6	11	91	11
	12	32	8	14	77	12	11	62	10	8	42	5	11	93	10
	13	33	7	16	73	13	12	60	12	7	43	5			
	11	34	9	16	78	11				8	41	7			

Na Tabela 3.13 nas instâncias MK01, MK02, MK05 e MK08 o DABC é melhor que as abordagens X-SM e AIA. Nas instâncias MK03 e MK07 o DABC é melhor que as demais abordagens menos a hGA. Na instância MK04 o DABC obteve melhor resultado em comparação as demais abordagens utilizadas nessa instância.

Nesse trabalho Li, Pan e Tasgetiren (2014), demonstram e concluem que sua abordagem é bastante eficaz e eficiente em comparação as abordagens utilizadas para comparação.

Tabela 3.13: Comparação de resultados da abordagem DABC usando o *benchmark* de Brandimarte (1993)

Instâncias	Abordagens	Cm	Wt	Wm
MK01	X-SM	42	162	42
	AIA	40	171	36
	hGA	40	167	36
	HTSA	40	167	36
	DABC	40	167	36
MK02	X-SM	28	155	28
	AIA	26	154	26
	hGA	26	151	26
	HTSA	26	151	26
	DABC	26	151	26
MK03	X-SM	204	852	204
	AIA	204	1207	204
	hGA	204	850	204
	HTSA	204	852	204
	DABC	204	850	204
MK04	X-SM	68	352	67
	AIA	60	403	60
	hGA	60	375	60
	HTSA	61	366	61
	DABC	60	374	60
MK05	X-SM	177	702	177
	AIA	173	686	173
	hGA	172	687	172
	HTSA	172	687	172
	DABC	172	687	172
MK07	X-SM	150	717	150
	AIA	140	695	140
	hGA	139	693	139
	HTSA	140	695	140
	DABC	139	693	139
MK08	X-SM	523	2523	523
	AIA	523	2723	523
	hGA	523	2524	523
	HTSA	523	2524	523
	DABC	523	2524	523

3.5 Trabalhos dissertados no período de 2015

3.5.1 *A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion*

Neste trabalho, Gao et al. (2015), desenvolveram a abordagem *Two-stage Artificial Bee Colony Algorithm* (TABC) para tratar o problema de programação da produção do tipo JSF com inserção de um novo *job*. A intenção desse trabalho está na minimização do critério de desempenho *makespan* (Cm) em um ambiente onde, ocorre a inserção de um novo *job* gerando a necessidade de reprogramação de operações no chão de fábrica. Os autores nesse trabalho, tratam o problema primeiramente sem a reprogramação (estágio um) e com a inserção de um

novo *job* (estágio dois) nas instâncias utilizadas. Nesse trabalho, quando ocorre a necessidade de reprogramação da produção, as operações dos novos *jobs* são inseridos na solução existente e a programação dessa nova solução é refeita utilizando os processos descritos no decorrer dessa seção de acordo com as informações e ideias apresentadas no presente trabalho.

Na intenção de atingir seus objetivos, Gao et al. (2015) utilizam em sua abordagem TABC os seguintes processos e características:

- A modelagem das soluções é representada na forma de 2 vetores um representando as operações a serem programadas e o outro as respectivas máquinas definidas para processarem as operações a serem programadas.

- As soluções iniciais neste trabalho, são geradas por meio de diferentes regras para garantir soluções com certo nível de qualidade. Para mais informações sobre essas regras consulte Gao et al. (2015).

- As abelhas empregadas neste trabalho, aplicam os seguintes operadores genéticos *partial-mapped crossover*, *order crossover* e *cycle crossover* para modificar as soluções alterando o vetor de sequência de operações. Depois de realizada a alteração das soluções por meio dos operadores de cruzamento, o autores utilizam um método de busca local baseado na ideia do método de caminho crítico relacionado as máquinas. Com base no caminho crítico, é apresentado um vetor no qual são armazenados cinco métodos para construção de novas vinhaças referentes as soluções usando: *1-insert*, *1-swap*, *2- insert*, *2-swap* e *1-insert* seguido de *1-swap*. A escolhas desses métodos ocorre por meio de uma roleta probabilista na qual, ocorre maior chances de ser selecionado o método que obtiver melhores resultados em relação ao demais.

- As abelhas espectadoras neste trabalho, selecionam as soluções das abelhas empregadas por meio do método de seleção por torneio. Após selecionar as soluções a serem alteradas, os mesmos processos utilizados pelas abelhas empregadas são reproduzidos pelas abelhas espectadoras a fim de atualizar as soluções encontradas pelas abelhas empregadas.

- Na fase da abelha exploradora, as soluções que atingirem o limite de tentativas para melhorar devem ser abandonadas. No lugar dessas soluções que serão abandonadas, ocorre a geração de uma nova solução que é construída por meio das regras mencionadas na geração da população inicial e está nova solução substituíra a velha solução que atingiu o limite de tentativas para melhorar.

Para avaliar a abordagem TABC Gao et al. (2015), utilizaram os *benchmarks* desenvolvidos por Kacem, Hammadi e Borne (2002) e Brandimarte (1993). As abordagens PVNS, KABCO, TSPCB, EABC e *Simple and Effective Evolutionary Algorithm* (SEA) (CHIANG; LIN, 2013) foram usadas para comparar os resultados gerados pela abordagem TABC. Nas Tabelas 3.14 e 3.15 são apresentados os resultados gerados nas instâncias de problemas fornecidos pelos *benchmarks* mencionados anteriormente.

Na Tabela 3.14 a abordagem TABC é melhor somente que a abordagem PVNS e somente na instância Kacem 5.

Tabela 3.14: Comparação de resultados usando a abordagem TABC no benchmark desenvolvido por Kacem, Hammadi e Borne (2002)

Instâncias	PVNS	KBACO	TSPCB	EABC	SEA	TABC
Kacem 1		11	11	11	11	11
Kacem 2	14	14	14	14	14	14
Kacem 3		11	11	11	11	11
Kacem 4	7	7	7	7	7	7
Kacem 5	12	11	11	11	11	11

Na Tabela 3.15 na instância MK01 o TABC é inferior a abordagem KBACO e apresenta mesmo resultado em relação as demais abordagens. Na instância MK02 o TABC é melhor que a abordagem KABCO e apresenta o mesmo resultado em relação as demais abordagens. Na instância MK03 todas abordagens apresentam o mesmo resultado. Na instância MK04 a abordagem TABC é melhor que as abordagens KABCO, TSPCB e SEA. Na instância MK05 as abordagens TSPBC e EABC apresentam resultados superior as demais abordagens. Na instância MK06 a abordagem TABC é melhor que as abordagens KBACO, TSPCB e SEA. Na instância MK07 a abordagem TABC é melhor que todas as abordagens menos a EABC. Na instância MK08 todas abordagens apresentam o mesmo resultado. Na instância MK09 o TABC apresentou melhor resultado que as abordagens KBACO, TSPCB e SEA. Na instância MK10 o TABC foi a abordagem que apresentou melhor resultado em relação as demais abordagens.

Gao et al. (2015), concluem que sua abordagem apresentou bons resultados ao tratar o problema de pesquisa proposto em seu trabalho. Além disso, os autores ressaltam a eficácia de sua abordagem ao tratar as instâncias utilizadas para comparação com as demais abordagens utilizadas.

Tabela 3.15: Comparação de resultados usando a abordagem TABC no benchmark desenvolvido por Brandimarte (1993)

Instâncias	PVNS	KBACO	TSPCB	EABC	SEA	TABC
MK01	40	39	40	40	40	40
MK02	26	29	26	26	26	26
MK03	204	204	204	204	204	204
MK04	60	65	62	60	61	60
MK05	173	173	172	172	173	173
MK06	60	67	65	60	65	60
MK07	141	144	140	139	140	139
MK08	523	523	523	523	523	523
MK09	307	311	310	307	311	307
MK10	208	229	214	208	225	202

3.6 Considerações Finais

Neste capítulo, foram apresentadas diversas abordagens que utilizam a técnica ABC para tratar o problema de programação da produção do tipo JSF. Nessa pesquisa bibliográfica realizada, foram abrangidos trabalhos dissertados durante o período de 2011 até 2016. Observa-se nos trabalhos apresentados neste capítulo que, as abordagens usando a técnica ABC apresentaram resultados similares ou melhores que os das outras abordagens encontradas na literatura utilizando os *benchmarks* desenvolvidos por Kacem, Hammadi e Borne (2002) e Brandimarte (1993). Além disso, é possível constatar nos trabalhos estudados que alguns métodos são utilizados em mais de uma abordagem de enxame de abelhas. Essas características são apresentadas nas Tabelas 3.16, 3.17 e 3.18 de modo que seja possível observar quais técnicas e métodos foram usados no desenvolvimento das abordagens estudadas.

Tabela 3.16: Métodos utilizados nos trabalhos estudados na revisão da literatura - PARTE 1

Características	(Zhout et al., 2011)	(Li, Pan e Gao, 2011)	(Li, Pan e Xie, 2011)	(Li et al., 2011)
Monoobjetivo				
Multiobjetivo	X	X	X	X
Regras de Inicialização da População	X		X	
Método de Seleção por Roleta	X			
Método de Seleção por Torneio		X	X	X
Operador genético de Cruzamento	X	X		
Operador genético de Mutação				
Soma Ponderada para mensurar as soluçoes				
Abordagem de Pareto para mensurar as soluçoes		X	X	
Fuzzy para mensurar as soluçoes				
Método de Caminho Crítico	X			
Método Swap		X		X
Método Insert		X		X
Método Reverse				
Vetor de estrutura de vizinhanças				
Gerar solução da abelha exploradora com regras específicas	X	X	X	X
TS			X	X
Arquivo de Pareto		X	X	
Método VNS				

Tabela 3.17: Métodos utilizados nos trabalhos estudados na revisão da literatura - PARTE 2

Características	(Wang et al., 2012a)	(Wang et al., 2012b)	(Liu et al., 2013)	(Wang et al., 2013)
Monoobjetivo	X			
Multiobjetivo		X	X	X
Regras de Inicialização da População	X	X		X
Método de Seleção por Roleta	X		X	
Método de Seleção por Torneio				X
Operador genético de Cruzamento	X	X	X	X
Operador genético de Mutação	X		X	
Soma Ponderada para mensurar as soluçoes				
Abordagem de Pareto para mensurar as soluçoes		X	X	
Fuzzy para mensurar as soluçoes				X
Método de Caminho Crítico	X	X		
Método Swap		X		
Método Insert				
Método Reverse				
Vetor de estrutura de vizinhanças				
Gerar solução da abelha exploradora com regras específicas		X		
TS				
Arquivo de Pareto		X	X	
Método VNS				X

Tabela 3.18: Métodos utilizados nos trabalhos estudados na revisão da literatura - PARTE 3

Características	(Li, Pan e Tasgetiren, 2014)	(Gao et al., 2015)
Monoobjetivo		X
Multiobjetivo	X	
Regras de Inicialização da População	X	X
Método de Seleção por Roleta		
Método de Seleção por Torneio	X	X
Operador genético de Cruzamento		X
Operador genético de Mutação		
Soma Ponderada para mensurar as soluçoes		
Abordagem de Pareto para mensurar as soluçoes	X	
Fuzzy para mensurar as soluçoes		
Método de Caminho Crítico		X
Método Swap	X	X
Método Insert	X	X
Método Reverse	X	
Vetor de estrutura de vizinhanças		X
Gerar solução da abelha exploradora com regras específicas	X	X
TS	X	
Arquivo de Pareto	X	
Método VNS		

Capítulo 4

PROPOSTA DO TRABALHO

Na atividade de programação da produção em ambientes de manufatura do tipo JSF, encontra-se os seguintes problemas aludidos na literatura:

1. Necessidade de tratar os sub-problemas de roteamento e de programação das operações;
2. Demanda-se alto custo computacional para tratar o problema. Além disso, ocorre-se a incerteza de encontrar uma solução ideal em tempo computacional considerável devido a característica *Non-Deterministic Polynomial Time* (NP)-Difícil do problema JSF;
3. Resolver os conflitos na harmonização dos multi-objetivos.

Devido as características dos problemas ocorridos no chão de fábrica supracitados referentes ao problema de programação da produção JSF, neste trabalho propõe-se a seguir uma solução para esses problemas.

