PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS

LEANDRO PEREIRA DOS SANTOS

ANÁLISE DA OTIMIZAÇÃO DA PROGRAMAÇÃO DE PRODUÇÃO PARA TRÁS EM SISTEMAS MONO-ESTÁGIO POR COLÔNIA DE FORMIGAS E SUA COMPARAÇÃO COM *BRANCH AND BOUND*.

CURITIBA
2008

LEANDRO PEREIRA DOS SANTOS

ANÁLISE DA OTIMIZAÇÃO DA PROGRAMAÇÃO DE PRODUÇÃO PARA TRÁS EM SISTEMAS MONO-ESTÁGIO POR COLÔNIA DE FORMIGAS E SUA COMPARAÇÃO COM *BRANCH AND BOUND*.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Engenharia de Produção e Sistemas.

Orientador: Prof. Dr. Guilherme Ernani Vieira

Curitiba
2008

Para minha esposa Renilda, que sempre esteve
presente nos momentos mais difíceis.
E para minha filha Heloísa, que além de compreender
os momentos de ausência de seu pai,
foi sua maior incentivadora.

AGRADECIMENTOS

A Deus, pois sem Ele nada neste mundo seria possível.

Aos meus pais, que sempre me conscientizaram sobre a importância da educação.

A todos os outros meus familiares, que sempre me incentivaram.

Ao meu orientador Professor Dr. Guilherme Ernani Vieira, pelo apoio e orientações.

Aos professores do PPGEPS, pelos ensinamentos transmitidos ao longo do projeto.

A PUC-PR, que ofereceu toda a estrutura necessária para o desenvolvimento do trabalho.

Aos meus amigos, pela paciência e descontração.

Ao Brasil, país onde nasci e que gostaria de alguma forma ajudar com os ensinamentos e experiências que adquiri.

RESUMO

O potencial da manufatura como fator preponderante no desempenho competitivo de uma empresa é cada vez mais nítido no mundo corporativo. Inserido nesse contexto está à programação de produção que, abordando as decisões de curto prazo, tem como principal objetivo atribuir a cada recurso (máquina ou linha de produção) um determinado número de tarefas e seqüenciá-las para que a produção seja a mais eficiente e econômica ("otimizada") possível. Atingir este objetivo configura-se numa tarefa árdua e complexa, pois as regiões de busca, compostas por várias possíveis soluções para o problema, são extremamente grandes, o que pode inviabilizar a utilização de técnicas exatas. Por isso, em muitos casos, renuncia-se a uma solução ótima propriamente dita em troca de soluções boas, mas que sejam viáveis no que se refere ao tempo (custo) computacional. Em várias situações industriais, a aplicação das chamadas técnicas heurísticas e metaheurísticas podem ser apropriadas. Nesse sentido, este trabalho tem como objetivo implementar e analisar a metaheurística conhecida como *Ant Colony Optimization* (ACO) na solução de problemas de otimização da programação de produção para trás (*backward scheduling*), em sistemas produtivos compostos de apenas um estágio de processamento, recursos paralelos e roteiros flexíveis. Para analisar a aplicabilidade dessa técnica, é feita uma comparação com a técnica de otimização *branch and bound.* Os indicadores utilizados para medir a qualidade das respostas são obtidos através do comprimento do programa (*makespan*) e em termos do tempo (custo) de processamento. Os resultados mostram que o *ACO* é uma técnica eficiente na resolução do problema de programação de produção proposto.

Palavras-chave: Otimização. Técnicas heurísticas. Programação de produção. Colônia de formigas.

ABSTRACT

The potential of manufacturing as a predominant factor in the competitive performance of a company is increasingly clearer in the corporate world. Production scheduling is part of this context. It addresses the short-term decisions, which main objectives are to allocate a number of tasks to resources (machines or production lines) and sequence them as efficient and economic as possible ("optimized"). Achieving this goal is a complex and arduous task, due to the large search space, made of innumerous possible solutions, which prevents the use of more accurate techniques. Therefore, in many cases, one renounces the search for an optimal solution in exchange for good answers that are reachable in acceptable computer time. In many industrial situations, the application of techniques called heuristics and metaheuristics may be more appropriate. This work implements and analyzes a metaheuristic known as Ant Colony Optimization (ACO) applied to solve production scheduling problems in manufacturing scenarios of only one processing phase (single operation), parallel resources, flexible routings and backward scheduling. To study the applicability of this technique, a comparison with branch-and-bound optimization is performed. The indicators used to measure response quality is given in terms of maximum completion times and computer processing time (effort). The results show that ACO is an efficient technique to solve the proposed production scheduling problem.

Key-words: Optimization. Heuristics techniques. Production scheduling. Ant colony.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC = Ant *Colony* (Colônia de Formigas)

ACO = *Ant Colony Optimization* (Otimização por Colônia de Formigas)

ACS = *Ant Colony System* (Sistema de Colônia de Formigas)

ANOVA = *Analysis of variance* (Análise de Variância)

BB = *Branch-and-bound*

BT = Busca *Tabu (***TS** = *Tabu Search*)

DOE = *Design of experiments* (Projeto de experimentos)

EV = Evaporação

GA = *Genetic Algorithm* (Algoritmo Genético)

GL = Graus de liberdade

JIT = *Just in time*

MRP = *Material Requirements Planning* (Planejamento das Necessidades de Materiais)

NF = Número de formigas

NP-*hard* = Não polinomial árduo

NVF = Número de viagens das formigas

OP = Ordem de produção

QAP = Quantidade adicionada de feromônio

QPI = Quantidade de feromônio inicial

AS = *Simulated Annealing* (Tempera Simulada)

SS = Soma de quadrados

SQ = Soma quadrática das respostas

TSP = *Travelling Salesman Problem* (Problema do Caixeiro Viajante)

VMR = Valorização da melhor resposta

SUMÁRIO

1 INTRODUÇÃO

A conjuntura econômica mundial globalizada exige que a competitividade das organizações esteja em crescente evolução. Cada vez mais as empresas precisam adaptar continuamente seus sistemas produtivos e de gestão, buscando sempre obter aumentos significativos em relação à sua produtividade. Neste sentido, Moura Jr. (1996) enfatiza o potencial da manufatura em influir decisivamente no desempenho competitivo de uma empresa e cita como exemplo o sucesso alcançado pelas empresas japonesas, obtido através da alta qualidade e dos baixos preços de seus produtos, na conquista de mercados antes dominados por empresas ocidentais. Esses resultados, segundo o autor, são conseqüência, acima de tudo, do alto desempenho em seus sistemas de manufatura.

Inserida no escopo da manufatura, encontra-se a programação da produção, que dentro do planejamento hierárquico da produção aborda as decisões de curto prazo e que, de acordo com Corrêa & Corrêa (2004), “consiste em alocar no tempo as atividades, obedecendo ao seqüenciamento definido e ao conjunto de restrições considerado”. Devido a sua grande importância, esta matéria tem sido objeto de investigação de vários estudos e também será o deste trabalho.

Partindo-se do princípio de que a programação da produção tem como principal objetivo atribuir a cada recurso (máquina ou linha de fabricação) determinadas tarefas e seqüenciá-las de forma que indicadores de gestão sejam otimizados, incluindo na maioria dos casos a produção, pode-se incontestavelmente afirmar que a busca constante por melhorias neste processo traz ganhos significativos às empresas. Assim, programar a produção de forma adequada nada mais é do que extrair o melhor resultado possível dos recursos disponíveis, reduzindo com isso os custos de produção e ao mesmo tempo atendendo os clientes de forma rápida e eficiente

Porém, a solução desta difícil equação, principalmente em casos reais, nos quais a disponibilidade de recursos é limitada e as restrições são imensas, se traduz muitas vezes em uma tarefa árdua, e técnicas adequadas são necessárias. Garey & Johnson (1979) classificam os problemas de programação de produção como NP-*hard*, o que,

pela teoria da complexidade, dificulta a obtenção de uma solução ótima através de algoritmos exatos. A dificuldade na utilização de técnicas exatas para a solução de tais problemas abre espaço para o uso dos chamados métodos aproximados. Esses métodos, conhecidos como heurísticas e metaheurísticas, procuram encontrar soluções aceitáveis, que eventualmente podem ser ótimas, com um tempo de processamento dos dados drasticamente menor.

A literatura sobre as várias metaheurísticas existentes é vasta e demonstra, de forma bastante convincente, que os resultados que estas podem trazer atendem as necessidades dos problemas propostos. Neste trabalho, a metaheurística, conhecida como *Ant Colony Optimization* (Otimização por Colônia de Formigas - ACO), será estudada mais profundamente, sobretudo no que diz respeito a sua utilização na resolução de problemas de programação da produção.

Para que se possa realizar uma avaliação de um determinado programa de produção, critérios para a análise da qualidade deste programa devem ser definidos. *Makespan* (tempo total de produção), tempo de fluxo da ordem de fabricação, pontualidade, porcentagem de utilização dos recursos, entre outros são alguns dos indicadores mais utilizados na literatura (REIS, 1996). Neste trabalho específico, a qualidade dos programas de produção gerados será analisada através dos *makespans* obtidos. Além disso, o tempo de processamento para a geração dos programas de produção será consisderado para a verificação da viabilidade da técnica proposta.

Desse modo, o presente trabalho busca responder as seguintes perguntas: **1) A metaheurística *ACO* é uma técnica viável para a solução de problemas de otimização da programação de produção para trás em sistemas mono-estágio? 2) Quão eficiente é esta técnica se comparada com a técnica *branch and bound*?**

1.1 Objetivos do trabalho

Este estudo pretende, de forma geral, verificar se a metaheurística *ACO* é uma técnica viável para a resolução de problemas de otimização da programação de produção para trás em sistemas produtivos compostos de apenas um estágio de processamento, recursos paralelos e com velocidade de produção diferente, além de roteiros flexíveis para os produtos. Para isso, sua eficiência será comparada com a de outra técnica baseada em *branch-and-bound*, através de uma comparação com respeito à qualidade da resposta e ao tempo computacional das duas técnicas.

Como objetivos específicos deste trabalho, apresentam-se os seguintes pontos:

- Obter maior conhecimento sobre problemas de otimização;
- Compreender e analisar as várias faces de um problema de programação de produção, tanto no que diz respeito a sua complexidade bem como aos métodos utilizados para sua solução;
- Obter conhecimento sobre as características gerais de técnicas heurísticas e metaheurísticas em relação a sua aplicabilidade e às formas de funcionamento;
- Entender em detalhes a estratégia de funcionamento da técnica *ACO*, inclusive suas limitações;
- Compreender os aspectos envolvidos na elaboração de um protótipo (*software*) que funciona fundamentado em uma inteligência baseada na técnica *ACO*.

1.2 Metodologia utilizada

Considerando-se o propósito deste trabalho, a caracterização quanto a sua metodologia pode ser definida como uma pesquisa aplicada, quantitativa e descritiva.

Pode ser classificado como uma pesquisa aplicada, pois se destina a gerar conhecimento para uma aplicação prática, buscando a solução de um problema de programação de produção.

É uma pesquisa quantitativa porque busca, através de indicadores numéricos, mensurar a qualidade de um programa de produção. Objetiva ainda avaliar a metaheurística *ACO* como técnica de solução de problemas de otimização da programação de um caso específico, utilizando para isso métodos quantitativos de medição e comparação.

Finalizando a classificação da pesquisa, pode-se, do ponto de vista dos procedimentos adotados, classificá-la como pesquisa descritiva, pois visa descobrir a existência de associações entre variáveis e ainda objetiva conhecer e interpretar a realidade sem nela interferir para modificá-la.

Para todo trabalho de pesquisa que envolva a realização de experimentos, sendo um experimento definido como

> (...) um teste ou uma série de testes nos quais são feitas mudanças propositais nas variáveis da entrada de um processo ou sistema de forma que possam ser observadas e identificadas as razões para mudanças na resposta de saída. (Calegare, 2001).

Assim, deve-se definir a melhor forma de condução ou a melhor forma de realização dos experimentos, garantindo que estes possam ser reproduzidos sob condições controladas, o que promove aos resultados obtidos a confiabilidade necessária para os projetos de pesquisa. Dessa forma, é necessário um planejamento ou delineamento dos experimentos.

O planejamento dos experimentos deste trabalho foi realizado, portanto, com dois objetivos distintos:

- Determinar quais variáveis controláveis (parâmetros de entrada) que mais influem na qualidade da solução. Essa análise é feita em uma etapa preliminar, que oferece a oportunidade de determinar o desempenho relativo dos diversos parâmetros do algoritmo, face ao problema considerado;
- Verificar se o *ACO* é eficiente para a resolução do problema proposto. Nesta fase de experimentos finais, o comportamento e o desempenho do modelo são avaliados por comparação direta, utilizando-se alguns cenários diferentes, comparando-se seus resultados com os de outra técnica bastante utilizada na literatura para a resolução de problemas otimização na área de pesquisa operacional, e que por ser uma técnica exata, tem como principal atributo encontrar a solução ótima para o problema. Esta é a técnica *branch and bound*.

Para que esses objetivos sejam atingidos, Montgomery (1991) indica um procedimento composto de sete etapas para o planejamento experimental e a análise dos resultados e que serão adotados neste projeto.

1. Reconhecimento e definição do problema;
2. Escolha das variáveis e das faixas de valores que essas variáveis serão avaliadas;
3. Escolha adequada da variável de resposta, de modo a garantir a objetividade da análise dos resultados;
4. Delineamento dos experimentos;
5. Execução dos experimentos;
6. Análise dos resultados;
7. Elaboração de conclusões e recomendações.

A etapa 1 é realizada no capítulo 2, onde o problema é definido e revisado com base na literatura sobre o assunto. As etapas 2 e 3 estão presentes no capítulo 3, em que são escolhidas as variáveis de influência no sistema bem como a variável de

resposta. Os valores das variáveis de influência são definidos no capítulo 4, que contém também as etapas 4, 5 e 6 do procedimento. A etapa 7 está contemplada no capítulo 5.

O delineamento dos experimentos para a análise quanto a influências das variáveis de configuração do sistema na qualidade das respostas para o problema proposto é feito através do experimento 2^k, que é a execução do algoritmo com todas as combinações possíveis entre essas variáveis. Com os resultados desta primeira parte dos experimentos, através da análise de variância, as primeiras conclusões podem ser tiradas.

O experimento 2^k, além de ser necessário para verificação sobre quais variáveis de configuração têm influência na variável de resposta, também mostra quais os níveis ideais para as variáveis de configuração no intuito de otimizar o valor da variável de resposta. Com essa configuração indicada pelo experimento 2^k, são feitas simulações em diferentes cenários de programação, e testes estatísticos são realizados para comparação da eficiência da metaheurística *ACO* com a técnica *branch and bound*, buscando verificar a eficiência da primeira.

1.3 Organização do trabalho

O capítulo 2 apresenta uma revisão sobre os assuntos centrais discutidos na pesquisa, a partir de trabalhos que têm usado técnicas heurísticas e metaheurísticas para a resolução de problemas complexos da mesma família do estudado neste trabalho. Demonstra ainda o método de funcionamento da metaheurística *ACO*, explicando detalhadamente sua lógica de atuação. O capítulo 3 demonstra de que forma a técnica foi aplicada na resolução do problema de programação de produção bem como a estruturação do protótipo utilizado no trabalho. No capítulo 4 é feita uma demonstração de como foram realizados os experimentos, além da análise dos resultados; tendo sempre como objetivo verificar a influência das variáveis de configuração na qualidade da solução gerada e a viabilidade do algoritmo implementado. No capítulo 5 apresenta-se uma síntese do trabalho, contemplando os objetivos e resultados obtidos com a pesquisa.

2 REVISÃO DA LITERATURA

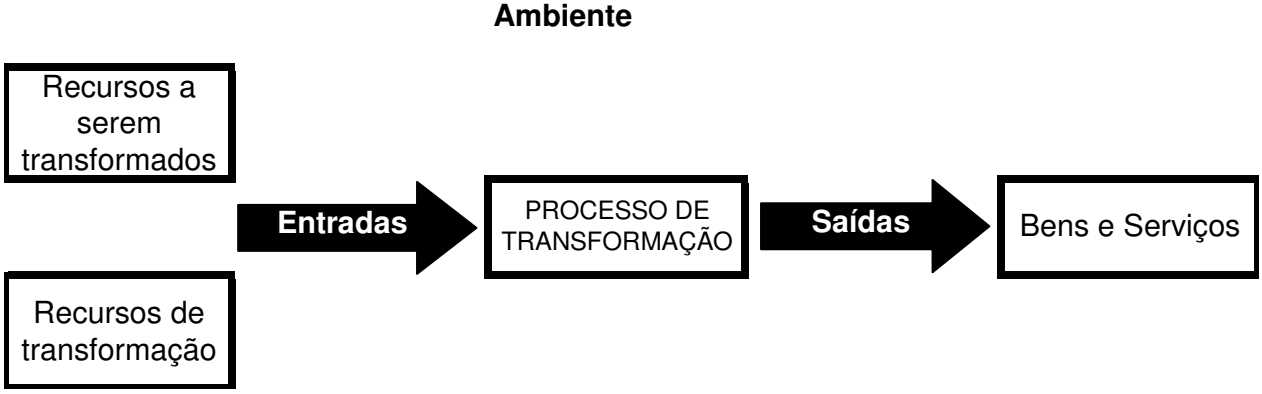
A aplicação de uma determinada técnica para a resolução de problemas complexos passa primeiramente pelo entendimento da problemática abordada pela pesquisa e também pela evolução no tempo no que diz respeito à utilização dessa técnica. Dessa forma, este capítulo tem como objetivo fazer uma revisão sobre conceitos envolvidos na questão da programação da produção, bem como de conhecimentos que precedem este assunto. Faz-se uma revisão de temas, como os sistemas de produção e suas classificações, a programação de produção **propriamente dita**, heurísticas e metaheurísticas aplicadas à resolução de problemas de otimização, e uma análise mais profunda e abrangente sobre a metaheurística *ACO*.

2.1 Sistemas de produção

De forma básica, o significado da palavra “sistema” pode ser descrita como uma inter-relação de partes, elementos ou unidades que fazem funcionar uma estrutura organizada. Nessa mesma linha, pode-se definir “produção” como “ato, ou efeito de produzir, criar, gerar, elaborar, realizar”. Aquilo que é produzido ou fabricado pelo homem e, especialmente, por seu trabalho associado ao capital e à técnica.

Para uma definição mais técnica do que é um sistema de produção, pode-se citar Riggs (1970) que, citado por Russomano (1998), afirma ser “(...) um processo planejado pelo qual elementos são transformados em produtos úteis, ou seja, um procedimento organizado para se conseguir a conversão de insumos em produtos acabados”.

Na Figura 1 é mostrado esquematicamente, de forma genérica e simplificada, o que é um sistema de produção.



Fonte: Slack *et al.* (1997)

Figura 1: Modelo de sistema de produção

Verifica-se então que um sistema de produção é uma estrutura que, de forma organizada, utiliza os recursos de produção para transformar outros recursos, por exemplo, matérias-primas em produtos finais.

2.2 Classificação dos sistemas de produção

A classificação dos sistemas de produção é importante ao permitir discriminar grupos de técnicas de programação de produção para cada tipo particular de sistema, o que racionaliza a escolha e a tomada de decisão sobre qual delas adotar em determinada circunstância.

