UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM ACO PARA A PROGRAMAÇÃO REATIVA DA PRODUÇÃO

MARCOS ABRAÃO DE SOUZA FONSECA

ORIENTADOR: PROF. DR. EDILSON REIS RODRIGUES KATO

São Carlos - SP
Junho/2010

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM ACO PARA A PROGRAMAÇÃO REATIVA DA PRODUÇÃO

MARCOS ABRAÃO DE SOUZA FONSECA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.
Orientador: Dr. Edilson Reis Rodrigues Kato.

São Carlos - SP
Junho/2010

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

F676aa

Fonseca, Marcos Abraão de Souza.
Uma abordagem ACO para a programação reativa da produção / Marcos Abraão de Souza Fonseca. -- São Carlos : UFSCar, 2010.
93 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2010.

1. Inteligência artificial. 2. Programação da produção. 3. Sistemas flexíveis de manufatura. I. Título.

CDD: 006.3 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Uma Abordagem *ACO* para a Programação Reativa da Produção"

MARCOS ABRAÃO DE SOUZA FONSECA

Dissertação de Mestrado 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

Membros da Banca:

Prof. Dr. Edilson Reis Rodrigues Kato
(Orientador - DC/UFSCar)

Prof. Dr. Orides Morandin Júnior (DC/UFSCar)

Profa. Dra. Jandira Guenka Palma
(UEL/Londrina)

São Carlos - SP
Junho/2010

Dedico este trabalho ao meu irmãozão Márcio.

AGRADECIMENTO

A minha avó D.ª Lurdes, que sempre me ajudou em momentos difíceis e nunca deixou que as barreiras lhe tirassem o sorriso. Ela é pra mim um grande exemplo.

A minha mãe por sua dedicação e carinho.

Ao meu pai pelos conselhos, atenção e por sempre me apoiar.

Ao meu irmão pelo carinho e sensibilidade, tenho grande orgulho de seu caráter. Confio a ele meus mais nobres sentimentos.

A minha irmã por compreender minha fase de ausência.

Agradeço a toda minha família, que mesmo estando distantes, souberam compreender a distância que nos une. Por simplesmente acreditarem em minhas capacidades e me apoiarem.

Ao meu orientador, Edilson, pelo auxílio e discussões, dando o suporte necessário para a realização desse trabalho.

Ao meu amigo Jesus por sua grande sabedoria e amizade, sempre disposto a ajudar e conversar. Levarei comigo aprazíveis lembranças e fortes sentimentos.

Aos meus grandes amigos peruanos. O que seria do mestrado sem vocês!. Agradeço pelo convívio prazeroso e pela enorme partilha de cultura. Serão inesquecíveis.

Ao meu amigo Vinícius pelas conversas, reflexões e muitas risadas. É um rapaz super organizado.

Aos meus amigos do laboratório de pesquisa, Flávio, Mayra, Wesley, Carlos, Ageu pela grande ajuda e companheirismo durante o mestrado.

Aos professores do Departamento de Computação.

Enfim, aos amigos que fiz no mestrado e todos aqueles que estão direta e indiretamente ligados.

A todo instante a existência principia; em torno de cada aqui, gira a esfera do acolá. O centro está em toda parte. Tortuosa é a senda da eternidade.

Friedrich Nietzsche - Assim falou Zaratustra

RESUMO

No contexto de Sistemas Automatizados de Manufatura, problemas de otimização combinatória, como determinar a programação da produção, têm sido foco de estudo em muitas pesquisas devido ao alto grau de complexidade para sua resolução. Diversos trabalhos apontam para o uso de metaheurísticas para o tratamento do problema, onde diferentes perspectivas de abordagens têm sido propostas visando encontrar soluções de qualidade em um curto espaço de tempo. Neste trabalho, é proposta uma abordagem baseada na metaheurística Otimização por Colônia de Formigas (*Ant Colony Optimization – ACO*) para o problema de programação reativa da produção em um FMS, com o objetivo de conciliar as características do problema com as características da metaheurística. Para isso, o problema é tratado em duas perspectivas, com base na modelagem e no método de busca. A modelagem do problema é caracterizada por uma descrição do problema em nível de operações, uma vez que a programação da produção está incluída neste contexto. Sobre o modelo é aplicado um método de busca construtiva baseado em *ACO* que usando o princípio de colaboração, estabelece uma relação entre as operações de forma que esta direcione a busca para regiões promissoras do espaço de soluções. O Objetivo deste trabalho é obter uma programação reativa em tempo de resposta aceitável, visando minimizar o valor de *makespan*. Resultados experimentais mostraram uma melhoria dos resultados até então obtidos por outras abordagens.

Palavras-chave: Programação Reativa da Produção, Sistemas Flexíveis de Manufatura, *Ant Colony Optimization, ACO.*

ABSTRACT

In the context of automated manufacturing systems, combinatorial optimization problems, such as determining the production schedule, have been focused in many studies due to the high degree of complexity to their resolution. Several studies point to use of metaheuristics for the problem dealt, where different approaches perspectives have been proposed in order to find good solutions in a short time. In this paper, we propose an approach based on Ant Colony Optimization metaheuristic (ACO) for the reactive production scheduling problem in an FMS aiming the combination of problem characteristics with metaheuristic characteristics. For this, the problem is addressed from two perspectives, based on modeling and the search method. The problem representation is characterized by a description of the problem at the operations level, since the production schedule is included in this context. On the model is applied a constructive search method based on ACO that using the collaboration principle, establishing a relationship between operations so that it lead the search for promising regions of the solution space. The goal of this work is to obtain a reactive programming in acceptable response time in order to minimize the makespan values. Experimental results showed an improvement of the results obtained so far by other approaches.

Keywords: Reactive Production Scheduling, Flexible Manufacturing Systems, *Ant Colony Optimization, ACO, graph representation*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG – *Algoritmos Genéticos*

BT – *Busca Tabu*

FJSS – *Programação em Job Shop Flexível (Flexible Job Shop Scheduling)*

FSS – *Programação em Flow Shop (Flow Shop Scheduling)*

FMS – *Sistema Flexível de Manufatura (Flexible Manufacturing System)*

JSS – *Programação em Job Shop (Job Shop Scheduling)*

OC – *Otimização Combinatória*

PDG – *Grafo parcial disjuntivo (Partial Disjunctive Graph)*

PN – *Redes de Petri (Petri Nets)*

SMP – *Problema em máquina única (Single Machine Problem)*

SIA – *Sistemas Imunes Artificiais*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

A busca pela competitividade no meio industrial tornou-se umas das características mais marcantes no processo de evolução dos sistemas produtivos. Especificamente, considerando o âmbito deste trabalho, os sistemas de manufatura acompanham essa característica onde novas tecnologias têm sido desenvolvidas e utilizadas no intuito de atingir esse propósito (AGUIRRE, 2007).

Como apresentado em Slack (1993), o sucesso competitivo é decorrente de determinados objetivos de desempenho, como qualidade, custo, flexibilidade, rapidez e confiabilidade. Sistemas produtivos procuram considerar esses objetivos, que são intrinsecamente ligados, em função de atingir vantagens competitivas, na medida em que são conciliados para produzir de acordo com as expectativas dos clientes.

Nesse contexto, os Sistemas Flexíveis de Manufatura (FMS – *Flexible Manufacturing System*) surgiram como uma das alternativas em resposta a essas expectativas, fornecendo diversos tipos de mecanismos como forma de garantir uma resposta rápida às mudanças e exigências relativas ao mercado. Dentre outras características, esse sistema possibilita uma produção com médio grau de variedade de produtos e com baixo volume de produção por tipo de peças, quando comparado a outros ambientes de manufatura como, por exemplo, sistemas dedicados (AGUIRRE, 2007).

Devido ao alto grau de flexibilidade, onde máquinas executam operações de mais de um tipo produto, o mesmo produto pode ser manufaturado por roteiros de produção distintos, entre outras características, o sistema torna-se complexo no que se refere aos seus aspectos de planejamento e controle (PINEDO, 2008).

Dentre as atividades compreendidas pelo planejamento de um sistema produtivo está a programação da produção, que de maneira geral é responsável por alocar os recursos disponíveis em um determinado tempo seguindo as restrições definidas, como por exemplo, a sequencia correta de operações, de forma que o resultado gerado minimize o tempo de processamento das todas as tarefas do cenário de produção (AGUIRRE, 2007; GOOVER, 2006).

Considerando-se a complexidade do sistema, a programação da produção torna-se uma tarefa difícil de ser realizada, uma vez que características como compartilhamento de recursos, paralelismo entre operações, concorrência, eventos inesperados, entre outros, implicam diretamente em sua definição.

Geralmente a programação obtida pode ser aplicada de maneira satisfatória em sistemas produtivos. No entanto, a natureza de problemas de programação envolve um processo dinâmico e contínuo. Com isso, a programação obtida pode não ser condizente com a situação real do ambiente, em que eventos inesperados ocorrem e uma nova programação deve ser definida, levando em consideração o cenário a ser avaliado. Portanto, uma programação reativa deve ser utilizada buscando conciliar uma nova solução considerando o estado atual de produção, de forma a manter a continuidade do sistema produtivo (TOGUYÉNI *et al.,* 2003; TANG e WANG, 2008).

A programação reativa considerada, baseia-se na especificação de uma nova programação da produção frente as mudanças ocorridas no cenário (como quebra de máquina ou falta de matéria-prima) e será referenciada ao longo do texto como reprogramação da produção.

Devido a essas características, o problema de programação da produção é considerado NP-Difícil (GAREY *et al.,* 1976) e têm sido tratado sobre duas perspectivas. Por meio de métodos exatos, onde o objetivo é encontrar a solução ótima, porém exigem um elevado esforço computacional, e métodos de aproximação, que não garantem a solução ideal, mas são capazes de encontrar boas soluções em tempo computacional aceitável (BLUM e ROLI, 2003; AGUIRRE, 2007).

Muitos estudos têm focado no segundo grupo de procedimentos, tendo como objeto de estudo as metaheurísticas. Metaheurísticas podem ser entendidas como procedimentos heurísticos para fins gerais, combinado com um conjunto de conceitos e definições de forma a serem aplicadas a diferentes problemas de otimização. Recentemente, um método de busca baseada em modelo, denominado Otimização por Colônia de Formigas (*Ant Colony Optimization – ACO*) tem sido uma das alternativas para a tentativa de resolução dos problemas de programação da produção (BLUM e SAMPELS, 2004; KUMAR *et al.,* 2003; LIOUANE *et al.,* 2007; ROSSI e DINI, 2007; HUANG e LIAO, 2008; PRAKASH *et al.,* 2008). Dentre as especificidades apresentadas nesses trabalhos, destacam-se várias propostas de modelagem, as quais, convencionalmente, procuram admitir as características inerentes ao problema como uma melhor forma de representá-lo e solucioná-lo.

No que se refere à ambientes dinâmicos, a metaheurística ACO têm sido empregada considerando o aprendizado colaborativo decorrido do processo de busca, o qual é usado como informações sobre o ambiente para a obtenção de soluções (SILVA *et al.,* 2008).

1.2 Objetivos

O objetivo deste trabalho é o de apresentar uma proposta de modelagem e busca usando uma abordagem *ACO* para solucionar o problema de programação reativa da produção, que seguindo as características do sistema produtivo, busque minimizar o valor de *makespan*.

Avaliar se a trilha de feromônio possui informações relevantes do processo de busca durante a etapa de programação da produção e se essas informações podem ser usadas em uma nova busca, de maneira a direcionar-la para regiões mais promissoras do espaço de soluções em uma etapa de reprogramação da produção.

Avaliar se a abordagem *ACO* proposta apresenta soluções para o problema de programação reativa em tempo de resposta aceitável para o contexto do ambiente de produção, ou seja, realiza a busca e apresenta resultados em poucos segundos.

1.3 Motivação e Justificativa

Um dos aspectos a se considerar para solução de problemas em sistemas complexos é quanto à representação do conhecimento do problema, no qual técnicas de modelagem são utilizadas visando esse propósito. No contexto atual, existe uma grande dificuldade de gerar um modelo que represente efetivamente as características do sistema e que esse facilite a aplicação de técnicas computacionais para a busca da solução. Diversas pesquisas têm surgido na tentativa de unir o aspecto de representação com a busca da solução do problema (MAGGIO, 2005; PRAKASH *et al.,* 2008).

Neste contexto, têm-se como motivação para este trabalho a descrição e o desenvolvimento de um modelo de representação para o problema de programação da produção, baseando-se nas características do Sistema Flexível de Manufatura (FMS) em consideração. Sobre esse modelo é aplicada uma abordagem baseada em Otimização por Colônia de Formigas, que considerando as especificidades do ambiente de produção, realiza a busca de uma possível solução para o problema.

Ainda considerando as características dinâmicas do problema, a abordagem realiza uma reprogramação a partir de um novo cenário especificado. O cenário é obtido após eventos inesperados, como quebra de máquina e falta de matéria prima. A reprogramação é determinada levando em consideração o aprendizado decorrido do processo de busca do cenário sem os eventos mencionados onde, de maneira colaborativa, a relação estabelecida entre os estados pode direcionar a soluções de qualidade para o cenário modificado.

Este trabalho está inserido em uma das linhas de pesquisa do grupo de TEAR, Laboratório de Pesquisa e Inovação em Tecnologias e Estratégias de Automação, a qual inclui explorar propostas inovadoras para a programação reativa da produção com técnicas de inteligência artificial. Dessa forma, este busca dar continuidade aos trabalhos desenvolvidos dentro desse tema de pesquisa.

1.4 Estrutura do Trabalho

Este documento está organizado da seguinte maneira:

No capítulo 2 são apresentados os conceitos fundamentais do trabalho, incluindo uma descrição geral do ambiente de produção assim como as características do problema em consideração.

No mesmo capítulo é introduzida uma descrição formal generalizada sobre problemas de otimização combinatória e em seguida alguns aspectos sobre metaheurísticas. Neste contexto, é estabelecido um enfoque na metaheurística *ACO*, sendo apresentada nessa seção a descrição dos principais procedimentos do algoritmo assim como algumas das principais variantes usadas na literatura.

A revisão de trabalhos correlatos é apresentada em seguida, buscando incluir algumas considerações de como *ACO* tem sido empregada na tentativa de resolução de problemas de programação e quais as principais questões discutidas nessas aplicações.

A partir das considerações imprescindíveis do capítulo 2, no capítulo 3 é apresentada a proposta de trabalho visando descrever a principal contribuição desse trabalho. Nesse capítulo é descrita a proposta de modelagem sob uma visão geral dos procedimentos, cabendo ao capitulo posterior, capítulo 4, o detalhamento da abordagem desenvolvida, discernindo suas principais especificidades e definições que condizem com a perspectiva de resolução dada ao problema considerado.

Para a avaliação da abordagem, são apresentadas ainda no capítulo 4 as definições do cenário avaliado, sendo os resultados apresentados e discutidos no mesmo capítulo.

No capítulo 5 são apresentadas as conclusões e as propostas para trabalhos futuros.

Por fim, de forma a facilitar o entendimento da metaheurística *ACO*, um exemplo de aplicação da técnica a um problema clássico de otimização é descrito no Apêndice A desse documento.

Capítulo 2

REVISÃO DA LITERATURA

2.1 Considerações Iniciais

Neste capítulo serão apresentados e discutidos os principais conceitos desse trabalho em uma perspectiva mais ampla, incluindo a caracterização do ambiente de aplicação e do problema considerado.

Em seguida é introduzida a descrição da metaheurística *ACO* servindo como fundamento geral para dos artigos relacionados. Outras considerações também são incluídas como forma de descrever as características do problema que levam a escolha da abordagem.

Na seção 2.6.1 são apresentadas algumas abordagens da literatura baseadas em *ACO*, visando indicar as principais questões quanto ao uso do método para o problema de programação e de como suas especificidades tem sido exploradas de maneira a condizer com uma abordagem eficiente para a tentativa de resolução do problema.

2.2 Caracterização do Problema

No âmbito de sistemas automatizados de manufatura, os Sistemas Flexíveis de Manufatura (FMS) têm se destacado ao longo das últimas décadas como uma

das alternativas para atender aos fatores de desempenho de sistemas produtivos como custo, qualidade, flexibilidade e confiabilidade de entrega (AGUIRRE, 2007).

Diversos níveis de automação e flexibilidade tornam estes sistemas complexos e visando propor soluções para os problemas característicos desse tipo de ambiente, é dado enfoque a essa tecnologia de manufatura neste trabalho.

2.2.1 Sistemas Flexíveis de Manufatura

Um Sistema Flexível de Manufatura (*Flexible Manufacturing System – FMS*) consiste de um sistema complexo controlado por computador integrado, com grupos de estações de processamento interconectadas por sistemas automáticos para movimentação de materiais e armazenamento. Este tipo de sistema representa uma configuração de grande flexibilidade no que se refere à inserção de novos produtos, *mix* de produção (variação de produtos), processamento das operações e manipulação de materiais (AGUIRRE, 2007; TEMPELMEIER e KUHN, 1993).

Normalmente um FMS tem sido descrito com os seguintes elementos:

- **Estações de Processamento**: correspondem as máquinas que realizam as operações do processo de manufatura. Cada máquina pode ter um elemento de armazenamento de materiais denominado *buffer*.
- **Sistemas de Movimentação e Armazenamento de Materiais:** são responsáveis pela manipulação de material, movendo as peças entre as estações de trabalho, locais de armazenamento e de carga e descarga. São compostos por robôs industriais, transportadores e veículos auto-guiados.
- **Sistema Central de Controle por Computador:** tem como função principal coordenar os demais elementos do FMS ligando questões de planejamento, em nível estratégico, com as operações e funções dos elementos de um FMS, em um nível de chão de fábrica.

Diversos trabalhos têm apresentado propostas de classificação para as flexibilidades oferecidas em um FMS sobre diferentes perspectivas (SHEWCHUK e MOODIE, 1998; CORREA e SLACK, 1996; TEMPELMEIER e KUHN, 1993). Alguns tipos de flexibilidade podem ser identificados por:

- **Flexibilidade da máquina:** corresponde a habilidade de uma máquina processar vários tipos de peças, de maneira que facilite essa troca em tempo viável para o contexto de produção.
- **Flexibilidade do roteiro:** é a habilidade incluída em um sistema de produção que permite a fabricação de determinados produtos por meio do uso de rotas alternativas de operações.
- **Flexibilidade da operação:** refere-se à facilidade de mudança das operações necessárias para fabricar a peça.
- **Flexibilidade do produto:** habilidade de introduzir ou modificar os produtos do sistema de produção.

Vários tipos de problemas estão inseridos nesse ambiente de produção, envolvendo questões de controle e planejamento. Na próxima seção é descrito o problema de programação da produção, o qual é tratado neste trabalho.

2.2.2 Programação da Produção

A programação da produção está inserida nas atividades de Planejamento e Controle da Produção (PCP), o qual ainda inclui diferentes níveis de definições para tomada de decisões a longo, médio e curto prazo. No nível estratégico, o plano de produção é estabelecido em longo prazo considerando estimativas de vendas e recursos disponíveis.

Baseando-se no plano de produção, o planejamento mestre da produção é definido em médio prazo, incluindo informações mais detalhadas das etapas de programação e execução das atividades.

A programação da produção é então definida na terceira atividade do PCP, a qual munida das informações do planejamento mestre da produção determina em curto prazo de tempo o detalhamento das operações que serão realizadas, incluindo entre outros, onde as operações dos produtos serão processadas, com quais recursos, e em qual tempo de inicio e término para cada ordem de produção (GROOVER, 2006; TUBINO, 2000).

Para determinar a programação em um FMS, diversas características do ambiente de produção podem ser consideradas, como limitações da capacidade de armazenamento dos *buffers*, roteiros de produção flexíveis, tempo de transporte,

tempo de processamento determinístico, tempo de configuração das máquinas (*setup*) e ainda outras questões de gerenciamento como necessidades de data de entrega, restrições do tamanho de lote, prioridade de fabricação, entre outras (DOMINGOS, 2004).

Essa distribuição de operações segue ainda restrições inerentes a classe do sistema produtivo em questão (como por exemplo, *Job Shop*, *Flow Shop*, entre outros), onde questões sobre as propriedades de cada máquina, operações e recursos devem ser consideradas (AGUIRRE, 2007). Devido a essa alta complexidade, algumas convenções são admitidas no intuito de diminuir a complexidade do problema (SANTOS, 2008; MAGGIO, 2005). Maiores detalhes sobre a simplificação do problema são apresentados no capítulo de resultados, no qual é descrito o cenário ao qual foi aplicada a abordagem.

2.2.3 Programação Reativa da Produção

Sistemas Flexíveis de Manufatura estão sujeitos a mudanças inesperadas do fluxo de produção, o que pode conduzir a uma degradação do desempenho do sistema (TOGUYENI *et al.,* 2003).

A natureza de sistemas produtivos é dinâmica, onde eventos típicos como quebra de máquina, falta de matéria prima, insuficiência na habilidade de transporte, falta de operadores, entre outros são propícios de ocorrerem. Dessa forma, no intuito de manter o fluxo de produção, a reprogramação da produção deve ser realizada.

Alguns trabalhos baseiam-se em casos passados como forma de minimizar o impacto do evento ocorrido no sistema produtivo, ou ainda em propostas de monitoramento do sistema para a recuperação na ocorrência de falhas, entre outras abordagens (TOGUYENI *et al.,* 2003; TANG e WANG, 2008).

2.3 Problemas de Otimização Combinatória

Problemas de otimização combinatória (OC) podem ser encontrados em diversos tipos de sistemas, cuja natureza é de importância prática ou teórica. Exemplos de problemas OC são: plano de custo mínimo para sistemas de

transporte, atribuição ideal de funcionários para um determinado conjunto de tarefas a serem executadas, um melhor esquema de roteamento de pacotes de dados na internet, sequencia ótima de tarefas a serem processadas em uma linha de produção, entre outros (BLUM e ROLI, 2003; DORIGO e STÜTZLE, 2004).

De maneira geral, problemas de OC envolvem a busca de um melhor conjunto de valores para variáveis discretas de tal forma que estes valores otimizem uma determinada função objetivo, respeitando as restrições definidas sobre essas variáveis (BLUM e ROLI, 2003; DORIGO e STÜTZLE, 2004). Formalmente um problema de otimização combinatória $P = (S, f, \Omega)$ pode ser definido como:

- S é o conjunto de soluções candidatas que representa o espaço de soluções. Este espaço de soluções é definido sobre um conjunto finito de variáveis discretas $X = \{x_1, x_2, ..., x_n\}$ e um conjunto Ω de restrições entre as variáveis.
- f é a função objetivo a ser minimizada na qual determina um valor da função objetivo $f(s)$ para cada solução candidata $s \in S$.