Para tratar o problema 1 relacionado a modelagem dos sub-problemas de programação e de roteamento, nesse trabalho sugere-se uma modelagem baseada nos trabalhos de Zhou et al. (2011), Li, Pan e Gao (2011), Li, Pan e Xie (2011), Li et al. (2011), Wang et al. (2012a, 2012b, 2013), Li, Pan e Tasgetiren (2014), Gao et al. (2015). Nesse modelo apresentado nos trabalhos supracitados, as informações são apresentadas na forma de dois vetores onde, um dos vetores armazena as programações referentes ao sequenciamento das operações nas máquinas disponíveis no chão de fábrica (i.e., sub-problema de programação) e o outro vetor armazena as máquinas definidas para processar as operações (i.e., sub-problema de roteamento) supracitadas referentes ao primeiro vetor mencionado.

Para tratar o problema 2 sugere-se nesse trabalho uma abordagem ABC híbrida. O ABC é sugerido por ser uma abordagem que tem apresentado bons resultados ao tratar o problema

abordado nesse trabalho. Além disso, por ser uma meta-heurística o ABC tende a encontrar boas soluções em tempo considerável para o problema JSF (CHAUDHRY; KHAN, 2016).

Para que a abordagem proposta alcance bons resultados a partir da sua população inicial, nesta pesquisa com base nos trabalhos de Zhou et al. (2011), Li, Pan e Liang (2010b), Wang et al. (2012a, 2012b, 2013), Li, Pan e Tasgetiren (2014), Gao et al. (2015), sugere-se o uso de regras de inicialização para que a abordagem proposta possa começar seu processo de busca partindo de soluções com certo nível de qualidade considerável relacionadas ao JSF.

Para tratar o sub-problema de roteamento do JSF, nesse trabalho, propõe-se o uso do operador genético de mutação. Com base nos trabalhos de Wang et al. (2012a), Liu et al. (2013) sustenta-se a ideia de que esse operador ajude para que a abordagem proposta nesse trabalho não fique presa em pontos ótimos locais referentes ao espaço de busca do problema JSF. Nesse trabalho esse método é proposto na fase das abelhas espectadoras. Essa utilização do operador genético nessa fase é proposto com base nas indicações realizadas por Karaboga (2005) de modo que o método proposto ajude a melhorar os resultados obtidos explorando novas regiões do espaço de busca e também fornecendo a possibilidade de recuperar informações que já foram perdidas e não podem ser encontradas na solução corrente em uso.

Para tratar o sub-problema de programação das operações nas máquinas nesse trabalho propõe-se um vetor de estruturas de vizinhança inspirado e baseado no trabalho de Gao et al. (2015). No trabalho de Gao et al. (2015) é utilizado um vetor que armazena as estruturas de vizinhança *swap* e *insert* com número de substituições igual a 1 troca e no máximo 2 trocas para cada estrutura. Para definir a estrutura que será utilizada, Gao et al. (2015) utilizam uma roleta probabilística o que de modo adaptativo seleciona uma estrutura a cada iteração com base na qualidade dessa estrutura ao realizar a programação das operações.

No presente trabalho, propõe-se um novo método baseado no apresentado por Gao et al. (2015). Nesse novo método é adicionado mais três estruturas de vizinhança sendo elas *swap* e *insert* com 3 substituições e a adição da estrutura *reverse* baseada no trabalho de Li, Pan e Tasgetiren (2014). Com a adição das novas estruturas é possível explorar soluções ainda não alcançadas, na tentativa de encontrar melhores resultados para o problema estudado.

Para comparar os resultados obtidos pela abordagem proposta (i.e., problema 3), sugere-se nesse trabalho, o uso do método dominância de Pareto (LI; PAN; XIE, 2011; LI; PAN; GAO, 2011; WANG et al., 2012b; LIU et al., 2013; LI; PAN; TASGETIREN, 2014). O método de Pareto não necessita de ajustes (i.e., pesos) para definir qual solução será melhor ou pior do que outra (i.e., transforma o problema em mono-objetivo), ele julga o conjunto de objetivos como um todo. Isso ajuda com que os resultados obtidos não se limitem a determinada região do espaço de

busca do problema. O método de Pareto nessa pesquisa irá mensurar o conjunto de objetivos Cm, Wm e Wt apresentados e definidos no capítulo de fundamentação teórica desse trabalho.

Por meio do presente trabalho, espera-se verificar se as sugestões propostas para abordagem ABC, nas etapas supracitadas, apresentaram melhores resultados que outras abordagens da literatura por meio dos *benchmarks* de Kacem, Hammadi e Borne (2002), Brandimarte (1993).

4.1 Metodologia

Com o objetivo de tratar os problemas estudados nesse trabalho, a seguir são apresentados os passos necessários para a implementação da abordagem proposta. Para facilitar o entendimento da abordagem proposta ilustra-se na Figura 4.1 o fluxograma dos passos a serem implementados para o desenvolvimento da abordagem sugerida.

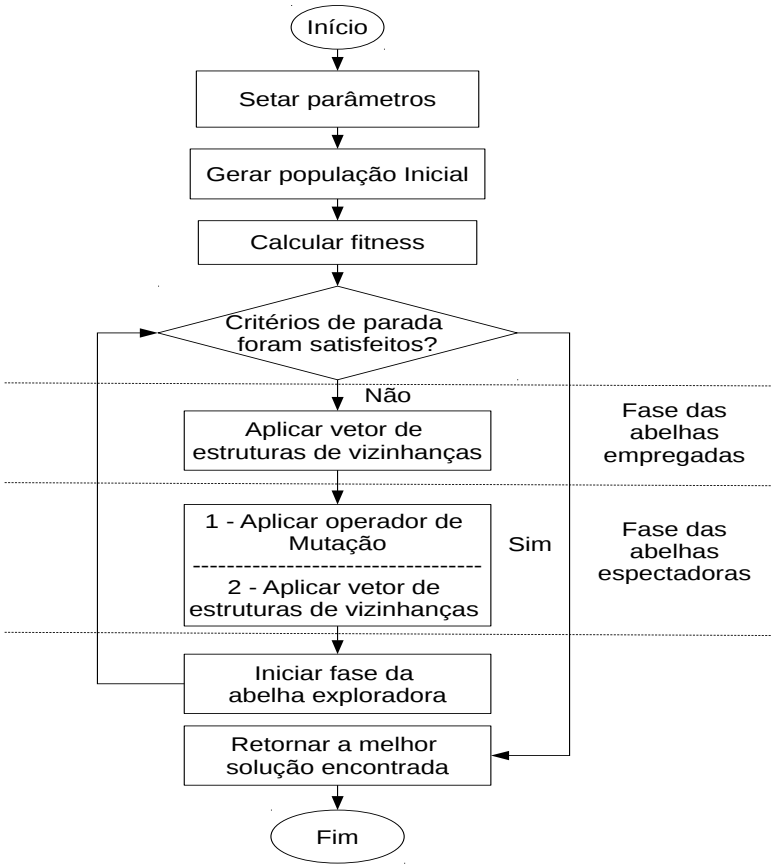

Figura 4.1: Fluxograma da abordagem proposta com as respectivas etapas do ABC sugerido

Antes de descrever as etapas do ABC proposto, é importante ressaltar que a configuração dos parâmetros do ABC sugerido não é mencionada nesse capítulo e sim no seguinte. Isso ocorre devido as características de cada problema estudado. As características de cada problema

faz com que seja necessário que o ABC utilize uma configuração específica para cada problema (i.e., instância) tratado dos *benchmarks* de Kacem, Hammadi e Borne (2002) e Brandimarte (1993).

4.1.1 Modelagem adotada para representar as soluções utilizadas pela abordagem proposta

Na literatura, geralmente utiliza-se duas formas para modelar as soluções para o problema JSF. A primeira forma é por meio de dois vetores separados no qual, um representa as máquinas a serem utilizadas para processar as operações e no outro vetor, representa-se a sequência de operações que serão processadas nas máquinas do primeiro vetor. A segunda modelagem é uma concatenação dos dois vetores mencionados na forma de um único vetor. Neste trabalho, adota-se a primeira forma de modelagem. Essa escolha foi realizada com base na quantidade de trabalhos apresentados na revisão da literatura que atingiram bons resultados.

Para ilustrar os métodos apresentados no decorrer desse capítulo, será utilizado o problema exemplo apresentado na Tabela 4.1. Na Tabela 4.1, X representa que uma determinada operação $O_{i,j}$ não pode ser processada por um determinado recurso M_k e abaixo desses recursos, ilustra-se o tempo de processamento gasto por cada operação $O_{i,j}$ nos mesmos. Na Figura 4.2, é ilustrado um exemplo da modelagem definida no parágrafo anterior com base no problema exemplo da Tabela 4.1.

Na modelagem da Figura 4.2, a solução é definida com duas regiões distintas, da primeira posição da solução até a sétima, tem-se a representação das máquinas a serem definidas para processar as operações (i.e., sub-problema de roteamento) e da oitava posição da solução até a décima quarta, tem-se a representação das operações a serem sequenciadas nas máquinas (i.e., sub-problema de sequenciamento).

Tabela 4.1: Problema de exemplo do tipo *Job Shop* Parcialmente Flexível

	Operações	Máquinas				
Jobs	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	4	5	1	2
	$O_{1,2}$	5	X	7	9	2
J_2	$O_{2,1}$	10	1	1	3	7
	$O_{2,2}$	1	8	2	X	7
	$O_{2,3}$	1	6	1	3	2
J_3	$O_{3,1}$	4	9	3	4	4
	$O_{3,2}$	X	1	2	8	10

A Figura 4.3 representa a solução de exemplo utilizada para representar a modelagem ilus-

trada na Figura 4.2.

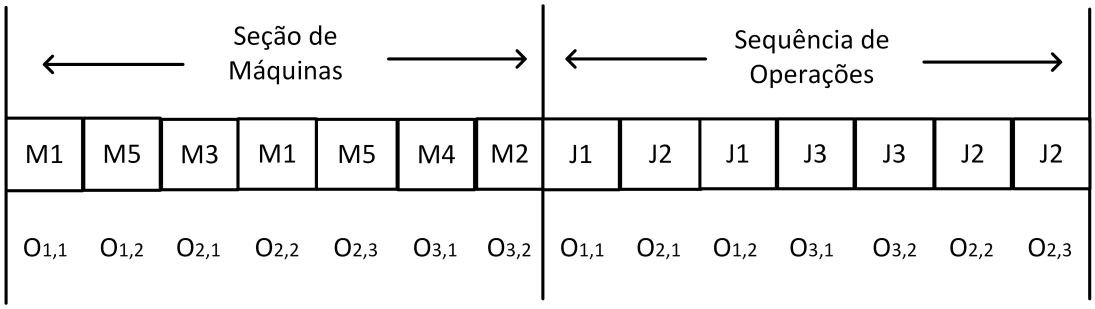

Figura 4.2: Exemplo da modelagem das soluções com base nos trabalhos de Li, Pan e Xie (2011), Li et al. (2011), Li, Pan e Gao (2011), Zhou et al. (2011), Wang et al. (2012b), Li, Pan e Tasgetiren (2014)

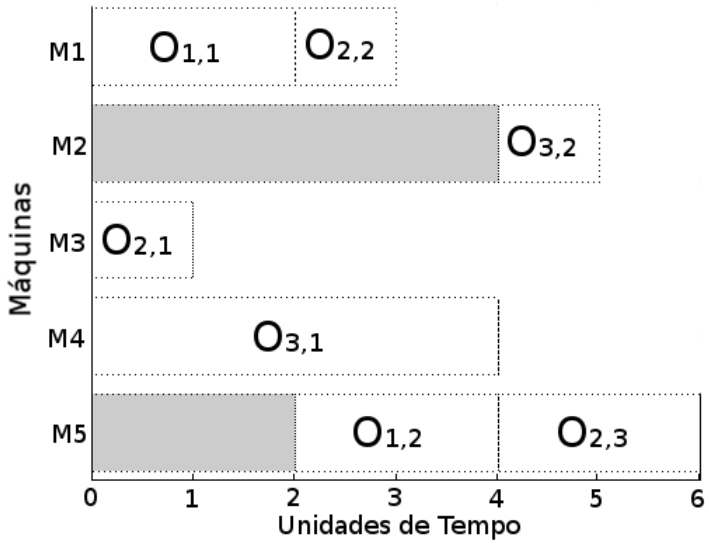

Figura 4.3: Representação da solução da Figura 4.2 por um gráfico de Gantt.