> As diferentes formas de classificação dos sistemas produtivos ajudam a entender o nível de complexidade necessário para a execução do planejamento e controle das atividades produtivas. O grau de padronização dos produtos, os tipos de operações necessárias e a natureza dos produtos são fatores determinantes para a definição das atividades do PCP (TUBINO, 1997).

A literatura disponível revela diversas formas de classificação dos sistemas de produção. Moreira (1998) descreve brevemente os elementos e interações de um sistema de produção e apresenta duas classificações para esses sistemas:

- Classificação tradicional: agrupa os sistemas de produção em função do fluxo do produto:
  a) Sistema de produção contínua, ou de fluxo em linha, cujo processo é focalizado no produto, apresentando uma seqüência linear em sua produção. Os produtos seguem uma seqüência prevista com uma alta eficiência e acentuada inflexibilidade. Apresenta-se de duas formas: produção em massa (linha de montagem) e processamento contínuo (indústrias de processo);
  b) Sistema de produção intermitente, cujo foco está no processo, apresentando uma produção feita em lotes ou por encomenda. O produto flui de forma irregular de um centro de trabalho a outro, e o equipamento é do tipo genérico, adaptando-se às características das diversas operações que estejam sendo feitas no produto. Esse sistema apresenta a flexibilidade como vantagem fundamental, mesmo perdendo em volume de produção em relação ao sistema contínuo;
  c) Sistema de produção de grandes projetos sem repetição: é aquele no qual não existe um fluxo do produto ou de produção e constitui-se em geral num único produto, seguindo uma seqüência de tarefas ao longo do tempo e, normalmente, de longa duração.

- Classificação cruzada de Schroeder**:** considera duas dimensões. De um lado, a dimensão tipo de fluxo do produto de maneira semelhante à classificação tradicional. De outro, a dimensão tipo de atendimento ao consumidor, na qual existem duas classes:
  a) Sistemas orientados para estoque: o produto é fabricado e estocado antes da demanda efetiva do consumidor;
  b) Sistemas orientados para a encomenda: as operações são relacionadas a um cliente em particular.

De forma um pouco mais ampla, Tubino (1997) identifica três critérios para classificar os sistemas de produção:

- Pelo grau de padronização;

  a) sistemas que produzem itens padronizados: bens ou serviços que apresentam alto grau de uniformidade e são produzidos em grande escala.

  b) sistemas que produzem itens sob medida: bens ou serviços desenvolvidos para um cliente específico.

- Pelo tipo de operação;

  a) processos contínuos: envolvem a produção de bens ou serviços que não podem ser identificados individualmente.

  b) processos discretos: envolvem a produção de bens ou serviços que podem ser isolados. Estes podem ser subdivididos em processos repetitivos em massa, que é a produção em grande escala de produtos padronizados; em processos repetitivos em lote, que é a produção em lotes de um volume médio de bens ou serviços padronizados; e processos por projeto, os quais buscam o atendimento de uma necessidade específica dos clientes.

- Pela natureza do produto;

  a) Manufatura de bens: quando o produto fabricado é tangível.

  b) Prestador de serviços: quando o produto gerado é intangível.

Dessa forma, o autor resume as principais características em relação à classificação dos sistemas de produção na tabela 1.

	Contínuo	**Repetivido em Massa**	**Repetitivo em Lotes**	**Projeto**
Volume do Produção	Alto	Alto	Médio	Baixo
Variedade de produtos	Pequena	Média	Grande	Pequena
Flexibilidade	Baixa	Média	Alta	Alta
Qualificação da MOD	Baixa	Média	Alta	Alta
Layout	Por Produto	Por Produto	Por Processo	Por Processo
Capacidade ociosa	Baixa	Baixa	Média	Alta
Lead Times	Baixo	Baixo	Médio	Alto
Fluxo de informações	Baixo	Médio	Alto	Alto
Produtos	Contínuos	Em Lotes	Em Lotes	Unitário

Fonte: Tubino (1997)

Quadro 1: Características dos sistemas de produção

Gaither & Frazier (2001) propõem que, ao se programar a manufatura, ela seja dividia em **manufatura focalizada no produto** e **manufatura focalizada no processo**. Para os autores, existem dois tipos gerais de produção focalizada no produto: em lote e contínua.

> A produção em lote muitas vezes é chamada de *flow shop*, porque os produtos fluem ao longo de roteiros lineares diretos. Lotes grandes de diversos produtos padronizados são produzidos no mesmo sistema de produção. Uma vez que os produtos são produzidos em lotes, o sistema de produção deve ser modificado quando um produto diferente deve ser produzido. Muitos fabricantes de produtos discretos utilizam esse tipo de produção. Na produção contínua, alguns produtos altamente padronizados são produzidos continuamente em volumes muito grandes, e são raras as preparações de máquinas. (GAITHER e FRAZIER, 2001).

Na manufatura focalizada no processo, ainda de acordo com Gaither; Frazier (2001), a qual é comumente chamada de *job shop*, existe uma grande variedade de tarefas a serem processadas e conseqüentemente várias escolhas de roteiros. Os centros de trabalhos são organizados em torno de tipos de funções, e as tarefas são processadas em lotes. Os autores citam algumas características que fazem de um *job shop* um problema complexo para se obter um programa. O primeiro, como produzem sob pedidos, atrasos na produção afetam diretamente os clientes. Outro aspecto é que, como os lotes de produção são geralmente pequenos, várias preparações de máquinas devem ser feitas. Em um ambiente *job shop*,

> Trabalhadores e máquinas são tão flexíveis que são capazes de ser atribuídos e reatribuídos a muitos pedidos diferentes. Nesse ambiente flexível, variável e mutável, os programas devem ser específicos e detalhados em cada centro de trabalho para trazer ordem a uma situação potencialmente agitada. (GAITHER e FRAZIER, 2001).

Essas são as classificações dos sistemas de produção mais freqüentemente citados na literatura que trata de problemas relacionados à programação da produção, e que, de uma forma ou de outra, ajudam a entender o contexto desse tipo de problema.

Assimilado então esse item, a próxima seção busca o entendimento do que trata a programação da produção, mostrando de forma mais detalhada seu relacionamento direto com o tipo de sistema de produção a ser aplicado.

2.3 Programação da produção

Os problemas de programação da produção estão inseridos em um contexto em que se submetem a uma lógica de processamento e de hierarquia das decisões a serem tomadas para responder as demandas de cada fase do processo. Entender esse contexto é fundamental para a clara definição do escopo do trabalho. Pensando nisso, esta seção buscará, através da análise da literatura existente sobre o assunto, esclarecer tais situações e mostrar em quais ambientes de produção se enquadram os problemas abordados.

2.3.1 Hierarquia no planejamento da produção

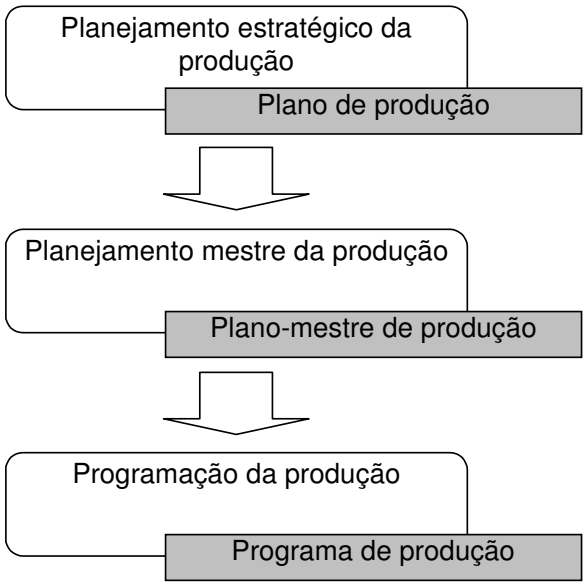
Um dos conceitos centrais na discussão sobre gestão da produção é o conceito de planejamento da produção. Isso porque as decisões, após serem tomadas, decorrem algum tempo até que se concretizem em ações concretas. Portanto, é necessário que se tenha uma visão do que vai ocorrer no futuro, para que as decisões tomadas tenham os efeitos desejados posteriormente, e isso é conseguido através do planejamento.

O processo de planejamento é continuado, de maneira que a cada momento deve-se ter uma noção do presente e uma projeção do futuro para que possíveis problemas sejam antecipados. À medida que o tempo passa, e se tendo como meta um horizonte de planejamento que atenda a necessidade da empresa, essa projeção do futuro deve ser atualizada com os novos dados que surgem a cada momento.

Porém, em uma mesma empresa, diferentes decisões são tomadas. Existem decisões que envolvem uma quantidade moderada de recursos. Estas podem ser tomadas de uma forma mais rápida. Entretanto, existem decisões que envolvem altos

investimentos e que definem o direcionamento da empresa. Obviamente, estas necessitam de maior tempo de discussão e, por isso, levam maior tempo para serem tomadas. Por essas razões é que surge a hierarquia do planejamento da produção, que divide as decisões referentes ao processo produtivo em tempos adequados para sua execução (TUBINO, 1997); (GAITHER e FRAZIER, 2001).

A forma esquemática da hierarquia de decisões do processo de planejamento da produção é ilustrada na Figura 2.



Fonte: Tubino (1997)

Figura 2: Hierarquia do planejamento da produção

Para Tubino (1997), o planejamento estratégico consiste em estabelecer um plano de produção em longo prazo, no qual se deve fazer uma comparação entre o que se estima vender e a capacidade produtiva prevista. Caso essa capacidade não esteja adequada a esta previsão de demanda, ela pode ser incrementada ou reduzida.

Para Rebouças (1988), o planejamento e a tomada de decisões neste nível tendem a ser em longo prazo, sendo de responsabilidade da mais alta cúpula da empresa, estabelecendo o rumo a ser tomado por esta, visando a obter o melhor resultado possível da relação empresa-meio ambiente.

Com base no planejamento estratégico da produção, o planejamento mestre da produção é elaborado, e deve desmembrar o plano de produção em programas mais detalhados em médio prazo.

> O planejamento mestre da produção coordena a demanda do mercado com os recursos internos da empresa de forma a programar taxas adequadas de produtos finais, principalmente aqueles que têm sua demanda independente (CORRÊA e CORRÊA, 2004).

Os autores colocam ainda alguns objetivos finais para esta parte do planejamento, tais como a maximização do serviço ao cliente, minimização de estoques e o máximo aproveitamento dos recursos produtivos.

Para Tubino (1997), o planejamento mestre da produção é responsável por fazer a conexão entre o planejamento estratégico da produção e as atividades operacionais da produção, separando os planos produtivos estratégicos de longo prazo em planos específicos de produtos acabados para o médio prazo, direcionando as etapas de programação e execução das atividades.

Russomano (1998) afirma que “o Plano Mestre da Produção é a determinação antecipada do programa de produção em médio prazo dos vários produtos que a empresa produz”. E dessa forma, utilizando os dados de estimativa de vendas, carteira de pedidos, disponibilidade de material, capacidade disponível, etc., estabelece com antecedência a melhor estratégia de produção a ser adotada pela empresa.

O último nível da hierarquia é o da programação da produção, que tem como primeiro objetivo satisfazer as metas de produção especificadas no nível anterior, que é o plano-mestre. Nesta parte, as tomadas de decisões são dadas no curto prazo e são em geral realizadas no dia-a-dia da empresa, cobrindo prazos de até algumas semanas (MOREIRA, 1998).

No intuito de definir exatamente em que consiste a programação da produção, Palomino (2001) afirma que,

> (...) consiste basicamente em determinar o tempo de início e término de cada ordem de produção para dar cumprimento aos compromissos estabelecidos no Plano Mestre de Produção. Tal determinação é

> conseqüência de: a) uma correta distribuição das operações necessárias aos diversos centros de trabalho (alocação de carga) e b) a determinação da ordem na quais as operações serão executadas (seqüenciamento). Estas duas questões geralmente estão sujeitas a dois tipos de restrições que são: o número de máquinas disponíveis no sistema, e as restrições tecnológicas que determinam a ordem na qual as tarefas (*jobs*) devem ser realizadas. Uma tarefa, no caso de um sistema de manufatura, representa um conjunto de operações necessárias para produzir um produto acabado. (PALOMINO, 2001).

Corrêa e Corrêa (2004) afirmam que a "programação de operações consiste em alocar no tempo as atividades, obedecendo ao seqüenciamento definido e ao conjunto de restrições considerado".

De forma resumida, pode-se dizer que na programação da produção é elaborado um cronograma detalhado que indica onde serão processadas as operações dos produtos de acordo com os recursos disponíveis, e o momento em que o processamento desses produtos deve começar e terminar (CARVALHO, 2003).

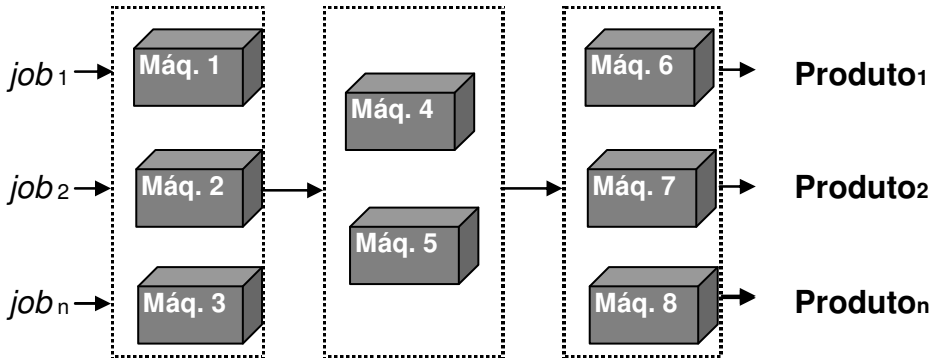
Analisando a problemática do seqüenciamento de produção, Blazewicz *et al.* (1996) mostram uma classificação e organização desses problemas baseados em três propriedades: características das máquinas, características das operações e recursos, e o critério de otimização do problema. Nesta classificação, dois conceitos são importantes: o conceito de operação, ou tarefa, e o conceito de *job*. Uma tarefa representa uma operação elementar que, para ser realizada, necessita de certo número de unidades de tempo e/ou recursos. Um *job* representa uma seqüência conhecida de uma ou mais tarefas, as quais compõem a seqüência tecnológica de fabricação de cada produto. Assim, num contexto de manufatura de produtos, um *job* pode representar a fabricação de um produto ou de um lote de uma família de produtos, que possuem a mesma seqüência tecnológica de fabricação.

Com base nessas propriedades, o autor define que, segundo as características das máquinas, os problemas podem ser classificados:

- Para máquinas paralelas, conforme ilustrado na figura 3, na qual se destacam os problemas:

- Com máquinas idênticas, que é o caso no em que todas as máquinas possuem a mesma velocidade;

- Com máquinas diferentes, caso no qual cada máquina possui uma determinada velocidade.

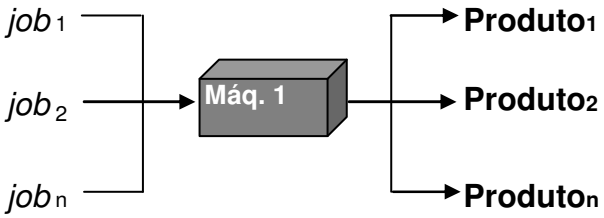


Fonte: Bustamante (2007)

Figura 3: Problema de seqüenciamento de máquinas paralelas

- Para máquinas especializadas, que executam um único tipo de tarefa, destacam-se os problemas de:

- Uma máquina: quando todas as tarefas são executadas em apenas uma máquina, conforme demonstrado na Figura 4. Neste caso particular, o conceito de tarefa coincide com o conceito de *job*, pois cada tarefa é independente de qualquer outra tarefa.

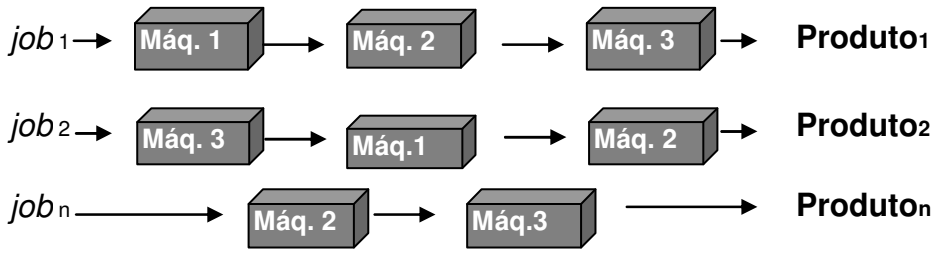


Fonte: Bustamante (2007)

Figura 4: Problema de seqüenciamento de uma máquina

- *Job shop* e *flow shop*. A definição formal de um problema tanto de *job shop* como de *flow shop* é praticamente a mesma. A diferença dos dois problemas está mais ligada às restrições que cada um apresenta.

A explicação de um problema de *job shop* pode ser mostrada da seguinte forma: quando um conjunto de máquinas executa tarefas sobre um conjunto de *jobs*, conforme a Figura 5, sendo cada *job* fabricado por uma seqüência própria de operações (tarefas). (BUSTAMANTE, 2007).



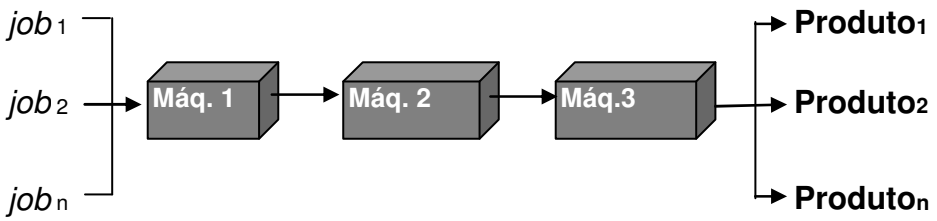
Fonte: Bustamante (2007)

Figura 5: Problema de seqüenciamento de um *job shop*

As restrições que devem ser respeitadas são:

- Operações não podem ser interrompidas e cada máquina pode processar apenas uma operação de cada vez;
- Cada *job* só pode ser processado em uma única máquina de cada vez;
- Cada *job* é fabricado por uma seqüência conhecida de operações;
- Não existe restrição de precedência entre operações de diferentes *jobs*;
- Não existe qualquer relação de precedência entre as operações executadas por uma mesma máquina;
- Nem todos os *J jobs* passam pelas *M* máquinas.

Quando um conjunto de máquinas executa tarefas sobre um conjunto de *jobs*, sendo que todos os *jobs* possuem a mesma seqüência de operações (tarefas) sobre as máquinas, tem-se então um problema *flow shop,* conforme apresentado esquematicamente na Figura 6 (BUSTAMANTE, 2007).



Fonte: Bustamante (2007)

Figura 6: Problema de seqüenciamento de um *flow shop*

Observa-se então que o *flow shop* nada mais é do que um caso especial do *job shop* e, deste modo, os modelos desenvolvidos para resolver problemas de *job shop* resolvem, conseqüentemente, os problemas *flow shop*.

Quanto às restrições desse problema, deve-se diferenciar em relação ao *job shop* apenas que:

- Cada *job* é executado por uma seqüência conhecida de operações sendo a mesma para todos os *jobs*.
- Todos os *jobs* passam necessariamente por todas as máquinas.