Soluções válidas são aquelas que satisfazem as restrições Ω e pertencem ao conjunto $S' \subseteq S$. A resolução do problema consiste em encontrar uma solução ótima global $s^* \in S'$, isso significa que para problemas de minimização s^* corresponde a solução de menor custo, tal que $f(s^*) \leq f(s)$ para todas as soluções $s \in S'$ (DORIGO e STÜTZLE, 2004).

Diversos métodos têm sido propostos para a solução de problemas de OC, os quais de maneira geral podem ser classificados como métodos exatos e métodos aproximados. Métodos exatos são aqueles que garantem a solução ótima do problema, sendo viáveis para instâncias pequenas do mesmo. Isso se deve ao alto custo computacional necessário para ser efetuada a busca. Devido a essa característica, métodos aproximados têm se destacado para aplicações em problemas reais. Métodos aproximados não garantem a solução ótima, porém são capazes de reduzir drasticamente o tempo de busca da solução, sendo amplo o suficiente para comportarem problemas de larga escala (BLUM e SAMPELS, 2003; DORIGO e STÜTZLE, 2004). Incluídas nessa classificação estão as metaheurísticas.

Metaheurísticas podem ser vistas como heurísticas de propósitos gerais, as quais podem ser aplicadas a diferentes problemas de otimização, através de modificações relativamente baixas sobre seu arcabouço de definições. Na seção a

seguir são descritas algumas das características discutidas em relação a esse tipo de método aproximado.

2.4 Metaheurísticas

Blum e Roli (2003) afirmam que, o termo metaheurística é sucessível a várias definições, porém possuem algumas propriedades fundamentais as quais os autores listam, como:

- Guiam o processo de busca;
- Têm como objetivo explorar de maneira eficiente o espaço de busca, procurando encontrar soluções próximas a ótima;
- Possuem técnicas que variam desde algoritmos simples a processos complexos de aprendizado;
- São algoritmos de aproximação normalmente não determinísticos;
- Podem incluir mecanismos de maneira a evitar que a exploração fique confinada em uma área do espaço de busca;
- Os conceitos envolvidos permitem um nível abstrato de descrição;
- Não são específicas para o problema;
- Podem fazer uso do conhecimento referente ao domínio do problema em forma de heurísticas, que por sua vez são controladas por uma estratégia em nível superior.
- As mais avançadas utilizam a experiência obtida na busca (incorporadas em algum sistema de memória) de forma a orientar a pesquisa.

Além dessas características, dois conceitos importantes, os quais de uma maneira explícita ou implícita, diversas metaheurísticas procuram abranger em seu processo de otimização, são as propriedades de diversificação e intensificação, que indicam a exploração do espaço de busca e a utilização efetiva de uma região promissora do espaço de soluções (BLUM e ROLI, 2003).

Na tentativa de estabelecer as principais propriedades de algumas metaheurísticas classificadas como inspiradas na natureza, Colorni *et al.,* (1996) propõem a utilização de quatro características. Estas características são condizentes quanto à manipulação e construção da solução, à utilização de algum

sistema de armazenamento e memorização, ao emprego de um ou diversos agentes para a busca da solução (neurônios, partículas, cromossomos, formigas, entre outros) e quanto à estrutura do espaço de soluções, visando outra perspectiva Blum e Roli (2003) sugerem uma classificação para as metaheurísticas de acordo com métodos baseados em população e métodos baseados em um ponto único que define a trajetória da busca (como por exemplo Busca Tabu).

Após uma descrição das principais metaheurísticas como Computação Evolutiva (*Evolutionary Computation – EC*) o qual inclui Algoritmos Genéticos (*Genetic Algorithms – GA*), *Iterated Local Search* (*ILS*), Têmpera Simulada (*Simulated Annealing – AS*), Otimização por Colônia de Formigas (*Ant Colony Optimization – ACO*), Busca Tabu (*Tabu Search – TS*), entre outras, Blum e Roli, (2003) propõem uma visão unificada para as metaheurísticas a partir da relação de diversificação e intensificação que as promovem.

Basicamente, o quadro de intensificação e diversificação proposto leva em consideração três aspectos: componentes guiados por função objetivo, componentes guiados por uma ou mais funções (além da função objetivo) e componentes que são completamente aleatórios. De acordo com os autores, as metaheurísticas podem ser analisadas sistematicamente a partir de um grau intermediário entre esses três aspectos.

O princípio de caracterização de metaheurísticas sugere a possibilidade de identificar processos específicos de exploração nos quais determinados mecanismos podem ser definidos como forma de aprimorar o método utilizado, assim como oferecer uma visão mais clara de suas propriedades que melhor condizem com o problema a ser tratado (Blum e Roli, 2003).

Nesse contexto, muitas abordagens têm surgido buscando considerar este aspecto de ajuste ou combinação de técnicas, visando suprir desvantagens que metaheurísticas “puras” apresentam. Exemplos bem sucedidos são os casos de algoritmos genéticos combinados com algum método de busca local (CHEN *et al.,* 2008; GONÇALVEZ *et al.,* 2003; ZIRIB *et al.,* 2007), onde a característica de busca exploratória do algoritmo genético em conjunto com o ajuste fino da busca em vizinhança promovem resultados mais significativos.

O mesmo ocorre em combinações da técnica de otimização por colônia de formigas com busca local como em Benbouzid *et al.,* (2008), Rossi e Dini (2007), Liouane *et al.,* (2007) e Huang e Liao, (2008). Uma vez que a solução obtida pelas

formigas é dada de maneira construtiva, essa pode ser melhorada por um ajuste de busca local promovendo uma melhor classificação das soluções (DORIGO e STÜTZLE, 2004). Essa técnica será discutida na próxima seção.

No que se referem a características do processo de exploração, podemos considerar como exemplo os Algoritmos Genéticos (AG), que empregam uma busca paralela e estruturada, direcionada a regiões promissoras do espaço de soluções sem que esta fique confinada em regiões de mínimos locais. Algoritmos genéticos utilizam mecanismos que exploram informações históricas para encontrar novos pontos de busca, dessa forma não são buscas simplesmente aleatórias não direcionadas (REZENDE, 2003). Com isso, AGs têm sido empregados a diversos tipos de problemas de otimização sendo uma alternativa a métodos convencionais para problemas de grande escala no que se refere ao espaço de soluções.

2.5 Otimização por Colônia de Formigas

Os estudos de algoritmos baseados em formigas derivaram da observação do comportamento de formigas reais em uma colônia. A partir dessa observação modelos foram criados, servindo como inspiração para projetos de algoritmos voltados para problemas de otimização (DORIGO e STÜTZLE, 2004; BLUM, 2005).

Os princípios de colaboração, auto-organização e comunicação estabelecidas pelas formigas podem ser visualizados sobre uma mesma perspectiva, nos quais estes compreendem a característica chave de algoritmos baseados em Otimização por Colônia de Formigas (*Ant Colony Optimization – ACO*). Dessa forma, por meio de colaboração, os agentes artificiais se auto-organizam para a resolução de um determinado problema, levando em consideração as modificações ao meio estabelecida pela forma indireta de comunicação (MANIEZZO *et al.,* 2004; DORIGO e STÜTZLE, 2004; DORIGO *et al.,* 1999).

De acordo com o comportamento observado, inicialmente as formigas estão livres para explorar o ambiente à procura de alimento, escolhendo percursos ao acaso. Na medida em que exploram o ambiente até a identificação da comida, as formigas depositam uma substância química denominada feromônio no trajeto percorrido. A trilha de feromônio permite que as formigas encontrem o caminho de

volta para a fonte de alimento (ou para o ninho).e além disso, pode ser usada por outras formigas para encontrar a localização das fontes de alimentos que foram encontradas por outras formigas da colônia (DORIGO *et al.,* 1999).

Considerando ainda essa casualidade, ocorre de algumas formigas escolherem um mesmo percurso, o que provoca um aumento na quantidade de feromônio do trajeto. Devido a essa substância agir influenciando as demais formigas a seguirem pelo mesmo caminho, a intensidade da substância estimula as demais formigas a escolherem esse mesmo trajeto novamente, e assim por diante até que finalmente ocorre uma convergência a um único caminho.

Este processo de *feedback* positivo é um exemplo de um comportamento de auto-organização das formigas (RODRIGUES, 2008; DORIGO e STÜTLZE 2004). As atividades de cada formiga são coordenadas por meio de *stigmergia* (MOURA e FERREIRA, 2003), uma forma indireta de comunicação feita por modificações no ambiente (através do feromônio). Dessa forma, a metaheurística *ACO* segue o uso de uma forma de *stigmergia* artificial para coordenar populações de agentes artificiais. (DORIGO *et al.,* 2000).

A metaheurística *ACO* foi formalizada em Dorigo e Di Caro (1999) e foi resultado de um esforço em se definir um único ponto de vista sobre as variantes do algoritmo *Ant System* (DORIGO, 1992) até então propostas. Como apresentado em Blum (2005), o modo geral de como *ACO* trabalha pode ser representado na Figura 2.1. Dado um problema de Otimização Combinatória a ser solucionado, um conjunto finito de soluções e seus componentes é derivado desse problema.

Da mesma forma, seguindo as características do problema, é derivado um modelo de feromônio no intuito de corresponder a um modelo probabilístico associado aos componentes das soluções. O modelo de feromônio é utilizado como forma de gerar probabilisticamente soluções para o problema em consideração a partir de um processo construtivo. Soluções candidatas são utilizadas para modificar os valores de feromônio do modelo e induzir as próximas amostragens em direção a soluções de alta qualidade.

Esses processos de construção e atualização do modelo seguem de maneira iterativa. Pressupõe-se que soluções de boa qualidade são constituídas de componentes de solução de boa qualidade. Com isso, o conhecimento adquirido sobre esses componentes pode ajudar na amostragem em direção a boas soluções.

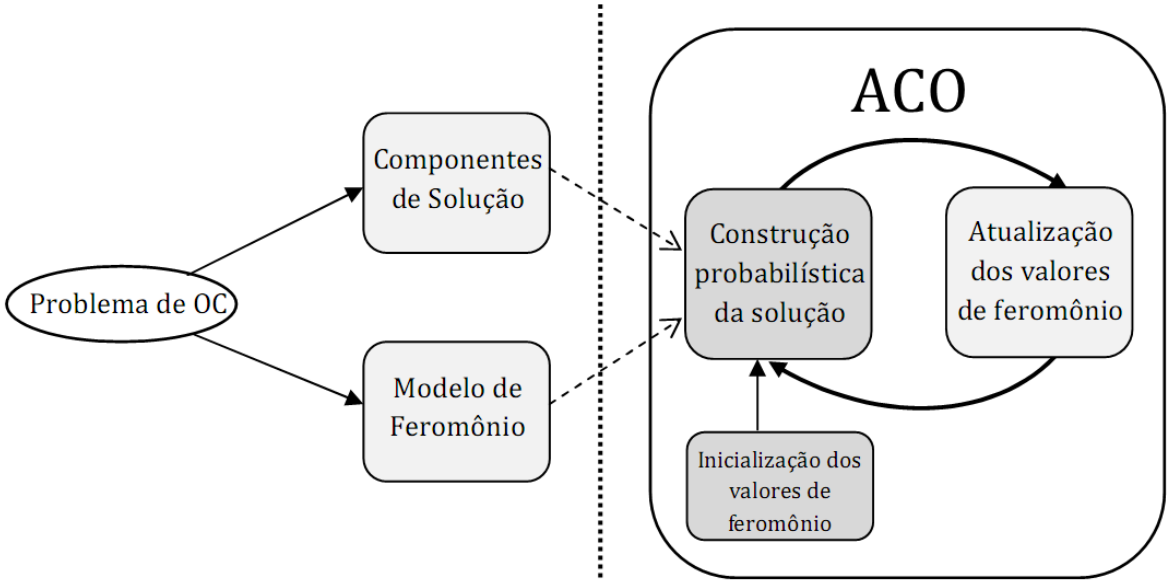

Figura 2.1 – Funcionamento da Metaheurística *ACO* (adaptado de BLUM, 2005)

2.5.1 Estrutura de Algoritmos ACO

Considerando as definições apresentadas em Blum e Dorigo (2005) e Blum, (2005), inicialmente serão apresentados os procedimentos básicos de um algoritmo *ACO*, em seguida nas subseções seguintes, esses procedimentos são especificados de acordo como sugerem algumas das variantes da metaheurística.

A metaheurística *ACO* pode ser definida como uma busca baseada em modelo (DORIGO e STÜTLZE, 2004). Essa classificação sugere que soluções candidatas são construídas usando um modelo probabilístico.

De acordo com as características apresentadas por Dorigo e Di Caro (1999) e seguindo o formalismo apresentado na seção 2.3, sobre problemas de otimização combinatória (OC), as formigas constroem soluções realizando percursos sobre um grafo completo $G = (C, L)$, sendo C o conjunto de vértices e L o conjunto de arestas que unem os vértices. As restrições Ω são implementadas sobre o modelo e são seguidas pelas formigas. Componentes $c_i^j \in C$ e conexões $l_i^j \in L$ podem ter associações a um modelo de trilha de feromônio. Os valores associados à trilha de feromônio são denotados por τ_i^j, isto é, os valores definidos para as arestas dos vértices *i* e *j*. O conjunto de todos os valores da trilha de feromônio é denotado por T. A notação $\mathfrak{C}$ representa o conjunto de todos os elementos c_i^j do conjunto de soluções (Blum e Dorigo, 2005; Blum, 2005).

Como um problema de OC pode ser modelado de diversos modos, diferentes modelos de feromônio podem ser definidos sobre esses modelos.

No algoritmo da Figura 2.2, são descritos os principais procedimentos da metaheurística *ACO*. Neste pseudocódigo é considerada uma instância *P* de um problema de OC com as definições apresentadas na seção 2.3:

```
1 - Algoritmo 1: Algoritmo ACO básico
2 -     Entrada: Uma instância P de um problema de OC.
3 -         P = (S,f,Ω)
4 -     InicializarValoresDeFeromônio(T)
5 -     s_bs ← NULL
6 -     Enquanto (condição de término não alcançada) Faça
7 -         Ɠiter ← ∅
8 -         Para j = 1, ..., n_a Faça
9 -             s ← ConstruirSolução(T)
10 -            s ← BuscaLocal(s) {opicional}
11 -            Se (f(s) < f(s_bs)) ou (s_bs = NULL) então s_bs ← s
12 -            Ɠiter ← Ɠiter U {s}
13 -        Fim Para
14 -        AplicarAtualizacaoDeFeromonio(T, Ɠiter, s_bs)
15 -    Fim Enquanto
16 -    Saida:Melhor caminho encontrado s_bs
```

Figura 2.2 – Algoritmo básico de ACO (adaptado de BLUM e DORIGO, 2005)

O algoritmo é descrito como segue. A cada iteração, n_a formigas constroem probabilisticamente soluções para o problema em consideração, usando dado modelo de feromônio. Então, opcionalmente um procedimento de busca local é aplicado as soluções construídas. Por fim, antes do inicio da próxima iteração, algumas das soluções são usadas para a realização do procedimento de atualização de feromônio. Maiores detalhes desses procedimentos são descritos a seguir.

InicializarValoresDeFeromonio(T): Inicialmente, todos os valores de feromônio são atribuídos um valor constante $c > 0$. Essa atribuição pode seguir especificações da variante *ACO* usada.

ConstruirSolução(T): Um algoritmo *ACO* é basicamente uma heurística construtiva usada para construir probabilisticamente as soluções. Uma heurística construtiva reúne uma sequência de elementos pertencentes a um conjunto de elementos finitos do espaço de soluções $\mathfrak{C}$. A construção da solução inicia com uma solução parcial vazia $s^P = \langle\emptyset\rangle$. Então, a cada etapa de construção, a solução parcial

é estendida pela adição de um componente factível do conjunto $\Re(s^P) \subseteq \mathbb{C}$. O processo de construção pode ser entendido como determinar um percurso (ou caminho) em no grafo *G = (C, L)*.

A escolha de um elemento c_i^j da solução *s* é determinada de maneira probabilística. Dessa forma, a probabilidade de sair do estado *i* e ir para o estado *j* é proporcional a $[\tau_i^j]^\alpha . [\eta(c_i^j)]^\beta$, onde η é a função heurística que associa um valor heurístico ao componente c_i^j e τ_i^j é o valor de feromônio depositado na aresta (*i, j*), ou ainda pode ser adicionado ao próprio elemento c_i^j. α e β são parâmetros positivos que cujos valores indicam o grau de importância para o valor de feromônio e para a heurística.

A função heurística é opcional, porém é necessária para atingir um melhor desempenho do algoritmo. Na maioria dos algoritmos *ACO* a probabilidade de escolha do próximo componente (também chamada de probabilidade de transição) é definida como:

$$p(c_i^j | s^P) = \frac{[\tau_i^j]^\alpha . [\eta(c_i^j)]^\beta}{\sum_{c_k^l \epsilon \Re(s^P)} [\tau_k^l]^\alpha . [\eta(c_k^l)]^\beta} \quad \forall c_i^j \epsilon \Re(s^P) \tag{2.1}$$

Onde a probabilidade p do elemento c_i^j é determinada considerando a vizinhança factível c_k^l.

BuscaLocal(*s*): Opcionalmente uma busca local pode ser aplicada como forma de melhorar as soluções parciais até então construídas.

AplicarAtualizaçãoDeFeromonio(T, $\Im iter$, s_{bs}): Uma vez que todas as n_a formigas tenham construído suas soluções, é aplicado o procedimento de atualização de feromônio com o objetivo de aumentar os valores de feromônio para os componentes que contribuíram com maior qualidade nas soluções. A maioria dos algoritmos ACO baseia-se na seguinte expressão:

$$\tau_i^j \leftarrow (1-\rho).\tau_i^j + \frac{\rho}{n_a} . \sum_{\{s \in \Im upd | c_i^j \in s\}} F(s) \tag{2.2}$$

Instâncias dessa regra de atualização podem ser definidas por diferentes especificações de $\Im upd$, a qual é um subconjunto de $\Im iter \cup \{s_{bs}\}$, onde $\Im iter$ é o

conjunto de soluções construídas na interação corrente e s_{bs} é a melhor solução global encontrada, isto é, a melhor solução encontrada até a iteração atual.

Com o objetivo de decrementar uniformemente os valores de feromônio é especificado o parâmetro $\rho \in (0,1]$, chamado de taxa de evaporação. A evaporação dos valores de feromônio é necessária como forma de evitar uma rápida convergência do algoritmo para uma região sub-ótima. $F(.)$ é denominada função de qualidade e corresponde ao valor adicionado as arestas pertencentes a sequencia de componentes escolhidos pelas formigas na etapa de construção. Quanto menor o valor de $f(s)$ maior o valor de $F(s)$. O fator $1/n_a$ não é frequentemente utilizado e é introduzido por um propósito matemático. Todavia, uma vez que o fator é constante, esse não modifica o comportamento qualitativo de um algoritmo *ACO*.

2.5.2 *Ant System* e seus Sucessores Diretos

Nessa seção será apresentada a abordagem *Ant System (AS)* assim como outros algoritmos *ACO* que são amplamente similares a AS, visando descrever características mais específicas quanto aos procedimentos anteriormente citados.

2.5.2.1 Ant System

Inicialmente, três diferentes versões para o algoritmo *AS* foram propostas (DORIGO *et al.,* 1991; COLORNI, *et al.*, 1992; DORIGO, 1992), sendo a referência atualmente usada ao algoritmo o trabalho apresentado em Dorigo, (1992) por ter apresentado um melhor desempenho em relação às demais propostas.

Basicamente os dois principais procedimentos de algoritmos *AS* são a construção das soluções e a atualização da trilha de feromônio. A cada etapa de construção, *k* formigas aplicam uma regra probabilística de escolha, denominada regra aleatória proporcional (*random proportional rule*), para a escolha do próximo estado a ser visitado seguindo a expressão (2.1). Cada formiga *k* mantém uma memória indicando os estados já visitados. Esta memória é usada para definir a vizinhança $\mathfrak{N}(s^P)$ do componente c_i^j.

Após todas as formigas terem construído seus percursos, a atualização de feromônio é realizada. Primeiramente é aplicada a evaporação dos valores de feromônio da trilha pela expressão:

$$\tau_i^j \leftarrow (1-\rho)\tau_i^j, \quad \forall (i,j) \epsilon\, L, \tag{2.3}$$

onde $0 < \rho \leq 1$ é um parâmetro que define a taxa de evaporação dos valores de feromônio. Após a evaporação, todas as *m* formigas depositam um valor de feromônio a trilha, no qual este é baseado no custo do caminho percorrido por cada formiga.

$$\tau_i^j \leftarrow \tau_i^j + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \qquad \forall (i,j) \epsilon\, L, \tag{2.4}$$

onde $\Delta\tau_{ij}^k$ é o valor de feromônio que a formiga *k* deposita sobre os arcos visitados. É definido por:

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k, & \text{se o arco } (i,j) \epsilon\, P^k \\ 0, & \text{caso contrário} \end{cases}, \tag{2.5}$$

onde P^k é o percurso percorrido pela formiga *k*. C^k é o custo do caminho P^k construído pela formiga *k*. Em geral, as arestas que são utilizadas por muitas formigas e as quais fazem parte de soluções de boa qualidade, recebem mais feromônio e são, dessa forma, mais propensos de serem escolhidos em futuras iterações do algoritmo (DORIGO e STÜTZLE, 2004).

2.5.2.2 *Ant System* baseado em *rank*

Proposta por Bullnheimer *et al.* (1997) a variante denominada *Rank-Based Ant System* (AS_{rank}) apresenta uma modificação para o procedimento de atualização da trilha, no qual somente um determinado numero de formigas podem depositar valores de feromônio que por sua vez levam em consideração o *rank* da formiga, isto é, quão bem colocada está uma solução em relação as demais. Assim, a atualização de feromônio é definida por:

$$\tau_i^j \leftarrow \tau_i^j + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^r + w\Delta\tau_{ij}^{bs}, \quad \forall(i,j)\epsilon\, L, \tag{2.6}$$

onde $\Delta\tau_{ij}^r$ é o valor de feromônio que a formiga *k* deposita sobre os arcos visitados considerando seu *rank r* e $w\Delta\tau_{ij}^{bs}$ é o valor do feromônio depositados pela melhor formiga global (*best-so-far ant*) (DORIGO e STÜTZLE, 2004).