4.1.2 Inicialização das fontes de alimentos (i.e., Gerar população inicial)

Disserta-se na literatura em alguns trabalhos como os de Zhou et al. (2011), Li, Pan e Liang (2010b), Wang et al. (2012a, 2012b, 2013), Li, Pan e Tasgetiren (2014), Gao et al. (2015), que a utilização de regras para criação das soluções iniciais, garantem uma população inicial com certo nível de qualidade. Devido a capacidade dessas regras para criação de uma população inicial boa, as regras a seguir serão empregadas na abordagem proposta com base no estudo dos autores supracitados:

1. Definir para cada operação a máquina que obtiver menor tempo de processamento *Shortest Processing Time* (SPT) (FERNANDES; FILHO, 2010). Caso ocorra empate no tempo de

processamento entre as máquinas, sortear de modo randômico entre essas máquinas qual irá realizar o processamento da operação em questão;

2. Definir para cara operação a máquina que obtiver maior tempo de processamento. Caso ocorra empate no tempo de processamento entre as máquinas, sortear de modo randômico entre essas máquinas qual irá realizar o processamento da operação em questão;

3. Definir as máquinas para as operações de forma randômica.

A geração das fontes de alimento iniciais é realizada com base no trabalho de Wang et al. (2012a) onde, em 30% do enxame de abelhas (i.e., população/soluções) utiliza-se a primeira regra supracitada e em mais 30% da população utiliza-se a segunda regra. No resto do enxame, aplicar a terceira regra de inicialização. Por meio dessas regras, as soluções iniciais são geradas a partir de diferentes pontos do espaço de busca. Desse modo a população inicial pode estar mais próxima do resultado desejado garantindo certo nível de qualidade a essas soluções geradas.

4.1.3 Função de fitness

A função de *fitness* do ABC tem por natureza o objetivo de mensurar a qualidade das soluções geradas por ele. Para que a abordagem proposta possa mensurar a qualidade das soluções geradas pelo ABC será utilizado nessa pesquisa a abordagem de Pareto descrita no capítulo 2. Nesse trabalho os critérios de chão de fábrica estudados são *Makespan* (C_m), *Maximum Workload* (W_m) e *Total Workload* (W_t).

A ideia utilizada para mensurar as soluções do ABC proposto será a mesma explicada no capítulo 2, ou seja, como exemplo imagine uma solução S_1 que possui o conjunto de objetivos C_1 composta com os critérios mencionados no parágrafo anterior. Agora imagine uma segunda solução S_2 que possui o conjunto de objetivos C_2 composta pelos mesmo critérios de S_1 supondo que ambas são diferentes.

Com base na descrição da abordagem de Pareto no capítulo 2 para que uma solução possa ser considerada melhor que outra deve-se ocorrer as seguintes situações:

1. As variáveis de decisão (i.e., critérios de desempenho) que compõem a solução S_1, devem ser no mínimo iguais as variáveis de decisão encontradas em S_2. Nesse caso ambas são boas por terem mesma qualidade mas isso não significa que ambas são iguais e o critério de definir qual é considerada melhor fica por conta do implementador dessa regra no problema em questão estudado.

2. Na solução S_1 deve haver pelo menos uma variável de decisão com valor menor (i.e., devido ser um problema de minimização) que o da variável equivalente presente na solução S_2. Por exemplo C(C_m,W_m,W_t) onde tem-se o seguinte caso C_1(1,2,2) e C_2(2,2,2);

3. O terceiro caso que pode ocorrer considera-se que S_1 é totalmente superior a S_2 em todas variáveis de decisão que a compõem;

Além das regras supracitadas, é possível se ter um conjunto de soluções e não somente uma solução do problema o que permite ao gestor da produção definir qual soluções é a melhor para o seu problema em questão fornecendo-lhe opções que podem ter características diferentes com qualidades semelhantes.

4.1.4 Fase das abelhas empregadas

Nesta fase, cada abelha empregada tem por característica a função de modificar a solução relacionada a ela como descrito na abordagem ABC proposta por Karaboga (2005). A modificação das soluções realizada nesta fase pelas abelhas empregadas, ocorre por meio do vetor de estruturas de vizinhança sugerido na proposta desse trabalho. A seguir, a implementação desse método é descrita de forma mais detalhada.

Para descrever a ideia do vetor de estruturas de vizinhança sugerido, será utilizado nesta seção a solução de exemplo ilustrada na Figura 4.2 baseada no problema da Tabela 4.1. Para representar essa solução, será utilizada a seguinte notação:

X_i**:** solução (fonte de alimento) da abelha *i*, onde (i=1, 2, 3,..., número de fontes de alimento (NFA));

V**:** vetor de estruturas de vizinhança composto por: V_1=(1-*swap*), V_2=(1-*insert*), V_3=(*reverse*), V_4=(2-*swap*), V_5=(2-*insert*), V_6=(*reverse*), V_7=(3-*swap*), V_8=(3-*insert*) e V_9 = (*reverse*). O número na frente de cada estrutura de vizinhança quando existir, significa a quantidade de trocas realizada por cada estrutura durante a modificação da solução a ser alterada;

M**:** conjunto de máquinas;

J**:** conjunto de *jobs*;

$fitness_i$**:** valor de aptidão/*fitness* da abelha *i*;

$fitnessV_k$**:** valor de aptidão referente a estrutura k do vetor de estruturas de vizinhança onde, (k=1, 2, 3,..., 9).

A solução de exemplo utilizada é composta pela seguinte sequência de operações: (vetor de sequência de operações) ($X_{i,8} = (O_{1,1},\ J_1)$, $X_{i,9} = (O_{2,1},\ J_2)$, $X_{i,10} = (O_{1,2},\ J_1)$, $X_{i,11} = (O_{3,1},\ J_3)$, $X_{i,12} = (O_{3,2},\ J_3)$, $X_{i,13} = (O_{2,2},\ J_2)$, $X_{i,14} = (O_{2,3},\ J_2)$). Nesta fase será utilizado somente a parte da solução exemplo relacionada a sequência de operações.

A decisão de tratar a parte da solução mencionada ocorre devido a essência original do ABC criado pelo autor dessa meta-heurística. Nessa fase o problema é tratado com relação a busca local e foca-se devido a isso na melhoria dos critérios de desempenho C_m e W_m nesta parte da solução do JSF.

Inicialmente, todo $fitnessV_k$ é setado com valor 1, e por meio de uma roleta probabilística como a utilizada na Equação 2.5, gera-se a probabilidade de cada estrutura de vizinhança V_k ser selecionada para atuar sobre a sequência de operações utilizada como exemplo a fim de otimizá-la. Na Figura 4.4 é ilustrado um exemplo dessa roleta proposta com as estruturas de vizinhança supracitadas. No exemplo da Figura 4.4 a estrutura de vizinhança com maior probabilidade de ser selecionada é a estrutura V_7 que representa 30% da roleta ilustrada.

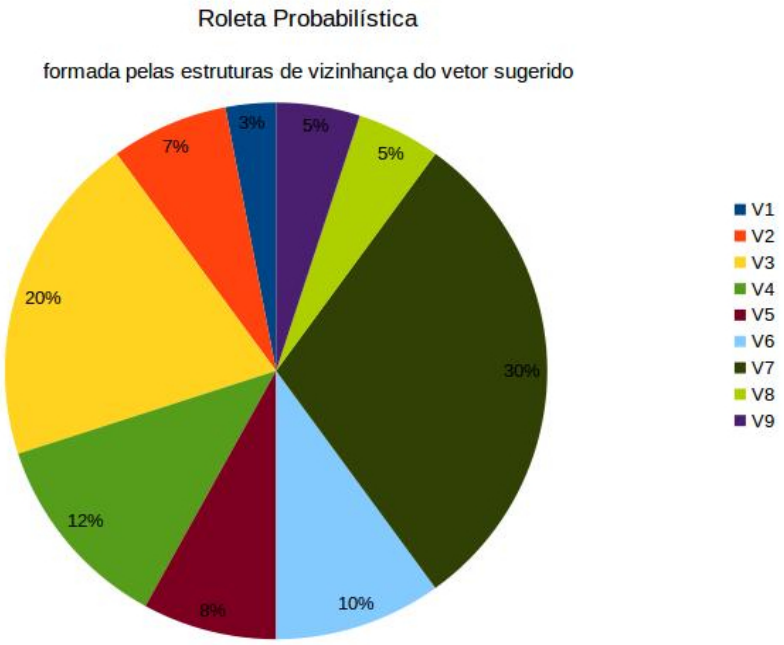

Figura 4.4: Exemplo de uma roleta probabilística representando o vetor de estruturas de vizinhança proposto

Se, ao utilizar uma estrutura V_k selecionada probabilisticamente e a nova solução X_i gerada por essa estrutura apresentar um valor de aptidão melhor, a aptidão $fitnessV_k$ da estrutura V_k é incrementada. Desse modo, a estrutura V_k torna-se mais apta a ser escolhida durante a próxima seleção realizada sobre o vetor de estrutura de vizinhanças. A seguir, são apresentados exemplos da aplicação das estruturas de vizinhança sobre a sequência de operações supracitada. Para

simplificar a ideia sobre cada uma, será utilizado um exemplo com apenas as estruturas de vizinhança que utilizam um movimento/substituição para facilitar o entendimento dessas estruturas e do método proposto.

Método de substituição *swap*: sorteia-se de modo randômico 2 posições referentes a sequência de operações. Neste exemplo, as posições escolhidas foram $X_{i,10}$ e $X_{i,13}$, 1ª passo da Figura 4.5. A estrutura de vizinhança *swap* é bastante simples, ela tem por função substituir o conteúdo de $X_{i,10}$ pelo conteúdo de $X_{i,13}$, 2º passo da Figura 4.5. Depois de realizada a troca, tem-se a seguinte nova sequência gerada: $X_{i,8} = (O_{1,1}, J_1)$, $X_{i,9} = (O_{2,1}, J_2)$, $X_{i,10} = (O_{2,2}, J_2)$, $X_{i,11} = (O_{3,1}, J_3)$, $X_{i,12} = (O_{3,2}, J_3)$, $X_{i,13} = (O_{1,2}, J_1)$, $X_{i,14} = (O_{2,3}, J_2)$), 3º e último passo da Figura 4.5. É importante que em toda substituição seja verificada e se necessário, refatorada a sequência/ordem das operações de cada *job*. Essa ação deve ser realizada para que não ocorra nenhuma irregularidade sobre as restrições impostas ao problema JSF estudado, ou seja, o de precedência das operações de cada *job*.

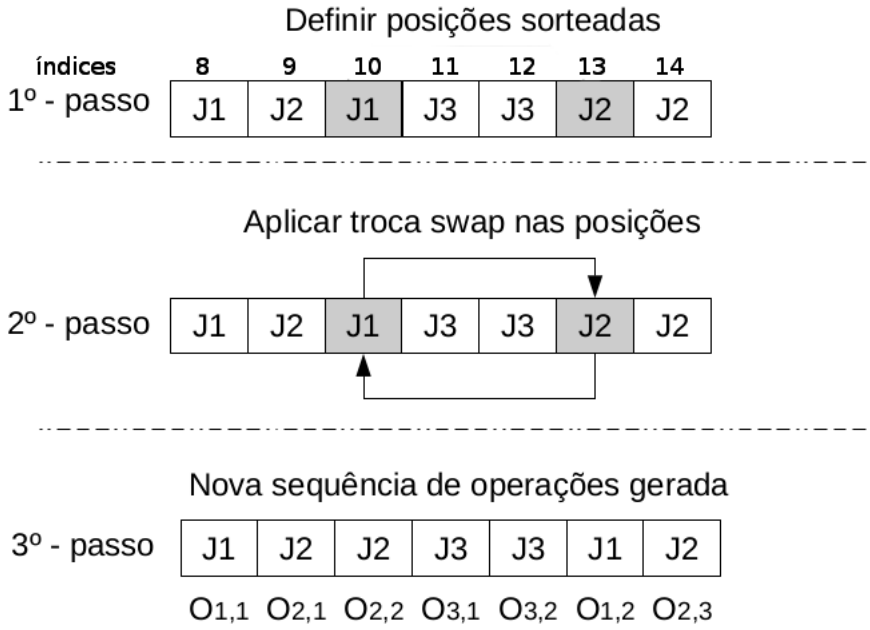

Figura 4.5: Exemplo de aplicação da estrutura de vizinhança *swap* com uma substituição