O critério de otimização difere em cada situação prática de acordo com os objetivos da empresa. Partindo-se do princípio que a tarefa da programação da produção é alocar no tempo o processamento de um conjunto de ordens, em uma seqüência adequada e em um determinado recurso de forma a otimizar algum indicador de qualidade desta programação, alguns indicadores são mais freqüentemente utilizados. O *makespam*, apesar de não ser o único existente, é o mais conhecido e amplamente encontrado na literatura (RODAMMER e WHITE, 1988). A soma dos tempos de fluxo e a soma dos tempos de espera são alguns dos critérios também utilizados, e que são baseados nos tempos de fim das operações (REIS, 1996).

As características do processo produtivo também impõem restrições ao problema de seqüenciamento, que devem ser respeitadas. O tempo de preparação de máquina para cada troca de produto é uma restrição específica para cada caso. Existem ainda

as restrições que são comuns à maioria dos problemas, como a de que a cada instante uma determinada tarefa só pode ser executada por no máximo uma máquina.

2.3.2 Definição formal do problema de programação

Ao se tratar de otimização combinatorial, o problema da programação de produção é considerado como dos mais difíceis de serem resolvidos. Isso porque as várias possibilidades existentes de soluções elevam em demasia a complexidade do problema, sendo considerado na literatura como um problema *NP-hard* (GAREY e JOHNSON, 1979). Essa complexidade é ainda mais visível em casos reais, nos quais a combinação de vários objetivos e restrições aumentam exponencialmente o espaço de busca, e encontrar um programa que seja ótimo se torna muito difícil.

A definição formal do problema de programação de produção que é mostrado nesta seção é encontrada de forma parecida nos trabalhos de Allahverdi (2006), Pradhan (2006), Udomsakdigool (2005), Reis (1996), entre outros.

O problema de programação da produção em geral é posto do seguinte modo:

- Há um conjunto de M máquinas, em que $M = \{m_1, m_2, ..., m_M\}$.
- Há um conjunto de N *jobs*, onde $J = \{J_1, J_2, ..., J_N\}$.
- Cada *job* J consiste em um conjunto de K operações, onde $J_j = \{o_{1j}, o_{2j}, ..., o_{kj}\}$.
- Cada operação de um determinado *job* J está relacionada com uma máquina do conjunto M. Então, $J_j = \{o_{1j}(m), o_{2j}(m), ..., o_{kj}(m)\}$. Aqui também se define o conceito de relação de precedência entre as operações. Quando se coloca que o conjunto das operações de um determinado *job* J é dado por $J_j = \{o_{1j}(m), o_{2j}(m), ..., o_{kj}(m)\}$, assume-se que será feito primeiramente a operação $o_{1j}(m)$. Em seguida, será executada a operação $o_{2j}(m)$, ou seja, necessariamente após a operação $o_{1j}(m)$. E assim sucessivamente.
- A cada operação o_{kj} associa-se um tempo de início da operação, denotado por s_{kj}. Neste trabalho, é considerado ainda que a operações são não-preemptivas, ou seja, quando começadas, não podem ter seu processamento encerrado antes que este esteja completo.

- A cada operação o_{kj} associa-se um tempo de término da operação, denotado por t_{kj}.
- Para cada operação o_{kj} há uma duração, ou tempo de processamento, denotado por p_{kj}, onde $p_{kj} = t_{kj} - s_{kj}$. A cada operação está associado ainda um tempo de *set up*, que invariavelmente está relacionado com a seqüência da operação no programa. Neste trabalho, será assumido que este tempo é o mesmo para qualquer seqüência definida, o que possibilita então considerar que este tempo pode ser incluído no tempo total de processamento da operação.

Alguns parâmetros, úteis para traduzir critérios de programação, são derivados dos anteriores e podem assim ser demonstrados:

Para cada *job J*, tem-se:

- Tempo de fim, ou *completion time*, C_j.
- Tempo de fluxo, ou *flow time*, $F_j = C_j - r_j$ (soma dos tempos de espera e processamento).
- Atraso absoluto, ou *tardiness*, $A_j = \max \{C_j - d_j, 0\}$.

A tarefa da programação de produção nada mais é do que encontrar uma seqüência de processamento para todas as operações de todos os *jobs*. O que se quer então é definir em qual máquina cada *job J* será processado e a seqüência das operações nestas máquinas. Quando cada máquina tiver sua seqüência definida, o programa de produção está pronto. Nesse caso, cada máquina *M* terá também seu tempo de término, ou *completion time*, denotado por C_m. Obviamente, esse valor é igual ao tempo de término da última operação a ser processada por esta máquina ($t_{kj}(m)$).

Cada critério corresponde a uma função custo que, para cada programa particular, terá um valor. Quanto menor esse valor, de melhor qualidade é o programa gerado. A procura do melhor programa para uma dada instância de determinado problema de programação traduzir-se-á na procura do programa cujos parâmetros originam o menor valor da função custo.

Esta parte do trabalho tinha como objetivo descrever, de modo formal, o problema de programação da produção. Com base neste entendimento, a próxima

seção procura demonstrar então os métodos utilizados para resolução deste tipo de problema, ou seja, problemas de otimização.

2.4 Métodos para solução de problemas de otimização

O conceito de otimização está diretamente relacionado com a obtenção do melhor resultado possível de determinada tarefa ou processo. Este melhor resultado pode estar explícito em diversos tipos de indicadores, podendo ser adotado a combinação de vários deles.

O método utilizado para a obtenção desse melhor resultado está relacionado ao tipo de problema em questão. Um dos métodos para encontrar a solução ótima para um problema de *scheduling* é o de usar um algoritmo que procura determinar o programa ótimo de uma forma exata, e que conseqüentemente apresenta a melhor solução. Contudo, a complexidade revela-se um obstáculo com este tipo de algoritmo, salvo em casos em que se introduzam simplificações que, não raro, são tão drásticas que se afasta do problema real. Em face disto, passou-se a dar maior atenção e importância a métodos aproximados, ou heurísticos, que não garantem a melhor solução, porém são viáveis para aplicações a casos reais de grande complexidade.

Os métodos de busca por soluções denominados exatos são aqueles que sempre encontram a melhor solução para o problema, desde que ela exista. Desse ponto de vista, o problema de *scheduling* pode ser considerado como um problema de otimização combinatória sujeito a restrições, em que se procura uma maneira de designar recursos, seqüenciar e temporizar tarefas. Exemplos de trabalhos nesta área são: Luche e Morabito (2004), que realizam um estudo de caso em uma empresa da indústria de grãos eletrofundidos, com o objetivo de aumentar a produtividade e melhorar o nível de serviço aos clientes, por meio da aplicação de modelos de programação linear inteira mista; Alle (2003), que utiliza técnicas de programação matemática mista inteira para programação da produção de plantas químicas contínuas; Miranda (2001), que faz a utilização de métodos de programação matemática para otimização da programação de produção de misturas de produtos na

indústria de petróleo; e Haddad (2006), o qual analisa a integração da programação matemática com ERP para resolver problemas de capacidade na programação de produção. A grande vantagem dessas técnicas é a garantia da solução ótima. No entanto, o que as torna inviáveis é seu custo computacional, principalmente em problemas complexos, como são os de *job shop* e *flow shop*.

A dificuldade da abordagem exata dos problemas de programação está na explosão combinatória do espaço de busca quando se adiciona uma variável. Segundo Cormen *et al.* (2001), um problema pode ou não ter um algoritmo exato para sua solução. Mesmo que exista este algoritmo, ele pode não encontrar a solução ótima em tempo hábil, ou seja, ele pode ser inviável.

Graves (1981) reconhece que "(...) há um distanciamento entre teoria e prática do *scheduling* da produção (...)" e que "(...) o desenvolvimento de melhores algoritmos e heurísticas é essencial para que a teoria do *scheduling* possa continuar a melhorar a prática do *scheduling*". O mesmo autor afirmava ainda haver uma grande necessidade "(...) de modelos de *scheduling* mais realísticos e de uma melhor compreensão da dinâmica inerente ao ambiente de *scheduling*".

Ainda, segundo Dorn (1994), há três tipos de problemas com os modelos analíticos / formais referidos :

- Os algoritmos são complexos em demasia para aplicações do mundo real;
- Os modelos exigem o conhecimento exato de durações e restrições técnicas;
- O esforço necessário para formalizar um novo problema é considerável.

Diante disso, tem-se verificado nas últimas décadas o surgimento de um conjunto de técnicas e algoritmos muito eficientes, mas que não garantem a solução ótima do problema. Estas técnicas são denominadas de heurísticas (GOLDBARG e LUNA, 2000).

> Uma heurística é uma técnica que busca alcançar uma boa solução utilizando um esforço computacional considerado razoável, sendo capaz

> de garantir a viabilidade ou a otimalidade da solução encontrada ou ainda, em muitos casos, ambas, especialmente nas ocasiões em que essa busca partir de uma solução viável próxima ao ótimo (GOLDBARG e LUNA, 2000).

Conforme sustenta Reeves (1993), sendo as heurísticas em geral mais flexíveis e capazes de tratar funções objetivo e/ou constrangimentos mais complicados, tornam possível terem-se modelos exatos ou pelo menos muito mais aproximados do problema real.

Já as metaheurísticas nada mais são do que subdivisões das heurísticas e, segundo Romero e Mantovani (2004), "(...) representa uma evolução em relação aos algoritmos heurísticos clássicos".

A distinção, que aqui se deve fazer, é que uma metaheurística possui grande abrangência, podendo ser aplicada a vários problemas de otimização combinatória. Como exemplo de metaheurísticas pode-se citar: ACO (*Ant Colony Optimization*), GA (*Genetic Algorithm*), AS (*Simulated Annealing*) e TS (*Tabu Search*). Já a heurística é a instanciação de uma metaheurística, ou seja, a aplicação da metaheurística em um problema específico de otimização. Por exemplo, Dorigo *et al.* (1991), denominou *Ant System* a heurística ACO aplicada à resolução do Problema do Caixeiro Viajante. Entretanto, essa heurística só ganhou dimensões de metaheurística quando Dorigo *et al*. (1999) a flexibilizou para tal.

De acordo com Romero e Mantovani (2004), a idéia fundamental de uma metaheurística é visitar apenas um conjunto reduzido do espaço de busca, em que esta visita deve ser feita de forma eficiente para que seja encontrada uma solução ótima ou quase ótima. Portanto, uma metaheurística é uma estratégia que especifica a forma em que devem ser realizadas as transições através do espaço de busca partindo-se de um ponto inicial ou de um conjunto deles. "Assim, a diferença entre as metaheurísticas é a estratégia usada por cada uma delas."

As heurísticas modernas (metaheurísticas) têm despertado crescente interesse na área de pesquisa operacional, devido aos resultados positivos encontrados em relação aos objetivos especificados, o que as torna uma alternativa cada vez mais

interessante. Alguns exemplos de metaheurísticas em diferentes áreas de atuação científica são descritos na seqüência.

Soares (2006), utilizando uma técnica de inteligência artificial (IA) conhecida como Algoritmo Genético, busca a otimização para o complexo problema de planejamento mestre de produção. Por meio da heurística, busca maximizar o nível de atendimento ao cliente e a utilização dos recursos existentes, e ao mesmo tempo, busca minimizar os níveis de estoque e a utilização de recursos extras, como contratações, horas extras, etc.

Um algoritmo, nomeado pelo autor *Progressive Bottlenec Improvement* (PBI), foi utilizado por Phandis *et al*. (2003) como técnica heurística para resolver um problema de *scheduling*. O trabalho apresenta o desenvolvimento de um algoritmo com inteligência PBI para a solução de um problema do tipo *flow shop*, com máquinas paralelas, e tem como objetivo obter uma programação que minimize o *makespan*. A heurística PBI é desenvolvida em duas fases. Na primeira é gerada uma seqüência inicial que identifica os estágios gargalos *(bottleneck*); na segunda fase a seqüência obtida na primeira é refinada baseada em certas características existentes. O trabalho mostra uma heurística simples que pode produzir soluções satisfatórias.

A análise do uso de têmpera simulada (S*imulated Annealing*) é feita por Ribas (2003), com o intuito de otimizar o planejamento mestre de produção. A heurística é uma técnica de IA de busca baseada no fenômeno físico da têmpera. O autor faz uma abordagem multiobjetivo na qual utiliza o algoritmo para tentar otimizar alguns objetivos: nível de atendimento ao cliente, estoques, tempos de trocas, horas extras, quantidade de estoques abaixo do nível de segurança e utilização dos recursos. O trabalho conclui que a técnica heurística sugerida (têmpera simulada) é uma ferramenta eficiente na busca de resultados mínimos de funções objetivo, provando assim a aplicabilidade dessa técnica ao problema de planejamento mestre de produção.

Mazzucco Jr. (1999) propôs e investigou a potencialidade de um método híbrido baseado na combinação de SA e AG em problemas de programação da produção em sistemas *job shop*. Os testes realizados com o algoritmo desenvolvido utilizaram um *benchmark* bastante usado na literatura sobre o assunto.

Buzzo e Moccellin (2000) fizeram um estudo de um método heurístico híbrido utilizando AG e SA para um problema de programação da produção em sistemas *flow shop* permutacional. O trabalho tem como principal objetivo analisar se a técnica híbrida é mais eficiente para resolver o problema em comparação como qualquer um dos métodos puros utilizadas isoladamente. O indicador de desempenho empregado para a comparação também é o *makespan*. Depois de desenvolver e testar 200 algoritmos híbridos, combinando-se diferentes operadores de cruzamento, temperaturas iniciais e parcelas da vizinhança analisada, o trabalho chega à conclusão que o método híbrido pode ser desenvolvido de maneira que agregue as características vantajosas dos métodos metaheurísticos puros.

Através de uma arquitetura multiagente, baseada em *Times Assícronos* (*A-Teams*), Passos e Fonseca (2005) propõem a utilização cooperativa e sinérgica dos seguintes algoritmos: Busca Tabu, AG, *Beam-Search* e heurística *Cheap-NEH*. A estratégia do trabalho busca resolver problemas de seqüenciamento da produção em sistemas *job* e *flow shops*, com limitação na oferta de recursos de uso compartilhado. O objetivo é seqüenciar as operações minimizando o tempo total de produção (*makespan*), respeitando as restrições do problema. O trabalho conclui que os algoritmos cooperam entre si apresentando resultados no que se refere à qualidade de solução e desempenho computacional superiores aos obtidos em trabalhos anteriores com os algoritmos utilizados isoladamente.

Campos Jr. (2000) realizou um estudo de caso em uma empresa do ramo de autopeças para a aplicação de AG para a solução de problemas de escalonamento. Vários experimentos e diversos escalonamentos em ambiente real foram obtidos e comparados com aqueles obtidos manualmente; melhorias não só na qualidade do escalonamento gerado foram sentidas, como também melhorias no tempo de trabalho demandado para a obtenção do escalonamento.

Castro (2001) aplicou AG para a solução de problemas de programação de produção em uma refinaria de petróleo, em um sistema de produção e armazenamento do gás liquefeito de petróleo. Além de uma solução eficaz para o problema de programação da produção, o trabalho demonstrou que o algoritmo consegue também

um aumento do horizonte de programação e a possibilidade de avaliação de soluções alternativas.

Colin e Shimizu (2000) aplicam um algoritmo que é a generalização do algoritmo de programação sugerido por Garey *et al*. (1988) e, através de uma estrutura computacional denominada fila de prioridade, buscam otimizar a programação de produção com penalidades distintas entre adiantamento e atraso, para o problema de uma única máquina. O objetivo é minimizar a soma das diferenças (adiantamentos ou atrasos) penalizadas das ordens. Os resultados demonstram que se ganha tempo de processamento.

A seguir estão descritas, e mostradas resumidamente, a estratégia de atuação de algumas das metaheurísticas mais utilizadas na literatura.

2.4.1 Algoritmo genético

Esta estratégia de busca de soluções ótimas está baseada em um fenômeno biológico e foi introduzida por Holland (1975) e posteriormente Goldberg (1989). Faz uso de uma analogia com o fenômeno natural da evolução dos organismos vivos. A idéia dos AGs é a de simular o desenvolvimento de uma população de indivíduos de uma espécie com características cada vez melhores, ou mais desejáveis, em face de uma seleção natural. Essas características são determinadas ao nível genético pelo modo como são combinados os cromossomos dos progenitores.

O objetivo deste algoritmo é explorar o espaço de busca na determinação de melhores soluções, permitindo que os indivíduos na população evoluam com o tempo. Cada passo do algoritmo no tempo, na escala evolutiva, é chamado de geração. Seguindo as idéias de Gauthier (1993), Goldber (1989) e Tanese (1989), o algoritmo executa continuamente o seguinte laço principal, partindo de uma geração inicial, normalmente criada aleatoriamente:

1 – avalia cada cromossomo da população através do cálculo de sua aptidão, utilizando a função de avaliação (etapa de avaliação);

2 – seleciona indivíduos que produzirão descendentes para a próxima geração, em virtude dos seus desempenhos apurados no passo anterior (etapa de seleção);

3 – gera descendentes através da aplicação das operações de cruzamento, *crossover* e mutação sobre os cromossomos selecionados no passo anterior, criando a próxima geração (etapa de reprodução);

4 – se o critério de parada for satisfeito, termina e retorna o melhor cromossomo até então gerado, senão volta ao passo 1.

Reeves (1993) relata aplicações dos algoritmos genéticos ao *scheduling,* em especial ao problema de *flow shop.* Neste caso, foram feitas comparações do desempenho do AG implementado com a de uma heurística tipo AS. Concluiu-se que havia uma equiparação dos dois métodos em geral, mas uma performance crescente do AG à medida que a dimensão do problema aumentava.

2.4.2 Têmpera simulada

Têmpera simulada é considerado um tipo de algoritmo conhecido como de busca local. Ele se constitui em um método de obtenção de soluções eficazes para problemas de otimização de difíceis resoluções. Desde sua introdução como método de otimização combinatorial, este é amplamente utilizado em diversas áreas, tais como projeto de circuitos integrados auxiliado por computador, processamento de imagem, redes neurais, etc. (MAZZUCCO JUNIOR, 1999).

O método têmpera simulada faz uso de uma analogia da Termodinâmica Estatística, no que diz respeito ao fenômeno da mudança de estado de energia de um sólido quando submetido a um processo de arrefecimento até convergir para um estado final (estado este que depende do modo como é feito o arrefecimento) (KIRKPATRICK, 1983), (LAARHOVEN, 1988), (AARTS, 1989).

Reis (1996) descreve a seqüência do algoritmo como segue. Num problema de otimização, um estado sólido corresponde a uma solução do problema, o nível de energia do sólido ao valor da função objetivo para uma solução e a temperatura a um parâmetro de controle que varia (diminui) em cada interação, servindo como um

contador de interações. O estado final corresponde à solução encontrada. O método começa com uma solução sub-ótima do problema e para certo valor do parâmetro de controle. Em cada interação esse parâmetro permanece constante. É escolhida, aleatoriamente, uma solução vizinha da solução atual, que será aceita como solução atual se o seu custo for menor. Caso contrário, a solução escolhida pode mesmo assim ser aceita se um valor aleatoriamente gerado for menor que o valor da função de probabilidade. Isso é repetido um número determinado de vezes na mesma interação, após o que o parâmetro de controle é decrementado, repetindo-se em seguida um novo ciclo até que o parâmetro de controle seja zero.

Dowsland (1993) descreve resumidamente aplicações com sucesso do algoritmo têmpera simulada à sequenciação no problema de *flow shop*: Ogbu (1990), Ogbu (1991) e Osman (1989).