2.5.2.3 *Max-Min Ant System*

Em *Max-Min Ant System* (*MMAS*) (STÜTZLE e HOOS, 2000) é introduzido quatro principais modificações em relação à abordagem AS. Primeiro, os melhores caminhos são fortemente explorados, uma vez que somente a melhor formiga da iteração corrente ou a melhor formiga global atualiza a trilha de feromônio. A segunda modificação procura aplicar um procedimento para evitar que as formigas fiquem confinadas em regiões sub-ótimas do espaço de soluções. Para isso, os valores de feromônio são limitados a um intervalo [τ_{min}, τ_{max}]. Em terceiro, os valores da trilha de feromônio são inicializados com o limite superior do intervalo, favorecendo uma exploração ampla já no inicio da busca. Por fim, os valores de feromônio são reinicializados a cada vez que a abordagem entra em estagnação (isto é, quando todas as formigas convergem para um mesmo caminho prematuramente) ou quando não há melhora da solução gerada após um determinado número de iterações.

Após todas as formigas terem construídos seus percursos utilizando a probabilidade definida em (2.1), é aplicada o procedimento de evaporação, como em AS, equação (2.3). Em seguida, o depósito do valor de feromônio é definido por:

$$\tau_i^j \leftarrow \tau_i^j + \Delta\tau_{ij}^{best}, \tag{2.8}$$

onde $\Delta\tau_{ij}^{best}$ é o valor de feromônio que a melhor formiga da iteração, ou a melhor formiga global deposita na trilha do percurso percorrido. Assim, a escolha da freqüência em que cada forma de atualização é realizada implica em quão agressiva será realizada a busca, ou seja, quando as atualizações são realizadas com mais freqüência pela melhor formiga global, as soluções tendem a se aproximar do valor da melhor solução encontrada, já quando as atualizações são realizadas pela

melhor formiga das iterações, o número de arcos que recebem feromônio é maior e a busca é menos direcionada (DORIGO e SÜTZLE, 2004).

2.5.3 Extensão Ant Colony System

Ant Colony System (*ACS* – DORIGO e GARBADELLA, 1997) difere de *AS* em três características principais. Primeiro, é utilizada uma regra de escolha mais agressiva, denominada regra pseudo-aleatória proporcional (*pseudorandom proportional rule*), na qual torna possível a escolha do melhor elemento dentre os vizinhos do estado corrente da etapa de construção. A regra é definida por:

$$j = \begin{cases} \text{argmax}_{l \in \mathcal{N}_i^k} \left\{ \tau_{ij} \left[\eta_{ij} \right]^{\beta} \right\}, & \text{se } q \leq q_0 \\ J, & \text{caso contrário} \end{cases} \qquad (2.9)$$

onde *q* é uma variável aleatória uniformemente distribuída em [0,1], q_0 é um parâmetro e *j* é determinado pela distribuição probabilística dado por (2.1).

Em *ACS*, somente a melhor formiga global deposita feromônio a cada iteração. Ao contrário de *AS*, os procedimentos de evaporação e depósito de feromônio são realizados somente no melhor percurso. Porém, adicionalmente a atualização global de feromônio, é aplicada uma atualização local de feromônio, isso significa que após a escolha do próximo elemento que constituirá a solução, a evaporação do valore de feromônio na aresta correspondente a essa transição é feita de imediato durante a etapa de construção. O efeito de uma regra de atualização local implica em um aumento da exploração e de forma prática evita que o algoritmo entre em estagnação (DORIGO e STÜTZLE, 2004).

2.6 Características do Problema x Características da Metaheurística

O problema de programação da produção em um Sistema Flexíveis de Manufatura (*FMS*) é caracterizado por apresentar diversas propriedades como concorrência, sincronismo, compartilhamento de recursos, paralelismo entre as operações, não determinismo, entre outras.

Um FMS quando tratado em nível de operações, as propriedades descritas inicialmente permanecem no sistema e com isso, técnicas de modelagem e métodos de busca têm sido propostas visando abranger essas características na tentativa de aproveitá-las para uma melhor forma de representação e de resolução do problema como apresentados nos trabalhos de Maggio (2005), Prakash *et al.* (2008).

O uso de uma abordagem construtiva que permita a escolha das operações considerando as especificidades discutidas anteriormente pode condizer com um método de exploração eficiente do espaço de soluções, no qual efetua a busca de maneira equivalente a natureza do problema em questão, onde a sequência de operações são determinadas ao longo das etapas de produção visando reduzir uma função objetivo.

Nesse contexto, uma vantagem de algoritmos construtivos em relação a abordagens que manipulam a solução inteira (como, por exemplo, algoritmos genéticos) está na facilidade de exploração sobre o modelo que represente as operações a serem executadas. Considerando os produtos que possuem roteiros alternativos, uma abordagem construtiva pode ser modelada possibilitando a escolha de operações alternativas durante o próprio processo de construção da solução. Já algoritmos que manipulam a solução inteira, aplicam procedimentos para modificar a solução como forma de explorar o espaço de soluções. Essa característica pode ocasionar em uma maior complexidade no que se refere ao modelo de representação e no tratamento das soluções.

Recentemente, um método de busca baseada em modelo, denominado otimização por colônia de formigas (*Ant Colony Optimization – ACO*), tem sido uma das alternativas para a tentativa de resolução dos problemas de programação da produção (BLUM e SAMPELS, 2004; KUMAR *et al.,* 2003; LIOUANE *et al.,* 2007; ROSSI e DINI, 2007; HUANG e LIAO, 2008; PRAKASH *et al.,* 2008).

Na próxima seção serão apresentados alguns trabalhos que fazem uso de ACO para o problema de programação, na qual serão apontadas as principais considerações desses trabalhos que se correlacionam com o tema de pesquisa.

2.6.1 Metaheurística Otimização por Colônia de Formigas Aplicada a Problemas de Programação

Em Blum e Sampels (2004) foi desenvolvida uma das primeiras abordagens *ACO* a qual pode apresentar resultados mais competitivos para o problema de programação em um *Job Shop (JSS),* sendo estes avaliados a partir de conhecidos *benchmarks* propostos na literatura (TAILLARD, 1993; BRUCKER *et al.* 1997). O método consiste em um *Max-Min Ant System* (*MMAS*) para o problema de programação.

De forma a gerar soluções factíveis (i. e. considerando a precedência das operações em relação ao grupo de tarefas e na mesma máquina), é utilizado um algoritmo *List Scheduler* (*LS*) onde a escolha das operações que pertencerão à lista é feita considerando como informação heurística uma das nove regras de prioridade apresentadas (*Short Processing Time* (*SPT*), *Earliest Starting Time* (*EST*), *Least Work Remaining*, entre outras).

A descrição do problema é dada por um grafo disjuntivo *G = (V, A, E),* onde *V* denota o conjunto das operações que compreendem a instância do problema, *A* denota os pares de operações *O, O'* em forma conjuntiva, isto significa que a operação *O* deve ser processada necessariamente antes de *O'*. Por fim *E* denota os pares de operações da mesma máquina e/ou do mesmo grupo, em forma disjuntiva. Para cada solução construída pelas formigas é aplicada uma busca local.

Como forma de avaliação a proposta é comparada com uma adaptação da Busca Tabu (BT) para o *JSS* desenvolvida em Nowicki e Smutnicki, (1996) onde ambas são aplicadas a *benchmarks*. Os resultados mostram que a abordagem *ACO* é eficiente na obtenção das soluções para o valor de *makespan*, onde foi possível melhorar os valores até então encontrados para 15 das 28 instâncias de problemas OSS.

Em um trabalho posterior, Blum (2005), considerando a ligação entre *ACO* e técnicas de busca em árvore, propõe um algoritmo híbrido baseado em *ACO* e *Bean Search* (BS) denominado Bean-ACO para o problema de programação em um *Open Shop* (OSS).

Dentre os princípios sugeridos pelo autor para se estabelecer esta combinação de técnicas, é considerada a maneira como ambos os métodos exploram o espaço de soluções, enquanto *BS* é geralmente determinística, a

metaheurística *ACO* atua de forma probabilística, usando a experiência adquirida durante o processo de busca em uma forma adaptativa. A combinação consiste em empregar uma *BS* de forma probabilista baseando-se na transição executada por cada formiga da abordagem *ACO*.

Diversas características são consideradas a partir do trabalho anterior em Blum e Sampels (2004) como, por exemplo, a representação do problema em grafos disjuntivos e o modelo de atualização do feromônio, isto é, a forma em que é empregado a atualização dos valores da trilha de feromônio deixada pelas formigas artificiais. Vale ressaltar que, por variar o problema tratado em relação ao trabalho anterior, a representação em grafos não complementa a notação dada para indicar os grupos de tarefas (*Jobs*).

A avaliação da abordagem se deu pela aplicação do método proposto a diversas instâncias do problema *OSS* em *benchmarks* da literatura e pela comparação de outras duas abordagens que utilizam Algoritmos Genéticos. A função objetivo foi o valor de *makespan.* Dentre as 80 instâncias aplicadas, o método foi capaz de melhorar 24 resultados dos valores até então conhecidos.

Sobre a perspectiva de elaboração do modelo de atualização de feromônio, Huang e Liao (2008) definem a trilha de feromônio a partir da decomposição do problema de programação em um *Job Shop* (*JSS*) para um problema de programação em máquina única (*Single Machine Problem - SMP*) uma vez que a abordagem proposta segue as características do algoritmo de *Shifting Bottleneck* (*SB*). Com isso, uma nova forma de construção das soluções é definida seguindo as especificidades de *SB*, porém é aplicado uma restrição entre as operações sucessoras pertencentes a mesma máquina. Isso quer dizer que uma operação em uma determinada máquina só pode ser processada após certas outras operações relacionadas (pertencentes à mesma máquina) ter concluído o seu processamento.

A abordagem denominada ACOFT, que tem como característica a combinação de *ACO* e Busca Tabu (BT), de maneira geral pode ser delineada como segue. Primeiramente é identificada a máquina de maior custo (designada como ponto crítico) por meio da regra *TML* (*Total Machine Loading*) que obtém a máquina que tem maior sobrecarga. Cada formiga constrói uma permutação de tarefas para a máquina selecionada utilizando uma regra de transição de estados que garante a escolha de operações respeitando as restrições de precedência. Além disso, a equação considera o grau de intensificação e diversificação por um parâmetro

definido q_0 ($0 \leq q_0 \leq 1$), o valor de feromônio e a função heurística que é dada pela regra do caminho mais longo de uma operação até a operação fictícia (*dummy*) que representa a operação final.

As operações *dummy* provêm da representação em grafos parciais disjuntivos (PDG) empregada para o problema, e indicam a operação inicial e final. A partir da representação em grafos, é aplicado um método de otimização das operações na programação. O método consiste em melhorar parte da solução construída (operações alocadas) reduzindo o tempo computacional por omitir as operações que não foram ainda alocadas às máquinas.

Uma vez que todas as formigas tenham construído suas soluções, a melhor solução é melhorada pela TS. Por fim, um método de atualização global do feromônio é aplicado considerando-se a solução dada pela TS. Como resultados, uma extensiva avaliação é apresentada preocupando-se em demonstrar a redução do tempo computacional por utilizar as características da representação do problema e a comparação entre outras 11 abordagens em 5 *benchmarks*, incluindo entre outros Taillard (1993) e Lawrence (1984).

Rossi e Dini (2007) abordam o problema de programação em um *Job Shop* com roteiros flexíveis. No trabalho, os autores descrevem o problema distinguindo *Job Shop* clássico de um *Job Shop* flexível, no qual esse último possui roteiros de produção distintos, e uma determinada operação pode ser executada por um grupo de máquinas idênticas.

É apresentada uma metodologia baseada em um modelo de representação em grafos disjuntivos para o suporte das características de flexibilidade de roteiros e tempo de setup separável (*separate setup*). Sobre o modelo é aplicado um algoritmo de complexidade mínima visando minimizar o *makespan.* A representação torna possível a busca por meio de uma rotina de busca baseada em *ACO*.

Segundo os autores, uma grande parte dos trabalhos da literatura inclui o tempo de setup incluído no tempo de operação, simplificando as análises e afetando a qualidade das soluções. Já quando consideram esses princípios, faltam investigações em programação FMS com suporte a roteiros flexíveis.

Dentre outras especificidades, é utilizado na abordagem o algoritmo *List Scheduler* (*LS*) como forma de gerar soluções válidas. Dessa forma a escolha de uma determinada operação em uma determinada máquina, descarta as demais arestas do grafo que correspondem as possibilidades de escolha à máquinas

alternativas para processamento da operação e descarta ainda, as arestas que unem as operações de outros *Jobs* que serão executadas nessa mesma máquina escolhida, aplicando assim, a restrição de compartilhamento de recursos. Os autores descrevem as características da variante *Ant Colony System* (*ACS*) e em função dessas a usa como método de busca. Na melhor solução é aplicada uma busca local com TB.

O problema é representado por um grafo $DG = (N, A, E_{jh})$, onde N são as operações do cenário a ser avaliado, A são as arestas conjuntivas que unem as operações. E_{jh} são as arestas disjuntivas que unem as operações na mesma máquina h do grupo de maquinas j. Soluções factíveis são determinadas por caminhos sem ciclos no grafo, isto é, caminhos que começam e termina na mesma operação. Ao ser integrado os tempos de setup, uma nova representação é indicada pelos autores, $WDG = (N_F, A, E_{jh}, W_N)$ onde W_N é um vetor de quatro dimensões que indica os pesos sobre os nós do grafos.

A avaliação é baseada na comparação dos resultados alcançados pela proposta com resultados de outras abordagens ACO, heurísticas para *Job Shop* Flexível com flexibilidade de roteiros e *benchmarks* considerando tempo de *setup* e flexibilidade de roteiro. Os resultados alcançados para determinados *benchmarks* são melhores que Colorni *et al.* (1994) (sendo este a primeira abordagem *ACO* para *Job Shop*). Também supera os resultados alcançados em Kumar *et al.* (2003) e Blum e Sampels (2004) com tempos de resposta semelhantes aos de Blum e Sampels (2004).

Kumar *et al.* (2003) aplicam uma abordagem baseada em um *Ant System* (*AS*), com algumas modificações, para o problema de programação em um FMS com roteiros alternativos. Cada uma das operações pode ser executada em mais de uma máquina, o que ocasiona em uma maior importância na determinação da melhor máquina para a execução das operações.

As modificações incorporadas consistem em mecanismos para a prevenção de convergência prematura e estagnação do algoritmo. O valor heurístico definido corresponde a regras heurísticas como LTP (*Longest Processing Time*) e LRPT (*Largest Remaining Processing Time*). O modelo de representação consiste em um grafo onde cada nó indica as operações dos *Jobs* em determinadas máquinas.

Neste contexto, a flexibilidade definida pelos autores é determinada pela existência de mais de um nó para a mesma operação do mesmo *Job*. Com isso, ao

ser escolhido um determinado nó, os demais nós que correspondem às máquinas alternativas são removidos da lista de nós a serem visitados pela formiga. Vale ressaltar que em outras variantes como *Max-Min Ant System* ou *Ant Colony System*, possuem por características próprias os mecanismos incorporados pelos autores.

Propondo uma abordagem para o problema de alocação as máquinas (*maquine loading problem*) em um FMS, Prakash *et al.* (2008) apresentam o uso de um modelo hierárquico para a representação do problema. Segundo os autores, metaheurísticas como Algoritmos Genéticos (AG), Têmpora Simulada (TS) e Sistemas Imunes Artificiais (SIA) têm convergência lenta e aprisionamento em mínimos locais para o problema em questão, o que direciona o enfoque do trabalho para demais métodos de busca alternativos para o problema.

A modelagem em grafos busca representar as propriedades do problema em questão, o qual é tratado em dois sub-níveis de resolução, no primeiro nível é resolvido o problema de alocação as máquinas, no qual os nós são representados como nós locais e indicam a sequencia de operações pertencentes a cada *Job*, e no segundo nível é determinada a sequencia de *Jobs* a serem produzidos, os quais são representados como nós globais.

Considerando o modelo, é aplicada uma abordagem *ACO* nesses dois níveis constituindo a forma hierárquica da proposta. Qualquer formiga está apta a depositar feromônio somente se seu desempenho é melhor que a média do desempenho das iterações anteriores. Dentre as definições aplicadas incluem-se a descrição de uma modificação do modelo de atualização do feromônio por considerar esses dois tipos de nós com visibilidade local e global, um procedimento de prevenção de convergência rápida, baseando-se em uma busca aleatória e probabilística que é variada de acordo com um parâmetro estabelecido e um procedimento para evitar estagnação, baseado na verificação dos valores de feromônio. Os resultados apresentados são constituídos da comparação de regras heurísticas (*SPT*, *LPT*, *LIFO*, *FIFO*), com outras abordagens baseadas em *AG*, *SA*, *AIS*, *TS*, porém sem referências na literatura, e com outros três trabalhos baseados em métodos heurísticos. De acordo com os autores a abordagem apresenta resultados satisfatórios para o tratamento do problema em questão.

Liouane *et al.,* (2007) utilizam uma modelagem em grafos onde esta é derivada de uma representação matricial. O método para a busca da solução consiste em uma abordagem ACO aplicada à matriz que representa o espaço de

soluções. A relação entre os nós (Operações/Máquinas) é determinada por arestas que atribuídas respeitando as máquinas disponíveis e as restrições entre precedências de operações. Apesar dessas considerações nenhum procedimento para as especificações estabelecidas são indicados no trabalho. Após as formigas construírem as soluções é aplicada uma Busca Tabu. Apesar de obterem resultados da programação em um *Job Shop* Flexível (*FJSS*) em um tempo polinomial, o tempo de obtenção de resposta para cada instância avaliada foi omitido na descrição dos resultados.

Seguindo o mesmo princípio de representação matricial Kato *et al.* (2009) propõem um modelo de representação para o problemas de programação em um *Job Shop* a partir das características do ambiente de aplicação definido. Dessa forma, considerando os produtos e seus respectivos roteiros de produção, cada nó do grafo definido consiste dos pares de produtos/roteiros que definem o cenário a ser analisado.

Cada formiga constrói uma solução factível a partir da navegação sobre o modelo, sendo as arestas do modelo definidas e redefinidas dinamicamente por considerar a probabilidade de escolha da metaheurística *ACO*. Dessa forma, visando manter coerência entre a sequencia de produção determinada pelas formigas, a escolha de um determinado produto com seu respectivo roteiro de produção eliminam as demais relações ao mesmo produto com roteiros de produção alternativos, ou seja, se o produto P_1 possui três roteiros alternativos R_1, R_2 e R_3 a escolha de R_2, por exemplo, elimina a probabilidade de escolha de R_1 e R_3.

Sobre a perspectiva de abordagens híbridas Benbouzid *et al.* (2008) aplicam uma busca local baseada em movimentos de inserção (removendo a tarefa da i-ésima posição e a inserindo na j-ésima posição) em todas soluções construídas pelas formigas para o problema de seqüenciamento em um *Flow Shop* ao contrário de Huang e Liao (2008), descrito anteriormente, que aplicam somente na melhor solução construída. Este primeiro procedimento pode gerar um esforço computacional elevado e ainda tornar-se inviável para problemas em que a obtenção de respostas em um curto tempo de execução seja também um critério considerado.

No intuito de realizar uma análise mais clara da aplicação de ACO em problema de programação em *Job Shop* dinâmico, e ainda por considerar a existência de poucos trabalhos sobre esse tema, Zhou *et al.* (2008) tratam a dinamicidade do sistema e não apenas o *Job Shop* com um conjunto estático de

Jobs. No trabalho, elementos são removidos e adicionados no espaço de soluções na medida em que novos *Jobs* chegam ao sistema. Com isso, a matriz de feromônio que representa as operações pertencentes a cada *Job* é atualizada de maneira manter os valores dos nós que permanecem e atribuir a inicialização dos novos nós que foram atribuídos. Dentre outras especificações, no trabalho os autores consideram o tempo de transporte como parte do tempo de processamento da operação.

A modelagem para o problema é estabelecida por meio de um grafo que representa as operações de cada *Job*. Arestas direcionadas unem a precedência entre as operações do mesmo *Job* e arestas bi-direcionadas unem as operações sem restrições. No modelo ainda são incluídas operações fictícias (*dummy*) operações *source* e *sink* que representam o estado inicial e estado final do modelo. O peso associado a cada aresta corresponde ao valor de feromônio e ao tempo de processamento de cada operação.

Propondo uma comparação sistemática entre as abordagens *ACO* e *AG*, Silva *et al.* 2008 aplicam ambas metaheurísticas ao problema de programação em um sistema logístico dinâmico. Após uma revisão considerando os trabalhos que tratam dos principais problemas na literatura como Caixeiro Viajante, Alocação Quadrática, Roteamento de Veículos e Programação em *Job Shop* utilizando ambas as técnicas, os autores listam as seguintes conclusões preliminares:

- Ambas as abordagens apresentam desempenhos similares para diferentes tipos de problemas de otimização.
- AG são geralmente mais rápidos do que *ACO*, entretanto para codificações mais complexas o esforço computacional de um AG pode elevar-se demasiadamente.
- *ACO* aparenta ter um melhor desempenho em problemas de busca em grafos disjuntivos, enquanto AG aparenta ter um melhor desempenho em problemas discretos que visam à seleção de uma melhor combinação de elementos pertencentes a um conjunto mais amplo.

De acordo com os resultados alcançados pelos autores, outro aspecto em relação às abordagens pode ser observado. Considerando um cenário dinâmico do sistema logístico, onde alguns pedidos de entrega até então considerados na busca são cancelados alterando dessa forma o espaço de busca, a re-programação do sistema torna-se mais eficaz com o uso de ACO.

Segundo os autores, uma vez que a matriz de feromônio representa o conhecimento adquirido durante o processo de busca, essa pode ser utilizada de forma a permitir a busca de boas soluções sem a necessidade de computar uma nova solução, isto é, sem a necessidade de efetuar a busca considerando apenas os valores da trilha na fase inicial de execução.

Os valores da trilha podem ser inicializados a partir do aprendizado adquirido durante a fase de exploração do espaço de soluções para que, ao efetuar uma nova busca sobre esse espaço modificado, a relação entre os estados até então definida pode indicar um caminho para outras regiões promissoras de solução, mesmo após o espaço de soluções ter sofrido modificações.