Método de substituição *insert*: nesta estrutura, sorteia-se também de modo randômico 2 posições referentes a sequência de operações. Neste exemplo, as posições escolhias foram $X_{i,8}$ e $X_{i,13}$, 1º passo da Figura 4.6. Para exemplificar a aplicação da estrutura *insert*, será utilizado a sequência de operações gerada por meio da estrutura *swap* supracitada. Na estrutura *insert*, a ideia é inserir o conteúdo da primeira posição escolhida $X_{i,8}$, na posição à frente da segunda posição escolhida $X_{i,13}$. Neste caso, o conteúdo de $X_{i,8}$ é movido para a posição $X_{i,14}$ que equivale a posição 13+1 da sequência de operações, 2º passo da Figura 4.6. Depois de

realizada a troca, tem-se a seguinte nova sequência gerada: $X_{i,8} = (O_{2,3}, J_2)$, $X_{i,9} = (O_{2,1}, J_2)$, $X_{i,10} = (O_{2,2}, J_2)$, $X_{i,11} = (O_{3,1}, J_3)$, $X_{i,12} = (O_{3,2}, J_3)$, $X_{i,13} = (O_{1,2}, J_1)$, $X_{i,14} = (O_{1,1}, J_1)$). Nessa sequência gerada, nota-se que ocorre a infração da restrição referente a sequência de operações dos *jobs* i.e., nunca uma operação com índice de operação maior deve ser processada antes de uma operação com índice menor (e.g., a operação $X_{i,13} = (O_{1,2}, J_1), X_{i,14} = (O_{1,1}, J_1)$ quebra a restrição referente a ordem de execução das operações então nesse caso deve-se ocorrer a seguinte mudança $X_{i,13} = (O_{1,1}, J_1), X_{i,14} = (O_{1,2}, J_1)$). Nesse caso é necessária a atualização da ordem de processamento dos *jobs* e ao realizar alteração, a sequência de operações deve ficar desse modo: $X_{i,8} = (O_{2,1}, J_2)$, $X_{i,9} = (O_{2,2}, J_2)$, $X_{i,10} = (O_{2,3}, J_2)$, $X_{i,11} = (O_{3,1}, J_3)$, $X_{i,12} = (O_{3,2}, J_3)$, $X_{i,13} = (O_{1,1}, J_1)$, $X_{i,14} = (O_{1,2}, J_1)$), 3º e último passo da Figura 4.6.

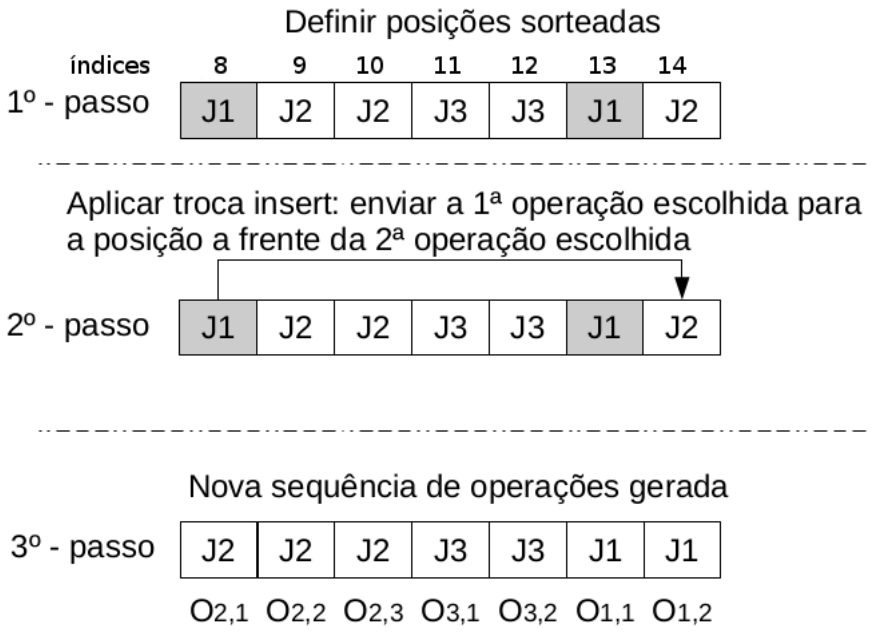

Figura 4.6: Exemplo de aplicação da estrutura de vizinhança *insert* com uma substituição

Método de substituição *reverse*: essa estrutura de vizinhança, diferentemente das anteriores, tem por característica inverter toda a sequência de operações. Aqui a sequência de operações iniciais é a sequência gerada pela estrutura *insert* na estrutura anterior apresentada. Partindo dessa sequência de operações, tem-se a seguinte nova sequência de operações ao realizar a inversão da sequência anterior: $X_{i,8} = (O_{1,2}, J_1), X_{i,9} = (O_{1,1}, J_1), X_{i,10} = (O_{3,2}, J_3), X_{i,11} = (O_{3,1}, J_3)$, $X_{i,12} = (O_{2,3}, J_2)$, $X_{i,13} = (O_{2,2}, J_2)$ e $X_{i,14} = (O_{2,1}, J_2)$. Do mesmo modo que na estrutura anterior, é necessário realizar a atualização da sequência da ordem das operações dos *jobs*. Com a atualização da ordem das operações, a sequência final gerada respeitando as restrições impostas ao problema JSF é a seguinte: $X_{i,8} = (O_{1,1}, J_1)$, $X_{i,9} = (O_{1,2}, J_1)$, $X_{i,10} = (O_{3,1}, J_3)$, $X_{i,11} = (O_{3,2}, J_3)$, $X_{i,12} = (O_{2,1}, J_2)$, $X_{i,13} = (O_{2,2}, J_2)$ e $X_{i,14} = (O_{2,3}, J_2)$, 1º e último passo

da Figura 4.7.

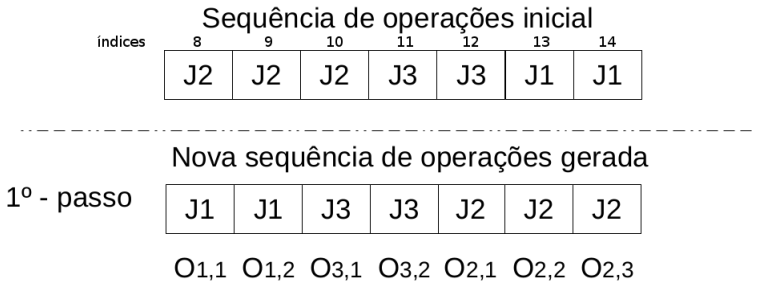

Figura 4.7: Exemplo de aplicação da estrutura de vizinhança *reverse*

4.1.5 Fase das abelhas espectadoras

Nesta fase, cada abelha espectadora deve selecionar uma solução das abelhas empregadas para modificá-la. Essa seleção realizada pela abelha espectadora neste trabalho, ocorre por meio do método de seleção por torneio onde, as soluções de três abelhas empregadas escolhidas de modo randômico são comparadas (i.e., utiliza-se a abordagem de Pareto para comparar os conjuntos de objetivos de cada uma das soluções selecionadas) e a solução entre as três que possuir melhor valor de aptidão, devera ser selecionada para sofrer uma alteração. Segundo Gao et al. (2015), Li, Pan e Tasgetiren (2014), a utilização desse método de seleção ajuda para que a população possa evoluir como um todo e não somente as melhores soluções (i.e., assim como ocorre no método de seleção por roleta).

Depois de selecionada uma solução pertencente a uma das abelhas empregadas, a abelha espectadora tem a função de tentar melhorar essa solução selecionada. Para melhorar essa solução, a abelha espectadora à tratará em duas etapas, sendo a primeira etapa referente ao sub-problema de roteamento (i.e., definição da máquina a ser utilizada no chão de fábrica para processar uma operação) e a segunda etapa se encarregará de tratar a otimização do sub-problema de sequenciamento das operações (i.e., ordem das operações nas máquinas).

Antes de iniciar os passos utilizados pela abelhas espectadora é importante informar que nessa fase o uso do operador genético é incorporado as abelhas espectadoras devido a característica dessa abelha no ABC. A abelha espectadora está relacionada a possibilidade de tratar o problema de forma mais global diferentemente da abelha empregada (KARABOGA, 2005). Por isso nessa fase também ataca-se a parte da solução relacionada ao roteamento das máquinas nas operações e o impacto dessa alteração das máquinas afeta a solução de forma muito grave uma vez que uma determinada máquina possa gastar menos tempo para processar uma operação enquanto outra gaste mais tempo em relação a está outra. Essa é uma das características que

tornam o JSF um problema ainda mais complexo que o JS.

Dando continuidade ao tratamento do problema pelas abelhas espectadoras, após a seleção da solução selecionada, nessa primeira etapa (i.e., etapa 1) será utilizado o operador genético de mutação com troca *swap* (i.e., operador genético encontrado em algoritmos genéticos) para rotear as máquinas nas operações. Nesta etapa, será utilizada a parte da solução definida como seção de máquinas composta por: ($X_{i,1} = (O_{1,1}, M_1)$, $X_{i,2} = (O_{1,2}, M_5)$, $X_{i,3} = (O_{2,1}, M_3)$, $X_{i,4} = (O_{2,2}, M_1)$, $X_{i,5} = (O_{2,3}, M_5)$, $X_{i,6} = (O_{3,1}, M_4)$, $X_{i,7} = (O_{3,2}, M_2)$) (i.e., representa um individuo em Algoritmos Genéticos), essa etapa é representada na parte (a) da Figura 4.8. De acordo com os fundamentos do operador de mutação, deve-se definir quanto dessa solução deve sofrer mutação. Para este exemplo, utiliza-se 14% dessa solução que equivale a quantidade de uma operação dessa solução ou um gene em algoritmos genéticos. De modo randômico, uma operação da seção de máquinas deve ser escolhida e nesse exemplo, a operação $O_{2,2}$ foi escolhida para sofrer mutação, essa etapa é representada na parte (b) da Figura 4.8. A operação $O_{2,2}$ (i.e., representa um alelo em Algoritmos Genéticos), só pode ser processada pelas seguintes máquinas M_2, M_3, M_5 (i.e., representa um alfabeto em Algoritmos Genéticos), essa etapa é representada na parte (c) da Figura 4.8. A máquina M_1 é a máquina corrente definida atualmente para processar a operação $O_{2,2}$. Ainda de modo randômico, umas das máquinas mencionadas deve ser escolhida para substituir a máquina atual M_1 da operação $O_{2,2}$. Supondo que a máquina escolhida foi a M_3, após a mutação, tem-se a seguinte nova seção de máquinas composta por: ($X_{i,1} = (O_{1,1}, M_1)$, $X_{i,2} = (O_{1,2}, M_5)$, $X_{i,3} = (O_{2,1}, M_3)$, $X_{i,4} = (O_{2,2}, M_3)$, $X_{i,5} = (O_{2,3}, M_5)$, $X_{i,6} = (O_{3,1}, M_4)$, $X_{i,7} = (O_{3,2}, M_2)$), essa etapa é representada na parte (d) da Figura 4.8 e ilustra a nova seção de máquinas gerada pelo operador genético de mutação proposto.

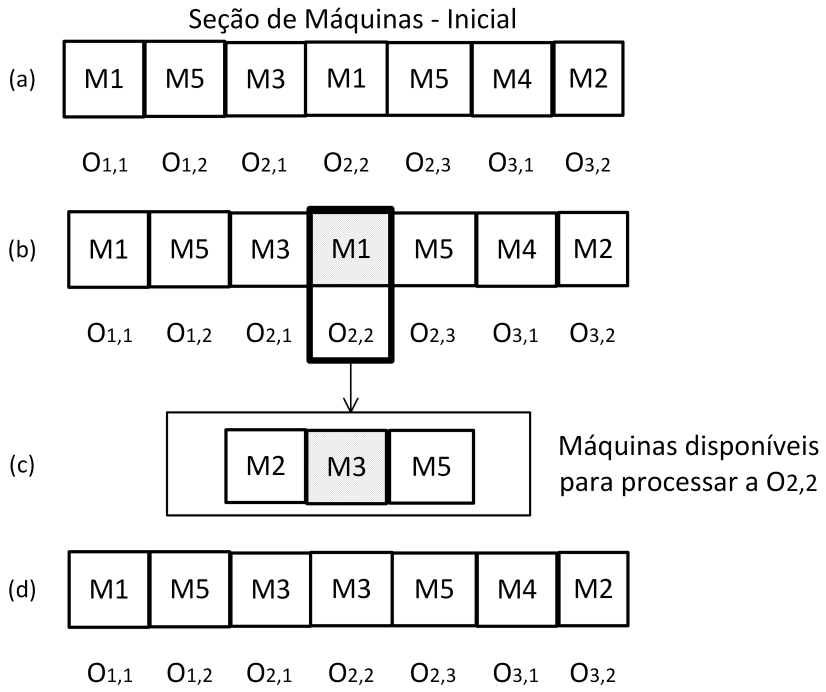

Figura 4.8: Exemplo do processo de mutação aplicado na seção de máquinas do vetor solução referente ao problema de roteamento do JSF

Depois de finalizada a etapa 1 referente ao sub-problema de roteamento, é necessário tratar o sub-problema de sequenciamento no qual foi apresentado na seção anterior. Neste trabalho, o processo aplicado pela abelha espectadora para tratar o sequenciamento das operações (etapa 2) é o mesmo adotado pela abelha empregada.