2.4.3 Busca *tabu*

O método busca *tabu* combina um algoritmo determinístico de avance interativo com a possibilidade de aceitar soluções que incrementem o custo. Dessa maneira, a busca é dirigida fora dos ótimos locais, e outras partes do espaço de soluções podem ser exploradas. À semelhança da têmpera simulada, também este método faz uso da noção de vizinhança e de uma analogia com um fenômeno natural que é, no caso, o uso de uma forma de memória (GLOVER, 1993).

Para que a busca seja guiada a não se encaminhar para soluções localmente ótimas, usa-se uma estrutura de memória designada por lista *tabu*, que contém em cada momento um número de soluções proibidas determinadas por uma história relativa a um número de interações anteriores. Alternativamente, a lista *tabu* pode memorizar, não as próprias soluções proibidas, mas os movimentos no espaço de soluções que levam a elas. (GLOVER, 1986), (HANSEN, 1986) e (GLOVER, 1993).

No método de busca *tabu*, em cada interação passa-se da solução atual para uma na vizinhança que não esteja da lista *tabu* e que melhore o valor da função objetivo ou a que menos o piora, se não houver nenhuma melhor do que a atual. Ainda

assim, existe um critério que determina se um movimento é admissível apesar de estar na lista tabu. Podem existir, no contexto deste método, estratégias de intensificação, para concentrar a procura em regiões do espaço de soluções mais promissoras ou de diversificação, para conduzi-las a regiões não exploradas (GLOVER, 1986), (HANSEN, 1986) e (GLOVER, 1993).

Glover (1993) descreve resumidamente várias aplicações com sucesso do busca *tabu* a variados problemas de *scheduling.*

A próxima seção se ocupará de explicar mais detalhadamente a estratégia de funcionamento da heurística que será utilizada neste trabalho, que tem como objetivo principal otimizar a programação da produção em um sistema de produção específico. Ao se entender a concepção geral do *ACO*, poder-se-á então estruturar o problema mais formalmente para a sua resolução.

2.5 A metaheurística ACO

Os métodos de otimização baseados nos princípios de grupos de criaturas que, através de atividades exercidas de forma cooperativa, conseguem encontrar soluções eficazes para problemas complexos, têm recebido crescente interesse nas últimas décadas, em virtude da sua eficiência na resolução destes problemas. Esta parte do capítulo tem como objetivo apresentar em linhas gerais a forma de estruturação e funcionamento de um desses métodos, conhecido como *ACO*.

Dorigo *et al*. (1999) fizeram a heurística *Ant System* tomar a forma de uma metaheurística, denominada *Ant Colony Optimization*, ou Otimização por Colônia de Formigas. Nesta metaheurística, uma colônia de formigas artificiais coopera entre si com o objetivo de encontrar boas soluções para problemas de otimização discreta. A essência da metaheurística *ACO* é a cooperação entre os agentes, no caso as formigas, na varredura do espaço de busca pela solução ótima.

2.5.1 A inspiração biológica

Algoritmos de colônia de formigas foram inspirados na observação de colônia de formigas reais. Esses insetos sociais vivem em colônias e seu comportamento é direcionado muito mais pela sobrevivência da colônia do que de cada individuo. As formigas reais são capazes de encontrar o caminho mais curto para uma fonte de alimento do formigueiro sem a utilização de dados visuais. Descobriu-se então que o meio usado entre os indivíduos para trocar informações relacionadas com o caminho mais curto a ser seguido, consiste no odor de feromônio.

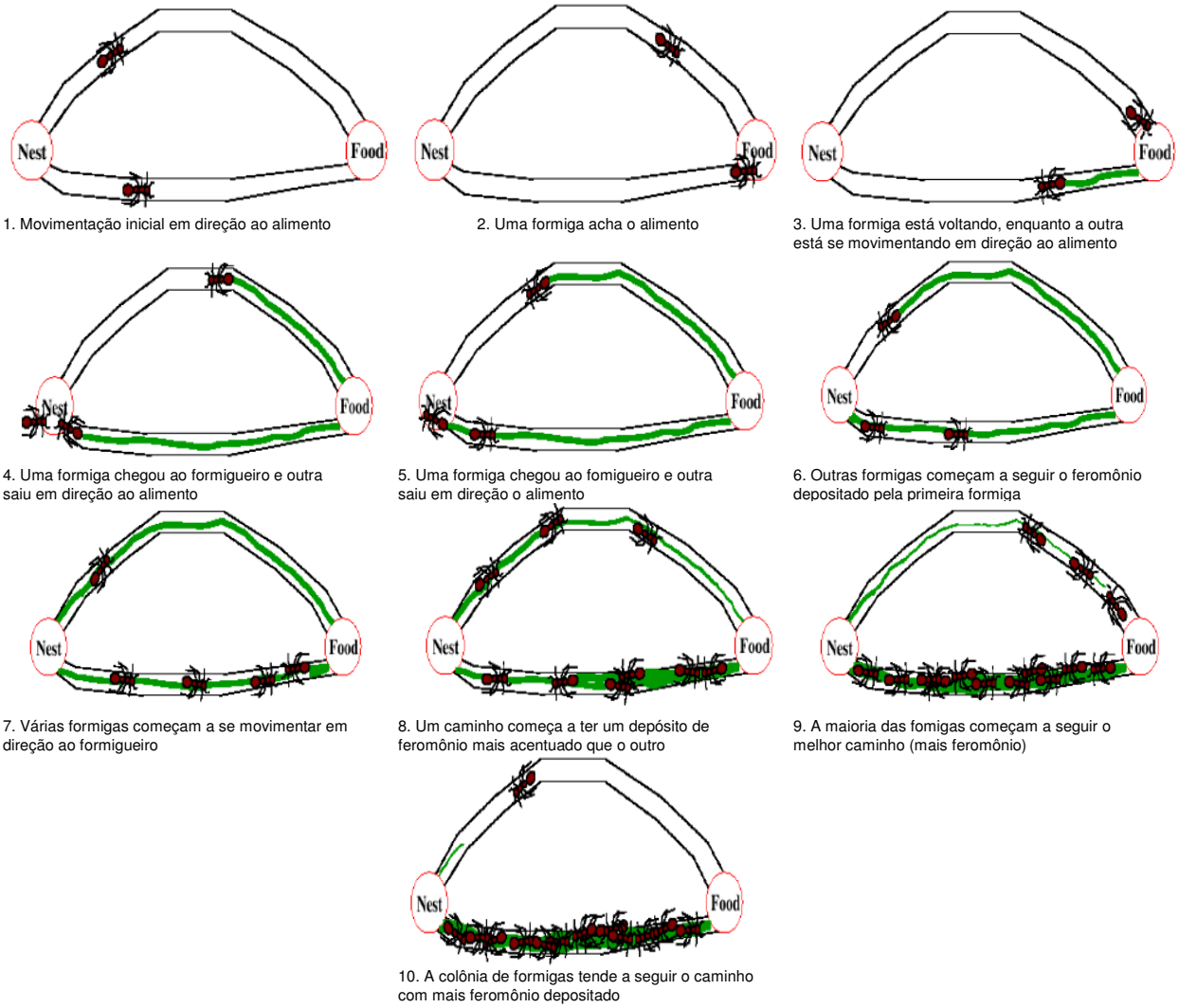
Enquanto se movimentam, as formigas depositam no solo, em quantidade variável, algum feromônio sobre a trilha, marcando o caminho com o odor dessa substância. Uma formiga encontrando uma trilha previamente marcada com o cheiro pode decidir com maior probabilidade seguir esta trilha, reforçando o odor com seu próprio feromônio. No entanto, uma formiga que se movimentando sozinha, escolhe o caminho de forma aleatória. Dessa maneira, a maioria das formigas tende a seguir o odor, que se torna mais intenso e mais propenso a ser seguido. Há então um comportamento coletivo do tipo auto-catalítico, em que a probabilidade de uma formiga escolher um determinado caminho aumenta com o número de formigas que escolheu previamente o mesmo caminho (DORIGO, 1991).

Coelho (2003) descreve de forma resumida a estratégia de trabalho das formigas da seguinte maneira:

> Primeiro, quando as formigas chegam a um ponto de decisão em que elas têm que decidir mover-se à direita ou à esquerda, as formigas selecionam aleatoriamente o próximo caminho e depositam feromônio no solo, sem ter a noção de qual é a melhor escolha. Depois de um pequeno período de tempo a diferença entre a quantidade de feromônio entre dois caminhos é suficientemente grande que influencia a decisão de novas formigas que estão no impasse da nova tomada de decisão por qual caminho seguir. Neste caso, as novas formigas escolhem o caminho com maior quantidade de feromônio (COELHO, 2003).

Uma evolução do comportamento cooperativo das formigas - de sua movimentação do ninho ou formigueiro (*nest*) até o alimento (*food*) - baseada na

atualização do feromônio, segundo Krink *et al*. (2002), citado por Coelho (2003), é apresentada na figura 7.



Fonte: Krink *et al.* (2002)

Figura 7: Evolução do comportamento das formigas

2.5.2 Diferenças e semelhanças entre formigas reais e artificiais

Como dito na seção anterior, a maioria das idéias utilizadas na metaheurística ACO provém do mundo real, ou seja, ocorre realmente nas colônias de formigas. As

características que conferem tal semelhança entre o natural e o artificial são, segundo Dorigo *et al*. (1999):

- Ambos os sistemas são constituídos por múltiplos agentes cooperando entre si.
- Ambos os sistemas utilizam um fator de cooperação, através do qual acontece a sinergia entre os agentes (feromônio).
- Os agentes dividem a mesma função em ambos os universos: o de busca de menor caminho entre uma origem e um destino.
- O comportamento estocástico e local dos agentes na busca por soluções.

Porém, existem diferenças entre os dois universos que precisam ser expostas, e que são importantes para o funcionamento da metaheurística. Segundo Dorigo *et al*. (1999), essas diferenças são:

- As formigas artificiais possuem movimentação discreta, sendo que seus movimentos consistem em origens e destinos discretos.
- As formigas artificiais possuem memória, para que não haja sobreposição de movimentos já executados.
- O depósito de feromônio no mundo artificial ocorre com base na qualidade da solução encontrada.
- Como a *ACO* consiste em um metaheurística, ela pode possuir artifícios para otimizar ainda mais a busca por soluções. Tais recursos podem ser:

  - *Lookahead*, que consiste em analisar não apenas a próxima vizinhança de movimentos, como também as vizinhanças subseqüentes.

  - Otimizações locais, na qual um agente ataca o problema localmente antes de decidir por qual caminho seguir. Essas otimizações locais recuperam uma seqüência de passos a seguir.

  - *Backtracking*, ou seja, otimizações em caminhos já escolhidos, a fim de melhorar a rota até então designada.

As considerações que estão sendo feitas nesta parte do trabalho sobre a metaheurística *ACO* foram retiradas de Dorigo *et al*. (1999), que apresentam ainda algumas idéias que precisam ser entendidas para que se possa aplicar a técnica.

A proposta da metaheurística é de um grupo de agentes inteligentes artificiais, as formigas, que compartilham informações através de um meio comum, o espaço de busca de soluções. Para haver cooperação, são necessárias algumas formigas no ambiente, pois só assim há sinergia entre o meio e as formigas. Para a otimização propriamente dita, é necessário um sistema com número adequado de formigas.

A construção de uma solução ocorre mediante a escolha, seguindo uma função probabilística do melhor vizinho determinado por tal função. Esta função, estocástica, é uma busca local guiada pela (a) memória interna de cada formiga, (b) feromônio, disponível através do sistema como um todo nas informações locais conhecidas *a priori.*

Cada agente possui um estado interno para armazenagem relacionado ao seu passado, utilizada para construir a solução final. Este estado interno é geralmente utilizado no cálculo da competitividade da solução. Ao encontrar sua solução, a formiga morre, ou seja, o agente inteligente é removido do sistema. Sua contribuição para o mesmo é constituída pala memória deixada para o grupo (feromônio), bem como pela solução encontrada, caso esta seja melhor do que as encontradas até então.

2.5.3 Formulação matemática para a metaheurística ACO

A formulação matemática e o algoritmo básico apresentado a seguir para a metaheurística *ACO* foi baseado em Dorigo *et al*. (1991). O autor utiliza-se do Problema do Caixeiro Viajante (*Travelling Salesman Problem, TSP*), para construir uma heurística baseada no comportamento de formigas.

O *Travelling Salesman Problem* consiste em um conjunto de localidades a serem visitadas, obrigatoriamente, apenas uma vez, por um agente qualquer que, após um ciclo, deve voltar a sua posição de origem. O objetivo é encontrar o caminho mais curto que contenha um *tour* por todas as cidades. Uma instância do *TSP* pode ser representada pelo grafo valorado *G(V,E)*, onde *V* representa o número de vértices (localidades) e *E* consiste no número de arestas (estradas) do grafo, cada aresta com

seu respectivo custo, (caminho, distância). Esta distância é denotada por d_{ij}, que indica a distância entre a cidade i e j ($d_{ij} = \left[(Xi - Xj)^2 + (Yi - Yj)^2\right]^{1/2}$).

Seja $b_i(t)(i = 1,...,n)$ o número de formigas na cidade i no tempo t e seja

$$m = \sum_{i=1}^{n} b_i(t) \tag{2.1}$$

o número total de formigas.

Cada formiga tem as seguintes características:

(i) Escolhe a próxima cidade a ser visitada com uma probabilidade que é função da distância e da quantidade de feromônio existente na aresta que conecta as duas cidades.

(ii) Para forçar as formigas a realizarem um roteiro factível, transições para cidade já visitadas são descartadas até que um *tour* seja completado.

(iii) Quando é completado um circuito cada formiga deposita uma certa quantidade de feromônio sobre cada aresta (i, j) visitada.

Seja $\tau_{ij}(t)$ a intensidade de feromônio na aresta (i, j) no tempo t. Cada formiga no tempo t escolhe a próxima cidade para onde irá no tempo $t+1$. Definindo uma interação do *ACO* como sendo os n movimentos realizados pelas m formigas no intervalo $(t, t+1)$, então as n interações de cada uma das formigas formam um ciclo, ou seja, cada formiga realizou um *tour* passando por todas as cidades. Em todos estes pontos a intensidade de feromônio é atualizada segundo a fórmula:

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \tag{2.2}$$

em que ρ é um coeficiente onde $(1-\rho)$ que representa a evaporação do feromônio entre os tempos t e $t+n$ e $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{\,k}$ onde $\Delta\tau_{ij}^{\,k}$ é a quantidade por unidade de comprimento de feromônio depositada na aresta (i, j) pela *k*-ésima formiga entre os

tempos t e $t+n$. O coeficiente ρ deve ser ajustado em um valor menor que 1 para evitar acúmulo ilimitado de substância. Normalmente a intensidade do odor no tempo 0, $\tau_{ij}(0)$ é ajustada como uma constante inteira positiva c.

A regra para satisfazer a restrição de que uma formiga visite n diferentes cidades, é associar a cada formiga um lista, chamada lista *tabu*, que armazena as cidades já visitadas e proíbe a formiga de visitá-las novamente antes que o *tour* tenha sido completado. Quando um *tour* é completado, a lista *tabu* é utilizada para calcular a solução atual da formiga (isto é, a distância do caminho percorrido). Defini-se $tabu_k$ o vetor que cresce dinamicamente e que contém a lista *tabu* da *k*-ésima formiga, e *tabu(s)* a *s*-ésima cidade visitada pela formiga *k* no corrente *tour*.

Chamando-se de visibilidade η_{ij} a quantidade $1/d_{ij}$, defini-se então a probabilidade de transição da cidade i para a cidade j pela *k*-ésima formiga como:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k \in Permitido_k} [\tau_{ik}(t)]^{\alpha}.[\eta_{ik}]^{\beta}} & \text{se } j \in Permitido_k \\ 0 & \text{caso contrário} \end{cases} \qquad (2.3)$$

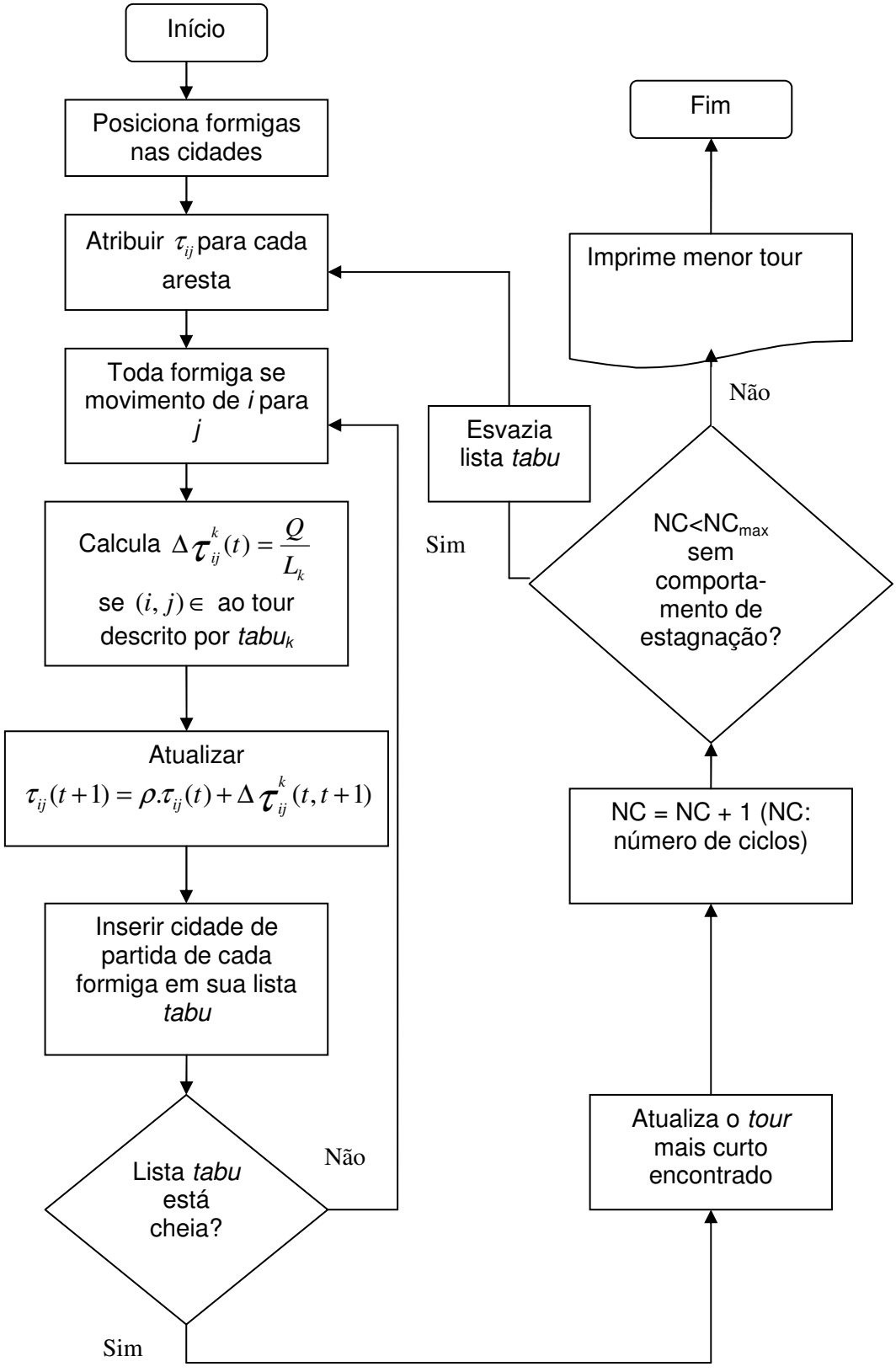
em que $Permitido_k = \{N - tabu_k\}$ e α e β são parâmetros que controlam a importância relativa do odor de feromônio *versus* a visibilidade. Desta forma, a probabilidade de transição é uma combinação entre visibilidade e intensidade do odor de feromônio no tempo t.