2.7 Considerações Finais

Como pode ser observado em trabalhos da literatura, abordagens *ACO* têm sido uma das alternativas para a tentativa de solução do problema de programação da produção. Diversas classes de sistemas têm sido tratadas sendo suas características exploradas de forma a contribuir para a representação e forma de resolução do problema. Normalmente, utilizam-se modelagens baseadas em grafos para abranger as especificidades do sistema em consideração. Os nós pertencentes ao grafo representam as operações do cenário a ser avaliado e as arestas, a relação existente entre essas operações. A exploração feita sobre os estados é realizada considerando os pesos associados a cada nó. Esses pesos podem sugerir uma relação com a função objetivo a ser minimizada, isto é, quando se trata de *makespan*, os valores heurísticos que normalmente são atribuídos as arestas correspondem ao tempo de execução de uma determinada operação em uma determinada máquina.

Para as abordagens *ACO* estudadas, não se observou uma ligação direta entre a variante do algoritmo escolhida e o problema em consideração. Dessa forma, uma análise quanto às propriedades apresentadas pela variante e quanto às características do problema é requerida, para então se estabelecer a escolha. Dentre as extensões de *ACO* encontradas na literatura, foi escolhida a variante *ACO*

Max-Min Ant System em razão de sua forma de exploração e indução da busca sobre o espaço de soluções.

Existe uma preocupação em relação à etapa de construção das soluções a qual métodos baseados em construção de soluções válidas são utilizados, garantindo dessa forma que as restrições aplicadas ao problema serão consideradas (ROSSI e DINI, 2007; BLUM e SAMPELS, 2004; BLUM, 2005). Do contrário, quando não são utilizados métodos desse tipo, outros procedimentos são definidos como forma de incluir a verificação de precedência entre operações ou demais restrições incluídas no problema.

Associado às definições do modelo de representação está a definição do modelo de atualização da trilha de feromônio. Como puderam ser observadas, diversas propostas têm sido empregadas visando proporcionar uma indução eficiente do processo de busca, para que regiões de melhor qualidade sejam atingidas pelo algoritmo. Essas propostas baseiam-se em informações sobre o problema ou mesmo na própria descrição da metaheurística e se trata de uma das questões mais importantes a se estabelecer para o algoritmo, a qual abrangerá o princípio de colaboração, característico em *ACO*.

Outro trabalho incluído na revisão bibliográfica e relacionado ao tema de pesquisa é o trabalho de Morandin *et. al.*, (2008). O trabalho, apesar de não se tratar de uma abordagem baseada em *ACO*, define o cenário ao qual a abordagem proposta neste trabalho é aplicada e dessa forma foi utilizado como um dos critérios de avaliação para os resultados obtidos.

Morandin *et. al.*, (2008) desenvolveram uma abordagem baseada em Algoritmos Genéticos (AG) cuja modelagem considera o cenário de produção em nível de roteiros de produção. Assim, as operações pertencentes aos roteiros são vistas como uma parte única no cromossomo que representa as soluções. As operações de mutação e cruzamento são realizadas a partir da manipulação dos roteiros de cada produto. Essa perspectiva motivou o desenvolvimento de uma abordagem ACO para o problema seguindo as mesmas características de modelagem baseada em roteiros de produção. Dessa forma, os trabalhos citados anteriormente Kato *et al.,* 2009 e Kato et al. 2010 foram os precursores da abordagem proposta nesta dissertação

Para a avaliação dos resultados, Morandin *et. al.*, (2008) realizaram uma comparação do AG proposto com outras quatro abordagens da literatura. São

discutidos e apresentados os resultados com variações do tamanho do cenário avaliado, nos quais o AG pode encontrar soluções de melhor qualidade em relação às outras abordagens em poucos segundos.

Capítulo 3

PROPOSTA DE UMA ABORDAGEM ACO PARA A PROGRAMAÇÃO REATIVA DA PRODUÇÃO

3.1 Considerações Iniciais

De acordo com os temas discutidos no capítulo de revisão bibliográfica, baseando-se nos trabalhos da literatura, algumas considerações tomam parte das principais preocupações no desenvolvimento de uma abordagem eficaz para o problema de programação da produção em um sistema de manufatura. Uma delas é a possibilidade de tratar do problema sobre dois aspectos, quanto ao modelo de representação e quanto ao método de solução, como discutido em Maggio (2005). Algumas vantagens sobre essa perspectiva estão na redução da complexidade computacional, por usar informações abrangidas no modelo de representação (PRAKASH *et al*. 2008; HUANG e LIAO, 2008) e uma vez que o modelo corresponda com as características do sistema de produção, este pode ser usado como forma de definir a estrutura do espaço de soluções do problema em consideração.

A definição da estrutura do espaço de soluções compreende as restrições do problema de forma que as soluções construídas ou manipuladas pela aplicação do algoritmo de busca sejam válidas e não violem as propriedades do sistema. Essa

característica tem sido apresentada de forma clara em alguns trabalhos (BLUM e SAMPELS, 2004; BLUM, 2005; ROSSI e DINI, 2007; HUANG e LIAO, 2008), utilizando algum tipo de formalismo em grafos ou pela aplicação de abordagens que seguem esse princípio de criar soluções coerentes como, por exemplo, o algoritmo *List Scheduling* (*LS*) (GIFFLER e THOMPSON, 1960).

Essa definição implica diretamente no modelo de atualização de feromônio empregado, sendo que, a atualização dos pesos de feromônio pode variar em relação ao problema e suas restrições, visando com isso, uma melhor forma de indução da exploração sobre o espaço de solução. Dessa forma, um conhecimento específico do problema pode ser usado para direcionar a busca para regiões promissoras do espaço de soluções, levando em consideração a característica colaborativa da abordagem *ACO*.

Diversas abordagens utilizam como modelo de representação a teoria dos grafos de maneira a atender as preocupações mencionadas anteriormente, porém para os conceitos envolvidos não existe um formalismo específico e cabe a cada abordagem propor as propriedades que abrangerão o problema como em (ROSSI e DINI, 2007; PRAKASH *et al.* 2008; ZHOU *et al*. 2008) que apesar de tratarem da mesma classe de problema (programação em um FMS) apresentam propriedades distintas para a modelagem em grafos.

O problema de programação da produção considerado está na alocação de recursos compartilhados no tempo, respeitando as restrições do problema, de forma que o resultado gerado seja a conclusão do conjunto de operações no menor tempo possível (AGUIRRE, 2007). De forma mais específica, as seguintes características são consideradas no problema (PINEDO, 2008).

- Os recursos são compartilhados e disputados de maneira concorrente.
- Pode existir paralelismo entre operações que utilizam recursos distintos. Por exemplo, a operação de um determinado produto pode estar sendo executada na Máquina 01, enquanto uma operação de outro produto pode estar sendo executada no mesmo tempo na Máquina 02.
- Cada produto possui um conjunto pré-definido de operações que denota o percurso por onde o produto passa até o seu acabamento. Esse percurso será tratado como roteiro de fabricação do produto.

- As operações devem seguir a precedência determinada pelo roteiro de fabricação.
- Existe a flexibilidade de roteiros de fabricação, a qual possibilita que um produto pode ser fabricado por uma sequencia de etapas de processamento distintas.

Devido à natureza dinâmica do ambiente, em que a ocorrência de eventos não programados é um fator imprevisível e a programação da produção estabelecida para um determinado cenário pode não corresponder a uma solução de qualidade perante as modificações emergidas, é incluída nesta abordagem um mecanismo para determinar a reprogramação da produção perante as alterações do cenário. Dentre os diversos tipos de eventos que podem ocorrer, neste trabalho são considerados a quebra de máquinas e falta de matéria prima.

A partir dessas considerações, define-se então uma abordagem de busca para determinar uma sequencia de operações ao longo do tempo que resultará no menor valor de *makespan*. A abordagem proposta é baseada em um método construtivo que torne possível atender as especificidades do problema. Dessa forma, as características descritas anteriormente são tratadas a cada etapa de escolha dos elementos que compõem a solução.

Seguindo os conceitos da metaheurística *ACO*, a relação estabelecida entre as operações indica um conhecimento colaborativo obtido durante o processo de busca. Como apresentado em Silva *et al.,* (2008), esse conhecimento pode ser explorado de forma a servir como referência para uma futura exploração do espaço de soluções a partir de supostas modificações no ambiente, dada pelas características dinâmicas mencionadas.

A descrição geral de cada procedimento é apresentada na próxima seção.

3.2 Descrição dos procedimentos

A abordagem é constituída de duas etapas. Na primeira, a programação da produção é tratada em nível de planejamento, sendo as especificações do cenário de produção fornecidas como entrada (quantidade de produtos, roteiros de

produção, recursos a serem utilizados) e a partir dessas, é efetuada a busca de uma solução visando minimizar o tempo total de produção. Na segunda etapa, é considerada a ocorrência de um evento inesperado, o que provoca mudanças no cenário até então considerado, com isso uma nova solução é obtida por meio de uma reprogramação da produção sobre o novo cenário especificado. A estrutura geral da abordagem em cada etapa é ilustrada na Figura 3.1.

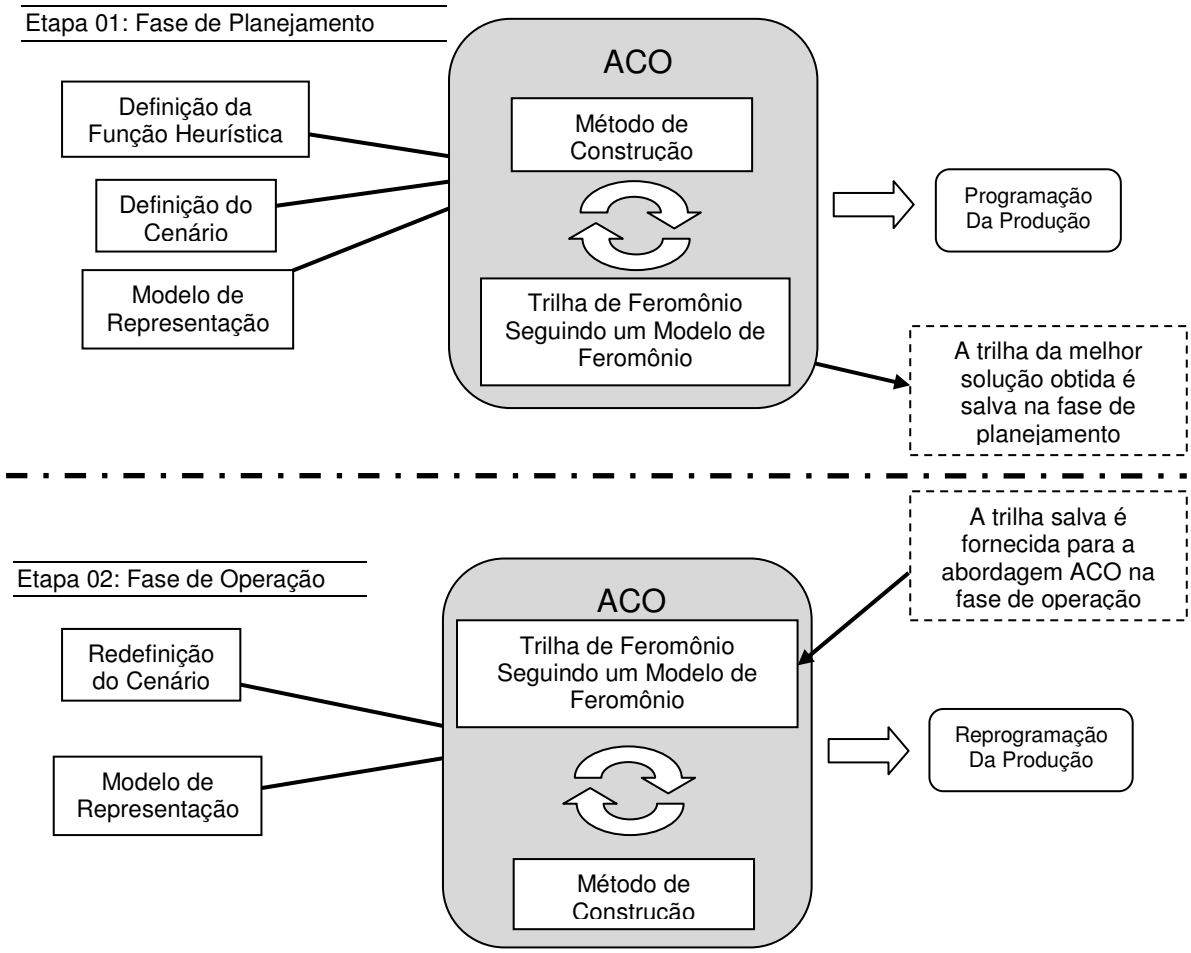

Figura 3.1 – Estrutura da abordagem proposta.

Para a aplicação da primeira etapa da abordagem, é necessária a definição do cenário a ser avaliado. Essa definição corresponde a uma instância dos valores que serão assumidos para o ambiente como: número de máquinas, produtos a serem fabricados, restrições quanto aos roteiros de produção, dentre outras convenções que são adotadas.

A partir desse cenário, é gerado o modelo de representação que indica as restrições definidas quanto às operações de cada produto. O modelo corresponde a uma representação em grafos cujos vértices denotam todas as operações pertencentes ao cenário. As arestas definem a relação entre os vértices de forma a possibilitar a navegação entre os nós. Cada formiga constrói um percurso escolhendo probabilisticamente cada operação ao longo do tempo, considerando as características do problema. O percurso, por sua vez, denota a solução do problema de programação.

Para o cálculo de probabilidade de escolha das operações, na etapa de construção das soluções, é necessária a definição de um modelo de feromônio e uma função heurística. Para a função heurística é aplicado um procedimento que determina o valor relativo de uma determinada operação em relação ao roteiro em que esta se encontra. Melhores detalhes sobre esse procedimento são descritos na próxima seção, na qual inclui um cenário exemplo para a demonstração de cada procedimento.

O modelo de feromônio é definido por meio de uma associação ao modelo de representação do problema, dessa forma, cada estado possui uma informação que é compartilhada entre as formigas para estabelecer o princípio de colaboração desejado. A atualização da trilha é realizada seguindo as diretrizes da variante *Max-Min Ant System*. Essa atualização é realizada após todas as formigas terem construído seus percursos.

Os procedimentos de atualização e construção da solução são realizados até que o número de iterações seja atingido. Por fim, é apresentada a solução que denota o melhor caminho percorrido pelas formigas. Os valores da trilha de feromônio que levaram a tal solução são salvos, uma vez que indicam certo aprendizado adquirido durante o processo de exploração dos estados. Os valores são usados na segunda etapa da abordagem, a de reprogramação, os quais podem corresponder a um conhecimento prévio do cenário, por estabelecerem relações entre as operações que levam a soluções de qualidade.

Para a segunda etapa, são consideradas as mesmas definições para a função heurística e o modelo de atualização. Nesta etapa é necessária a redefinição do cenário de produção devido à ocorrência de algum evento que interrompa o fluxo de execução das operações, seja por quebra de máquina ou falta de matéria prima.

Esta modificação implica diretamente no modelo de representação e no modelo de feromônio, onde os nós que pertencem aos estados inválidos são descartados para ambos os modelos.

Como mencionado, os valores da trilha de feromônio nessa etapa, são inicializados com os valores que indicam a melhor solução encontrada para o cenário da primeira etapa. O intuito dessa atribuição é o de verificar se o conhecimento prévio obtido na primeira etapa pode favorecer a busca da etapa de reprogramação, direcionando a exploração para regiões promissoras logo no inicio da execução do algoritmo.

As etapas de construção e atualização da trilha de feromônio seguem as mesmas definições da primeira etapa e ao término desses procedimentos, a reprogramação para o novo cenário é definida e apresentada.

No próximo capítulo, os procedimentos descritos serão novamente referidos, porém tendo como base um cenário de exemplo, onde os detalhes mais específicos poderão ser mais bem contextualizados.

Capítulo 4

APLICAÇÃO DA ABORDAGEM E ANÁLISE DE RESULTADOS

4.1 Considerações Iniciais

Neste capítulo serão detalhados os procedimentos descritos no capítulo anterior. Para isso, inicialmente é apresentada a descrição geral do cenário, incluindo algumas convenções adotadas. O cenário corresponde a uma instância pequena do problema como forma de ilustrar cada procedimento. Após o detalhamento da abordagem, são apresentadas as análises realizadas e a comparação com outras abordagens para o problema em consideração.

4.2 Descrição do problema

O problema de programação da produção em pode ser definido como segue. Considere um conjunto de N produtos $P_1, P_2 \ldots P_N$ a serem manufaturados em K máquinas $M_1, M_2 \ldots M_K$. Os produtos são divididos em famílias e cada família usa uma matéria prima diferente. Cada produto P_i é construído de uma seqüência definida O_{ji} de operações que representam seu roteiro de produção. Cada produto pode ter um ou mais roteiros de fabricação com isso, determinadas operações podem ser executadas em máquinas distintas e com diferentes tempos de processamento.

A restrição que indica a qual máquina é destinada a execução da operação é estabelecida pelos roteiros de produção R_{ir}, sendo *i* o índice do produto e *r o* índice do roteiro. Cada roteiro pode variar em números de operações, sendo NR_{ir} o número total de operações do roteiro *r* do produto *i*. Na Tabela 4.1 é ilustrado um exemplo de definição de roteiros para três produtos e a especificação da matéria prima utilizada para a fabricação do produto. No exemplo, os produtos P_1 e P_2 possuem dois roteiros de produção cada um e o produto P_3 apenas um roteiro, os produtos P_1 e P_3 usam a matéria prima MP_A e o produto P_2 a matéria prima MP_B. Os valores indicados na relação entre os produtos e suas respectivas operações são expressos pelas máquinas que constituem cada roteiro. Dessa forma, o roteiro R_2 do produto P_2 indica que, para que o produto esteja completo é necessário que as operações sejam executadas nas máquinas M_2, M_4, M_5 e M_6 ou então em seu roteiro alternativo R_1 com as máquinas M_1, M_4, M_5 e M_6.

Tabela 4.1 – Exemplo de Definição dos Roteiros de Produção

MP_A	P_1	R_{11}	M_1	M_2	M_3	M_4	M_5
		R_{12}	M_1	M_2	M_3	M_6	-
MP_B	P_2	R_{21}	M_1	M_4	M_5	M_6	-
		R_{22}	M_2	M_4	M_5	M_6	-
MP_A	P_3	R_{31}	M_3	M_5	M_6	-	-

Os tempos de processamento das operações TO_{jir} do produto P_i no roteiro R_{ir} na máquina M_k são pré-definidos. Na Tabela 4.2, é ilustrado um exemplo desta descrição considerando as especificações de roteiros da Tabela 4.1. Os valores indicados na relação entre as operações e roteiros são expressos em unidades de tempo (u.t).

Tabela 4.2 – Tempo de Processamento das Operações.

MP_A	P_1	R_{11}	434	452	400	472	460
		R_{12}	434	452	400	421	-
MP_B	P_2	R_{21}	458	485	402	435	-
		R_{22}	443	485	402	435	-
MP_A	P_3	R_{31}	469	432	445	-	-

As restrições a seguir também são consideradas para o problema visando uma simplificação do modelo de representação. Estas convenções influenciam no grau de complexidade do problema permitindo assim, a definição de um modelo que se comporte dentro de um universo observável:

- Cada máquina executa somente uma operação por vez;
- Não é permitido preempções das operações;
- O tempo de transporte de um recurso para uma máquina é considerado junto com o tempo de execução da operação;
- O tempo de configuração não é considerado;
- A ordem de execução das operações para cada produto é fixa e não pode ser alterada;
- O *buffer* de cada máquina é ilimitado;
- Cada produto pode possuir um ou mais roteiros de fabricação.

Para a característica dinâmica do problema são considerados os eventos de quebra de máquina e falta de matéria prima (impedindo a produção de determinado produto). Com isso, após a especificação de um determinado cenário do problema, a programação da produção é determinada sobre esse cenário. Em seguida um dos eventos é disparado e uma nova especificação do cenário é estabelecida.

Assim como apresentado em Silva *et al.* (2008), neste trabalho será avaliada a viabilidade da experiência adquirida pela abordagem ACO durante o processo de busca, onde as informações do aprendizado são utilizadas para estabelecer a reprogramação da produção sobre o novo cenário estabelecido após a ocorrência do evento.

O problema então consiste em alocar todas as operações, de seus respectivos produtos às maquinas, o que por conseqüência nos permite determinar a seqüência de execução de cada operação de acordo com uma função objetivo. A função objetivo utilizada neste trabalho é o *makespan*, denotado (PINEDO, 2008):

$$TC_{max} = TC_{ji} \tag{4.1}$$

Onde C_{Max} denota o valor da última operação *j* do último produto P_i. TC_{ji} é o tempo em que a operação *j* do produto P_i está completa.

4.3 Detalhamento dos procedimentos da abordagem

4.3.1 Definição do Cenário

O cenário para a aplicação da abordagem é definido por meio das informações do sistema de produção como número de produtos a serem fabricado, número de máquinas disponíveis, roteiros de produção de cada produto e tempo de execução das operações de um determinado produto em cada máquina. Para exemplificar os demais procedimentos serão usadas as Tabelas 4.1 e 4.2, apresentadas anteriormente, que correspondem às definições dos roteiros de três produtos e o tempo de execução de cada operação, respectivamente.

4.3.2 Modelagem de representação do problema

A modelagem proposta procura seguir as especificidades discutidas em abordagens desenvolvidas com a metaheurística *ACO* (DORIGO e DI CARO, 1999). Dessa forma, o espaço de soluções para o problema pode ser representado por um grafo *G = (V, A, E)*, onde *V* denota o conjunto de vértices cujos valores indicam todas as operações do cenário de produção mais as operações falsas (*dummy*) de início e fim (representadas pelos símbolos Ø e *). *A* denota o conjunto de arcos que unem os vértices para a construção das soluções respeitando as restrições de precedência entre as operações. *E* denota o conjunto de arcos disjuntivos entre as operações como forma de permitir o paralelismo na execução das operações. Na Figura 4.1 é apresentado um exemplo dessa descrição considerando *G' = (V,A)* e o cenário da Tabela 4.1.