4.1.6 Fase da abelha exploradora

Nesta abordagem, a abelha exploradora segue as mesmas características apresentadas na versão inicial da abordagem ABC proposta por Karaboga (2005). Com a geração randômica de uma nova solução, torna-se possível a exploração de soluções ainda não visitadas no espaço de busca do problema tratado como um todo. A nova solução gerada deve estar conforme a modelagem ilustrada na Figura 4.2 respeitando as restrições impostas para o problema estudado neste trabalho.

4.2 Validação da Proposta

A validação deste trabalho, ocorrerá por meio de comparações entre os resultados gerados pela abordagem proposta com os resultados apresentados por outras abordagens encontradas

na literatura. Para poder realizar as comparações, serão utilizados *benchmarks* conhecidos na literatura já utilizados em outros trabalhos referentes a atividade de programação em ambientes do tipo JSF. Os *benchmarks* utilizados são fornecidos por: Kacem, Hammadi e Borne (2002) e Brandimarte (1993). O *benchmark* de Kacem, Hammadi e Borne (2002), fornece ambientes que variam entre 5 a 10 máquinas, 4 a 15 *jobs* e 12 a 56 operações. O *benchmark* de Brandimarte (1993), fornece ambientes que variam entre 6 a 15 máquinas, 10 a 20 *jobs* e 55 a 240 operações.

4.3 Delimitações do Trabalho

As seguintes restrições a seguir são impostas aos ambientes dos *benchmarks* utilizados para estudo:

- Cada máquina suporta processar apenas uma operação por vez;
- As máquinas não quebram;
- As matérias primas referentes as operações dos *jobs* são ilimitadas;
- Não ocorrerá preempções (i.e., uma máquina não pode parar de processar uma operação durante o processo e iniciar outra sem que está esteja finalizada) entre as operações dos *jobs* ou mesmo entre os *jobs*;
- A ordem de execução das operações é fixa e não pode ser alterada;

Capítulo 5

ANÁLISE DE RESULTADOS E DISCUSSÃO

Demonstra-se por meio deste capítulo, os testes realizados para avaliar a abordagem proposta nesse trabalho em relação as outras encontradas na literatura. Para a realização dos testes efetuados nesse trabalho, foram utilizados os *benchmarks* desenvolvidos por: Kacem, Hammadi e Borne (2002) e Brandimarte (1993). O *benchmark* desenvolvido por Kacem, Hammadi e Borne (2002), fornece ambientes para simulação que variam entre 5 a 10 máquinas, 4 a 15 *jobs* e 12 a 56 operações. O *benchmark* desenvolvido por Brandimarte (1993), fornece ambientes que variam entre 6 a 15 máquinas, 10 a 20 *jobs* e 55 a 240 operações.

A abordagem proposta e os testes realizados nesse trabalho, foram implementados utilizando a linguagem de programação Julia. A linguagem Julia foi adotada devido as suas características computacionais sendo elas simplicidade e possuir diversas funções matemáticas que facilitam o desenvolvimento entre outras razões que podem ser encontradas no site oficial da linguagem em (https://julialang.org) além de ser uma linguagem *open source*.

Para cada instância dos *benchmarks* utilizados a abordagem será executada 30 vezes consecutivas. Essa quantidade de execuções que a abordagem será executada foi definida com base nos trabalhos estudados e apresentados no capítulo de revisão da literatura. Os testes foram executados sobre o ambiente computacional Intel(R) Core(TM)i5-4210U CPU @1.70GHz 2.40GHz.

Os valores adotados para a configuração dos parâmetros da abordagem proposta foram definidos de modo manual (i.e., tentativas com valores alternativos) e depois de diversos testes realizados os valores foram definidos para cada instância com base na configuração que apresentaram melhores resultados para cada instância.

Nas seções a seguir os gráficos de Gantt apresentados paras a instâncias dos *benchmarks* utilizados foram ilustrados somente para as instâncias de Kacem, Hammadi e Borne (2002)

devido ao valor de *makespan* baixo obtido nas instâncias de Kacem, Hammadi e Borne (2002). Nas instâncias de Brandimarte (1993) os valores de *makespan* foram grandes em comparação as instâncias de Kacem, Hammadi e Borne (2002). O valor muito grande de *makespan* dificulta na visualização das programações de forma visual por meio da ferramenta gráfico de Gantt. Então o critério para exibição dos gráficos de Gantt foi realizado com base no menor valor de *makespan* C_m seguido do critério W_m seguido pelo critério W_t obtidos pela abordagem proposta.

A seguir são apresentadas as instâncias de problemas fornecidas por Kacem, Hammadi e Borne (2002) e Brandimarte (1993) utilizadas para avaliação da abordagem proposta e os seguintes objetivos foram utilizados *makespan* (Cm), tempo de trabalho máximo das máquinas (Wm) e o tempo total de produção de todas as máquinas (Wt).

5.1 Instância A fornecida por Kacem, Hammadi e Borne (2002)

Essa instância segundo Kacem, Hammadi e Borne (2002) é considerada de pequena escala e ela é composta por 4 *jobs*, 5 máquinas e possui 12 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 2 minutos para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.1 com os respectivos valores adotados para cada parâmetro utilizado nessa abordagem.

Tabela 5.1: Configuração dos parâmetros da abordagem proposta para a Instância A

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
10	10	10	30	50

Na Tabela 5.2 é apresentado o resultado obtido pela abordagem proposta em comparação com as abordagens dos autores Zhang et al. (2009) *Particle Swarm Optimization and Tabu Search* (PSO+TS), Xing, Chen e Yang (2009c) *Simulation Modeling* (SM), Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) e Li, Pan e Gao (2011) *Pareto-based Discrete Artificial Bee Colony algorithm* (PDABC).

Na Tabela 5.2 é possível observar por meio da cor dos blocos mais escuros que todas as abordagens utilizadas nessa instância almejaram resultados bastante similares com relação aos melhores resultados. A abordagem PDABC atingiu o menor resultado no critério W_m, em contrapartida, os resultados dos outros objetivos dessa mesma solução foram ruins, se comparado com os resultados das demais abordagens nessa tabela.

Na Tabela 5.3 ilustra-se que para cada objetivo do JSF da "Instância A" a abordagem manteve um padrão de resultados consistente no qual é possível observar que não ocorreu nenhuma

Tabela 5.2: Comparação de resultados da Instância A - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
PSO+TS	11	10	32
SM	12	8	32
HTSA	11	10	32
	12	8	32
PDABC	11	9	33
	11	10	32
	12	8	32
	13	7	33
Abordagem proposta	11	10	32

alteração durante o processo de execução da abordagem.

Tabela 5.3: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância A

Critérios de Desempenho	Média	Desvio Padrão
C_m	11	0
W_m	10	0
W_t	32	0

Na Figura 5.1 ilustra-se o resultado obtido pela abordagem proposta na forma de um gráfico de Gantt, o que possibilita de forma visual observar o fluxo de execução das operações dos *jobs* nas máquinas referentes ao problema da "Instância A".

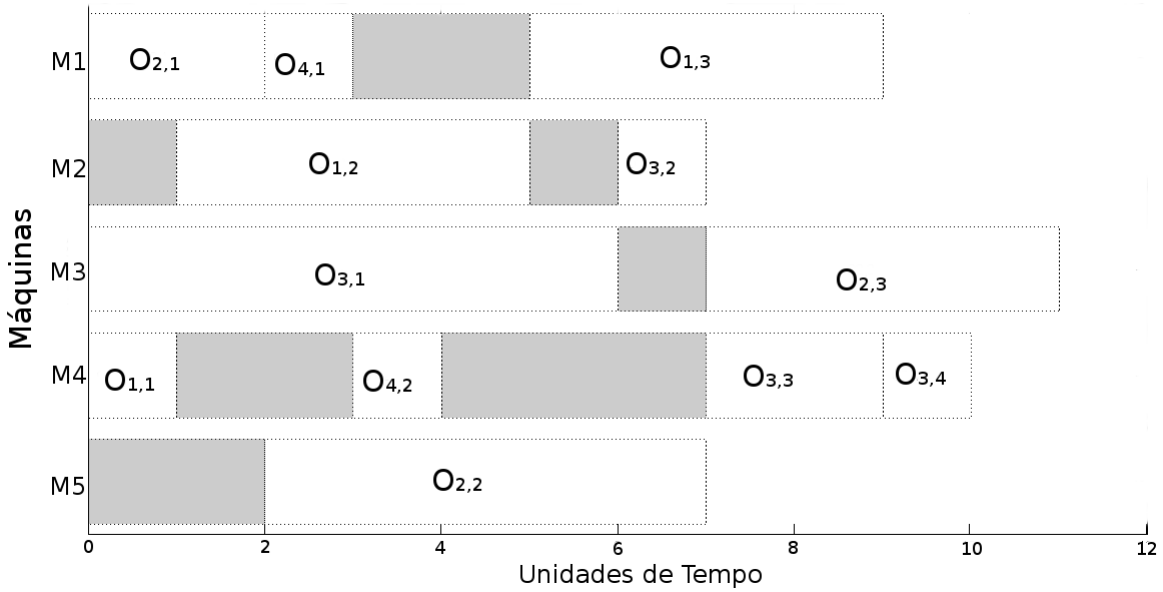

Figura 5.1: Programação da Instância A obtida pela abordagem proposta com os seguintes resultados ($C_m = 11, W_m = 10, W_t = 32$)

5.2 Instância B fornecida por Kacem, Hammadi e Borne (2002)

Essa instância segundo Kacem, Hammadi e Borne (2002) é considerada de média escala e ela é composta por 8 *jobs*, 8 máquinas e possui 27 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 2 minutos para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.4 com os respectivos valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.4: Configuração dos parâmetros da abordagem proposta para a Instância B

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
20	10	5	30	50

Na Tabela 5.5 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Zhang et al. (2009) *Particle Swarm Optimization and Tabu Search* (PSO+TS), Xing, Chen e Yang (2009c) *Simulation Modeling* (SM), Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) e Li, Pan e Gao (2011) *Pareto-based Discrete Artificial Bee Colony algorithm* (PDABC), Xia e Wu (2005) *Particle Swarm Optimization and Simulated Annealing* (PSO+SA), Moslehi e Mahnam (2011) *Multi-Objective Particle Swarm Optimization and Local Search* (MOPSO+LS) e Li, Pan e Tasgetiren (2014) *Discrete Artificial Bee Colony algorithm* (DABC).

Na Tabela 5.5 é possível observar que o resultado ($C_m = 15, W_m = 12, W_t = 75$) é bastante comum e aparece para todas as abordagens utilizadas nessa instância. Com relação aos conjuntos de objetivos é possível notar que todas as abordagens apresentaram boas soluções, mas avaliando de forma separada cada objetivo observa-se que a abordagem proposta apresenta superioridade no objetivo C_m em relação a abordagem PSO+SA. Em relação ao objetivo W_m as abordagens DABC e MOPSO+LS apresentaram melhores resultados, mas nessas soluções os outros objetivos apresentaram resultados ruins (i.e., com maiores valores de unidade de tempo) se comparado as outras soluções encontradas pelas outras abordagens incluindo a proposta nesse trabalho. Em relação ao objetivo W_t a abordagem proposta apresentou melhor resultado em comparação as abordagens PSO+TS, SM e HTSA, mas assim como mencionado no objetivo anterior, os resultados dos outros objetivos para essas soluções foi pior se comparado as outras soluções encontradas para as abordagens utilizadas nessa instância para comparação.