Segundo Dorigo (1991), existem diferentes formas de computar o valor $\Delta\tau_{ij}^{k}(t,t+1)$. Uma delas é denominada pelo autor de *Ant-cycle* que é calculada segundo a equação:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{se a } k\text{-ésima formiga passa pela aresta } (i,j) \text{ em seu } \textit{tour} \text{ entre os tempos } (t,t+1)\\ 0 & \text{caso contrário}\end{cases} \tag{2.4}$$

em que Q é uma constante e L_k é o comprimento do caminho percorrido pela *k*-ésima formiga.

Com o que foi explanado, pode-se criar um fluxograma básico do algoritmo *ACO* para que se tenha uma visão mais clara da estratégia, conforme ilustrado na Figura 8.



Fonte: Dorigo *et al.* (1991)

Figura 8: Fluxograma do algoritmo *ACO*

Este é um fluxograma básico ilustrativo da técnica *ACO*. Dependendo do problema a ser tratado, podem ser propostas outras funções principalmente para a probabilidade de movimento das formigas e atualização do cheiro das trilhas.

2.5.4 Alguns trabalhos aplicando *ACO* na programação de produção

A metaheurística ACO, apesar de ser relativamente nova, tem sido objeto de estudo freqüente no meio acadêmico, sendo aplicada para a resolução de vários problemas de otimização. O problema da programação de produção tem sido também bastante utilizado como ambiente para a aplicação desta metaheurística.

Nesta seção serão apresentadas algumas pesquisas realizadas neste sentido, confirmando a boa qualidade dos resultados encontrados por esta técnica ao ser utilizada na resolução de problemas de programação de produção.

Chiu e Chang (2008) propõe um método de duas fases, baseado em *AC*, para a otimização de problemas de programação de um *Job Shop* Flexível. Na primeira fase, o *AC* é utilizado primeiramente para determinar em qual máquina uma determinada operação será executada. Logo após, é feito o seqüenciamento destas operações em cada máquina. Na segunda fase do método proposto, pesos diferentes para os critérios adotados para formar a função objetivo são adotados. Os critérios são os seguintes: tempo total de trabalho das máquinas, carga máxima de todas as máquinas e *makespan*. Os resultados dos experimentos realizados em três cenários diferentes e comparados com outras técnicas demonstram que o método proposto correspondeu com as expectativas dos autores, apresentando resultados satisfatórios.

Liao e Liao (2008), apresentam um algoritmo baseado em *AC*, para a otimização de um problema de programação da produção em um cenário bastante específico, denominado manufatura ágil (*agile manufacturing*). Neste sistema de manufatura, existem duas fases: usinagem e montagem. Na fase da usinagem, há uma única máquina utilizada para realizar todas as operações. Na fase da montagem, existem várias estações de montagem idênticas para a realização ou produção dos itens. Os autores fazem um comparativo da técnica baseada em *AC* com um outro algoritmo

baseado em *Branch-and-bound*. Os experimentos demonstram que, principalmente no que diz respeito a custo computacional, os resultados são melhores no algoritmo que utiliza *AC*.

Shyu *et al.* (2008) propõe uma aplicação do *ACO* para um problema de programação de produção do tipo *flows shop*, com duas máquinas. O objetivo é a composição de um programa que minimize a soma do tempo total necessário para a produção de todos os *jobs*. Os autores formulam de forma gráfica o problema de *flow shop*, e demonstram matematicamente a equivalência entre as duas formas de expor o problema. Da mesma forma, é demonstrado de que maneira o *ACO* pode ser utilizado para a resolução do problema proposto. No problema abordado, um conjunto de N *jobs* devem ser processados em 2 máquinas, ou seja, cada *job* possui duas operações. Cada operação requer um tempo de preparação, que é pré-determinada e não depende da seqüência a ser seguida. Uma série de experimentos computacionais é executada para a verificação da eficiência do algoritmo, utilizando-se para esta verificação então, a comparação com trabalhos anteriores a respeito do mesmo problema. Os resultados numéricos do trabalho mostram que a técnica adotada melhora os resultados anteriormente encontrados.

Rajendran e Ziegler (2004) tem como propósito a utilização e análise de dois algoritmos baseados em *ACO* para resolução de problemas de programação de produção, em sistemas de produção do tipo *flow shop*, utilizando-se para isto a comparação do tempo total de fluxo dos *jobs* encontrado com outros trabalhos existentes e que tratam do mesmo assunto. O primeiro algoritmo é baseado em idéias já existentes sobre a utilização da metaheurística *ACO* e o segundo algoritmo é um desenvolvimento próprio dos autores. Os autores avaliam os resultados dos algoritmos implementados, comparando seus resultados numéricos com os resultados das melhores heurísticas reportados por Liu e Reeves (2001), e Rajendran e Ziegler (2004) de 90 *benchmarks* de problemas de *flow shop*, que são extraídos de Taillard (1993). Logo após a demonstração formal do problema de *flow shop* permutacional, a metaheurística *ACO* é explanada e adaptada a resolução deste tipo de problema. Os resultados do trabalho mostram que o algoritmo baseado em ACO tem seus resultados

com uma qualidade superior em relação às heurísticas reportadas em estudos passados e que foram considerados no trabalho para comparação.

Figlali *et al.* (2007) chama a atenção para a importância de uma combinação adequada entre os parâmetros de controle do algoritmo, para que melhores qualidades de soluções sejam encontradas. Mostra que na maioria dos trabalhos, os parâmetros de controle são determinados por alguns experimentos iniciais não sistemáticos e a interação entre estes parâmetros não são levados em consideração. O estudo se propõe então a utilizar-se da técnica de delineamento de experimentos para a definição de parâmetros para um algoritmo *ACS* utilizado em problemas de programação em sistemas de produção do tipo *Job Shop*. Os efeitos e as interações dos parâmetros são interpretados com base nos resultados dos experimentos computacionais. O trabalho faz uma descrição detalhada da estratégia de funcionamento do algoritmo *ACS*, dando especial atenção aos parâmetros de controle deste algoritmo, já que este é o principal foco da pesquisa. Depois disto, um delineamento de experimentos é realizado. Os experimentos computacionais são realizados e uma análise dos dados é feita. A principal conclusão é que nenhuma interação entre os parâmetros de controle do algoritmo *ACS* trazem efeitos significantes sobre o valor do *makespan*.

Yagmahan e Yenisey (2007) argumentam que a maioria dos estudos sobre *flow shop* focam apenas na minimização do *makespan*, apesar de existir outros importantes critérios a serem otimizados. Por isso, o trabalho tem como objetivo considerar o problema de programação de *flows shop* como um problema multiobjetivo, e propõe o estudo e a análise mais especificamente da avaliação não apenas dos resultados do *makespan*, mas também do tempo total de fluxo do *jobs* e do tempo de parada das máquinas. Além disso, os autores propõem a utilização de um algoritmo baseado em *ACO* para a solução deste problema multiobjetivo de *flow shop*. Para verificação dos resultados apresentados pelo algoritmo implementado, comparações com a heurística (NEH) proposta por Nawaz, Enscore e Ham (1983) e também com a heurística (HC), proposta por Ho e Chang (1991). Os experimentos computacionais permitiram aos autores concluirem que o algoritmo criado, baseado em *ACO*, apresenta melhores

resultados que os algoritmos NEH e HC comparando-se tanto os critérios individualmente quanto quando comparado o resultado da função multiobjetivo.

Merkle e Middendorf (2000) criaram um algoritmo, baseado em *ACO*, para solução de um problema de programação de produção de uma única máquina. A diferença entre trabalhos anteriores é que há uma nova forma das formigas se guiarem pelo odor do feromônio. Neste algoritmo, o que os agentes levam em consideração é o somatório do feromônio em determinada aresta. Isto permite que elas façam um exame do feromônio utilizado por suas parceiras, o que lhes permite tomar decisões de forma mais adiantada. O trabalho traz um detalhamento de como esta estratégia é aplicada ao problema de programação de uma única máquina, realiza os experimentos computacionais, analisa seus resultados e finalmente concluí que o algoritmo desenvolvido apresente relevantes avanços em relação aos trabalhos pesquisados.

Bauer *et al.* (1999) realizaram um trabalho que tem como objetivo a aplicação da técnica ACO para a resolução de um problema de programação de produção com uma só máquina. O critério adotado para verificação da eficiência do algoritmo criado foi o de minimização do tempo total de atraso de todas as ordens. No procedimento, já adotado em outros trabalhos, os autores integram o *Modified Due Date (MDD).* Dessa forma, os *jobs* são programados de forma interativa. Assim, depois que um trabalho é programado, todos os trabalhos não-programados restantes são classificados outra vez em ordem ascendente, porém de acordo com as datas desejadas modificadas. O *job* que tiver a data desejada mais baixa modificada é adicionado à segunda seqüência gerada.

Van der Zwaan e Marques (1999) realizam uma pesquisa que tem como objetivo a aplicação do algoritmo *ACS* em um problema de programação de produção de um *job shop*. Além disto, buscam obter um ajuste apropriado dos parâmetros de controle do algoritmo, para que uma qualidade eficaz das soluções seja alcançada. Os autores separam o espaço de busca dos melhores valores para os parâmetros em dois sub-espaços. O primeiro contendo os parâmetros de transição (α e β). O segundo, contendo a constante de evaporação do feromônio e o número de formigas. Ao observar as simulações realizadas para os problemas, os autores sugerem que os parâmetros de

concentração de feromônio e a quantidade inicial desta substância, têm pouca importância no desempenho do algoritmo. Segundo os autores, os experimentos realizados com a pesquisa permitem afirmar que, para os problemas tratados e que tem estudos anteriores possibilitando assim comparações entre os resultados, os parâmetros α e β determinam a taxa de convergência para uma solução satisfatória. O parâmetro constante de evaporação também se mostrou importante, pois se bem ajustada, ela pode guiar a pesquisa para espaços de melhor solução, fazendo com que o algoritmo não acabe por apresentar soluções ótimas locais. De forma geral, o trabalho mostra que o algoritmo com os parâmetros ajustados, converge satisfatoriamente para soluções satisfatórias para o problema de *job shop*.

Lin e Lu (2004) realizam um trabalho que tem como objetivo não somente a utilização do algoritmo *ACO* para a solução de um problema de programação da produção, mas também propor a inclusão de duas novas características inspiradas em formigas reais; e assim desenvolver um novo algoritmo baseado em *ACO* que possa melhorar a qualidade das soluções. O algoritmo desenvolvido é aplicado em dois problemas de *flow shop*. No primeiro, o objetivo é minimizar a somatória do tempo total de operação de cada *job*. No segundo, o objetivo é minimizar a somatória do tempo total de operação de cada *job* e também minimizar o *makespan*. A primeira nova característica proposta pelos autores é a inclusão de um novo tipo de feromônio no cálculo da probabilidade de uma formiga ir de *i* para *j*. Este novo feromônio ditará a importância de um *job j* em relação a uma posição *l* na programação. A outra nova característica proposta pelos autores á a inclusão de um critério de dominância, semelhante ao usado em técnicas *branch-and-bound*, o que aplicado na técnica *ACO*, não permitiriam que as formigas desperdiçassem tempo em caminhos não promissores. Os resultados numéricos do trabalho mostram que o novo algoritmo *ACO* desenvolvido se mostra mais eficiente em relação ao algoritmo genérico *ACO*, principalmente em problemas de grande escala.

Pradhan e Lam (2006) têm como objetivo a comparação entre dois algoritmos, um baseado em AG e outro baseado em *ACO,* para minimização do *makespan* durante uma triagem de estresse ambiental (*environmental stress screening* - *ESS*). Esta

triagem tem como propósito estimular a ocorrência de falhas nos produtos quando submetidos a tais estresses. O problema consiste então em definir a melhor combinação dos *jobs* que cabem em uma câmara do *ESS* em um determinado tempo, para que o tempo total de processamento dos testes seja mínimo. Os resultados numéricos encontrados com os experimentos computacionais, que continham diferentes tamanhos de grupos (câmaras), mostram que ambos os algoritmos obtêm resultados próximo do ótimo em grupos de 10, 20 e 40 *jobs*. Para grupos maiores, de 80 e 100 *jobs*, os resultados mostram que o algoritmo baseado em AG tem um custo computacional menor que o algoritmo baseado em *ACO*. Contudo, este último apresenta soluções mais consistentes em termos de qualidade em relação ao primeiro.

Merkle e Middendorf (2005) têm como objetivo a aplicação de um algoritmo baseado em *ACO* para a solução de um problema permutacional de programação. O problema se refere à programação de produção em uma única máquina, em que o indicador usado para a medição da eficiência do algoritmo é a soma do total dos adiantamentos e do total dos atrasos. Para isto, os autores propõem uma mudança em relação ao *ACO* tradicional. No novo algoritmo, cada formiga determina de forma aleatória o próximo lugar a ser visitado, ou seja, em qual seqüência o *job* vai ser processado na programação. Os autores defendem que este procedimento tem como vantagem o fato de cada lugar ter a mesma probabilidade de ser escolhido primeiro para se alocar o *job*. Após a demonstração formal de como o procedimento é executado, testes e experimentos computacionais são executados, e a conclusão é que uma combinação entre o *ACO* tradicional e o novo *ACO* proposto no trabalho, trazem resultados satisfatórios em comparação a problemas em que foi aplicado apenas o *ACO* tradicional.

Ying e Liao (2003) propõem a utilização de uma abordagem fundamentada em *ACS* para a resolução de problemas de programação da produção, porém com um foco maior para a verificação da robustez da técnica. O problema tratado é o de uma máquina única em que o critério de verificação utilizado para verificação da qualidade do algoritmo é o atraso total médio dos *jobs*. Os parâmetros listados e analisados pelos autores para verificação da robustez do algoritmo implementado, e que foram testados

em experimentos computacionais foram o desvio padrão médio das soluções encontradas em relação às soluções ótimas, o desvio padrão máximo, o número de interações para encontrar a solução ótima e o tempo computacional médio. Os resultados numéricos mostram que as soluções encontradas por essa abordagem são bastante consistentes, com probabilidades pequenas de que as soluções encontradas por essa técnica sejam ruins.

Ying e Lin (2006) têm como objetivo em sua pesquisa, aplicar uma abordagem baseado em *ACS* para resolução de um problema de programação da produção. O problema considerado é o de um *flow shop* híbrido com multi-estágios e com múltiplos processadores de tarefas, ou seja, um problema em que cada estágio tem mais de um processador, sendo eles idênticos e paralelos, com a mesma função, flexibilizando então a programação da produção. O critério de verificação utilizado para as análises é o *makespan*. Os experimentos computacionais realizados permitem aos autores concluir que o algoritmo proposto, fundamentado em *ACS,* oferece significativas melhorias em termos de resultados, se comparados com outros algoritmos, fundamentados em BT e AG, presentes na literatura mais relevante.

Agrawal e Tiwari (2006) realizaram um trabalho que tem como problema a ser estudado um modelo misto estocástico de balanceamento de linha de desmontagem em forma de “U” e seqüenciamento da produção nesta linha. Para tal, os autores propõem uma abordagem colaborativa fundamentado em colônia de formigas. A característica que distingui este novo algoritmo proposto dos anteriores é a manutenção de duas colônias de formigas paralelas, que identificam suas seqüências de forma independente uma da outra, mas que utilizam a informação obtida de forma colaborativa para guiar seus passos futuros. Delineamento de experimentos e análise de variância foi aplicado aos resultados obtidos com o algoritmo implementado e mostraram que o novo algoritmo apresenta soluções consistentes e melhores do que o algoritmo tradicional *ACO.*

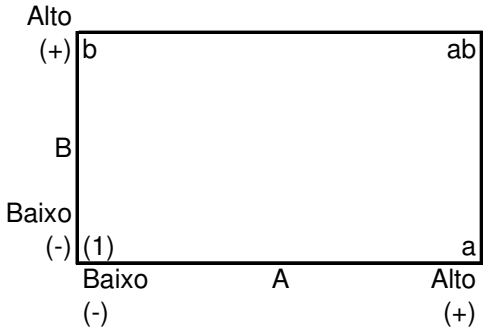
2.6 Algumas técnicas estatísticas utilizadas

Para a realização dos experimentos, bem como para a análise dos resultados obtidos, algumas técnicas estatísticas são necessárias. Uma breve revisão sobre as mesmas é realizada nesta seção.

2.6.1 O experimento 2^k

Montgomery; Runger (2004) afirmam que quando se quer avaliar vários fatores, um experimento fatorial deve ser utilizado. Num experimento fatorial todas as combinações possíveis dos níveis dos fatores são investigadas. Um caso particular dos experimentos fatoriais, bastante utilizados em trabalhos de pesquisa, é o planejamento fatorial onde k fatores são analisados com somente dois níveis. Este caso é chamado de planejamento fatorial 2^k.

O nível mais simples de um planejamento 2^k é o planejamento 2^2, que corresponde a duas variáveis controláveis, A e B, com dois níveis possíveis, nível alto denotado por "+" e nível baixo denotado por "-". Este tipo de planejamento pode ser representado geometricamente por um quadrado, mostrado na Figura 9. Tem-se então 2^2 possibilidades de corridas ou combinações de tratamento.



Tratamento	A	B
(1)	-	-
a	+	-
b	-	+
ab	+	+

Montgomery; Runger (2004)

Figura 9: Fatorial 2^2

Para representar as combinações de tratamento geralmente são utilizadas séries de letras minúsculas. Se uma letra estiver presente, o fator correspondente é corrido em nível alto naquela combinação de tratamentos. Se ela estiver ausente é corrido com nível baixo. A combinação de tratamentos com todos os fatores em nível baixo é representada por (1). Para o exemplo da Figura 9 os efeitos de interesse para o planejamento 2^2 são os efeitos principais A e B com fator de interação de segunda ordem AB. Para estimar o efeito principal do fator A, deve-se fazer a média das observações do lado direito do quadrado na Figura 9, estando a em nível alto e subtrair a média das observações do lado esquerdo do quadrado, estando A em nível baixo, conforme equação:

$$A = \bar{y}_{A+} - \bar{y}_{A-} = \frac{a+ab}{2n} - \frac{b+(1)}{2n} = \frac{1}{2n}\left[a+ab-b-(1)\right] \tag{2.5}$$

Onde n é o número de observações.