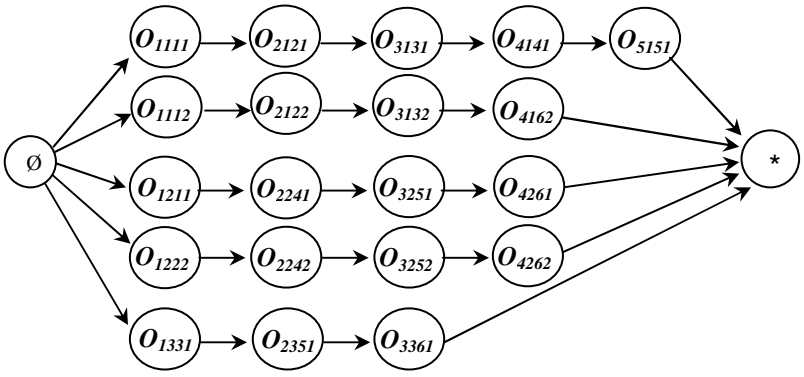

Figura 4.1 – Modelo de representação do problema.

Os valores indicados nos vértices correspondem ao número da operação, o produto a máquina e o roteiro ao qual a operação pertence. Por exemplo, o estado O_{2122} é referente a segunda operação do produto P_1 na máquina M_2 do segundo roteiro de produção do produto P_1.

Por questão de simplicidade, na representação apresentada na Figura 4.1, o conjunto de arcos *E* do grafo *G = (V,A,E)* não é descrito. Esse conjunto consiste das arestas que unem cada uma das operações a todas as outras operações do cenário, uma vez que partindo de qualquer operação é possível iniciar qualquer outra operação de outro produto de forma paralela (respeitando restrições).

4.3.3 Definição da função heurística

O valor heurístico é definido visando propor um direcionamento da busca logo no início da execução do algoritmo, sendo que todos os valores de feromônio inicializam iguais e dessa forma induzem de forma proporcional a todos os possíveis estados (Dorigo e Stützle). O valor heurístico corresponde a uma informação estática inerente ao problema, que normalmente procura condizer com a função objetivo definida. Dessa forma, considerando as definições do problema apresentado, o valor heurístico pode ser definido por:

$$d_{jir}=\frac{TO_{jir}}{\sum_{l=1}^{NRir} TO_{lir}}, \quad \forall j \in P_{ir} \tag{4.2}$$

Onde, os valores TO_{jir} são dados pela Tabela 4.2. e as demais expressões são denotadas nas seções anteriores.

De acordo com a expressão (4.2), é possível definir um valor relativo entre uma determinada operação e o roteiro de produção ao qual esse se encontra. Com isso, operações de roteiros distintos que utilizam a mesma máquina podem variar seu grau de informação heurística por considerar o roteiro ao qual a operação se encontra. O resultado da aplicação da expressão (4.1) ao grafo da Figura 4.1 é ilustrado na Figura 4.2.

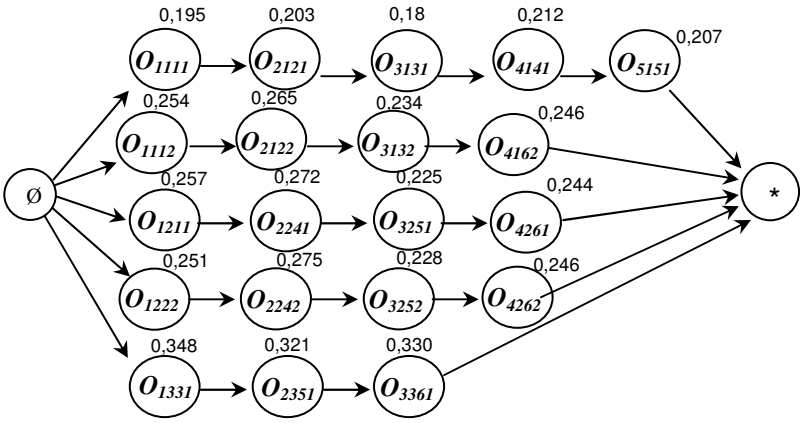

Figura 4.2 – Informação heurística adicionada aos estados do modelo.

No exemplo, as operações O_{1111} e O_{1112} utilizam ambas a máquina 1 e pertencem ao mesmo produto P_1. Por usarem a mesma máquina, o tempo de processamento é igual e basear-se nesse valor pode não tão significante quanto os valores obtidos em d_{jir}. Como é ilustrado na Figura 4.1, a operação O_{1112} é mais significante que a operação O_{1111} uma vez que o roteiro ao qual esta se encontra é menor em termos de tempo total necessário para executar todas as operações do roteiro.

4.3.4 Modelo de feromônio

O modelo de feromônio é baseado na estrutura do espaço de soluções da Figura 4.1. Os valores da trilha correspondem aos valores atribuídos a cada elemento do espaço de estados. São inicializados seguindo as diretrizes da variante *Max-Min Ant System* (STÜTZLE e HOOS, 2000).

A atualização da trilha é realizada após todas as formigas terem construído seus percursos. Dessa forma, cada estado escolhido pela melhor formiga (melhor global ou melhor da iteração) terá seu valor de feromônio gradativamente atualizado em função da qualidade em que contribui na otimização da função objetivo (4.1).

A evaporação da trilha precede o procedimento de depósito é tem como objetivo reduzir os valores dos estados que não pertencem ao melhor percurso encontrado. É realizado seguindo um parâmetro ρ, que especifica a taxa em que os valores serão reduzidos.

4.3.5 Atributos da formiga

Para realizar a construção das soluções para o problema de programação de forma válida, cada formiga deve considerar o paralelismo entre as operações, o compartilhamento das máquinas do cenário e a precedência entre as operações do roteiro. Cada formiga constrói uma solução de forma individual e possui os seguintes atributos:

- **Tempo de produção decorrido:** inicialmente todas as formigas são inicializadas no tempo de produção zero, e de acordo com o andamento do fluxo de operação, o tempo é incrementado.
- **Registro da operação atual de cada produto**: uma vez que cada produto pode estar em estágios de produção distintos, esse registro armazena a operação atual de cada um em relação ao seu roteiro de produção.
- **Controle de alocação das máquinas:** para efetivar uma determinada escolha de operação do espaço de estado, a formiga deve verificar se a máquina está disponível no tempo de produção corrente. Do contrário, a operação deve aguardar o incremento do tempo para a liberação da máquina. Cada formiga controla esses procedimentos de forma individual.
- **Percurso percorrido:** uma vez que as restrições mencionadas foram satisfeitas e determinado estado está apto a ser escolhido, esse é armazenado nessa estrutura, para a posterior atualização da trilha.
- **Estados não visitados:** considerando os estados atuais de cada produto, os próximos estados são obtidos pela estrutura da Figura 4.1.

A exploração efetuada pelas formigas é realizada a partir do controle desses atributos em conjunto com algumas considerações da etapa de construção. A etapa de construção é descrita a seguir.

4.3.6 Construção das soluções

Para a definição do procedimento de construção das soluções é necessário considerar as seguintes propriedades:

- **Flexibilidade de roteiros:** considerando os roteiros de produção de um mesmo produto, se existir uma sequencia de operações iniciais iguais em

mais de um roteiro, pode-se assumir que o roteiro de produção é indefinido até o final dessa sequencia. Dessa forma, a construção da solução pode seguir em todos os roteiros alternativos até a necessidade de sua definição. Por exemplo: os estados O_{1111}, O_{2121}, O_{3131} da Figura 4.1 utilizam as mesmas máquinas dos estados O_{1112}, O_{2122}, O_{3132} e são as operações iniciais do produto P_1. Neste estágio de produção, o roteiro é ainda indefinido e somente na quarta operação é que se estabelece a operação que define o roteiro. Com isso, a formiga pode percorrer pelos estados iniciais, assumindo-os como único, e garantindo maior possibilidade de escolha.

- **Definição da vizinhança:** para determinar as próximas operações de um produto em determinado estágio de produção, é necessária a definição dos próximos estados desse produto. Esses estados são obtidos por meio de uma consulta à estrutura do espaço de soluções (Figura 4.1), sendo a vizinhança o resultado de todos os próximos estados de todos os produtos. Assumindo que todos os produtos estão no estado inicial Ø. A vizinhança de cada produto corresponde aos estados destacados na Figura 4.3.

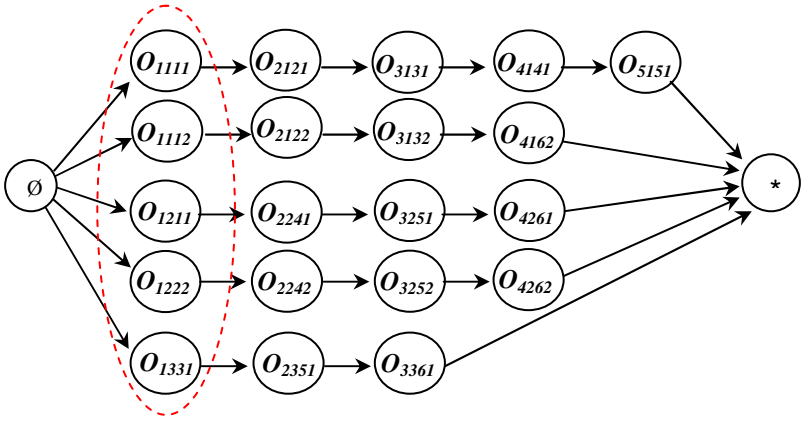

Figura 4.3 – Vizinhança de cada produto no estado inicial Ø.

Como pode ser observado na Figura 4.3, a escolha de um determinado estado implica na atualização da vizinhança de todos os demais produtos para a primeira etapa de construção das soluções. Por exemplo, supondo que o produto P_2 está no estado O_{1222}, ocupando a máquina M_2, as demais operações do mesmo produto terão probabilidade zero de serem escolhidas, de forma a não interferirem na escolha de outros produtos em

máquinas distintas. Nesse exemplo, a operação O_{1211} terá probabilidade zero e não interferirá nas operações O_{1112} e O_{1111} do produto P_1 que utilizam a operação M_1.

- **Cálculo de probabilidade:** para determinar a probabilidade da escolha do próximo estado de um produto em relação a sua vizinhança, devem-se considerar também como vizinhos os estados cujas operações são executadas na mesma máquina desses estados da vizinhança. A probabilidade é calculada em relação ao recurso que está sendo compartilhado, possibilitando que operações de produtos com alta qualidade de valor heurístico e de valor de feromônio tenham maiores chances de serem escolhidas para processamento. Considerando os estados em destaque da Figura 4.3, a probabilidade do produto P_2 sair do estado inicial Ø e escolher uma das operações disponíveis, no caso O_{1222} e O_{1211}, deve considerar que as operações O_{1111} e O_{1112} do produto P_1 ocupam a mesma máquina M_1 e dessa forma possui um grau de influência na decisão do próximo estado O_{1211} do produto P_2. Na Figura 4.4 são destacados os estados que utilizam a máquina M_1 e os estados vizinhos do produto P_2 no estado inicial.

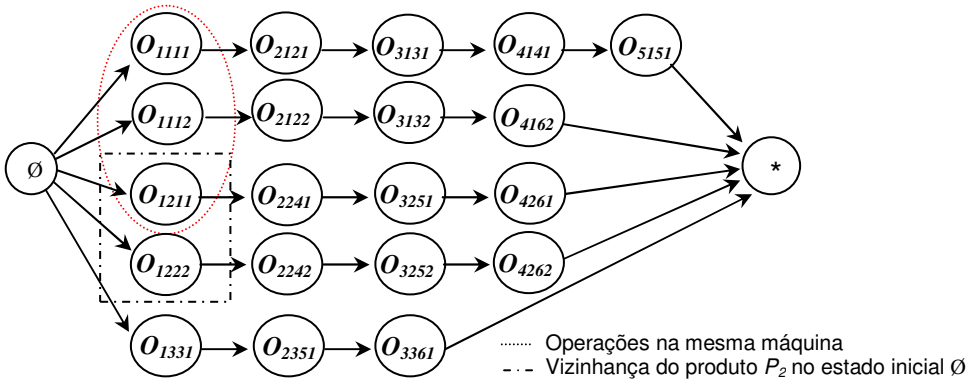

Figura 4.4 – Probabilidade dos estados na mesma máquina.

Dessa forma, o estado O_{1211} do produto P_2 tem probabilidade relativa em função das operações do produto P_1. E o cálculo de probabilidade deve considerar esse fator.

O procedimento de construção é executado até que todas as formigas tenham construído seus percursos. O percurso representa a ordem de escolha dos elementos do espaço de estados ao longo do período de produção e portando, denota a programação da produção, estabelecendo o tempo de início e fim de execução de cada operação e respeitando suas restrições.

No fluxograma da Figura 4.5 são apresentados os principais processos da etapa de construção. Note que a mudança de estado primeiro leva em consideração a probabilidade de escolha e só em seguida verifica se a máquina está ocupada. Essa característica permite que estados com alta probabilidade fiquem em fase de espera, aguardando que a máquina em uso seja desocupada.

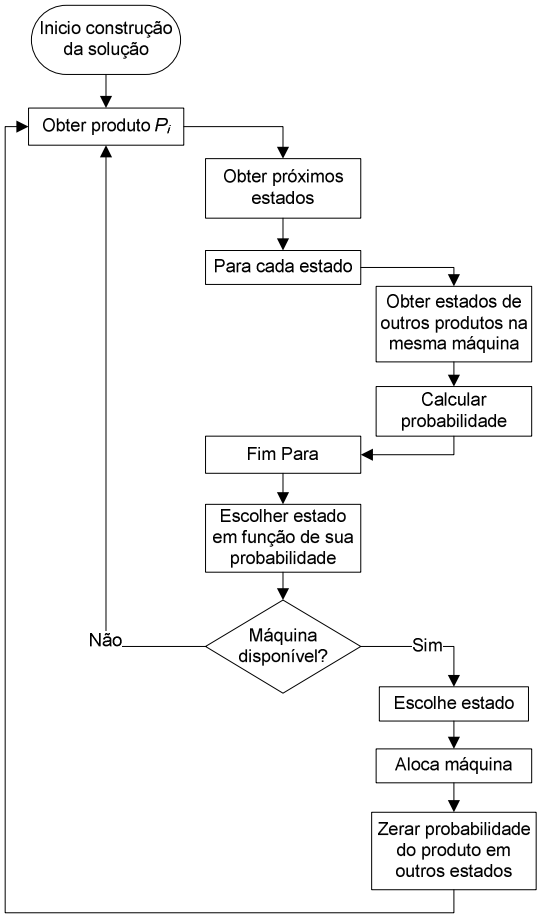

Figura 4.5 – Fluxograma do procedimento de construção das soluções.

4.3.7 Programação da produção

A programação da produção é obtida por meio da melhor solução encontrada durante as iterações do algoritmo. A descrição de cada estado escolhido inclui o tempo de inicio e fim de cada operação dos produtos do cenário e pode ser

representada por um gráfico de Gantt com é ilustrado na Figura 4.6. Para o exemplo, são consideradas as especificações das Tabelas 4.1e 4.2.

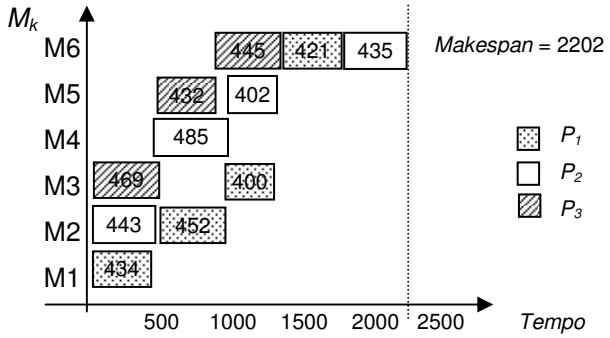

Figura 4.6 – Representação da programação em gráfico de Gantt.

4.3.8 Redefinição do cenário

Para a etapa de reprogramação da produção é realizado o tratamento de dois eventos que interrompem o fluxo normal de execução das operações, sendo esses a quebra de uma determinada máquina e falta de matéria prima.

Nessa etapa, é necessária a redefinição do cenário de maneira a refletir nas alterações ocorridas no ambiente de produção. A redefinição é baseada em informações de entrada para a abordagem onde ocorrências de falta de matéria prima e ou máquina quebrada são informadas.

A partir dessas informações, os dados do cenário da fase de planejamento são ajustados e uma redefinição do modelo de representação também é necessária. Na Figura 4.7 é ilustrada redefinição do modelo considerando que a máquina M_2 está quebrada.

Como pode ser observada na figura, a retirada da máquina do cenário de produção implica no cancelamento do roteiro de fabricação dos produtos que fazem uso da máquina. Para o caso de não haver roteiro alternativo para os produtos, a fabricação desses é então impossibilitada.

Para a ocorrência de falta de matéria prima, primeiramente é identificada quais tipos de matéria prima estão em falta. Com as informações do cenário de produção é possível identificar todos os produtos que utilizam a matéria prima em falta e a partir da identificação, os produtos são removidos do cenário de produção, assim como todas as operações dos roteiros de cada produto identificado.

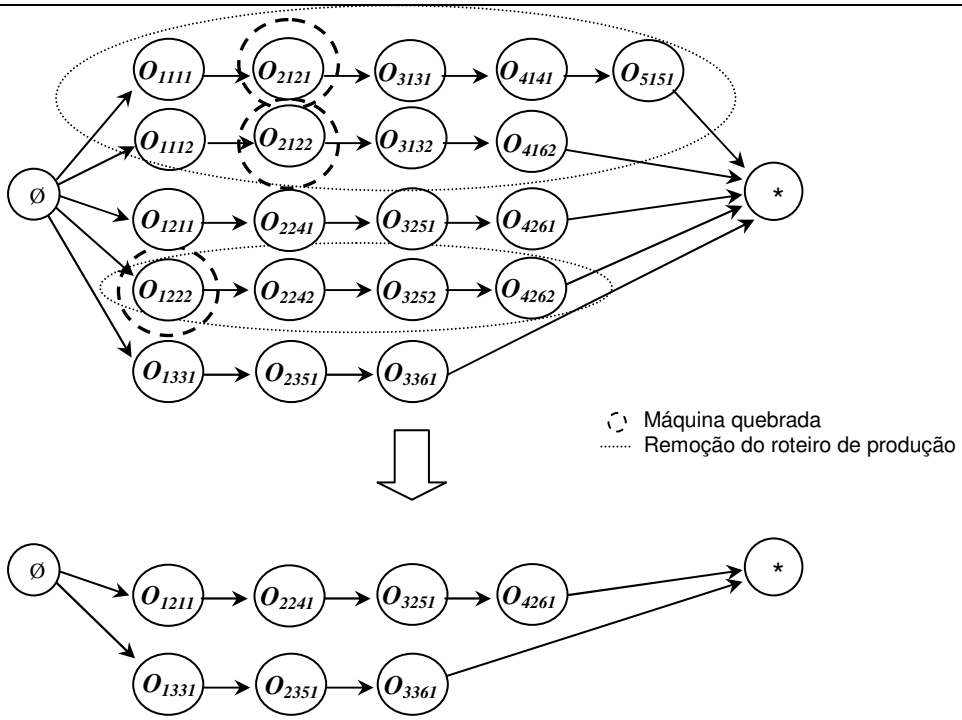

Figura 4.7 – Processo de redefinição do modelo de representação.

4.3.9 Reprogramação da produção

Para a reprogramação da produção são consideradas as definições do novo cenário, a função heurística e o modelo de feromônio definidos na etapa de planejamento.

Partindo do princípio de que a relação estabelecida entre os estados do cenário da etapa de planejamento pode indicar um conhecimento sobre o problema, os valores da trilha de feromônio são usados na fase de reprogramação. O mecanismo consiste em inicializar os valores da trilha de feromônio na etapa de re-planejamento com os valores que levaram a uma melhor solução no cenário de planejamento.

Uma vez que determinados estados da trilha de feromônio da etapa de planejamento podem correspondem a estados eliminados da fase de operação, é necessário um ajuste na trilha de maneira a remover tais estados.

A partir dessas considerações a reprogramação da produção é obtida.

Nas próximas seções são apresentados os resultados experimentais obtidos, incluindo as especificações do cenário avaliado, análise dos parâmetros fornecidos a metaheurística *ACO* e comparação dos resultados com outras abordagens para o problema de programação da produção em Sistemas Flexíveis.

4.4 Análise dos parâmetros ACO

Baseando-se na revisão dos trabalhos relacionados, a parametrização adotada para abordagens *ACO* normalmente são determinadas por meio de procedimentos experimentais empíricos. Algumas abordagens como Prakash *et al.,* (2008) e Huang e Liao, (2008) fazem menção aos experimentos aplicados e adotam uma determinada configuração de parâmetros para todas instâncias de testes. Liouane *et al.* (2007) apresentam apenas uma variação entre os parâmetros *α, β*. Benbouzid *et al.* (2008) analisam, além de *α*, *β,* o coeficiente de evaporação e o número de ciclos utilizados.

Na tentativa de investigar a parametrização de algoritmos *Ant System* para a programação da produção em *job shop* Figlali *et al.* (2009) afirmam que soluções melhores são obtidas quando são definidos valores altos para o número de formigas e para o número de iterações, e valores baixos para *β*.

Dorigo e Stützle (2004) sugerem algumas configurações para abordagens aplicadas ao problema do caixeiro viajante, considerando uma revisão da literatura para as principais variantes do algoritmo *AS*, incluindo *MMAS*.

Seguindo essas considerações, a avaliação foi realizada a partir de diversas configurações de parâmetros sendo adotada a melhor configuração para ser aplicada as instâncias dos problemas.

Inicialmente é necessária a definição do cenário ao qual foi aplicada a avaliação. A descrição do cenário é apresentada na Tabela 4.3.

Tabela 4.3 – Cenário de produção com 9 máquinas e 9 produtos.

Produtos		Roteiros de produção
P_1	R_{11}	$M_1\ M_2\ M_4\ M_5\ M_7\ M_9$
	R_{12}	$M_3\ M_4\ M_5\ M_6\ M_8\ M_9$
P_2	R_{21}	$M_1\ M_2\ M_3\ M_4\ M_5\ M_6\ M_7$
	R_{22}	$M_2\ M_3\ M_5\ M_7\ M_8\ M_9$
P_3	R_{31}	$M_4\ M_5\ M_6\ M_7\ M_8$
	R_{32}	$M_2\ M_3\ M_7\ M_8\ M_9$
P_4	R_{41}	$M_2\ M_3\ M_4\ M_5\ M_6\ M_7$
	R_{42}	$M_1\ M_5\ M_6\ M_8\ M_9$
P_5	R_{51}	$M_4\ M_5\ M_6\ M_8\ M_9$
	R_{52}	$M_1\ M_2\ M_3\ M_5\ M_6$
P_6	R_{61}	$M_2\ M_4\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{62}	$M_1\ M_3\ M_6\ M_7\ M_8\ M_9$
P_7	R_{71}	$M_1\ M_2\ M_4\ M_5\ M_6\ M_9$

	R_{72}	$M_1\ M_2\ M_3\ M_7\ M_8\ M_9$
P_8	R_{81}	$M_4\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{82}	$M_3\ M_4\ M_5\ M_7\ M_8\ M_9$
P_9	R_{91}	$M_3\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{92}	$M_2\ M_4\ M_6\ M_7\ M_8\ M_9$

Os parâmetros a serem avaliados são: a influência relativa dos valores de feromônio (α), a influência relativa dos valores da função heurística (β), taxa de evaporação (ρ), número de formigas (n) e número máximo de iterações (NC).