Tabela 5.5: Comparação de resultados da Instância B - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
PSO+SA	15	12	75
	16	13	73
PSO+TS	14	12	77
	15	12	75
SM	14	12	77
	15	12	75
HTSA	14	12	77
	15	12	75
PDABC	14	12	77
	15	12	75
	16	13	73
DABC	14	12	77
	15	12	75
	16	13	73
	16	11	77
MOPSO+LS	14	12	77
	15	12	75
	16	13	73
	16	11	78
Abordagem proposta	14	12	77
	15	12	75
	16	13	73

Na Tabela 5.6 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que a abordagem não distanciou-se muito dos bons resultados desse problema.

Na Figura 5.2 ilustra-se o resultado ($C_m = 14, W_m = 12, W_t = 77$) obtido pela abordagem proposta na forma de um gráfico de Gantt, o que possibilita de forma visual observar o fluxo de execução das operações dos *jobs* nas máquinas referentes ao problema da "Instância B".

Tabela 5.6: Informações estatísticas das 30 execuções para cada critério de desempenho na instância B

Critérios de Desempenho	Média	Desvio Padrão
C_m	14,233	0,568
W_m	12,067	0,254
W_t	76,533	1,137

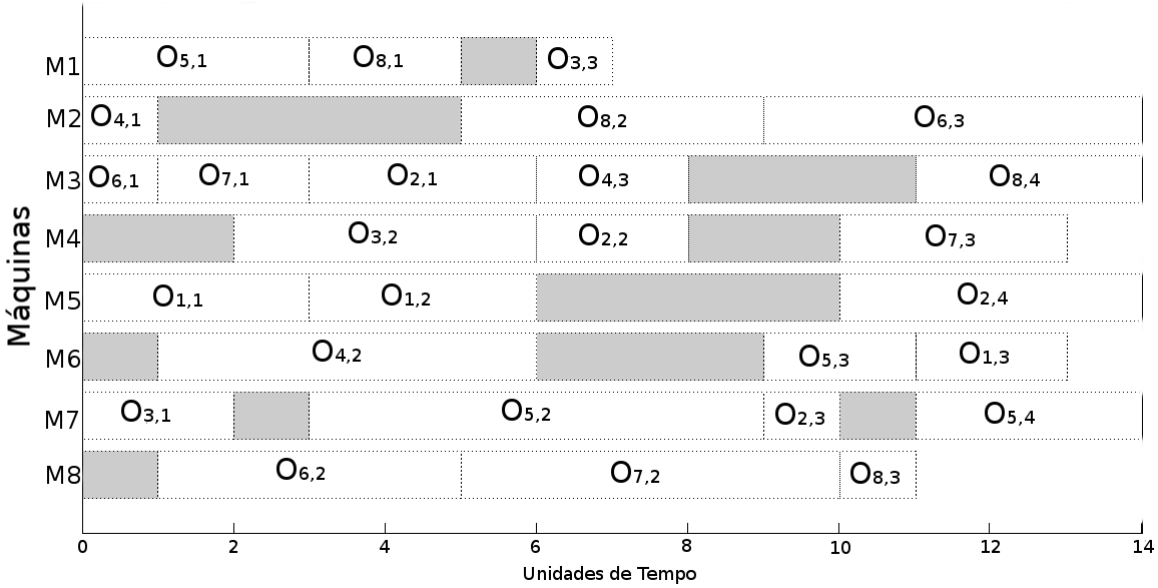

Figura 5.2: Programação do resultado ($C_m = 14, W_m = 12, W_t = 77$) para a Instância B obtida pela abordagem proposta

5.3 Instância C fornecida por Kacem, Hammadi e Borne (2002)

Essa instância, segundo Kacem, Hammadi e Borne (2002) é considerada de média escala e ela é composta por 10 *jobs*, 7 máquinas e possui 29 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 2 minutos para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.7 com os respectivos valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.7: Configuração dos parâmetros da abordagem proposta para a Instância C

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
20	10	5	30	50

Na Tabela 5.8 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Xing, Chen e Yang (2009c) *Simulation Modeling* (SM), Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) e Li, Pan e Gao (2011) *Pareto-*

based Discrete Artificial Bee Colony algorithm (PDABC) e Li, Pan e Tasgetiren (2014) *Discrete Artificial Bee Colony algorithm* (DABC).

Na Tabela 5.8 analisando os conjuntos de objetivos de todas as soluções é possível notar que, os resultados de as todas abordagens foram bastante similares quando analisados em conjunto. Separadamente, iniciando-se uma análise a partir do objetivo C_m, é possível observar que todas abordagens almejaram o melhor resultado. No objetivo W_m é possível observar que a abordagem proposta apresentou melhor resultado nesse objetivo quando comparada a abordagem PDABC. Em relação ao objetivo W_t é possível observar que os melhores resultados foram almejados pelas abordagens PDABC e DABC. Vale ressaltar que, para o objetivo W_t das abordagens PDABC e DABC, os outros objetivos apresentam resultados muito piores em relação as outras soluções das demais abordagens comparadas incluindo a abordagem proposta.

Tabela 5.8: Comparação de resultados da Instância C - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
SM	11	10	62
	11	11	61
HTSA	11	10	62
	11	11	61
PDABC	12	11	61
	11	11	63
	12	12	60
DABC	11	10	62
	11	11	61
	12	12	60
Abordagem proposta	11	10	62
	11	11	61
	12	11	61

Na Tabela 5.9 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que a abordagem não distanciou-se muito dos bons resultados desse problema.

Tabela 5.9: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância C

Critérios de Desempenho	Média	Desvio Padrão
C_m	11,033	0,183
W_m	10,933	0,254
W_t	61,067	0,254

Na Figura 5.3 ilustra-se a programação do resultado ($C_m = 11, W_m = 10, W_t = 62$) obtido

pela abordagem proposta por meio do gráfico de Gantt.

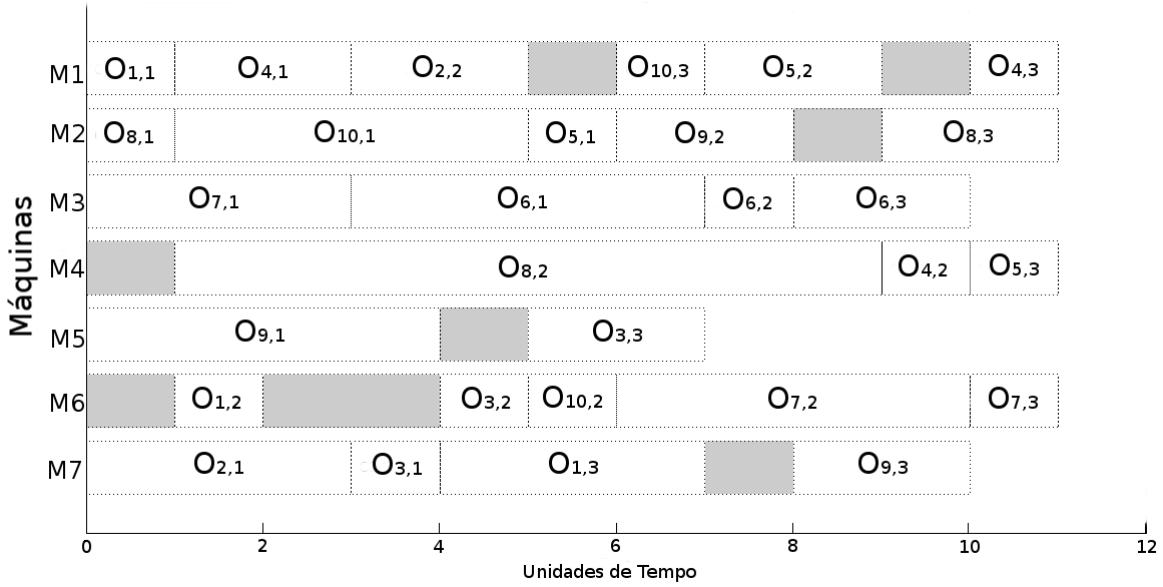

Figura 5.3: Programação do resultado ($C_m = 11, W_m = 10, W_t = 62$) **para a Instância C obtida pela abordagem proposta**

5.4 Instância D fornecida por Kacem, Hammadi e Borne (2002)

Essa instância segundo Kacem, Hammadi e Borne (2002) é considerada de média escala e ela é composta por 10 *jobs*, 10 máquinas e possui 30 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 13 minutos para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.10 com os respectivos valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.10: Configuração dos parâmetros da abordagem proposta para a Instância D

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
100	10	5	20	100

Na Tabela 5.11 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Zhang et al. (2009) *Particle Swarm Optimization and Tabu Search* (PSO+TS), Xing, Chen e Yang (2009c) *Simulation Modeling* (SM), Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) e Li, Pan e Gao (2011) *Pareto-based Discrete Artificial Bee Colony algorithm* (PDABC), Xia e Wu (2005) *Particle Swarm Optimization and Simulated Annealing* (PSO+SA), Moslehi e Mahnam (2011) *Multi-Objective Particle Swarm Optimization and Local Search* (MOPSO+LS) e Li, Pan e Tasgetiren (2014) *Discrete Artificial Bee Colony algorithm* (DABC).

Na Tabela 5.11 é possível observar que todas as abordagens apresentaram bons resultados ao tratar o problema quando comparados os objetivos de cada solução como um conjunto. Separadamente, no objetivo C_m todas as abordagens alcançaram o melhor resultado para esse objetivo. No objetivo W_m a abordagem proposta obteve melhor resultado quando comparada as abordagens PSO+SA e PSO+TS. No objetivo W_t as abordagens PDABC, MOPSO+LS e DABC obtiveram melhor resultado em relação as demais abordagens. No caso do objetivo W_t com melhores resultados vale ressaltar que os outros objetivos tiveram piores resultados quando comparados a outras soluções das demais abordagens. Ainda nessa instância quando comparado os resultados da abordagem proposta com a abordagem PSO+SA fica evidente que a abordagem proposta é superior a PSO+SA quando os objetivos são analisados de forma conjunta no caso com relação a primeira solução da abordagem proposta tem-se W_m e W_t sendo melhores que os resultados obtidos pela abordagem PSO+SA e com relação a segunda solução encontrada pela abordagem proposta tem-se o objetivo W_t sendo superior ao da abordagem PSO+SA.

Analisando os resultados dos objetivo de forma conjunta da abordagem PSO+TS também é possível notar determinada superioridade da abordagem proposta quando comparada a ela. Com relação a primeira solução da abordagem proposta tem-se o objetivo W_t superior ao da abordagem PSO+TS. Em comparação a segunda solução da abordagem proposta tem-se W_m com melhor resultado quando comparado ao da abordagem PSO+TS.

Na Tabela 5.12 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que a abordagem não distanciou-se muito dos bons resultados dessa instância.

Tabela 5.12: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância D

Critérios de Desempenho	Média	Desvio Padrão
C_m	7	0
W_m	5,367	0,490
W_t	42,633	0,490

Na Figura 5.4 ilustra-se a programação do resultado ($C_m = 7, W_m = 5, W_t = 43$) obtido pela abordagem proposta por meio do gráfico de Gantt.