De modo análogo, o efeito principal de B é encontrado calculando a média das observações no topo do quadrado, estando B em nível alto e subtraindo a média das observações na parte inferior do quadrado, estando B em nível baixo:

$$B = \bar{y}_{B+} - \bar{y}_{B-} = \frac{b+ab}{2n} - \frac{a+(1)}{2n} = \frac{1}{2n}\left[b+ab-a-(1)\right] \tag{2.6}$$

Para calcular o valor da interação de segunda ordem AB calcula-se a diferença entre as médias das diagonais do quadrado representado na Figura 9, conforma equação:

$$AB = \frac{ab+(1)}{2n} - \frac{a+b}{2n} = \frac{1}{2n}\left[ab+(1)-a-b\right] \tag{2.7}$$

As grandezas entre colchetes nas equações 2.5, 2.6 e 2.7 são denominadas de contrastes. Assim, por exemplo, o contraste A é:

$$Contraste_A = ab - b - (1) \tag{2.8}$$

Nestas equações os coeficientes dos contrastes são sempre +1 ou -1. Para determinar o sinal de cada combinação de tratamentos para um contraste em particular utilizam-se tabelas como a do exemplo representado na Tabela 1.

Combinação de Tratamentos	Efeito Fatorial			
	I	*A*	*B*	*AB*
(1)	+	-	-	-
a	+	+	-	-
b	+	-	+	-
ab	+	+	+	+

Tabela 1: Sinais para os efeitos do planejamento 2^2

Deve-se aqui destacar o fato de a coluna de sinais AB se obtida a partir do produto das colunas A e B. Para gerar o contraste utilizando a Tabela 1, basta multiplicar os sinais na coluna apropriada pelas combinações de tratamentos e somá-las. Por exemplo, para o contraste AB:

$$Contraste_{AB} = [(1)] + [-a] - [b] + [ab] = ab + (1) - a - b \tag{2.9}$$

De acordo com Ribas (2003), a partir do contraste pode-se estimar o valor do efeito de determinado fator através da equação:

$$Efeito_p = \frac{2}{n2^{k-1}}(Contraste_p) \tag{2.10}$$

Onde:

n é o número de observações registradas;

k é o número de fatores, parâmetros, variáveis;

P é o parâmetro cujo efeito se deseja;

2.6.2 O teste F, o teste t e a análise de variância

Segundo Lapponi (2005), "O teste de hipóteses da diferença das médias de duas populações é freqüentemente utilizado para determinar se é ou não razoável concluir que as médias das duas populações são diferentes". Segundo o autor, as premissas iniciais do teste de hipóteses para diferenças entre médias podem ser apresentadas da seguinte forma:

- Há duas populações diferentes independentes, denominadas X_1 e X_2, com médias μ_1 e μ_2 e variâncias σ_1^2 e σ_2^2, sendo que ambas as populações medem a mesma variável.
- Uma amostra aleatória é extraída de cada população. As duas amostras têm tamanhos n_1 e n_2 e médias $\overline{X_1}$ e $\overline{X_2}$.
- A diferença das duas médias $\overline{X_1} - \overline{X_2}$ é uma nova variável aleatória maior do que zero se $\overline{X_1} > \overline{X_2}$, e menor do que zero se $\overline{X_1} < \overline{X_2}$.

Na distribuição de freqüências da diferença das médias $\overline{X_1} - \overline{X_2}$:

- O valor esperado, ou média, de $\overline{X_1} - \overline{X_2}$ é igual à diferença das médias das populações, $E(\overline{X_1} - \overline{X_2}) = \mu_1 - \mu_2$.
- A variância de $\overline{X_1} - \overline{X_2}$ é igual a $\sigma^2_{\overline{X_1} - \overline{X_2}} = \sigma^2_{\overline{X_1}} + \sigma^2_{\overline{X_2}}$, pois as variáveis são independentes.

As hipóteses do teste a ser aplicado têm a seguinte forma:

$H_0 : \mu_1 - \mu_2 = 0$

$H_1 : \mu_1 - \mu_2 \neq 0$

Lapponi (2005) afirma ainda que, se o número de amostras retiradas das populações é grande ($n > 30$), o teste de hipótese a ser utilizado deve ser o *teste Z*. Se o tamanho das amostras for pequeno, que é o caso deste trabalho, o teste a ser realizado deve ser o *teste t*. Porém, para que o *teste t* seja aplicado, deve-se antes verificar se as variâncias das duas amostras são diferentes ou não. Para isto, utiliza-se o *teste F*, que é um teste de hipóteses utilizado para verificar se as variâncias de duas populações com distribuição normal são diferentes, ou para verificar qual das duas populações com distribuição normal tem mais variabilidade.

De acordo com Lapponi (2005), a Análise de Variância (ANOVA) é um procedimento de teste de hipótese utilizado para comparar as médias de mais de dois tratamentos ou populações.

Ainda segundo Lapponi (2005), o objetivo da análise de variância é avaliar se as diferenças observadas entre as médias das amostras são estatisticamente diferentes. Para isto, a variabilidade total das amostras é dividida em duas partes ou fontes de variabilidade, em que a primeira parte dessa variabilidade é proveniente das populações serem diferentes, chamada variabilidade *entre*. Quanto maior for à variabilidade *entre*, mais forte é a evidência de as médias das populações serem diferentes. A segunda parte de variabilidade é causada pelas diferenças dentro de cada amostra, denominada variabilidade *dentro*. Quanto maior for a variabilidade *dentro*, maior será a dificuldade para concluir se as médias das populações são diferentes.

O teste de hipótese da análise de vari6ancia é estabelecido como:

- A hipótese nula H_0 afirma que as k populações tem a mesma média.
- A hipótese alternativa H_1 afirma que nem todas as médias das k populações são iguais.

A distribuição *F* conduzirá a decisão de aceitar ou rejeitar a hipótese nula, comparando o *F* observado (F_0), calculado com a expressão:

$$F_0 = \frac{Variância\_entre}{Variância\_dentro} = \frac{S_b^2}{S_w^2} \qquad (2.11)$$

com o *F* crítico (F_c), correspondente ao nível de significância α adotado.

O F_0 mede a variabilidade entre por unidade de variabilidade dentro, ou quantas vezes a variabilidade das médias das amostras é maior do que a variabilidade amostral. Se $F_0 > F$c, rejeita-se a hipótese de nulidade H_0, ou seja, existem evidências de diferença significativa entre pelo menos um par de médias das populações, ao nível α de significância escolhido. Caso contrário, não se rejeita a hipótese de nulidade H_0, ou seja, não há evidências de diferença significativa entre as populações ao nível α de significância escolhido.

3 IMPLEMENTANDO ACO NA PROGRAMAÇÃO DA PRODUÇÃO

A explanação sobre a metodologia utilizada no trabalho, que foi realizada na seção 1.2, afirmava que se deve fazer uma escolha adequada da variável de resposta, de modo a garantir a objetividade da análise dos resultados. Com os conceitos relativos à definição formal do que é um problema de programação de produção, da metodologia de funcionamento da metaheurística *ACO* e também de como esta técnica tem sido aplicada para a resolução de problemas de programação de produção, pode-se então partir para um maior entendimento e aprofundamento sobre o problema específico desta pesquisa, para que este objetivo seja atendido. Assim, a presente seção tem o objetivo de demonstrar como a metaheurística *ACO* será estruturada para sua utilização no problema estudado no trabalho; formular um pseudocódigo para o algoritmo e também explanar brevemente o funcionamento do *software* utilizado para a realização dos experimentos, e que se utiliza da inteligência da metaheurística *ACO.*

3.1 Estrutura básica do ACO aplicado ao problema de programação da produção para trás

Com base em Dorigo *et al.* (1996), Ventresca e Ombuki (2004) e Mazzucco Jr. (1999), a representação do problema de programação de produção pode ser feita através de um grafo disjuntivo. Este grafo pode ser definido como $Q = (V, A, E)$, onde V é o conjunto dos vértices do grafo, que corresponde ao conjunto das operações a

serem programadas, e que aqui é representado por O. São adicionadas ainda ao conjunto V duas operações fictícias simbolizadas com 0 e $N+1$, ou seja $V=\{0,O,N+1\}$.

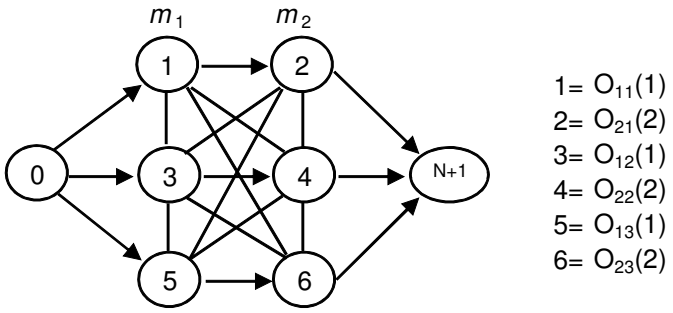
O conjunto A é formado pelos arcos que conectam operações consecutivas do mesmo *job*, os arcos que conectam a operação 0 à primeira operação real de cada *job* e a última operação de cada um deles à operação $N+1$. O conjunto A pode ser então expresso da seguinte maneira (MAZZUCCO Jr.,1999),

$$A=\{(v,w)/\,v,w\in O, v\rightarrow w\}\cup\{(0,w)/\,w\in O \text{ e não existe } v\in O/\,v\rightarrow w\ \}\cup$$
$$\{(v,N+1)/\,v\in O \text{ e não existe } w\in O/\,v\rightarrow w\ \}$$

Finalmente, o conjunto E é formado pelas arestas que conectam duas operações a serem realizadas pela mesma máquina, que pode ser expresso como $E=\{\{v,w\}/\,M_v=M_w\}$, (MAZZUCCO Jr.,1999).

Cada arco é possuidor de um par de números $\{\tau_{ij},\eta_{ij}\}$, que são respectivamente a concentração de feromônio e a visibilidade. Este último podendo ser considerado como o tempo de processamento da operação no nó *j*.

A Figura 10 é um grafo representativo de um problema de programação de produção. Nele, as operações numeradas com 1, 3 e 5 são executadas por uma mesma máquina m_1. Cada uma dessas operações pertence a um determinado *job*, J_1, J_2 e J_3 respectivamente. Da mesma forma, pode-se afirmar que as operações 2, 4 e 6 são executadas pela mesma máquina m_2 e pertencem aos *jobs* J_1, J_2 e J_3 respectivamente. As operações inicial e final, rotuladas em 0 e *N+1,* respectivamente são fictícias, ou seja, não são executadas, porém encontram-se na seqüência de operações de todos os *jobs*. Elas existem apenas para que o grafo, por ser orientado, tenha uma operação inicial e final, mas como não possuem um tempo de processamento, não influenciam na programação.



Fonte: Dorigo *et al*. (1996)

Figura 10: Grafo *Ant System* para 3 *jobs* e 2 máquinas

Os nós numeradas de 1 a 6 representam as operações a serem agendadas. Cada operação desta possui um número índice, que representa a operação e a que *job* ésta pertence. O número entre parênteses ao lado diz em qual máquina a operação tem que ser processada. O vértice 1 por exemplo, corresponde a operação $O_{11}(1)$. Isto quer dizer que esta é a operação 1 (primeira operação) do *job* 1 e deve ser processado na máquina 1. O vértice 2 corresponde a operação $O_{21}(2)$, e isto quer dizer que esta é a operação 2 (segunda operação) pertence ao *job* 1 e que deve ser processado na máquina 2. E isto acontece com todos os vértices, que correspondem cada qual a uma determinada operação,.

Um conjunto de orientações de todas as arestas do grafo da figura 10 transforma-o em um grafo orientado, que representa uma das soluções possíveis do problema modelado. Da mesma forma, um conjunto de orientações de todas as arestas do grafo, define, para cada máquina em M, uma ordenação ou permutação das operações por ela processadas.

No exemplo da Figura 10, supondo-se que uma solução possível para o problema ilustrado seja $\pi = (0,1,4,2,5,3,6,N+1)$. Isso quer dizer que na máquina 1 (m_1), serão processadas nesta ordem os *jobs* J_1, J_3 *e* J_2. E na máquina 2 (m_2) serão processados nesta ordem, os *jobs* J_2, J_1 *e* J_3.

Na utilização do *ACO* para a resolução de um problema de programação de produção, o que se busca de forma resumida, é que um conjunto de agentes, no caso formigas, saia do seu ninho, aqui representado pelo vértice inicial, e vá até a fonte de

comida, aqui representada pelo vértice final do grafo, passando por todos os vértices, que representa no caso da programação de produção, agendar todas as operações.

No início do processo, cada formiga é equipada com 3 listas: G, que contém os nós que não foram visitados. S, que contém os nós que podem ser visitados na próxima interação; e *tabu*, contendo os nós já visitados. Inicialmente, todas as formigas se encontram na origem do grafo e possuem em suas listas assim:

$G_k = \{O_{11}, O_{22}, O_{31}, O_{12}, O_{21}, O_{31}\}$
$S_k = \{O_{11}, O_{22,} O_{31}\}$
$tabu_k = \{\}$

O processo fica em interação até que $G_k = Ø$. No final, a ordem dos nós percorridos é dado pela seqüência da lista *tabu*, e é a solução proposta pela formiga *k*.

3.2 Modelagem ACO para o problema proposto

O problema considerado nesta pesquisa consiste na otimização da programação de um sistema produtivo composto de apenas um estágio de processamento, recursos paralelos e roteiros flexíveis.

Afirmar que determinado sistema produtivo possui somente um estágio de processamento significa dizer que uma unidade de qualquer produto considerado terá apenas uma operação, ou seja, passará por somente um estágio de processamento, ou mesmo, que passará por apenas uma máquina. Dito de outra forma, cada *job J* é formado por apenas uma operação, sendo $J_j = \{o_{1j}\}$. Há então um conjunto de *N jobs*, onde $J = \{J_1, J_2, ..., J_N\}$ a ser programados.

O sistema produtivo tratado aqui é caracterizado também por possuir recursos paralelos, ou seja, o único estágio do processo produtivo é composto por mais de uma máquina ou recursos produtivos, de forma mais geral. Há um conjunto de *M* máquinas ou recursos onde $M = \{m_1, m_2, ..., m_M\}$. Cabe acrescentar ainda que as máquinas são diferentes, apresentando uma velocidade de produção diferente para o mesmo produto.

Outra característica importante do sistema produtivo considerado, e que é totalmente complementar às duas características citadas anteriormente, é que os produtos possuem roteiros flexíveis. Isso quer dizer que um determinado produto possui mais de um roteiro a ser seguido. Cada *job J* pode ser processado em qualquer uma das *M* máquinas, ou seja, cada *job J* possui um roteiro flexível. Esquematicamente, $J_1 = \{RF_1\}$, $J_2 = \{RF_2\}$ e $J_N = \{RF_N\}$, onde RF_1 é o roteiro flexível do *job* 1, RF_2 é o roteiro flexível do *job* 2, e RF_N é o roteiro flexível do *job* N. Cada roteiro flexível é composto por um conjunto de operações similares. Esquematicamente, $RF_1 = \{o_{11}(1), o_{11}(2), o_{11}(M),\}$, $RF_2 = \{o_{12}(1), o_{12}(2), o_{12}(M),\}$, e $RF_N = \{o_{1N}(1), o_{1N}(2), o_{1N}(M),\}$, onde $o_{11}(1)$ representa a operação 1 (única) do *job* 1 na máquina 1, $o_{12}(1)$ representa a operação 1 (única) do *job* 2 na máquina 1, e $o_{1N}(1)$ representa a operação 1 (única) do *job* N na máquina 1.

Adicionalmente, o problema proposto possui um sistema de programação de produção *backward*, ou para trás. Neste sentido, cada *job J* possui uma data que precisa estar disponível no estoque, e esta data será sempre respeitada. O problema consiste então em ordenar todas as operações a serem executadas de forma a minimizar o tempo total para a execução de todas as operações e ainda, respeitar a data limite de entrega de todos os *jobs*.

O tempo de preparação da máquina, ou tempo de *setup*, será considerado como fixo para todos os produtos independentemente da máquina na qual o mesmo será produzido. Isto equivale a dizer que este tempo não influência a qualidade da programação, já que é o mesmo independente da seqüência da programação.

E como característica final do problema, deve-se citar que o indicador responsável por mostrar o quão satisfatória é a solução do programa de produção gerado será o *makespan*, que representa o comprimento do programa de produção. Dizendo de outra forma, é o tempo de fim do último *job* programado menos o tempo de início do primeiro *job* programado. Outra forma de ser calculado o *makespan* é através do tempo de fim da última operação a ser processada na última máquina a terminar sua seqüência menos o tempo de início da primeira máquina a começar sua seqüência.

De forma resumida, o problema proposto neste trabalho consiste em definir a programação de produção em um sistema de produção com um estágio de

processamento, máquinas paralelas e diferentes, produtos com roteiros flexíveis, programação *backward*, tempos de *setup* fixos e que terá como indicador de qualidade da programação gerada o *makespan*.

O problema proposto é mostrado esquematicamente na Figura 11.

Como exemplo de casos reais em que o sistema de produção proposto se enquadra, têm-se as fábricas de resinas com mais de um reator, fábricas de papel decorativo, onde se têm várias linhas de impressão, plantas de produção de painéis de madeira, em que se pode encontrar mais de uma linha de produção e linhas de revestimento de painéis de madeira.

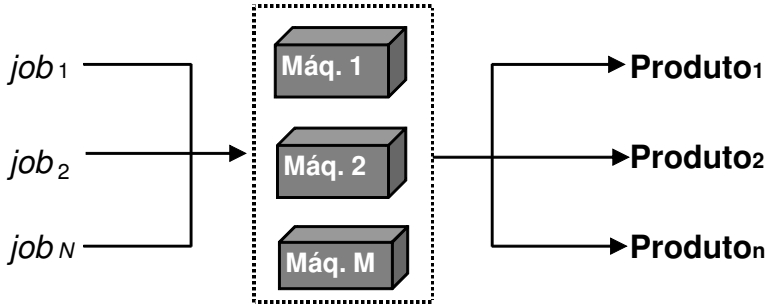


Figura 11: Problema de programação de produção com um estágio de processamento, recursos paralelos e roteiros flexíveis

3.3 O pseudocódigo *ACO* para o problema proposto

O processo de criação de um programa de produção, que parte da necessidade de execução de determinadas operações em certo número de recursos (máquinas ou linhas de produção) exige, principalmente em casos complexos, a utilização de um *software*, capaz de realizar tal tarefa. Este *software*, configurado para atender tipos específicos de sistemas de produção, utiliza-se de determinada técnica ou inteligência para a realização desta.

Para que a inteligência seja utilizada para a resolução do problema de programação tratado neste trabalho, algumas considerações precisam ser feitas. A primeira é que o problema de programação da produção para ser resolvido com a metaheurística *ACO* precisa ser transformado em um grafo. A Figura 12 traz uma representação em forma de grafo do problema proposto. Este grafo contém os nós inicial e final, que para temos de programação de produção são fictícios. Todos os outros nós correspondem a operações a ser programadas. As arestas correspondem ao tempo ou duração da operação a ser agendada, sendo esta a operação que se encontra no final da aresta. As operações estão divididas em grupos, que representam o roteiro flexível de cada *job*. Não existem arestas ligando as operações do mesmo grupo, ou seja, do mesmo *job*, pois estas são operações similares, o que significa dizer que quando qualquer uma dessas operações for programada, as outras deste mesmo grupo serão eliminadas. Porém, existem as arestas não direcionadas que ligam os grupos de operações, que indicam que todos os *jobs* precisam ser programados, mas que não existe nenhuma restrição quanto a este seqüênciamento.