Nas Figuras 4.8, 4.9 4.10 são apresentadas as variações de n e NC considerando os valores de α, β e ρ sugeridos em Dorigo e Stützle, (2004). O intervalo de avaliação corresponde a [5 – 100] para n e [50 – 200] para NC.

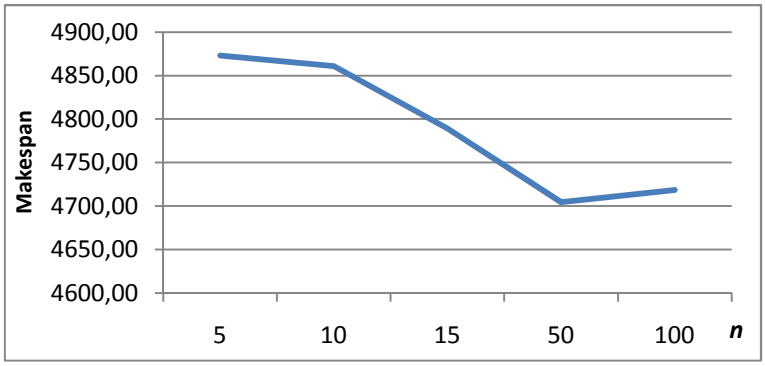

Figura 4.8 – Variações de *n*.

Como sugere os resultados da Figura 4.8, é necessário especificar um valor alto para o número de formigas. Contudo, buscou-se conciliar o número de formigas com tempo gasto na obtenção de resposta. Na Figura 4.9 são apresentados os resultados, sendo especificado o tempo em segundos. O valor resultante para esta configuração foi de $n = 15$.

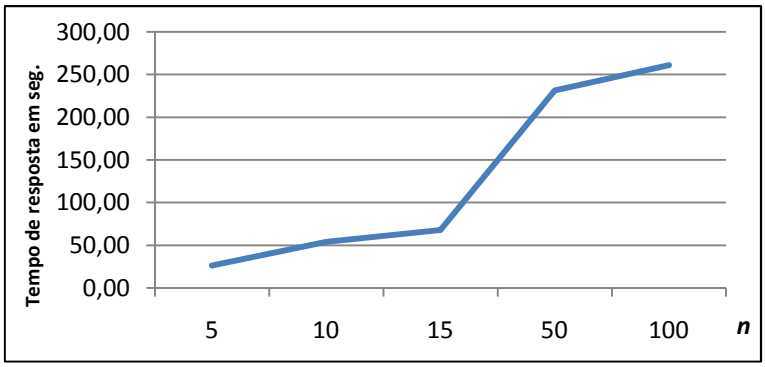

Figura 4.9 – Variação de *n* considerando tempo de resposta.

Para os valores do intervalo de *NC* da Figura 4.10 é considerado *n* = 5. De acordo com os resultados, para *NC* = 100 a abordagem obteve um bom resultado para os valores de *makespan*, porém com um número maior de *n*, pode se especificar um número menor de iterações visando a redução do tempo computacional. Na Figura 4.11 é ilustrado o comportamento das soluções obtidas pelo algoritmo para 100 iterações considerando o melhor resultado obtido em cada iteração. Os valores 50 e 75 para *NC* serão considerados para a avaliação dos próximos parâmetros.

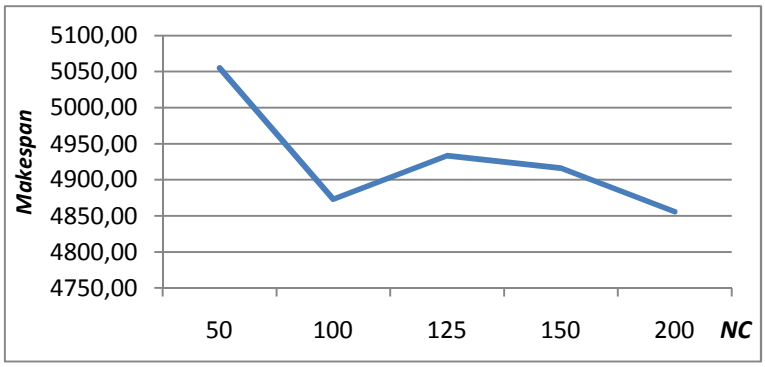

Figura 4.10 – Variações de *NC*.

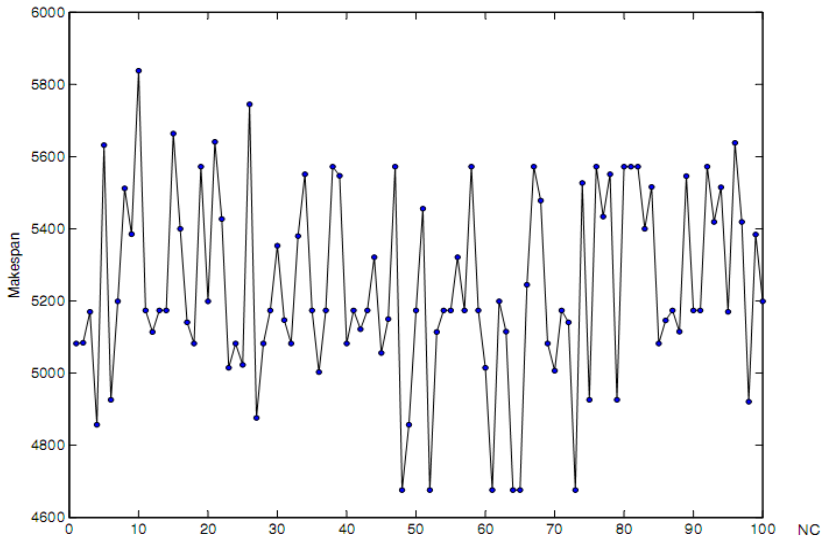

Figura 4.11 – Comportamento da melhor solução a cada iteração do algoritmo para *NC* = 100.

Na Figura 4.12 são apresentas algumas variações para o valor de α. Para os valores dos demais parâmetros são considerados β = 1, ρ = 0,02, *n* = 10 e *NC* = 50.

Observando os resultados verificou-se que para valores mais baixos de α, a qualidade das soluções tendem a melhorar.

Para a variação dos valores de β na Figura 4.13 são considerados os parâmetros α = 2, ρ = 0,02, *n* = 10 e *NC* = 50.

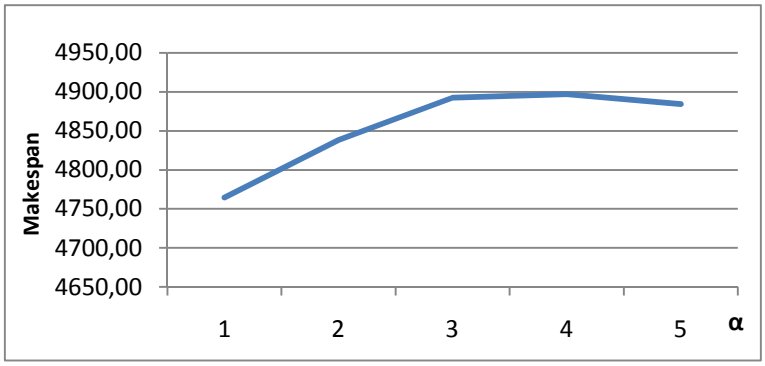

Figura 4.12 – Variações de α.

Observando os resultados da Figura 4.13, verificou-se que para valores mais altos de β a qualidade das soluções tendem a melhorar.

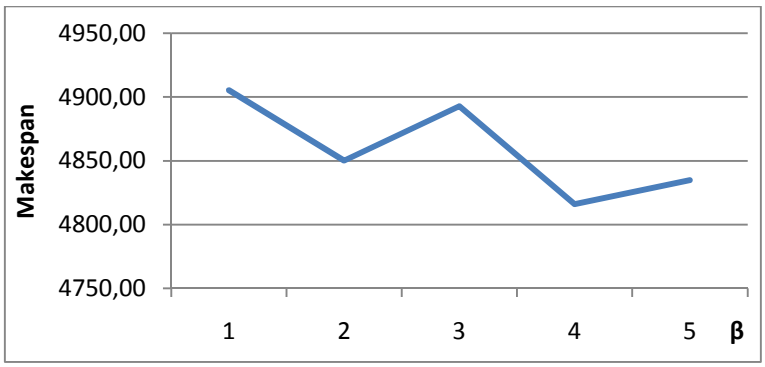

Figura 4.13 – Variações de β.

Para a definição dos valores da taxa de evaporação inicialmente definiu-se os parâmetros α = 2, β = 1, *n* = 5 e *NC* = 100. O Resultado é apresentado na Figura 4.13. No entanto, não houve uma tendência clara do comportamento da abordagem para essa configuração. Com isso, foi aplicada a configuração sugerida pela avaliação dos demais parâmetros ao mesmo intervalo de ρ (Figura 4.15).

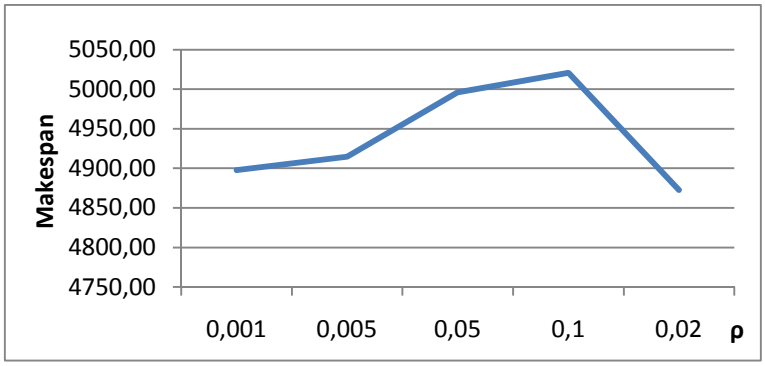

Figura 4.14 – Variações de ρ.

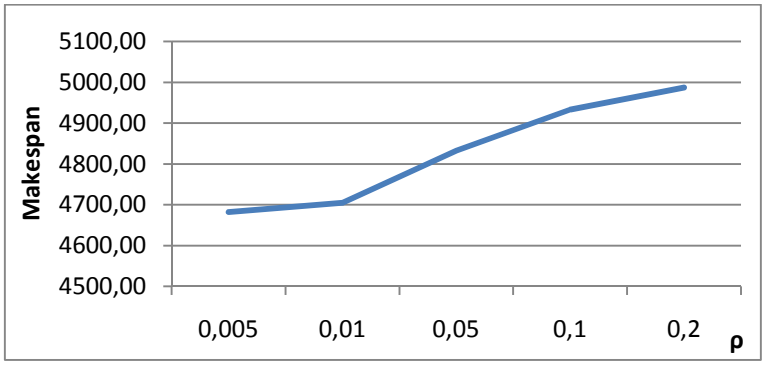

Figura 4.15 – Variações de ρ para os demais parâmetros definidos.

A configuração sugerida pela avaliação dos demais parâmetros corresponde a α = 2, β = 4, *n* = 15 e *NC* = 50. Para essa configuração, valores baixos de ρ tenderam a apresentar melhores resultados (Figura 4.15).

Dessa forma, a configuração resultante corresponde aos valores α = 2, β = 4, ρ = 0.005, *n* = 15 e *NC* = 50. Essa configuração será utilizada para a avaliação com outras abordagens.

Para a análise da diversidade dos percursos de cada formiga verificou-se que não ocorre estagnação. Dessa forma, a maioria das formigas mantém valores de *makespan* diferentes, o que sugere a exploração de percursos distintos. Na Figura 4.16 são apresentados os valores de *makespan* da última iteração do algoritmo para *n* = 15.

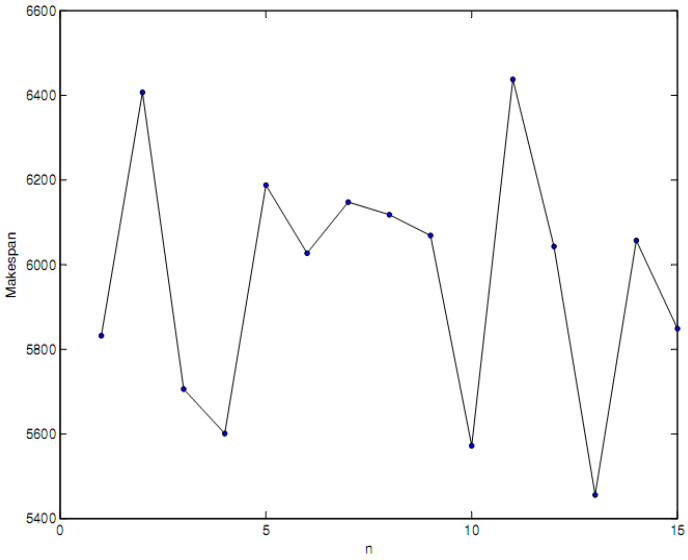

Figura 4.16 – Diversidade dos valores de *makespan* para *n* = 15.

4.5 Avaliação da abordagem

4.5.1 Considerações iniciais

Para a avaliação da abordagem proposta, foram realizados experimentos e os resultados foram comparados com a abordagem proposta em Morandin *et. al.*, (2008) e Kato *et. al.*, (2010), sendo a primeira baseada em Algoritmos Genéticos (AG) e a segunda em Otimização por Colônia de Formigas (*ACO*), ambas aplicadas ao mesmo problema de programação.

Todas as três abordagens foram codificadas usando o software Matlab. Os parâmetros foram obtidos pelas definições descritas nos próprios trabalhos e serão apresentadas a seguir. Os testes foram realizados em um computador Core 2 duo com 2GB de memória.

4.5.2 Definição dos cenários para a avaliação

Os cenários foram gerados seguindo as especificações da Tabela 4.4. O cenário apresentado na seção anterior (Tabela 4.3), também foi gerado seguindo essas especificações. Os tempos de processamento de cada operação foram gerados aleatoriamente e são apresentados na Tabela 4.5.

Tabela 4.4 – Especificação dos cenários.

#	Produtos	Máquinas	Número de Roteiros	Numero de Operações
1	P_i, = 9	M_k = 9	[2,5] para cada P_i	[5,7] máquinas

Tabela 4.5 – Temos de processamento para os cenários.

Máquinas /Produtos	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
M_1	428	439	453	403	481	446	414	491	458
M_2	423	433	474	436	440	495	457	419	486
M_3	459	487	417	410	477	474	452	435	416
M_4	433	405	447	410	442	448	426	491	454
M_5	467	447	486	400	450	469	493	495	452
M_6	461	497	496	468	468	408	408	452	438
M_7	464	495	459	489	436	454	457	477	484
M_8	455	469	489	439	486	424	497	452	435
M_9	418	439	480	457	435	482	445	408	416

Para cada cenário avaliado foi definida uma quantidade de 35 testes. No primeiro cenário (ver Tabela 4.3) foram aplicadas as três abordagens sem considerar a ocorrência de eventos no cenário. Na abordagem proposta por Morandin *et. al.,* (2008) foram utilizados os seguintes valores para os parâmetros: tamanho da população = 30, taxa de cruzamento = 0,8 (80%), taxa de mutação = 0,05 (5%) e como o critério de parada de 100 gerações. Na abordagem proposta por Kato *et. al* (2010) foram utilizados os seguintes valores para os parâmetros: influência do valor de feromônio = 1, influência do valor heurístico = 2, taxa de evaporação = 0,02 número de formigas = 50 e número de iterações = 75.

Na Tabela 4.6 são apresentados os resultados de cada teste. As colunas indicam o número do teste, o *makespan* obtido e tempo de resposta calculado em segundos por cada abordagem. Os valores de *makespan* estão expressos em unidades de tempo (u.t).

Tabela 4.6 – Resultados obtidos para o primeiro cenário.

#	Proposta ***Makespan***	Proposta Tempo (seg.)	Morandin ***Makespan***	Morandin Tempo (seg.)	Kato ***Makespan***	Kato Tempo (seg.)
1	4676	28,81	4676	5,50	4676	22,58
2	4676	28,09	4745	5,50	4676	22,54
3	4676	28,42	5026	5,68	4878	22,59
4	4676	28,73	5035	5,52	4676	22,74
5	4676	27,55	5473	5,44	4676	22,74
6	4676	28,72	5082	5,63	4676	22,93
7	4921	28,99	5192	5,43	4768	22,97
8	4676	28,01	5015	5,49	4768	22,83

9	4676	28,95	4979	5,62	4676	22,79
10	4676	28,54	5108	5,63	4676	22,70
11	4676	28,66	5123	5,63	4676	22,71
12	4676	28,50	5374	5,47	4676	22,78
13	4676	28,40	5097	5,70	4676	22,77
14	4951	27,42	5035	5,56	4676	22,41
15	4676	28,08	5115	5,60	4676	22,56
16	4676	28,52	5145	5,67	4673	22,41
17	4676	28,53	5475	5,69	4676	28,41
18	4676	27,31	5155	5,60	4673	22,53
19	4676	28,53	5050	5,54	4676	22,31
20	4676	29,10	4901	5,68	4754	22,24
21	4676	28,91	4989	5,57	4673	27,20
22	4676	28,28	5037	5,71	4715	24,46
23	4676	27,90	4965	5,47	4673	23,08
24	4676	28,03	5018	5,56	4768	22,10
25	4676	28,18	5072	5,55	4676	24,34
26	4676	28,69	5177	5,46	4673	21,05
27	4676	28,76	4673	5,62	4673	22,20
28	4676	29,38	5096	5,62	4676	24,43
29	4676	28,66	4925	5,65	4676	24,75
30	4676	28,77	5166	5,60	4676	22,27
31	4676	28,39	5123	5,69	4676	23,23
32	4676	28,18	5155	5,50	4673	20,63
33	4676	28,40	5115	5,72	4660	22,43
34	4676	28,56	5247	5,54	4745	24,81
35	4676	28,68	5082	5,56	4673	22,71
Média	**4690,9**	**28,45**	**5075,5**	**5,56**	**4693,8**	**23,09**

Como pode ser observada, a média obtida pela abordagem proposta foi de 4690,9 com um desvio padrão de 61,34 em um tempo quase seis vezes maior que Morandin, porém obtido na faixa de poucos segundos e aceitável para o contexto de planejamento. O valor mínimo encontrado foi de 4676 e o máximo 4951, que apesar de pior, está abaixo da média dos valores obtidos por Morandin.

A média obtida por Morandin foi de 5075,5 com um desvio padrão de 173,5 devido à característica menos direcionada do método de busca utilizado por apresentar alta variabilidade para o resultado de cada teste realizado. Os valores variam entre o mínimo de 4673 e o máximo de 5475.

Os resultados obtidos por Kato se assemelham ao da proposta. Com uma média de 4693,8 e desvio padrão de 44,64. Apesar de divergirem na modelagem do problema, ambas as abordagens baseadas em *ACO* efetuaram uma busca mais direcionada sobre o espaço de soluções, direcionando a exploração para regiões mais promissoras e com soluções de melhor qualidade.

O número de soluções possíveis para os cenários avaliados pode ser obtido pela expressão 3.3:

$$n! \prod_{i=1}^{n} NR_i \tag{3.3}$$

Onde *n* é o numero de produtos do cenário e NR_i é o número de roteiros do produto P_i.

4.5.3 Avaliação da programação reativa

Para a avaliação do comportamento da abordagem proposta em um contexto de replanejamento, foi instanciado um novo cenário considerando as especificações da Tabela 4.3. Os tempos de processamento são os mesmos da Tabela 4.4 e na Tabela 4.7 são especificadas a divisão dos produtos em famílias e suas respectivas matérias prima. De acordo com as informações da Tabela 4.7 os produtos P_6 e P_8 utilizam a matéria prima MP_D enquanto que os produtos P_1 e P_3 a matéria prima MP_A

Tabela 4.7 – Especificação das matérias prima de cada produto dos cenários.

Matéria prima	Produto
MP_A	P_1, P_3
MP_B	P_2
MP_C	P_4, P_5
MP_D	P_6, P_8
MP_E	P_7, P_9

A descrição do novo cenário gerado é apresentada na Tabela 4.8.

Tabela 4.8 – Cenário de produção para o problema 02.

Produtos		Roteiros de produção
P_1	R_{11}	M_1 M_2 M_4 M_5 M_7 M_9
	R_{12}	M_3 M_4 M_5 M_6 M_8 M_9
	R_{13}	M_2 M_4 M_5 M_6 M_8 M_9
	R_{14}	M_1 M_3 M_5 M_6 M_8 M_9
P_2	R_{21}	M_1 M_2 M_3 M_4 M_5 M_6 M_7
	R_{22}	M_1 M_2 M_3 M_4 M_5 M_8 M_9
	R_{23}	M_1 M_2 M_3 M_4 M_5 M_6 M_8
	R_{24}	M_2 M_3 M_5 M_7 M_8 M_9
P_3	R_{31}	M_4 M_5 M_6 M_7 M_8
	R_{32}	M_2 M_3 M_7 M_8 M_9

P_4	R_{41}	M_1 M_3 M_4 M_5 M_6 M_7
	R_{42}	M_2 M_3 M_4 M_5 M_6 M_7
	R_{43}	M_1 M_5 M_6 M_8 M_9
	R_{44}	M_1 M_5 M_6 M_7 M_8 M_9
	R_{45}	M_2 M_5 M_6 M_8 M_9
P_5	R_{51}	M_4 M_5 M_7 M_8 M_9
	R_{52}	M_1 M_2 M_3 M_5 M_6
	R_{53}	M_1 M_2 M_3 M_7 M_8
P_6	R_{61}	M_3 M_4 M_5 M_6 M_7 M_8 M_9
	R_{62}	M_2 M_4 M_5 M_6 M_7 M_8 M_9
	R_{63}	M_1 M_3 M_6 M_7 M_8 M_9
	R_{64}	M_2 M_4 M_6 M_7 M_8 M_9
P_7	R_{71}	M_1 M_2 M_4 M_5 M_6 M_9
	R_{72}	M_1 M_2 M_3 M_7 M_8 M_9
P_8	R_{81}	M_4 M_5 M_6 M_7 M_8 M_9
	R_{82}	M_3 M_4 M_5 M_7 M_8 M_9
	R_{83}	M_2 M_4 M_5 M_7 M_8
P_9	R_{91}	M_1 M_5 M_6 M_7 M_8 M_9
	R_{92}	M_3 M_5 M_6 M_7 M_8 M_9
	R_{93}	M_2 M_4 M_6 M_7 M_8 M_9

Para este cenário é considerada a ocorrência de eventos inesperados, sendo necessária uma reprogramação da produção que reflita as modificações causadas no ambiente.