Tabela 5.11: Comparação de resultados da Instância D - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
PSO+SA	7	6	44
PSO+TS	7	6	43
SM	7	6	42
	8	5	42
HTSA	7	5	43
	7	6	42
	8	5	42
PDABC	7	5	43
	8	7	41
	8	5	42
MOPSO+LS	7	5	43
	7	6	42
	8	5	42
	8	7	41
DABC	7	5	43
	7	6	42
	8	5	42
	8	7	41
Abordagem proposta	7	5	43
	7	6	42

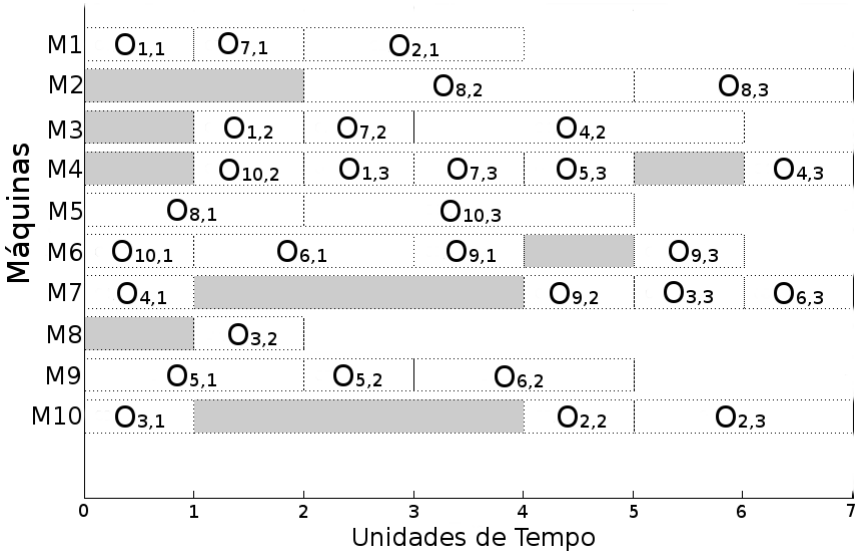

Figura 5.4: Programação do resultado ($C_m = 7, W_m = 5, W_t = 43$) para a Instância D obtida pela abordagem proposta

5.5 Instância E fornecida por Kacem, Hammadi e Borne (2002)

Essa instância é considerada de grande escala pois é composta por 15 *jobs*, 10 máquinas e possui 56 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 5 horas para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.13 com os respectivos valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.13: Configuração dos parâmetros da abordagem proposta para a Instância E

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
100	100	3	10	800

Na Tabela 5.14 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Zhang et al. (2009) *Particle Swarm Optimization and Tabu Search* (PSO+TS), Xing, Chen e Yang (2009c) *Simulation Modeling* (SM), Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) e Li, Pan e Gao (2011) *Pareto-based Discrete Artificial Bee Colony algorithm* (PDABC), Xia e Wu (2005) *Particle Swarm Optimization and Simulated Annealing* (PSO+SA), Moslehi e Mahnam (2011) *Multi-Objective Particle Swarm Optimization and Local Search* (MOPSO+LS) e Li, Pan e Tasgetiren (2014) *Discrete Artificial Bee Colony algorithm* (DABC).

Na Tabela 5.14 é possível notar que, a abordagem proposta apresentou melhores resultados que as abordagens PSO+SA e PSO+TS quando comparadas as soluções em conjunto. Separadamente ao analisar os objetivos é possível observar que, para o objetivo C_m, a abordagem proposta apresentou resultado superior em relação a abordagem PSO+SA. No objetivo W_m a abordagem proposta apresentou-se superior em comparação as abordagens PSO+SA, PSO+TS e PDABC. Em relação ao objetivo W_t a abordagem proposta apresentou resultado superior somente em comparação a abordagem PSO+TS.

Na Tabela 5.15 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que, os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que os resultados são bastante consistentes e apresentam pouca variação. Para tanto a abordagem proposta distância-se muito pouco da média de cada objetivo.

Tabela 5.14: Comparação de resultados da Instância E - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
PSO+SA	12	11	91
PSO+TS	11	11	93
SM	11	10	93
	11	11	91
HTSA	11	10	93
	11	11	91
PDABC	12	11	91
	11	11	93
DABC	11	10	93
	11	11	91
MOPSO+LS	11	11	91
	12	10	93
Abordagem proposta	11	10	93
	11	10	94
	11	11	91
	12	11	91

Tabela 5.15: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância E

Critérios de Desempenho	Média	Desvio Padrão
C_m	11,800	0,407
W_m	10,867	0,346
W_t	91,300	0,702

Na Figura 5.5 ilustra-se a programação do resultado ($C_m = 11, W_m = 10, W_t = 93$) obtido pela abordagem proposta por meio do gráfico de Gantt.

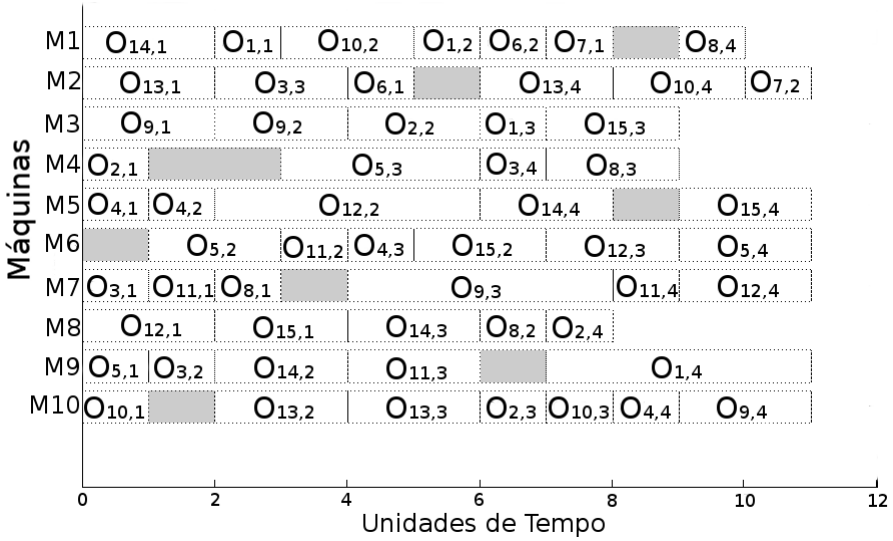

Figura 5.5: Programação do resultado ($C_m = 11, W_m = 10, W_t = 93$) para a instância E obtida pela abordagem proposta

5.6 Instância MK01 fornecida por Brandimarte (1993)

Essa instância é considerada de grande escala e ela é composta por 10 *jobs*, 6 máquinas e possui 55 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 19 minutos para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.16 com os respectivos valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.16: Configuração dos parâmetros da abordagem proposta para a Instância E

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
50	10	5	30	100

Na Tabela 5.17 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) , *Discrete Artificial Bee Colony algorithm* (DABC), Xing, Chen e Yang (2009a) *Xing Search Method* (X-SM), Bagheri et al. (2010) *Artificial Immune Algorithm* (AIA) e (GAO et al., 2007) *Hybrid of Genetic Algorithm* (hGA).

Na Tabela 5.17 é possível observar que analisando os objetivos de modo conjunto a abordagem proposta apresentou melhores resultados quando comparada as abordagens X-SM e AIA. Separadamente, é possível notar que no objetivo C_m, a abordagem proposta apresentou melhores

resultados quando comparada a abordagem X-SM. No objetivo W_m a abordagem proposta também gerou melhores resultados quando comparada a abordagem X-SM. Em relação ao objetivo W_t nota-se que os melhores resultados foram obtidos pelas abordagens X-SM e DABC.

Tabela 5.17: Comparação de resultados da Instância MK01 - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
X-SM	42	42	162
AIA	40	36	171
hGA	40	36	167
HTSA	40	36	167
DABC	40	36	167
	40	38	162
	40	37	164
Abordagem proposta	40	36	167
	40	36	168
	42	42	163
	42	42	164

Na Tabela 5.18 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que, os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que os resultados são bastante consistentes e apresentam pouca variação. Para tanto a abordagem proposta distância-se muito pouco da média de cada objetivo.

Tabela 5.18: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância MK01

Critérios de Desempenho	Média	Desvio Padrão
C_m	40,400	0,184
W_m	37,000	2,274
W_t	166,433	1,501

5.7 Instância MK08 fornecida por Brandimarte (1993)

Essa instância é considerada de grande escala e ela é composta por 20 *jobs*, 10 máquinas e possui 225 operações.

Nessa instância o tempo computacional total gasto foi de aproximadamente 9 horas para realizar as 30 execuções por meio da configuração apresentada na Tabela 5.19 com os respectivos

valores adotados para cada parâmetro da abordagem proposta.

Tabela 5.19: Configuração dos parâmetros da abordagem proposta para a Instância MK08

População	Limite	Taxa de mutação (%)	Número de iterações do vetor de estruturas de vizinhança	NMCF
50	10	3	30	700

Na Tabela 5.20 são apresentados os resultados obtidos pela abordagem proposta em comparação com as abordagens dos autores Li, Pan e Liang (2010a) *Hybrid Tabu Search Algorithm* (HTSA) , *Discrete Artificial Bee Colony algorithm* (DABC), Xing, Chen e Yang (2009a) *Xing Search Method* (X-SM), Bagheri et al. (2010) *Artificial Immune Algorithm* (AIA) e (GAO et al., 2007) *Hybrid of Genetic Algorithm* (hGA).

Tabela 5.20: Comparação de resultados da Instância MK08 - os objetivos onde cada bloco estiver com a cor de fundo mais escura representam os melhores resultados de cada objetivo individualmente

Abordagens	C_m	W_m	W_t
X-SM	523	523	2524
AIA	523	523	2723
hGA	523	523	2524
HTSA	523	523	2524
DABC	523	523	2524
	524	524	2519
Abordagem proposta	253	523	2524
	523	523	2526

Na Tabela 5.20 é possível observar que a abordagem proposta apresenta um melhor conjunto de objetivos quando comparada com a abordagem AIA. Analisando os objetivos separadamente é possível verificar que nos objetivos C_m e W_m todas as abordagens apresentaram os mesmos valores ótimos para essa instância. Em relação ao objetivo W_t a abordagem DABC obteve o melhor resultado em comparação as demais abordagens. Com relação a abordagem proposta é possível observar que no objetivo W_t a abordagem proposta apresentou melhor resultado quando comparada com a abordagem AIA.

Na Tabela 5.21 por meio da média e do desvio padrão obtidos dos objetivos é possível notar que, os valores alcançados durante as 30 execuções estão muito próximos aos melhores resultados de cada objetivo separado e também que os resultados são bastante consistentes e apresentam pouca variação. Para tanto a abordagem proposta distância-se muito pouco da média de cada objetivo menos em relação ao W_t que apresentou uma dispersão alta em relação aos outros objetivos estudados.

Tabela 5.21: Informações estatísticas das 30 execuções para cada critério de desempenho na Instância MK08

Critérios de Desempenho	Média	Desvio Padrão
C_m	523,167	0,379
W_m	523,000	0,000
W_t	2526,367	5,255

Capítulo 6

CONSIDERAÇÕES FINAIS

Na área de planejamento e controle da produção, o problema de programação de operações do tipo JSF, tem sido foco de estudo por diversos pesquisadores e tem chamado bastante a atenção devido a sua importância na área industrial e também por sua característica NP-Difícil que explicitamente demonstra a dificuldade de resolução deste problema em relação a resultados e custo computacional.

Neste trabalho, foi proposta e apresentada uma abordagem ABC para tratar o problema de programação supracitado. Na abordagem proposta, foi utilizado um vetor de estruturas de vizinhanças adaptativo que define uma forma de tratar o sub-problema de programação das operações nas máquinas (i.e., atividade de sequenciamento) de acordo com a eficácia da estrutura selecionada para está atividade. Por meio desse vetor adaptativo com diferentes estruturas de vizinhança, a abordagem proposta conseguiu atingir nos testes realizados os resultados ótimos mais atuais no que diz respeito aos objetivos relacionados a minimização dos critérios *makespan* (C_m) e tempo de produção gasto nas máquinas (W_m) assim como esperado. Outro ponto importante nessa abordagem, está na utilização de um operador genético de mutação, que teve a finalidade de realizar o roteamento das máquinas nas operações na fase da abelha espectadora. O operador de mutação realizou a atividade de definir qual máquina iria processar qual operação (i.e., atividade de roteamento). Por meio desse operador genético, a abordagem foi capaz de minimizar o tempo total de produção (W_t) assim como esperado. A utilização desse operador foi crucial na fase da abelha espectadora porque, na fase dessa abelha, aleatoriamente uma solução dentre um conjunto era escolhida de acordo com sua qualidade e essas soluções (fontes de alimento) eram então, alteradas pelos métodos propostos supracitados. Além dessas atualizações na meta-heurística ABC proposta, foi utilizado o método dominância de Pareto para tratar os conflitos de multiobjetivo do problema estudado.

A abordagem proposta, foi avaliada e comparada seguindo tendências atuais (i.e., com-

paração de resultados) devido a dificuldade de reprodução de outros trabalhos. Para os testes realizados, foram utilizadas diferentes escalas do problema em estudo para se obter uma melhor avaliação sobre a abordagem proposta. Nas comparações realizadas, é demonstrada a eficácia da abordagem proposta em relação as demais abordagens utilizadas e a sua superioridade em relação a algumas das abordagens utilizadas para comparação.