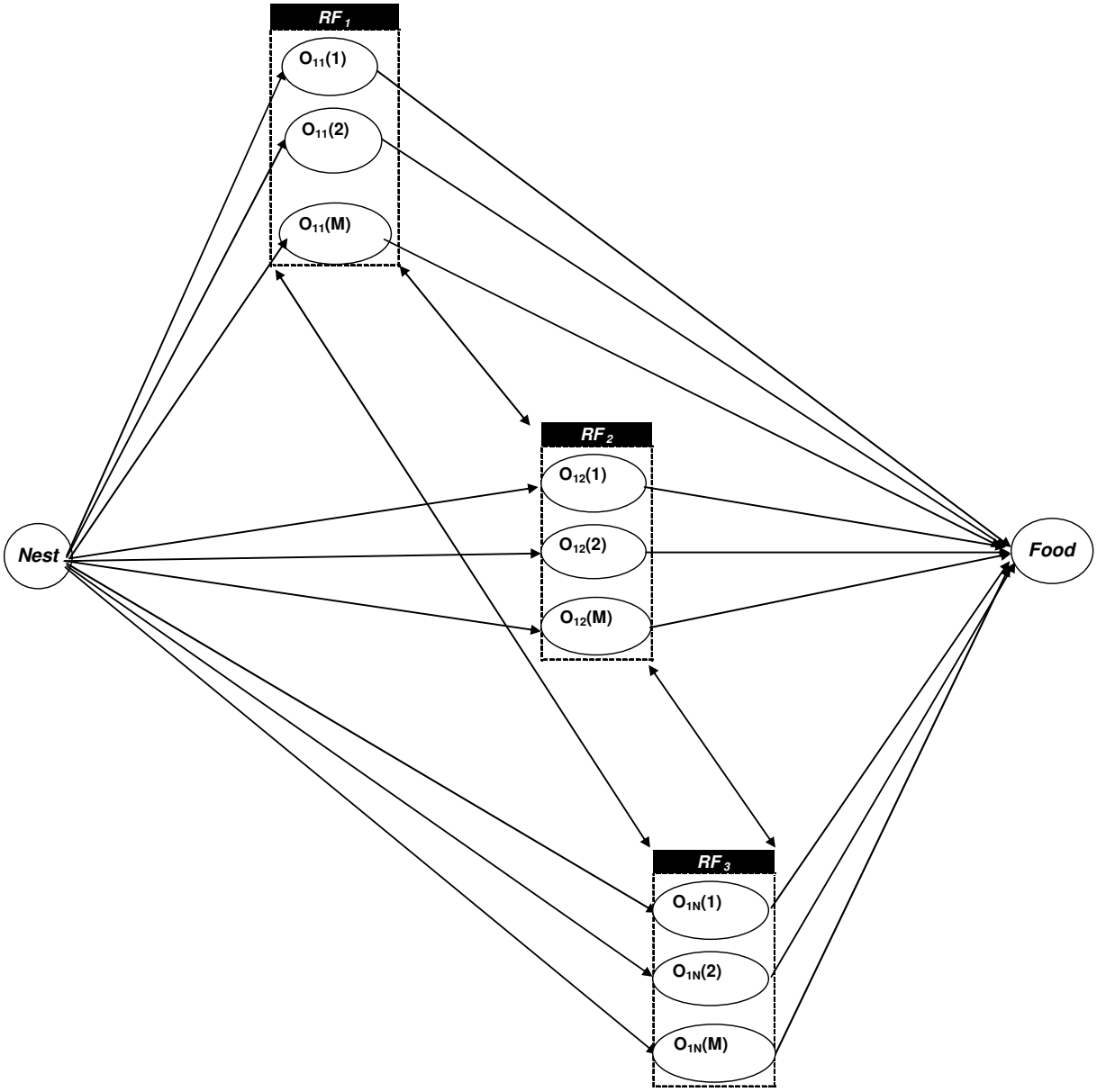


Figura 12: Grafo de um problema de programação de produção com um estágio de processamento, recursos paralelos e roteiros flexíveis

No problema proposto, cada *job* possui um roteiro flexível, o que quer dizer que o roteiro de cada *job* permite que o mesmo seja processado em qualquer um dos recursos existentes. Cada recurso corresponde a um roteiro para o *job* em questão. No

caso particular deste trabalho, cada roteiro é composto de apenas uma operação, ou seja, uma operação é equivalente a um roteiro.

Dessa forma, pode-se dizer que cada roteiro flexível possui um conjunto de operações similares, ou seja, se uma destas operações for á escolhida, as outras deste mesmo roteiro flexível são eliminadas. Quando determinado *job* é designado a ser processado em determinado recurso, o mesmo foi designado a seguir este roteiro. As outras operações referentes aos demais roteiros que o *job* poderia ter seguido são eliminadas.

Supondo um cenário composto de 6 *jobs* (J_1, J_2, J_3, J_4, J_5, J_6) e três máquinas (m_1, m_2 e m_3). O sistema é formado por apenas um estágio de processamento, o que significa dizer que cada *job* deverá passar por apenas uma das máquinas. Então, o J_1, por exemplo, possui o roteiro flexível (RF_1), que é composto por três operações similares. Pode seguir pelo roteiro 1 (equivalente a m_1), pode seguir pelo roteiro 2 (equivalente a m_2), ou pode seguir pelo roteiro 3 (equivalente a m_3). Esquematicamente: $J_1 = \{O_{11}(1), O_{11}(2), O_{11}(3)\}$, onde $O_{11}(1)$ representa a operação 1 do *job* 1 na máquina 1, $O_{11}(2)$ a operação 1 do *job* 1 na máquina 2, e $O_{11}(3)$ a operação 1 do *job* 1 na máquina 3, lembrando que quando o J_1 é designado a ser processado em qualquer uma destas máquinas, ou seja, "escolhe" qualquer uma destas três operações, as outras duas são eliminadas. A lógica serve para todos os *jobs*.

Modelando o problema em forma de grafo, tem-se que cada *job* (J_n) possui um roteiro flexível (RF_N). Cada roteiro deste *job* corresponde a uma operação deste job e é representado por um nó no. Estes nós são "virtuais" pois somente serão confirmados quando o *job* é designado para determinada máquina, ou seja, escolhe um destes nós, sendo os outros pertencentes ao mesmo roteiro flexível eliminados.

As formigas saem então do ninho (*Nest*) com o objetivo de chegar à comida (*Food*). Porém, antes disto, todos os *jobs* precisam ser programados, o que significa dizer que uma formiga ao sair no *Nest*, antes de chegar ao *Food,* tem que agendar pelo menos uma operação de cada roteiro flexível.

Uma instância deste problema é mostrada a seguir juntamente com uma das possíveis soluções. Os dados do problema são mostrados na Tabela 2.

	Quantidade (unidades)	Velocidade Máquina 1 (unidades/hora)	Máquina 2 (unidades/hora)	Máquina 3 (unidades/hora)
Job 1	610	10	17	21
Job 2	990	16	19	11
Job 3	1580	18	13	7
Job 4	350	12	14	18
Job 5	480	13	9	19
Job 6	730	15	11	16

Tabela 2: Para uma instância do problema de programação de produção

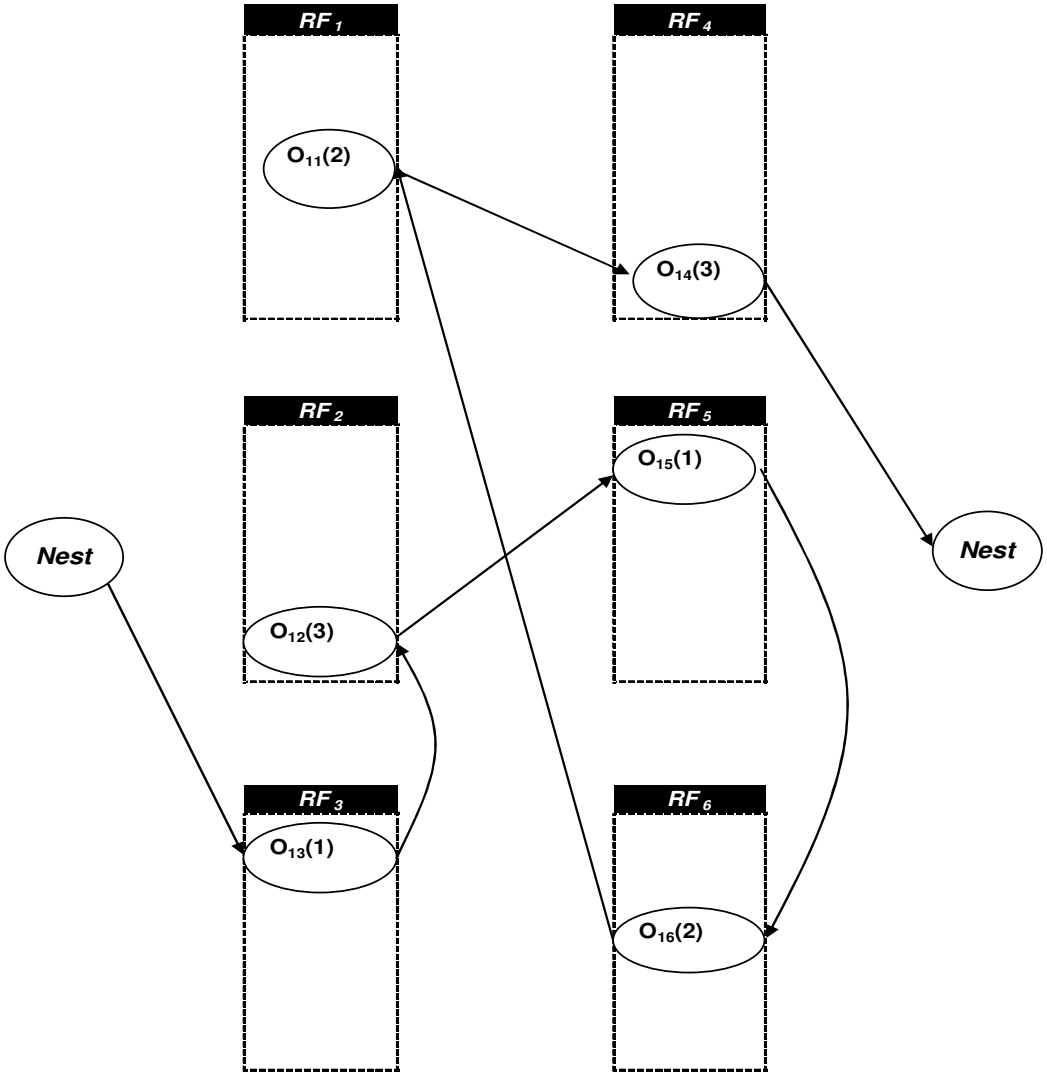


Figura 13: Grafo solução problema de programação de produção

Parte-se da premissa de que é necessário programar 6 *jobs* (OPs) em 3 máquinas com velocidades de produção diferentes. Cada *job* é formado por uma quantidade de determinado produto. Como os tempos de *setups* são fixos, pode-se considerar que os mesmos já estão inclusos nas velocidades de produção. O objetivo é minimizar-se o *makespan*.

Supondo que a formiga que encontrou a melhor solução para o problema proposto seja a que fez o caminho indicado no grafo da Figura 13 {*Nest*, $O_{13}(1)$, $O_{12}(3)$, $O_{15}(1)$, $O_{16}(2)$, $O_{11}(2)$, $O_{14}(3)$, *Food*}. Isto que dizer que esta formiga agendou primeiramente a operação do *job* 3 na máquina 1, a operação do *job* 2 na máquina 3, a operação do *job* 5 na máquina 1, a operação do *job* 6 na máquina 2, a operação do *job* 1 na máquina 2 e finalmente a operação do *job* 4 na máquina 3. O *makespan* para esta solução seria 5,7 dias, conforme mostrado na Figura 14.

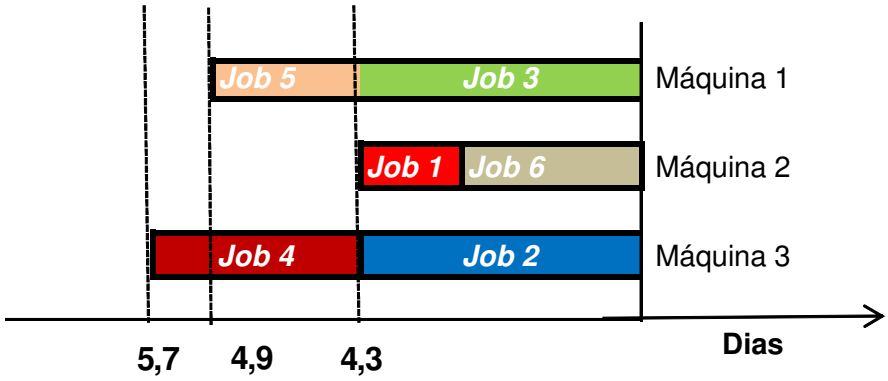


Figura 14: Possível solução para a instância do problema de programação de produção.

O exemplo mostrado anteriormente é muito simples e poderia ter sido feito até sem a utilização de uma inteligência mais avançada. Porém, o mesmo serve apenas como exercício para o entendimento do problema, que tem na solução a seguinte explicação: na máquina 1 serão produzidos os *jobs* 3 e 5. O *job* 3 é programado primeiro, lembrando que a programação é *backward*, porque a formiga passa primeiro

pela operação do RF_3, correspondente ao *job* 3, e só depois passa pela operação do RF_5, correspondente ao *job* 5.

Esta lógica está traduzida em um pseudo-código para o algoritmo da metaheurística *ACO* utilizada, que é mostrado no Quadro 2. Basicamente este pseudo-código consiste em um procedimento geral que mantém o algoritmo em execução até que critérios de parada sejam satisfeitos. Esse procedimento geral contem procedimentos específicos para cada função do algoritmo.

O primeiro procedimento específico é responsável por criar o sistema, o que consiste de forma geral a ler os dados de entrada, correspondentes as variáveis de configuração do sistema.

O segundo procedimento específico é o responsável pela movimentação das formigas e é onde se encontra a essência da inteligência da metaheurística *ACO*. Isto porque na decisão de cada formiga sobre qual operação deve ser a próxima a ser programada se encontra o ponto fundamental que vai determinar se o programa de produção tem qualidade satisfatória ou não. Como esta decisão leva em consideração o tempo de processamento da operação a ser programada e a quantidade de feromônio existente na aresta que liga a operação que acaba de ser programada pela formiga e a próxima operação a ser programada, este procedimento se configura num procedimento de fundamental importância para a eficiência do algoritmo.

O terceiro procedimento específico do algoritmo é o que analisa a solução encontrada por cada formiga e determina a quantidade de feromônio que deve ser depositado no caminho percorrido por cada uma. Como isto vai influenciar na probabilidade de outras formigas também seguirem por este caminho, este procedimento também se mostra determinante para a qualidade das respostas geradas.

O último procedimento específico do algoritmo corresponde ao momento em que a cada viagem que chega ao fim de cada formiga, o feromônio do sistema como um todo é evaporado em certa proporção.

Cada destacar ainda que cada um destes procedimentos são formados por procedimentos internos a cada um deles e que aqui são demonstrados apenas de forma geral para que fique entendida a lógica de funcionamento do algoritmo.

```
procedimento Ant_Colony_Optimization
        procedimento cria_sistema; (dados de entrada)
        fim procedimento
                enquanto critério_de_término_não_satisfeito (Formigas e viagens)
                        agendar_operações
                                movimenta_formigas;
                                analisar_solução;
                                evapora_feromônio_sistema;
                        fim agendar_operações
                fim enquanto
        procedimento movimenta_formigas
                enquanto Lista_nós_factíveis ≠ Ø
                        para k de 1 até W faça
                                movimenta_k_de_i_para_j  (com probabilidade P^k_ij )
                                tira_ o_1j(m)_de_ Lista_nós_factíveis
                                coloca_ o_1j(m)_em_Lista_tabu_k
                        fim para
                fim enquanto
        fim procedimento
        procedimento analisar_solução
                para k de 1 até W faça
                        calcular_custo_solução_corrente_através_tabu_k
                        se solução_corrente = melhor_solução
                                convergência ++;
                                se convergência > 20
                                        feromônio_todas_arestas = 1;
                                        convergência = 0;
                                fim se
                        fim se
                        se solução_corrente < melhor_solução
                                melhor_solução = solução_corrente;
                                Atualiza_feromônio_arestas_solução_corrente; (importância)
                        fim se
                        se solução_corrente < 1,05 * melhor_solução
                                Atualiza_feromônio_arestas_solução_corrente; (0,95*importância)
                        fim se
                fim para
        fim procedimento
        procedimento evapora_feromônio_sistema
                se Lista_nós_factíveis = Ø
                        Atualiza_feromônio_sistema_iminuindo_evaporação
                fim se
        fim procedimento
fim procedimento
```

Quadro 2: Algoritmo *ACO* para o problema proposto

Foi com esta estrutura que foi criado o *software* utilizado para as análises dos resultados do presente trabalho.

3.4 Estrutura do software implementado

O *software* implementado tem como objetivo testar de forma mais real, utilizando-se para isto simulações de problemas de programação de produção, a eficiência da metaheurística ACO mais especificamente em um ambiente composto de um sistema de produção com um estágio de processamento, máquinas paralelas e com velocidades diferentes, produtos com roteiros flexíveis, programação *backward*, tempos de *setup* fixos e que tem como indicador de qualidade da programação gerada o *makespan.*

Para cada cenário a ser simulado, há que se fazer uma série de configurações do *software.* Primeiramente realiza-se o cadastro dos produtos, recursos, roteiros e ordens das ordens de produção a serem programadas. A partir daí, os valores das variáveis controláveis do processo são inseridos e o algoritmo pode ser executado.

Para que haja um entendimento sobre as variáveis controláveis do algoritmo e principalmente o que elas significam, uma revisão da estratégia de funcionamento da metaheurística *ACO* deve ser feita.

Uma determinada quantidade de formigas sai do seu ninho com destino a fonte de comida. Existem vários caminhos a ser seguidos por estas formigas, e aqui se supõe que em todos estes caminhos já existe uma quantidade inicial de feromônio, substância esta utilizada pelas formigas para se comunicarem a respeito do menor caminho a ser seguido. Cada formiga fará um determinado número de viagens do ninho até a fonte de comida. Em cada uma destas viagens, as formigas depositam no caminho que acabaram de realizar uma determinada quantidade de feromônio. Esta quantidade será uma quantidade padrão caso o caminho percorrido pela formiga não apresente melhorias em relação ao melhor já existente até o momento ou receberá uma quantidade maior de feromônio caso o caminho percorrido pela formiga em questão seja mais curto do que o existente até então. Além disso, há que se adicionar ainda que

a quantidade de feromônio existente em todos os caminhos é constantemente diminuída, devido à evaporação da substância.

Com isso em mente, podem-se definir então quais serão as variáveis controláveis do algoritmo, que estão sendo consideradas neste trabalho. São elas:

- ➔ Valorização da melhor resposta (VMR). Esta variável indica por qual valor a quantidade padrão de feromônio depositada pela formiga deve ser multiplicada e posteriormente adicionada ao caminho percorrido por esta formiga, caso este caminho encontrado seja mais curto do que o existente até o momento.
- ➔ Número de formigas (NF). É a quantidade de formigas que farão o trajeto do ninho até a fonte de comida.
- ➔ Quantidade de feromônio inicial (QPI). É a quantidade de feromônio que já existe em todos os caminhos antes que as formigas comecem a realizar suas viagens.
- ➔ Quantidade adicionada de feromônio (QAP). É a quantidade padrão de feromônio que cada formiga depositará no caminho que acaba de percorrer cada vez que a mesma chegue na fonte de comida.
- ➔ Número de viagens de cada formiga (NVF). É o número de vezes que cada formiga fará o trajeto entre o ninho e a fonte de comida.
- ➔ Percentual de evaporação (EV). É a quantidade de feromônio que é perdida por cada caminho á medida que o tempo passa.