Os resultados obtidos pela abordagem proposta foram comparados com os resultados obtidos pela abordagem apresentada em Morandin *et. al.* (2008), para isso, um procedimento de reajuste do cenário foi incorporado a abordagem de maneira a corresponder com as características tratadas.

Para a primeira avaliação é considerada a ocorrência de quebra de máquina. Na etapa de programação todas as máquinas estão disponíveis e para o teste realizado na etapa de reprogramação considerou-se que a máquina M_1 estava quebrada, dentre as alterações, houve a remoção dos roteiros dos produtos que usavam a máquina M_1 e o produto P_7 ficou impossibilitado de ser fabricado, uma vez que todos os roteiros de fabricação do produto P_7 usam a máquina M_1.

Foram considerados quatro tipos de configurações para a abordagem proposta. As configurações consistem na redução do número de iterações do algoritmo como forma de avaliar se os valores de feromônio, obtidos na etapa de planejamento, podem direcionar a busca para soluções de qualidade já no inicio das iterações da etapa de reprogramação. Dessa forma foram definidas três configurações para o número de iterações (NC), sendo essas: para a configuração 01 NC = 50, igualando-se ao número de iterações da primeira etapa, para a

configuração 02 NC foi reduzido para a metade, sendo igual a 25 e para a configuração 03 NC é definido com o valor de 10 iterações.

Na Tabela 4.9 é apresentado o resumo dos 35 testes realizados para o cenário do problema 02 considerando a etapa de reprogramação. Os valores correspondem aos resultados obtidos por cada abordagem nas etapas de programação (Etapa 01) e reprogramação (Etapa 02).

Tabela 4.9 – Resumo dos resultados obtidos para a programação e reprogramação da primeira ocorrência.

	Morandin		**Proposta Conf. 01**		**Proposta Conf. 02**		**Proposta Conf. 03**	
	Etapa 01	Etapa 02	Etapa 01	Etapa 02	Etapa 01	Etapa 02	Etapa 01	Etapa 02
Média *Makespan*	4895,1	4956,3	4769,9	4725,2	4797,6	4835,8	4785	4919
Melhor solução	4511	4526	4526	4526	4611	4526	4526	4526
Pior solução	5325	5447	4995	4970	5028	5015	5068	5068
Desvio Padrão	199,6	212,26	187,22	176,3	116,04	169,99	182,29	153,65

Para uma melhor forma de visualização, os mesmos valores da Tabela 4.9 são apresentados no gráfico da Figura 4.17.

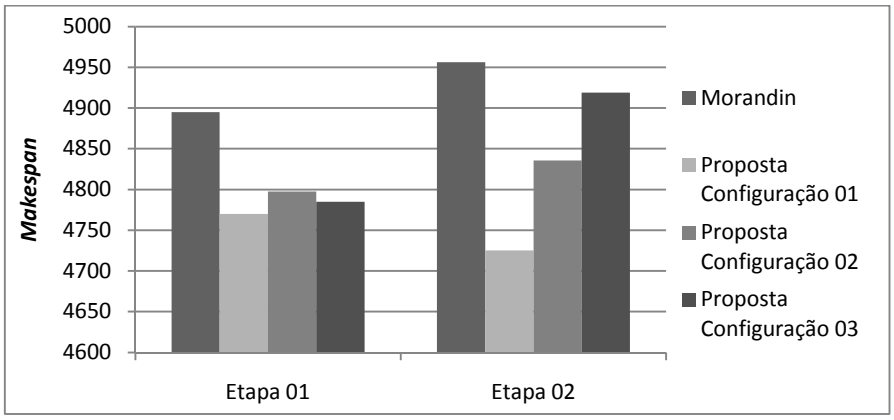

Figura 4.17 – Comparação entre as abordagem para as etapas de programação e reprogramação da produção.

De acordo com os valores da Figura 4.17, na medida em que o número de iterações para a etapa de reprogramação se eleva, os valores de *makespan* tendem a ser minimizados, porém é provável que no decorrer das iterações os valores de feromônio se ajustem ao novo cenário e os valores definidos inicialmente, a partir da Etapa 01, tenham pouca influência.

É possível observar que a abordagem proposta consegue obter um melhor *makespan* para ambas as etapas quando comparado com os resultados de Morandin, mesmo com a redução do cenário, a abordagem baseada em AG tendeu a ter piores resultados. Essa característica pode ser em função da modelagem do

AG, na qual produtos com apenas um roteiro ficam impossibilitados de sofrerem mutação. Esse fato reduz a capacidade de exploração da abordagem, e determinadas regiões do espaço de soluções podem não serem atingidas.

Para poucas iterações, os resultados obtidos estão mais distantes dos valores obtidos com a configuração 01 (50 iterações), e a influência dos valores iniciais de feromônio não se tornou tão clara ao ponto de direcionar a busca para regiões de melhor qualidade. No entanto, outra avaliação foi aplicada de maneira a verificar a influência dos valores iniciais de feromônio atribuídos para a etapa de reprogramação.

A abordagem foi aplicada à etapa de reprogramação (Etapa 02), considerando as configurações de NC = 50, NC = 25 e NC = 10 descritas anteriormente. Para a verificação foi considerada uma aplicação da abordagem sem a inicialização de feromônio da Etapa 01, ou seja, sem conhecimento prévio do cenário, e com inicialização dos valores de feromônio a partir dos valores obtidos na melhor solução da Etapa 01. Os resultados são apresentados na Figura 4.18.

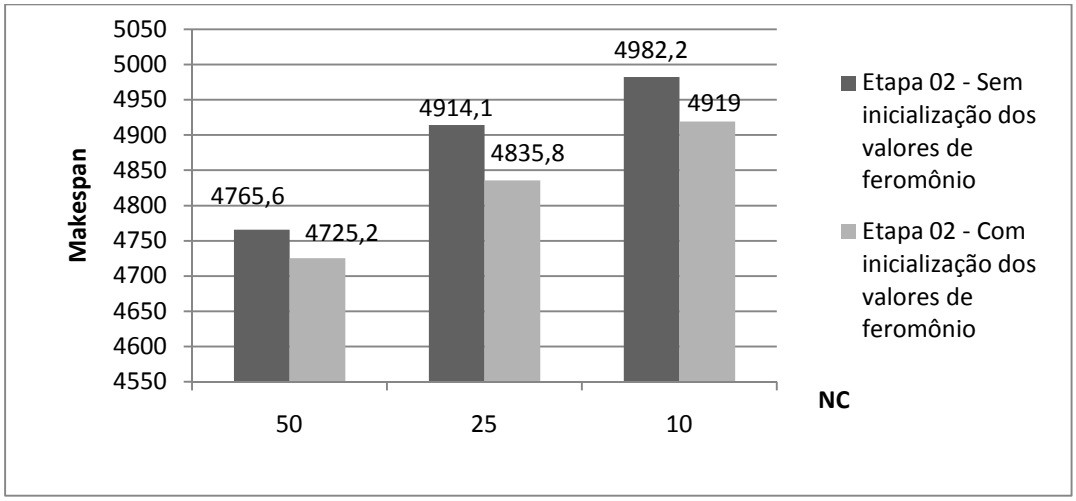

Figura 4.18 – Influência dos valores de feromônio para a etapa de reprogramação.

A partir dos resultados é possível observar que o mecanismo proposto pode indicar informações favoráveis para a qualidade da solução obtida, por considerar um conhecimento prévio para a etapa de busca e pode ser mais bem explorado de maneira a atingir regiões promissoras do espaço de soluções.

Para a segunda avaliação é considerada a ocorrência de falta de matéria prima. No teste foi considerado que a matéria prima MP_C estava em falta e com isso, impediu a produção dos produtos P_4 e P_5.

Seguindo o mesmo princípio da primeira avaliação, foram definidas três configurações para a segunda etapa da abordagem, que consistiu em estabelecer valores distintos para o número de iterações. Os valores definidos foram NC = 50 para a configuração 01, NC = 25 para a configuração 02 e NC = 10 para a configuração 03. Na Tabela 4.10 é apresentado o resumo dos 35 testes realizados para o cenário do problema 02 considerando a etapa de reprogramação. Os valores correspondem aos resultados obtidos por cada abordagem nas etapas de programação (Etapa 01) e reprogramação (Etapa 02).

Tabela 4.10 – Resumo dos resultados obtidos para programação e reprogramação da segunda ocorrência.

	Morandin		**Proposta Conf. 01**		**Proposta Conf. 02**		**Proposta Conf. 03**	
	Etapa 01	Etapa 02	Etapa 01	Etapa 02	Etapa 01	Etapa 02	Etapa 01	Etapa 02
Média *Makespan*	4895,1	4580,2	4772	4478,3	4814,9	4507,8	4795,5	4523,2
Melhor solução	4511	4228	4550	4282	4540	4488	4609	4488
Pior solução	5325	4768	5010	4510	4990	4547	5001	4599
Desvio Padrão	199,6	107,6	141,9	49, 07	129,1	18,6	111,8	33, 5

Para uma melhor forma de visualização, os mesmos valores da Tabela 4.10 são apresentados no gráfico da Figura 4.18.

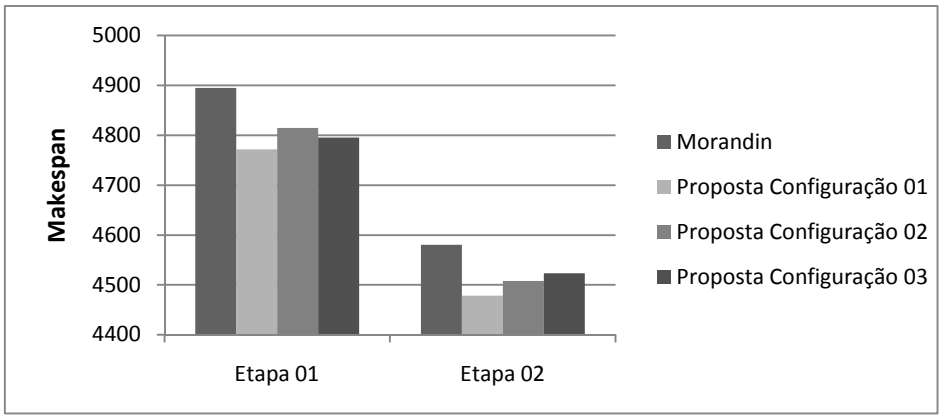

Figura 4.19 – Influência dos valores de feromônio para a etapa de reprogramação.

Baseando-se nos resultados obtidos, é possível observar que ocorre o mesmo princípio discutido na primeira avaliação, a inicialização dos valores de feromônio para a segunda etapa, não chega a ser determinante quando são executadas poucas iterações do algoritmo. Porém, de acordo com os resultados apresentados no gráfico da Figura 4.20, o uso do aprendizado adquirido pelas formigas durante o processo de busca pode ser significativo para a redução do tempo de exploração. Neste exemplo, considerando NC = 25 e ainda a inicialização dos valores de feromônio, os resultados estiveram muito próximos dos resultados

obtidos com NC = 50, sendo estes 4507,8 para o primeiro caso e 4478,3 para o segundo caso. Dessa forma, mesmo reduzindo o valor de NC, que por conseqüência reduz o tempo de execução do algoritmo, os resultados ainda se mantêm com valores bons e próximos. O tempo de execução é apresentado na Figura 4.21.

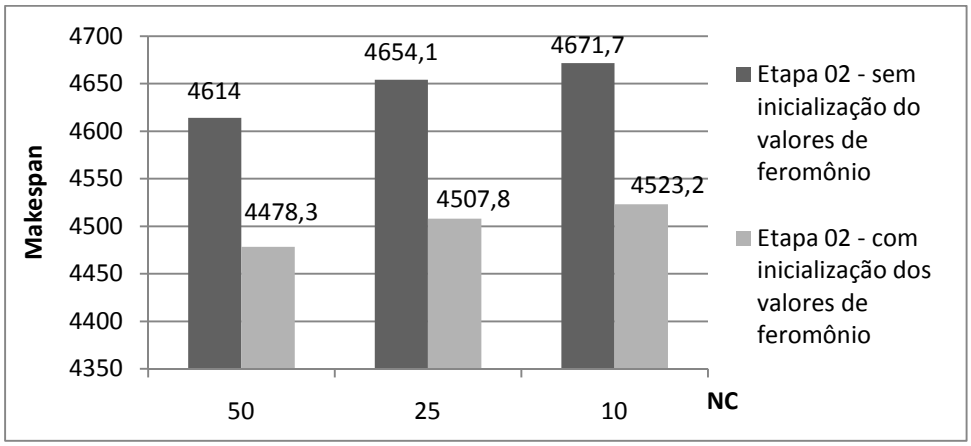

Figura 4.20 – Influência dos valores de feromônio para a etapa de reprogramação.

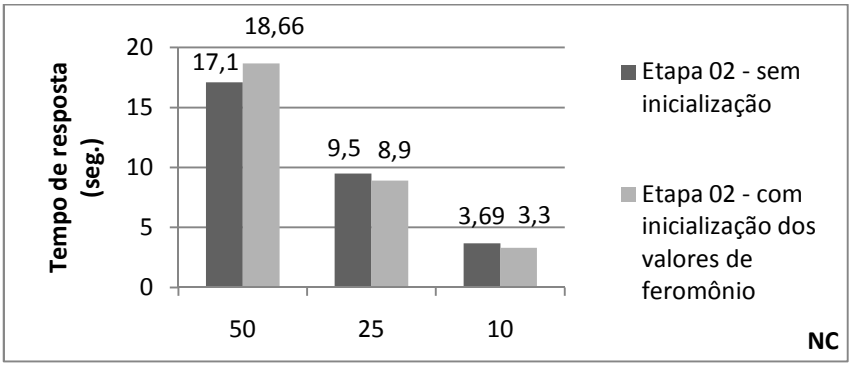

Figura 4.21 – Tempo de execução da abordagem para a etapa de reprogramação.

4.6 Considerações finais

Neste capítulo foram detalhados os procedimentos da abordagem proposta e em seguida definidos os cenários para aplicação e avaliação da abordagem. Os resultados obtidos foram comparados com Morandin *et. al.* (2008) e Kato *et. al.,* (2010), sendo considerado o valor de *makespan* obtido e o tempo de resposta com critérios de avaliação.

No próximo capítulo são apresentadas as principais conclusões sobre a abordagem proposta, assim como sugestões para trabalhos futuros.

Capítulo 5

CONCLUSÕES

O processo de evolução dos sistemas de manufatura é caracterizado por um índice elevado de competitividade, em que ambientes cada vez mais complexos buscam atuar em níveis estratégicos frente às exigências de mercado.

Neste contexto, a automatização e flexibilidade tornam-se fundamentais para o sucesso de tais sistemas, em que a possibilidade de se adequar a mudanças de forma mais rápida, mantendo qualidade e reduzindo custos, são cada vez mais necessárias.

Contudo, uma vez que este grau de automação e flexibilidade se eleva, os sistemas produtivos tornam-se cada vez mais complexos, apresentando problemas de difícil resolução para os profissionais das áreas de planejamento e controle da produção, o que têm motivado diversas pesquisas atuarem nestes dois campos.

Este trabalho tem como foco o problema de programação da produção em sistemas flexíveis, o qual é tratado em duas perspectivas, quanto ao modelo de representação e quanto ao método de busca utilizado.

A modelagem do problema é baseada em nível de operações que compõem o cenário do sistema produtivo, permitindo estabelecer uma correlação entre o modelo e as características da natureza do problema.

Sobre o modelo é aplicada uma abordagem baseada em Otimização por Colônia de Formigas (*Ant Colony Optimization – ACO*) que considerando as restrições e características do problema como roteiros flexíveis de produção, concorrência, compartilhamento de recursos, entre outras, constrói a solução para o problema de programação da produção estabelecendo uma relação colaborativa

entre as operações do cenário. Essa correlação é então usada para um panorama reativo do sistema, em que eventos inesperados podem ocorrer e uma reprogramação da produção deve ser definida de maneira a manter o fluxo de produção.

O objetivo deste trabalho foi o de propor uma abordagem para a programação reativa da produção visando reduzir o valor de *makespan*. O método foi testado e seus resultados avaliados em comparação a outras abordagens propostas (Morandin *et. al.,* 2008 e Kato *et. al*., 2010), tendo como critério a minimização do valor de *makespan* e o tempo para obtenção da resposta.

A partir dos resultados obtidos, pode-se constatar que a abordagem é eficaz para o problema em consideração, por cumprir com os objetivos dentro dos requisitos especificados, ou seja, efetuar a resolução do problema aderindo procedimentos às características do problema, e com isso, conseguir obter um melhor *makespan* em relação às outras abordagens comparadas.

Pode-se observar também, que a relação colaborativa estabelecida entre as operações, característico da metaheurística empregada, pode direcionar a busca para regiões promissoras e o conhecimento adquirido durante o processo de exploração condiz com uma informação relevante sobre o ambiente ao qual esta se aplica.

Para o ambiente de aplicação avaliado, a abordagem proposta foi capaz de reduzir o valor de *makespan* para a etapa de programação da produção*,* comparado a outras abordagens, em um tempo de resposta aceitável e para o tratamento da dinamicidade do problema, constatou-se que o mecanismo proposto, usando a trilha de feromônio como conhecimento agregado, apresentou resultados significativos para a etapa de reprogramação, por possibilitar a redução do tempo de execução do algoritmo.

5.1 Trabalhos futuros

Com base nas indicações da literatura, diversas melhorias podem ser sugeridas para a abordagem proposta. Dentre elas, diversos trabalhos apontam para o uso de busca local para o refinamento das soluções obtidas pelas formigas. No

que se refere às especificidades discutidas, podem-se considerar as diferentes formas de exploração empregada pelas abordagens.

De fato, o processo de construção das soluções empregado em ACO usa uma vizinhança distinta de abordagens baseadas em busca local, com isso, a probabilidade da busca local melhorar as soluções construídas pelas formigas é elevada (DORIGO e STÜTLE, 2004; HERTZ e WIDMER, 2003). Como demonstram os trabalhos de Liouane *et al.,* (2007), Benbouzid *et al.,* (2008) e Zhou *et. al.,* (2008) esta combinação apresenta melhorias significativas.

No que se refere ao modelo de representação para sistemas produtivos, as Redes de Petri têm sido uma alternativa conveniente para modelar as características de sistemas automatizados de manufatura. Dentre outras propriedades, a ferramenta de modelagem inclui um respaldo analítico que permite sua aplicação a diversos tipos de sistemas onde estratégias podem ser aliadas ao uso dos conceitos envolvidos.

Dessa forma, uma modelagem em Redes de Petri para o problema em consideração pode ser conciliada com a abordagem proposta neste trabalho, unindo o formalismo do modelo de representação com as especificidades discutidas para o método de construção das soluções (CIUFUDEAN et al., 2006; ZHIFANG e YO, 2008).

Outra sugestão para trabalhos futuros é a inclusão de outros problemas que estão diretamente ligados ao problema de programação, como exemplo pode-se considerar a programação de manutenção preventiva das máquinas, como apresentado em Ruiz *et. al.,* (2007) *ACO* obteve resultados mais significativos para o problema quando comparado a outras metaheurísticas.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIRRE, L. A. **Enciclopédia de Automática Controle e Automação**. Vol. 1. São Paulo: Editora Blucker, 2007.

BENBOUZID-SITAYEB, F.; AMMI, I.; VARNIER, C.; ZERHOUNI, N. Applying ant colony optimization for the joint production and preventive maintenance scheduling problem in the flowshop sequencing problem. IN: 3RD INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES: From Theory To Applications, Ictta, States, Apr 7-11, 2008.

BLUM, C. Ant colony optimization: Introduction and recent trends, **Physics of Life Reviews**, Vol. 2, December, p. 353-373, 2005.

BLUM, C. e DORIGO, M. Search bias in ant colony optimization: on the role of competition-balanced systems. **IEEE Transactions on Evolutionary Computation**, v. 9, n. 2, p. 159-174, 2005.

BLUM, C. e ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Comput. Surv.** v. 35, n. 3, p. 268-308, Sep. 2003.

BLUM, C. e SAMPELS, M. An ant colony optimization algorithm for shop scheduling problems. **Journal of Mathematical Modelling and Algorithms** v. 3, p. 285-308 2004.

BULLNHEIMER, B.; HARTL R. F.; e STRAUSS, C. A new rank based version of the ant system: A computational study. **Technical report**, Institute of Management Science, University of Vienna, Austria. Later published in **Central European Journal for Operations Research and Economics**, 7(1):25-38, 1997.

BRUCKER, P.; HURINK, J.; JURISCH, B.; WÖSTMANN B. A branch & bound algorithm for the open-shop problem. Discrete Applied Mathematics, v. 76, p. 43–59, 1997.

CHEN, Y.-W.; LU, Y.-Z.; YANG, G.-K. Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling. **Int. J. Adv. Manuf. Technol.**, v. 36, p. 959-958., Springer, London, April, 2008.

CIUFUDEAN, C.; GRAUR, A.; FILOTE, C.; TURCU, C.; POPA, V. Diagnosis of complex systems using ant colony decision Petri nets. Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on, v. 8 p. 20-22 April 2006.

CORREA, H. L. SLACK, N. Framework to analyse flexibility and unplanned change in manufacturing systems, **Computer Integrated Manufacturing Systems**, v. 9, Issue 1, p. 57-64, February 1996.

COLORNI, A.; DORIGO, M.; MAFFIOLI, A.; MANIEZZO, V.; RIGHINI, G.; TRUBIAN, M.; Heuristics from nature for hard combinatorial optimization problems. In: **International Transactions Operational Research**, v. 3, p. 1 - 21, 1996.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. An investigation of some properties of an ant algorithm. In R. Männer & B. Manderick (Eds.), **Proceedings** of PPSN-II, Second INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, p. 509–520, Amsterdam, Elsevier, 1992.