6.1 Trabalhos Futuros

Com base nos bons resultados apresentados pela abordagem proposta, seria interessante a sua utilização e comparação com outras abordagens em outros problemas de carácter específicos ligados a manufatura flexível como, por exemplo, aqueles relacionados a manutenção preventiva de máquinas (WANG; YU, 2010) e também de reprogramação da produção (GAO et al., 2015). Outro fator interessante a ser estudado futuramente, seria o acréscimo de mais objetivos ao problema estudado neste trabalho.

Outro trabalho futuro estendido desta pesquisa, poderia ser a atualização e adição de novas estruturas de vizinhança ao vetor de vizinhanças proposto nesse trabalho.

REFERÊNCIAS

ABBASS, H. A. Mbo: Marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In: IEEE. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. 2001. v. 1, p. 207–214.

ABBASS, H. A. A single queen single worker honey bees approach to 3-sat. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2001. p. 807–814.

ABBASS, H. A. An agent based approach to 3-sat using marriage in honey-bees optimization. *INTERNATIONAL JOURNAL OF KNOWLEDGE BASED INTELLIGENT ENGINEERING SYSTEMS*, UNKNOWN, v. 6, n. 2, p. 64–71, 2002.

ADAMS, J. et al. Estimation of the number of sex alleles and queen matings from diploid male frequencies in a population of apis mellifera. *Genetics*, Genetics Soc America, v. 86, n. 3, p. 583–596, 1972.

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, Elsevier, v. 187, n. 3, p. 985–1032, 2008.

ARNOLD, J. Tony. *Administração de materiais: uma introdução. São Paulo: Atlas*, 1999.

BAGHERI, A. et al. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Generation Computer Systems*, Elsevier, v. 26, n. 4, p. 533–541, 2010.

BANHARNSAKUN, A.; SIRINAOVAKUL, B.; ACHALAKUL, T. Job shop scheduling with the best-so-far abc. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 25, n. 3, p. 583–593, 2012.

BRANDIMARTE, P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, Springer, v. 41, n. 3, p. 157–183, 1993.

BRUCKER, P.; BRUCKER, P. *Scheduling algorithms*. : Springer, 2007.

CHAUDHRY, I. A.; KHAN, A. A. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, Wiley Online Library, v. 23, n. 3, p. 551–591, 2016.

CHIANG, T.-C.; LIN, H.-J. A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics*, Elsevier, v. 141, n. 1, p. 87–98, 2013.

COLLETTE, Y.; SIARRY, P. *Multiobjective optimization: principles and case studies*. : Springer Science & Business Media, 2013.

CORRÊA, H. L. Administração de produção e operações: manufatura e serviços: uma abordagem estratégica/henrique l. *Corrêa, Carlos A. Corrêa.–2ª Ed. São Paulo: Atlas*, 2006.

DEB, K. *Multi-objective optimization using evolutionary algorithms*. : John Wiley & Sons, 2001.

DRIAS, H.; SADEG, S.; YAHI, S. Cooperative bees swarm for solving the maximum weighted satisfiability problem. In: SPRINGER. *International Work-Conference on Artificial Neural Networks*. 2005. p. 318–325.

FERNANDES, F.; FILHO, M. *Planejamento e controle da produção: dos fundamentos ao essencial*. : Atlas, 2010. ISBN 9788522458714.

FILIPPINI, R. Trade-off and compatibility between performance: definitions and empirical evidence. *International Journal of Production Research*, Taylor & Francis, v. 36, n. 12, p. 3379–3406, 1998.

FRAZIER, G.; GAITHER, N. Administração da produção e operações, 8ª edição. *São Paulo-SP, editora Pioneira*, 2005.

GAO, J. et al. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, Elsevier, v. 53, n. 1, p. 149–162, 2007.

GAO, K. Z. et al. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert Systems with Applications*, Elsevier, v. 42, n. 21, p. 7652–7663, 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, p. 117–129, 1976.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of metaheuristics*. : Springer Science & Business Media, 2006.

HOLLAND, J. H. Genetic algorithms. *Scientific american*, v. 267, n. 1, p. 66–72, 1992.

KACEM, I.; HAMMADI, S.; BORNE, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, IEEE, v. 32, n. 1, p. 1–13, 2002.

KARABOGA, D. *An idea based on honey bee swarm for numerical optimization*. 2005.

KARABOGA, D.; AKAY, B. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, Elsevier, v. 214, n. 1, p. 108–132, 2009.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, Springer, v. 39, n. 3, p. 459–471, 2007.

KARABOGA, D.; BASTURK, B. On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, Elsevier, v. 8, n. 1, p. 687–697, 2008.

KARABOGA, D. et al. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, Springer, v. 42, n. 1, p. 21–57, 2014.

KARABOGA, N. A new design method based on artificial bee colony algorithm for digital iir filters. *Journal of the Franklin Institute*, Elsevier, v. 346, n. 4, p. 328–348, 2009.

KRAUSE, D. R.; PAGELL, M.; CURKOVIC, S. Toward a measure of competitive priorities for purchasing. *Journal of Operations Management*, Elsevier, v. 19, n. 4, p. 497–512, 2001.

LEI, D. A genetic algorithm for flexible job shop scheduling with fuzzy processing time. *International Journal of Production Research*, Taylor & Francis, v. 48, n. 10, p. 2995–3013, 2010.

LI, J.; PAN, Q.; XIE, S. Flexible job shop scheduling problems by a hybrid artificial bee colony algorithm. In: IEEE. *Evolutionary Computation (CEC), 2011 IEEE Congress on*. 2011. p. 78–83.

LI, J.-Q.; PAN, Q.-K.; GAO, K.-Z. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 55, n. 9-12, p. 1159–1169, 2011.

LI, J.-q.; PAN, Q.-k.; LIANG, Y.-C. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, Elsevier, v. 59, n. 4, p. 647–662, 2010.

LI, J.-q.; PAN, Q.-k.; LIANG, Y.-C. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, Elsevier, v. 59, n. 4, p. 647–662, 2010.

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, p. 1111–1132, 2014.

LI, J.-q. et al. A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. *International Journal of Computers Communications & Control*, v. 6, n. 2, p. 286–296, 2011.

LIU, Z. et al. Solving multi-objective flexible job shop scheduling with transportation constraints using a micro artificial bee colony algorithm. In: IEEE. *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*. 2013. p. 427–432.

MONTGOMERY, D. C. *Design and analysis of experiments*. : John Wiley & Sons, 2008.

MOSLEHI, G.; MAHNAM, M. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, Elsevier, v. 129, n. 1, p. 14–22, 2011.

NAJID, N.; DAUZERE-PERES, S.; ZAIDAT, A. A modified simulated annealing method for flexible job shop scheduling problem. In: IEEE. *Systems, Man and Cybernetics, 2002 IEEE International Conference on*. 2002. v. 5, p. 6–pp.

PEZZELLA, F.; MORGANTI, G.; CIASCHETTI, G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, Elsevier, v. 35, n. 10, p. 3202–3212, 2008.

PHAM, D. et al. The bees algorithm–a novel tool for complex optimisation. In: *Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference (3-14 July 2006)*. 2006.

PINEDO, M. Scheduling: theory, algorithms, and systems. *Prentice Hall, USA*, 2002.

PULIDO, G. T.; COELLO, C. A. C. The micro genetic algorithm 2: towards online adaptation in evolutionary multiobjective optimization. In: SPRINGER. *International Conference on Evolutionary Multi-Criterion Optimization*. 2003. p. 252–266.

RINGUEST, J. L. *Multiobjective optimization: behavioral and computational considerations*. : Springer Science & Business Media, 2012.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. *Administração da produção*. : Atlas, 2009.

TEODOROVIĆ, D.; DELL'ORCO, M. Bee colony optimization–a cooperative learning approach to complex transportation problems. In: *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*. 2005. p. 51–60.

TUBINO, D. F. *Planejamento e controle da produção: teoria e prática*. : Editora Atlas SA, 2009.

WANG, L. et al. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 60, n. 1-4, p. 303–315, 2012.

WANG, L. et al. An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 60, n. 9-12, p. 1111–1123, 2012.

WANG, L. et al. A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. *International Journal of Production Research*, Taylor & Francis, v. 51, n. 12, p. 3593–3608, 2013.

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

WANG, X. et al. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 51, n. 5-8, p. 757–767, 2010.

WEDDE, H. F.; FAROOQ, M.; ZHANG, Y. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In: SPRINGER. *International Workshop on Ant Colony Optimization and Swarm Intelligence*. 2004. p. 83–94.

XIA, W.; WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, Elsevier, v. 48, n. 2, p. 409–425, 2005.

XING, L.-N. et al. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, Elsevier, v. 10, n. 3, p. 888–896, 2010.

XING, L.-N.; CHEN, Y.-W.; YANG, K.-W. An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, Springer, v. 20, n. 3, p. 283–293, 2009.

XING, L.-N.; CHEN, Y.-W.; YANG, K.-W. Multi-objective flexible job shop schedule: design and evaluation by simulation modeling. *Applied Soft Computing*, Elsevier, v. 9, n. 1, p. 362–376, 2009.

XING, L.-N.; CHEN, Y.-W.; YANG, K.-W. Multi-objective flexible job shop schedule: design and evaluation by simulation modeling. *Applied Soft Computing*, Elsevier, v. 9, n. 1, p. 362–376, 2009.

YANG, X.-S. Engineering optimizations via nature-inspired virtual bee algorithms. In: SPRINGER. *International Work-Conference on the Interplay Between Natural and Artificial Computation*. 2005. p. 317–323.

ZHANG, G. et al. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, Elsevier, v. 56, n. 4, p. 1309–1318, 2009.

ZHOU, G. et al. An effective artificial bee colony algorithm for multi-objective flexible job-shop scheduling problem. In: *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*. : Springer, 2011. p. 1–8.

GLOSSÁRIO

ABC+TS – *Artificial Bee Colony Algorithm and Tabu Search*

ABC – *Algoritmo de Colônia de Abelhas*

AGV – *Automated Guided Vehicle*

AIA – *Artificial Immune Algorithm*

AL+CGA – *Approach by Localization and Controlled Genetic Algorithm*

AL – *Approach by Localization*

BA – *Bees Algorithm*

BCO – *Bee Colony Optimization*

CGA – *Controlled Genetic Algorithm*

DABC – *Discrete Artificial Bee Colony Algorithm*

DIGA – *Decomposition Integration Genetic Algorithm*

EABC – *Effective Artificial Bee Colony Algorithm*

EPABC – *Enhanced Pareto-based Artificial Bee Colony Algorithm*

HABC – *Hybrid Artificial Bee Colony Algorithm*

HBMO – *Honney-Bees Mating Optimization*

HPABC – *Hybrid Pareto Based Artificial Bee Colony Algorithm*

HTSA – *Hybrid Thin-Slot Algorithm*

JSF – *Job Shop Flexível*

JSPF – *Job Shop Parcialmente Flexível*

JSTF – *Job Shop Totalmente Flexível*

JS – *Job Shop*

KABCO – *Knowledge-Based Ant Colony Optimization Algorithm*

LEGA – *Learnable Genetic Architecture*

MMABC – *Multiobjective Micro Artificial Bee Colony Algorithm*

MMGA – *Multiobjective Micro Genetic Algorithm*

MOPSO+LS – *Multiobjective Particle Swarm Optimization and Local Search Algorithm*

MPOX – *Modified Precedence Operation Crossover*

NFA – *Número de Fontes de Alimentos*

NMCF – *Número Máximo de Ciclos por Forrageamento*

NP – *Non-Deterministic Polynomial Time*

P-DABC – *Pareto Discrete Artificial Bee Colony*

PCP – *Planejamento e Controle da Produção*

PEGA – *Pezzella's Genetic Algorithm*

PEP – *Planejamento Estratégico da Produção*

PMP – *Planejamento Mestre da Produção*

PSO+SA – *Particle Swarm Optimization and Simulated Annealing*

PSO+TS – *Particle Swarm Optimization and Tabu Search*

PSO – *Particle Swarm Optimization*

PVNS – *Parallel Variable Neighborhood Search*

SEA – *Simple and Effective Evolutionary Algorithm*

SM – *Simulation Modeling*

SPT – *Shortest Processing Time*

TABC – *Two-stage Artificial Bee Colony Algorithm*

TSPCB – *Tabu Search Algorithm With a Fast Public Critical Block Neighborhood Structure*

TS – *Tabu Search*

VNS – *Variable Neighbourhood Search*

X-LS – *Xing and Local Search Algorithm*

X-SM – *Xing Search Method*

hGA – *Hybrid Genetic Algorithm*