COLORNI A, DORIGO M, MANIEZZO V, TRUBIAN M. Ant system for job-shop scheduling. **Belg J Oper Res, Stat Comput Soc;** v. 34, p. 39–53, 1994.

DOMINGOS, J. C. **Proposta de um procedimento de programação *on-line* da produção de sistemas flexíveis de manufatura baseado em lógica *fuzzy***. Dissertação de Mestrado – Departamento de Computação. Universidade Federal de São Carlos, São Carlos, 2004.

DORIGO, M. Optimization, learning and natural algorithms, **PhD thesis**, Dipartiment di Elettronica, Politecnico di Milano, Italy, 1992.

DORIGO, M. e BLUM, C. 2005. Ant colony optimization theory: a survey. **Theor. Comput. Sci**. v. 344, n. 2-3, p. 243-278, Elsevier Science Publishers Ltd. 2005.

DORIGO, M. e DI CARO, G. “Ant Colony Optimisation: A new metaheuristic”, **Proceedings** of the Congress on Evolutionary Computation, p. 1470 - 1477, IEEE Press, 1999.

DORIGO, M. DI CARO, G. GAMBARDELLA, L. M. Ant algorithms for discrete optimization **Artificial Life** v. 5, p. 137 - 172, MIT Press, 1999.

DORIGO M., Di CARO G. e STÜTZLE T., *Special issue on "Ant Algorithms"*, **Future Generation Computer Systems**, v.. 16, n. 8, 2000.

DORIGO, M., MANIEZZO, V., e COLORNI, A. Positive feedback as a search strategy. **Technical report** 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.

DORIGO, M., MANIEZZO, V., e COLORNI, A. The ant system: optimization by a colony of cooperating agents. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-Part B*,* v. 26, p. 29-41, 1996.

DORIGO, M. e STÜTZLE, T. **Ant Colony Optimization**. The MIT Press, July , 2004.

FERNANDES, M. C. Um Avaliador de cenários simulados para re-seqüenciamento da produção em sistemas automatizados de manufatura usando lógica nebulosa. Dissertação de Mestrado – Departamento de Computação. Universidade Federal de São Carlos, São Carlos, 2004.

FIGLALI, N; ÖZKALE, C; ENGIN, O.; FIGLALI, A. "Investigation of ant system parameter interactions by using design of experiments for job-shop scheduling problems". **Computers & Industrial Engineering**, v. 56, n. 2, p. 538-559, 2009.

GONÇALVES, J. F., MENDES, M. J. J.; RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. **European Journal of Operational Research**, Elsevier, v. 167, n 1, p. 77-95, November, 2003.

GAREY, M. R. JOHNSON, D. S. SETHI, R. "The complexity of the flowshop and jobshop scheduling" **Mathematics Operations Research** v. 1, p. 117-129, May, 1976.

GIFFLER, B. THOMPSON, G. L. Algorithms for solving production scheduling problems. **Operations *Research***, v. 8, p. 487–503, 1960.

GROOVER, M. P. **Fundamentals of Modern Manufacturing**: Materials, Processes, And Systems. 3 ed. Editora John Wiley & Sons, 2006.

HERTZ, A. e WIDMER, M. Guidelines for the use of meta-heuristics in combinatorial optimization, **European Journal of Operational Research**, p. 247–252, 2003.

HUANG, K-L. e LIAO, C-J Ant colony optimization combined with taboo search for the job shop scheduling problem. **Computers & operations research**. v. 35, n. 4, p. 1030-1046, Elsevier Science, Oxford, 2008.

KATO, E. R. R.; MORANDIN, O. Jr.; Fonseca, M. A. S. Ant colony optimization algorithm for reactive production scheduling problem in the job shop system. In: IEEE

INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2009, San Antonio, TX, USA, p. 2199-2204, 2009.

KATO, E. R. R.; MORANDIN, O. Jr.; Fonseca, M. A. S. A max-min ant system modeling approach for production scheduling in a FMS. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2010, Istanbul, Turkey, in press.

KUMAR, R.; TIWARI, M.; K.; SHANKAR, R Scheduling of flexible manufacturing systems: an ant colony optimization approach. **Journal of Engineering Manufacture,** v. 217, n.10, p. 2041-2975. 2003 **Proceedings** of the Institution of Mechanical Engineers, Part B:

LAWRENCE S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Pittsburgh, PA: Graduate School of Industrial Administration, Carnegie Mellon University; 1984.

LIOUANE, N.; SAAD, I.; HAMMADI, S.; BORNE, P. Ant systems & local search pptimization for flexible job shop scheduling production. **International Journal of Computers, Communications & Control**. v. I.2, n. 2. P. 174-184, 2007.

NOWICKI E, SMUTNICKI C. A fast Taboo search algorithm for the job shop problem. Management Science, v. 42, p. 797–813, june, 1996

MAGGIO, E. G. R. **Uma heurística para a programação da produção de sistemas flexíveis de manufatura usando modelagem em redes de Petri**. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de São Carlos, São Carlos, 2005.

MANIEZZO V., GAMBARDELLA L. M, LUIGI F. Ant colony pptimization In: ONWUBOLU, G. C., BABU, B. V. **New Optimization Techniques in Engineering**. Springer, p.101 -121, 2004.

MORANDIN, O. Jr.; KATO, E. R. R.; DERIZ, A. C.; SANCHES, D. S. A search method using genetic algorithm for production reactive scheduling of manufacturing systems in: Industrial Electronics, 2008. ISIE 2008. IEEE INTERNATIONAL SYMPOSIUM, p. 1843-1848, July, 2008.

MOURA, L. PEREIRA, H. G. Aprendendo com a stigmergia, a auto-organização e as redes de cooperação. III CONFERÊNCIA INTERNACIONAL SOBRE TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO NA EDUCAÇÃO. Centro de Competência Nónio Séc. XXI, Universidade do Minho, 2003.

PINEDO M. L. **Sheduling**: Theory, Algorithms, and Systems. Prentice-Hall 3rd ed., 2008.

PRAKASH, A.; TIWARI, M. K.; SHANKAR R. Optimal job sequence determination and operation machine allocation in flexible manufacturing systems: an approach using adaptive hierarchical ant colony **Journal of Intelligent Manufacturing,** v. 19, n. 2, p. 161-173, April, 2008.

RESENDE, S. O. **Sistemas inteligentes: Fundamentos e aplicações**. Ed. Manole Ltda, 1ed., p. 255-246, 2003.

RODRIGUES, S. B. A metaheurística colônia de formigas aplicada a um problema de roteamento de veículos: caso da Itaipu Binacioanal In: XL SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2008, João Pessoa. XL simpósio brasileiro de Pesquisa Operacional, p. 724-733, 2008.

ROSSI, A. e DINI, G. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. **Robot. Comput.-Integr. Manuf**. v. 23, n. 5, p. 503-516, Oct. 2007.

SANTOS, L. P. **Análise da otimização da programação da produção para trás em sistemas mono-estágio por colônia de formigas e sua comparação com *branch and bound***. Dissertação de Mestrado em Engenharia de Produção e Sistemas. Pontifícia Universidade Católica do Paraná, Curitiba, 2008.

SCHULZ, P. G. **Creative Design in Optimization - Metaheuristics Applied to Multi-modal Continuous Functions**, (Master's thesis) Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2006.

SHEWCHUK, J. P. e MOODIE, C. L. Definition and Classification of Manufacturing Flexibility Types and Measures The Int. Journal of Flexible Manufacturing Systems, v. 10, p. 325-349, 1998.

SILVA, C. A., SOUSA, J. M.,; RUNKLER, T. A. Rescheduling and optimization of logistic processes using GA and ACO. **Eng. Appl. Artif. Intell**. v. 21, n. 3, p. 343-352, Apr, 2008.

SLACK, N.; CHAMBERS, S.; HARLAND, C.; HARRISON, A.; JOHNSTON, R. **Administração da Produção**, p 55 – 72, Atlas, 1993.

STÜTZLE, T. e HOOS, H. H. “MAX–MIN Ant System” **Future Generation Computer Systems**, v. 16, n. 9, p. 889–914, 2000

TAILLARD E. Benchmarks for basic scheduling problems. **European Journal of Operations Research**, v. 64, p. 278–85, 1993.

TANG, L. e WANG, X. A predictive reactive scheduling method for color-coating production in steel industry **The International Journal of Advanced Manufacturing Technology.** v. 35, p. 633-645, 2008

TEMPELMEIER, H. e KUHN, H. **Flexible Manufacturing Systems**: Decision Support for Design and Operation. John Wiley & Sons, Inc. 1993.

TOGUYÉNI, A. K. A. BERRUET, P. CRAYE, E. Models and Algorithms for Failure Diagnosis and Recovery in FMSs. **International Journal of Flexible Manufacturing Systems**, v. 15 p. 57 – 85, 2003.

TUBINO, D. F. **Manual de planejamento e controle da produção.** 2 ed. São Paulo: Atlas, 2000.

ZHIFANG, S. e YU, C. Modeling and scheduling for semiconductor wafer fabrication systems. **System Simulation and Scientific Computing**, 2008. ASIA SIMULATION CONFERENCE - 7TH INTERNATIONAL CONFERENCE. v. 10-12 p.1177-1182, Oct. 2008.

ZHOU, R.; LEE; H. P.; NEE A. Y.C. Applying Ant Colony Optimization (ACO) algorithm to dynamic job shop scheduling problems. **Int. J. Manufacturing Research**, v. 3, n. 3, 2008.

ZRIBI, N.; KACEM, I.; El KAMEL, A.; BORNE, P. Assignment and Scheduling in Flexible Job-Shops by Hierarchical Optimization. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, Part C: Applications and Reviews, v. 37, n. 4, p. 652 – 661, July, 2007.

Apêndice A

EXEMPLO DE APLICAÇÃO DE ALGORITMO OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

Para o exemplo de aplicação é considerado o Problema do Caixeiro Viajante - PCV (*Traveling Salesman Problem - TSP*) o qual é amplamente discutido em um capítulo do livro Ant Colony Optimization de DORIGO e STÜTZLE, 2004.

O problema consiste em definir a rota mais curta que visite um determinado conjunto de cidades exatamente uma vez, partindo e retornando ao nó inicial. Formalmente, o PCV pode ser representado por um grafo completo *G* = (*N, A*) sendo *N* o conjunto de vértices que indicam as cidades e *A* o conjunto de arestas unindo os vértices. Um peso d_{ij} é associado a cada arco (*i, j*) ∈ *A,* representando o valor da distância entre a cidade *i* e a cidade *j,* sendo *i* e *j* ∈ *N.*

O objetivo é encontrar um circuito Hamiltoniano de tamanho mínimo sobre o grafo *G*, onde o circuito Hamiltoniano é um caminho com cada um dos vértices visitados somente uma vez. Assim, uma solução ótima para o problema é uma permutação π dos índices {1, 2,..., *N*} tal que o tamanho do percurso obtido em $f(\pi)$ é mínimo, onde $f(\pi)$ é dado por:

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \quad (1)$$

Um algoritmo ACO pode ser aplicado ao PVC considerando o modelo de representação em grafo. A trilha de feromônio é associada com os arcos do grafo e dessa forma, τ_{ij} se refere ao valor de feromônio entre a cidade *j* diretamente após a cidade *i.* A informação heurística é escolhida como $\eta_{ij} = 1/d_{ij}$ sendo que o valor é inversamente proporcional a distância direta entre a cidade *j* e a cidade *i.*

Percursos são construídos a partir da aplicação do seguinte procedimento construtivo simples: (1) escolher, de acordo com algum critério, a cidade inicial a qual uma formiga é posicionada; (2) usar os valores heurísticos e de feromônio para a construção probabilística do percurso, a partir de um processo iterativo o qual adiciona as cidades ainda não visitadas, até que todas as cidades tenham sido visitadas; e (3) voltar para a cidade inicial. Após todas as formigas terem completado a construção de seus percursos, elas podem atualizar os valores da trilha de feromônio.

Para exemplificar a aplicação do algoritmo ACO é considerado o seguinte cenário para o PVC: 5 cidades (*N* = 5); os valores da distância entre as cidades estão representados no grafo da Figura 1. Nesse exemplo os valores das arestas são simétricos, dessa forma $d_{ij} = d_{ji}$.

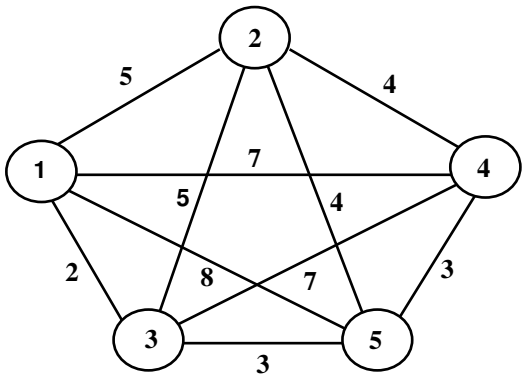

Figura 1. – Modelo de representação do problema e valores da distância entre as cidades.

Os valores da distância entre as cidades são representados em uma matriz de dimensão NxN, com é indicado na Tabela 1. Seguindo a mesma estrutura, os valores da trilha de feromônio também são representados em uma matriz NxN. Para a descrição dos demais procedimentos, são considerados os seguintes parâmetros para o algoritmo ACO: *m* = 3 (número de formigas), α = 1(influencia dos valores de feromônio), β = 2 (influência dos valores heurísticos), ρ = 0.5 (taxa de evaporação).

Tabela 1. Valores da distância entre as cidades.

	1	**2**	**3**	**4**	**5**
1	0	5	2	7	8
2	5	0	5	4	4
3	2	5	0	7	3
4	7	4	7	0	3
5	8	4	3	3	0

Cada formiga $k \epsilon m$ mantém uma memória contendo as cidades visitadas na ordem em que foram visitadas. Essa memória é utilizada para definir a vizinhança factível na etapa de construção das soluções, ou seja, define as cidades ainda não visitadas. A memória é ainda utilizada para o cálculo do tamanho do percurso, sendo esse o resultado da soma das distâncias entre cada cidade visitada.

Considerando que as formigas *k=1* foi inicializada na cidade *3*, *k=2* foi inicializada na cidade 1 e *k=3* foi inicializada na cidade 5, temos os seguintes valores armazenados na estrutura de cada formiga (Tabela 2):

Tabela 2. Valores da distância entre as cidades.

Formiga	Estado Atual	Nós não visitados	Nós visitados	Somatório da distância
k=1	3	1; 2; 4; 5	3	0
k=2	1	2; 3; 4; 5	1	0
k=3	5	1; 2; 3; 4	5	0

Para a escolha do próximo estado, cada formiga efetua uma escolha probabilística considerando os valores da trilha e da informação heurística. O cálculo de é realizado considerando a expressão (2).

$$p_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha} . \left[\eta_{ij}\right]^{\beta}}{\sum_{l \epsilon N_i^k} \left[\tau_{ij}\right]^{\alpha} . \left[\eta_{ij}\right]^{\beta}} \quad (2)$$

Onde, $\left[\tau_{ij}\right]^{\alpha}$ é o valor do feromônio entre a cidade *i* e *j,* α é a influência do valor de feromônio. $\left[\eta_{ij}\right]^{\beta}$é o valor heurístico dado por $1/d_{ij}$, onde d_{ij} é o valor da distância entre as cidades (Tabela 1), β é a influência do valor heurístico. N_i^k é a vizinhança factível da *k*-ésima formiga, ou seja as cidades ainda não visitadas. Para a inicialização dos valores da trilha é considerado um valor constante igual a 0.5 para todas as arestas. Assim, o resultado do cálculo da expressão (1) considerando o cenário da Figura 1 é apresentado na Tabela 3.

Tabela 3. Probabilidade de transição da cidade *i* para a cidade *j*.

	1	**2**	**3**	**4**	**5**
1	0	0,020	0,125	0,010	0,007
2	0,020	0	0,020	0,031	0,031
3	0,125	0,020	0	0,010	0,054
4	0,010	0,031	0,010	0	0,054
5	0,007	0,031	0,054	0,054	0

Dessa forma, os estados da vizinhança de cada formiga possuem as seguintes probabilidades de escolha (Tabela 4):

Tabela 4. Probabilidade de transição da cidade *i* para a cidade *j* para cada formiga.

Formiga	Estado Atual	Nós não visitados	Probabilidade de cada estado vizinho
k=1	3	1; 2; 4; 5	0.125; 0.020; 0.010; 0.054
k=2	1	2; 3; 4; 5	0,020; 0,125; 0,010; 0,007
k=3	5	1; 2; 3; 4	0,007; 0,031; 0,054; 0,054

Para a seleção é usado um método análogo ao da roleta, em que cada probabilidade possui uma fatia na roleta com tamanho proporcional ao seu valor. Por exemplo, para a vizinha da formiga k=1 na cidade 3 a divisão da roleta é apresentado na Figura .

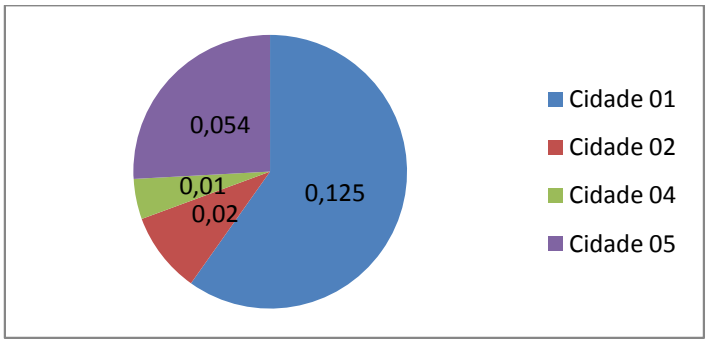

Figura 2. – Divisão da roleta.

Para a segunda etapa de construção as informações de cada formiga são atualizadas como é indicado na Tabela 5.

Tabela 5. Valores da distância entre as cidades.

Formiga	Estado Atual	Nós não visitados	Nós visitados	Somatório da distância
k=1	2	1; 4; 5	3; 2	5
k=2	3	2; 4; 5	1; 3	2
k=3	4	1; 2; 3	5; 4	3

A etapa de construção segue sucessivamente até que todas as formigas tenham completado seus percursos. As etapas são indicadas nas Tabelas 6, 7 e 8.

Tabela 6. Valores da distância entre as cidades.

Formiga	Estado Atual	Nós não visitados	Nós visitados	Somatório da distância
k=1	1	4; 5	3; 2; 1	10
k=2	5	2; 4	1; 3; 5	5
k=3	2	1; 3	5; 4; 2	7

Tabela 7. Valores da distância entre as cidades.

Formiga	Estado Atual	Nós não visitados	Nós visitados	Somatório da distância
k=1	4	5	3; 2; 1; 4	17
k=2	4	2	1; 3; 5; 4	8
k=3	3	1	5; 4; 2; 3	12

Tabela 8. Valores da distância entre as cidades.

Formiga	Estado Atual	Nós não visitados	Nós visitados	Somatório da distância
k=1	5	-	3; 2; 1; 4; 5	20
k=2	2	-	1; 3; 5; 4; 2	12
k=3	1	-	5; 4; 2; 3; 1	14

E para a etapa final o valor do tamanho do percurso considera ainda a distância entre a última e a primeira cidade. O resultado final da distância percorrida dado por cada formiga é apresentado na Tabela 9.

Tabela 9. Tamanho do percurso.

Formiga	Somatório da distância
k=1	23
k=2	17
k=3	22

A atualização dos valores da trilha de feromônio é realizada após todas as formigas terem construído seus percursos. Primeiro é realizada a evaporação seguindo a expressão (3):

$$\tau_{ij} \leftarrow (1-\rho)\,\tau_{ij}, \quad \forall (i,j) \epsilon\, A, \tag{3}$$

Assim, os valores da trilha que para a primeira iteração consistiam do valor 0.5 será atualizado pelo produto de (1 – 0.5), sendo 0.5 o parâmetro da taxa de evaporação. O resultado do cálculo da expressão 2 é apresentado na Tabela 10.

Tabela 10. Valores da trilha de feromônio.

	1	**2**	**3**	**4**	**5**
1	0	0.25	0.25	0.25	0.25
2	0.25	0	0.25	0.25	0.25
3	0.25	0.25	0	0.25	0.25
4	0.25	0.25	0.25	0	0.25
5	0.25	0.25	0.25	0.25	0

Após a evaporação, todas as *m* formigas depositam um valor de feromônio a trilha. Esse valor é baseado no custo do caminho percorrido por cada formiga.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall(i,j)\epsilon\, A, \tag{4}$$

onde $\Delta\tau_{ij}^{k}$ é o valor de feromônio que a formiga *k* deposita sobre os arcos visitados. É definido por:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, & \text{se o arco (i, j)}\epsilon\ \mathrm{P^k}\ , \\ 0, & \text{caso contrário} \end{cases} \tag{5}$$

onde P^k é o percurso percorrido pela formiga *k*. C^k é o custo do percurso P^k construído pela formiga *k*.

Assim, considerando o valor do tamanho do percurso igual a 23, dado pela formiga k=1, o resultado da aplicação da expressão (4) é apresentado na Tabela 11.

Tabela 11. Atualização da trilha de feromônio pela formiga k=1.

	1	**2**	**3**	**4**	**5**
1	0	0.293	0.25	0.293	0.25
2	0.293	0	0.293	0.25	0.25
3	0.25	0.293	0	0.25	0.293
4	0.293	0.25	0.25	0	0.293
5	0.25	0.25	0.293	0.293	0

O mesmo ocorre para as formigas k=2 e k=3. E o resultado final da atualização da trilha para a primeira iteração do algoritmo é apresentado na Tabela 12. É importante destacar a diferença entre as iterações do algoritmo e as etapas de construção, sendo que a cada iteração, todas as etapas de construção são realizadas e o procedimento de atualização são realizados.

Tabela 12. Atualização da trilha de feromônio pelas formigas k=1, k=2 e k=3.

	1	**2**	**3**	**4**	**5**
1	0	0.351	0.353	0.293	0.295
2	0.351	0	0,338	0.353	0.250
3	0.353	0,338	0	0.308	0.351
4	0.293	0.353	0.308	0	0.396
5	0.295	0.250	0.351	0.396	0

As informações da melhor formiga são salvas sempre que um menor valor do tamanho do percurso é encontrado.

Após a atualização da trilha, todas as *m* formigas efetuam novamente a construção das soluções desde a etapa inicial, porém considerando os novos valores da trilha de feromônio. Esse processo segue sucessivamente até que o número de iterações especificado como parâmetro seja alcançado. O resultado final apresentado pelo algoritmo consiste do menor valor obtido para o tamanho do percurso no decorrer das iterações.