Após a inserção destes dados, o *software* pode ser executado.

A lógica básica de funcionamento do *software* pode ser verificada na Figura 15, que consiste de forma resumida a ler os dados de entrada, criar o sistema, o que consiste em criar as formigas com suas respectivas listas *tabus* e lista de nós factíveis, e criar os grafos para cada formiga, com seus respectivos nós e arestas. Enquanto o critério de término, que está baseado no número de formigas e o número de viagens que cada formiga deve fazer, não é atingido, a parte mais importante do algoritmo é realizada - a programação das operações - e que tem sua lógica descrita de modo resumido na Figura 16.

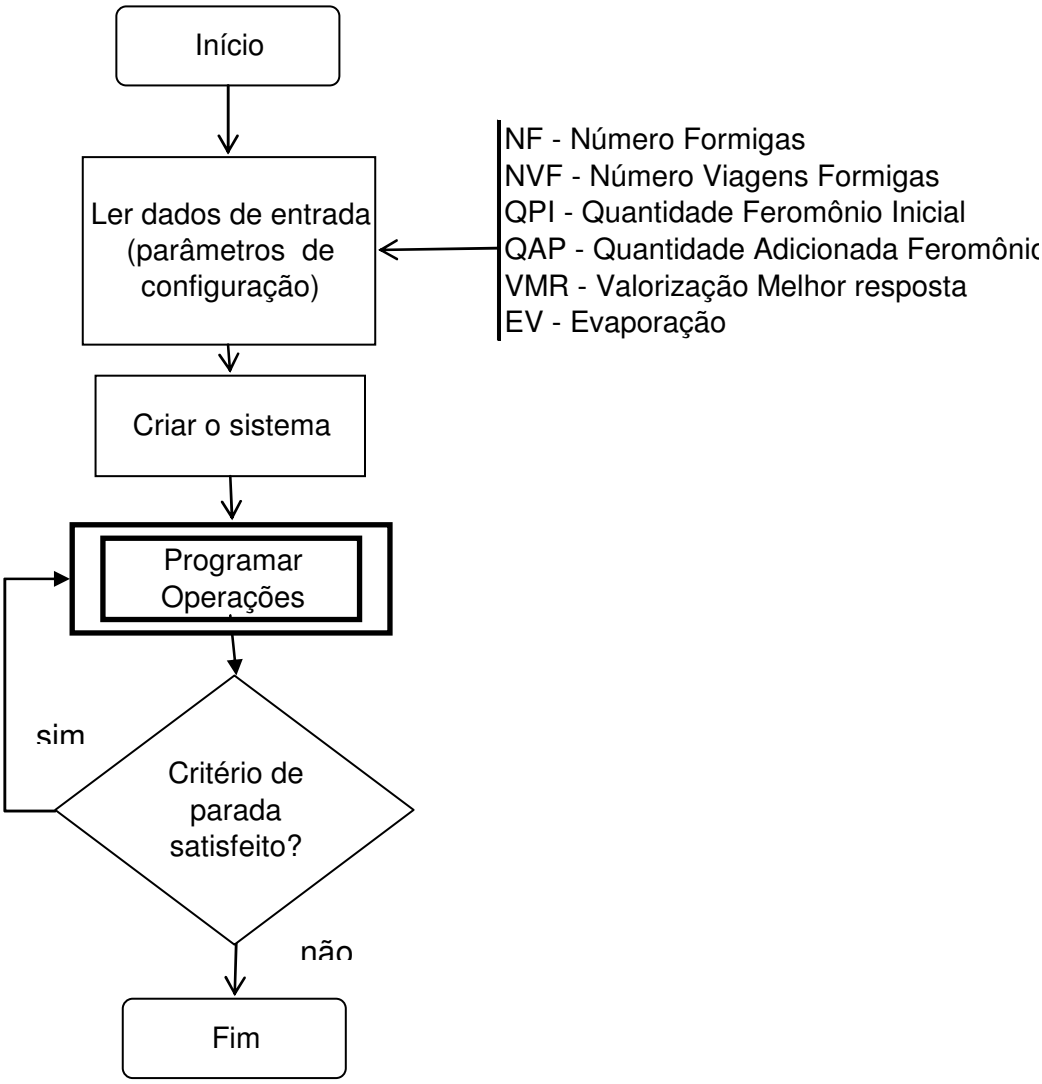


Figura 15: Fluxograma geral do *software*

O procedimento de programar operações é onde a inteligência do algoritmo é colocada em prática, ao por em interação todos os dados de entrada do sistema.

Basicamente o que se está propondo com a técnica *ACO* é utilizar a inteligência de uma colônia de formigas, que conseguem encontrar caminhos curtos entre seu ninho e a fonte de comida através de comunicação entre os agentes. O objetivo principal ao utilizar esta técnica para a resolução de problemas de programação de produção é utilizar esta comunicação a fim de alocar cada ordem de produção em determinada máquina e definir a seqüência de fabricação dessas ordens, para que o

tempo total para a conclusão do programa de produção gerado seja mínimo. A lógica consiste então em cada formiga sair do ninho e ir programando todas as ordens de produção do sistema até chegar à comida. Cada ordem de produção é escolhida para ser a próxima a ser agendada com uma probabilidade, a qual depende do tempo de processamento desta ordem e da informação quanto a quantidade de feromônio existente na aresta que liga a operação que acaba de ser agendada pela formiga e a próxima que a mesma vai agendar. Esta informação depende tanto da quantidade de feromônio depositada por outras formigas que agendaram esta operação e também de quanto feromônio já foi evaporado nesta aresta.

Como a quantidade de feromônio depositada no caminho que a formiga percorreu só ocorre quando a mesma chega na comida, o que equivale dizer que só ocorre quando a formiga tem uma solução para o problema, verifica-se que na primeira viagem de todas as formigas a informação quanto a concentração de feromônio nas arestas não vai interferir nas decisões das formigas, já que a quantidade inicial de feromônio é a mesma em todos as arestas.

Quando todas as formigas chegam à comida, uma certa quantidade de feromônio será depositada em seu caminho, e obedece a critérios que dependem da solução encontrada. Esta lógica está mostrada na Figura 17. Quando todas as formigas tiverem completado todas as suas viagens, o algoritmo chegou ao seu final.

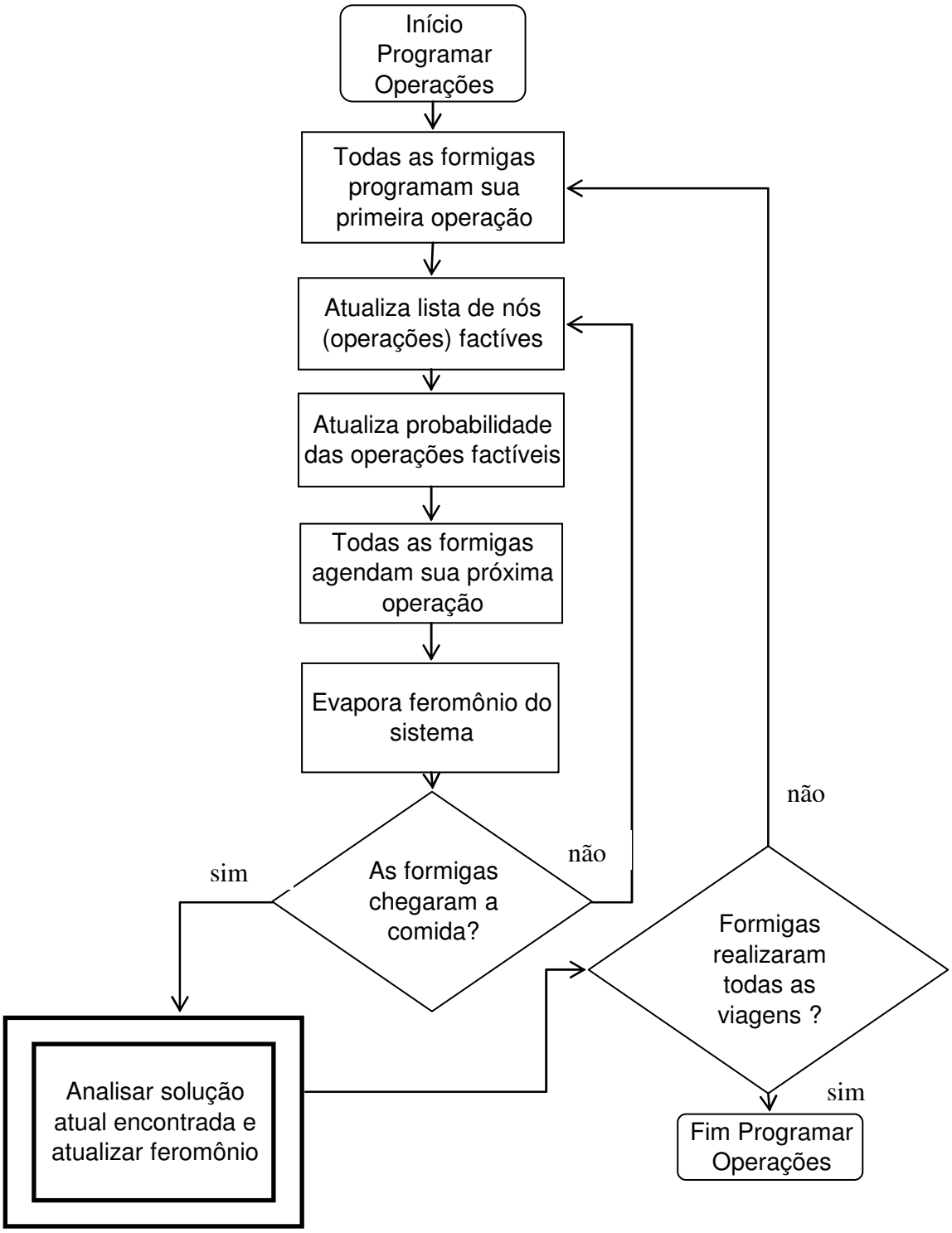


Figura 16: Fluxograma do processo de programação das operações

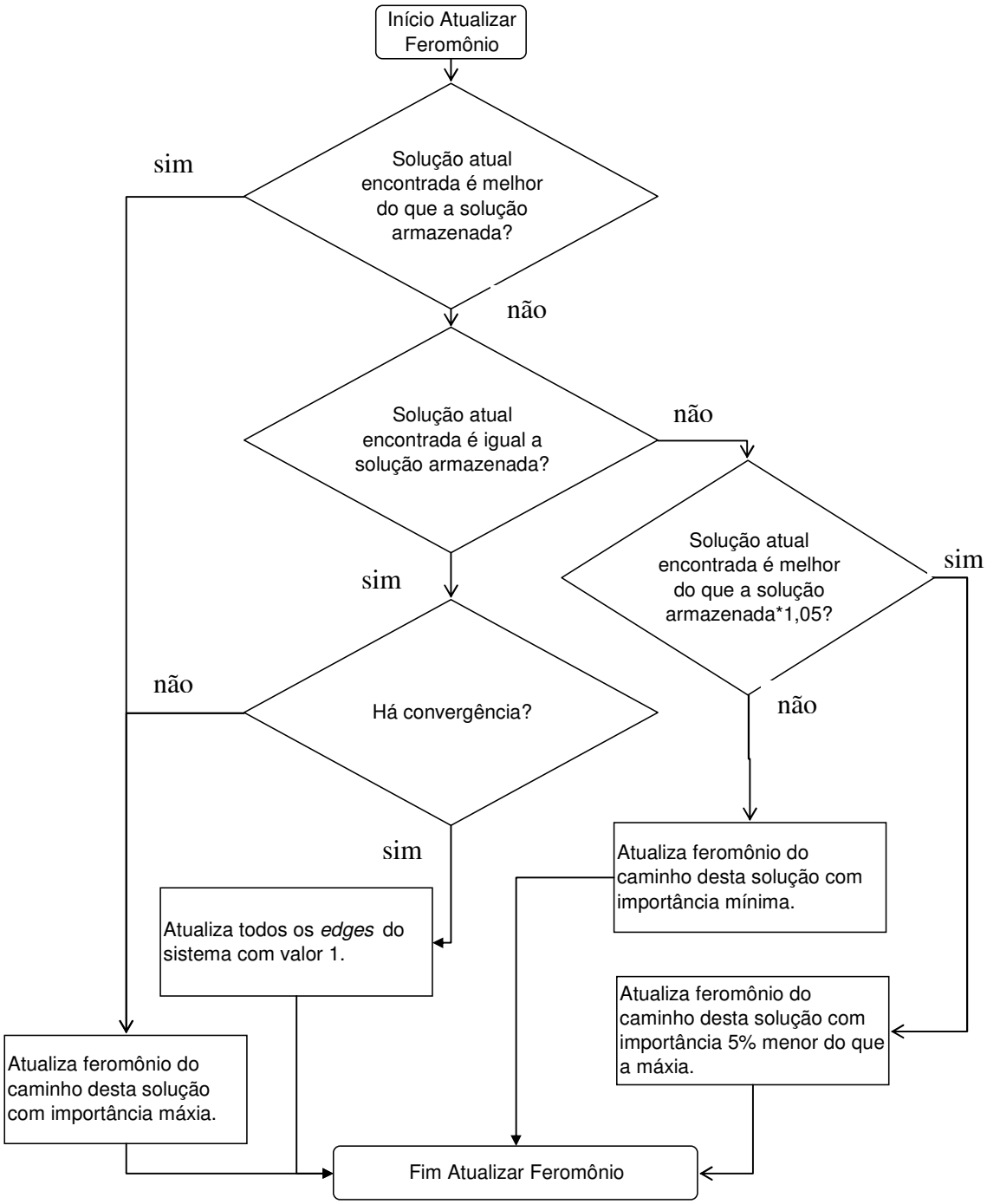


Figura 17: Fluxograma do processo de atualização de feromônio

Como já foi dito, quando cada formiga chega ao ninho, a mesma deposita uma quantidade de feromônio no caminho que realizou. Esta quantidade de feromônio vai depender única e exclusivamente da qualidade da solução obtida. O que se quer com

isto é fazer com que as formigas que tenha conseguido caminhos menores, ou qualidades melhores de soluções, passem as informações relativas a este caminho para ajudar as outras formigas. Isto é feito com um depósito adicional de feromônio neste caminho, o que aumenta a probabilidade das outras formigas também optarem por este caminho.

Em relação ao critério de quanto feromônio depositar, a lógica depende de dois dados de entrada, a valorização da melhor resposta e quantidade de feromônio depositada. Se a solução encontrada é melhor ou igual a melhor solução até o momento, a o depósito de feromônio no caminho será o produto das duas variáveis de entrada. Se for apenas 5% pior do que a melhor resposta até o momento, terá um depósito de feromônio neste caminho 5% menor do que a do caso anterior. E finalmente se a resposta for mais do que 5% pior do que a melhor resposta até o momento, este caminho receberá uma quantidade mínima de feromônio, ou apenas uma vez a quantidade de feromônio depositada. Além disso, para que não haja uma convergência prematura para uma solução qualquer, o sistema utiliza um contador para contabilizar a quantidade de soluções iguais a melhor solução até o momento. Quando este número chega a 20, todo o sistema é atualizado com a quantidade inicial de feromônio. O valor 20 para este “gatilho” foi determinado ao longo da implementação e testes com o software, por ser um nível que garante que as respostas não podem ser melhoras a partir daí e também não trazem grande custo no que diz respeito ao custo computacional para a técnica.

Esta é a descrição básica do *software* implementado, e que foi utilizado para a realização dos experimentos, que são explicados mais detalhadamente na próxima seção.

4 Planejamento, realização e análise dos experimentos

Como explicado na seção 1.2, este trabalho pode ser classificado como experimental, pois o procedimento técnico baseia-se em simulações e análise da influência de certas variáveis de configuração sobre os cenários específicos de produção.

Neste sentido, o objetivo principal da fase experimental deste trabalho da pesquisa é avaliar na prática a aplicação da metaheurística *ACO* na resolução de um problema de programação de produção para trás, em um sistema produtivo mono-estágio, recursos paralelos e roteiros flexíveis.

As seções seguintes têm o objetivo de explanar como os experimentos foram realizados, bem como realizar as análises para cada objetivo proposto, que são primeiramente verificar a influência das variáveis de configuração do *ACO* na qualidade da solução do problema; e, em um segundo momento, analisar a eficiência do algoritmo baseado em *ACO* na resolução do problema estudado neste trabalho.

4.1 Análise da influência das variáveis de configuração do ACO

O objetivo desta primeira análise é observar se alterações no valor das variáveis ou parâmetros de configuração do sistema trazem alterações no desempenho do algoritmo, afetando dessa forma a qualidade da solução e o tempo computacional necessário para a obtenção desta solução. Para a realização desta análise foi utilizado o experimentos 2^k onde são avaliadas todas as combinações dos dois níveis de cada variável de configuração do sistema.

4.1.1 O experimento 2^k adotado na fase de análise da influência das variáveis de configuração do *ACO*

As variáveis de configuração do *ACO* que estão sendo analisadas neste trabalho quanto a sua influência na qualidade da solução do problema proposto são: valorização

da melhor resposta (VMR), número de formigas (NF), quantidade de feromônio inicial do sistema (QPI), quantidade adicionada de feromônio ao encontrar-se as soluções (QAP), número de viagens de cada formiga (NVF) e percentual de evaporação (EV), totalizando seis variáveis. Para isto, são feitas simulações com dois níveis de valores para cada uma destas variáveis, sendo uma quantidade mínima e uma quantidade máxima.

No experimento fatorial 2^k são realizados ensaios para todas as combinações possíveis dos dois níveis das 6 variáveis. Para cada cenário tem-se então $2^6 = 64$ resultados para o problema. Pode-se definir o delineamento dos experimentos 2^6 conforme representado na Tabela 3, onde o sinal “+” representa o nível mais elevado para a variável, e “–” o nível menor.

	Combinações de Tratamentos	A	B	C	D	E	F
1	(1)	-	-	-	-	-	-
2	A	+	-	-	-	-	-
3	B	-	+	-	-	-	-
4	AB	+	+	-	-	-	-
5	C	-	-	+	-	-	-
6	AC	+	-	+	-	-	-
7	BC	-	+	+	-	-	-
8	ABC	+	+	+	-	-	-
9	D	-	-	-	+	-	-
10	AD	+	-	-	+	-	-
11	BD	-	+	-	+	-	-
12	ABD	+	+	-	+	-	-
13	CD	-	-	+	+	-	-
14	ACD	+	-	+	+	-	-
15	BCD	-	+	+	+	-	-
16	ABCD	+	+	+	+	-	-
17	E	-	-	-	-	+	-
18	AE	+	-	-	-	+	-
19	BE	-	+	-	-	+	-
20	ABE	+	+	-	-	+	-
21	DE	-	-	+	-	+	-
22	ACE	+	-	+	-	+	-
23	BCE	-	+	+	-	+	-
24	ABCE	+	+	+	-	+	-
25	DE	-	-	-	+	+	-
26	ADE	+	-	-	+	+	-
27	BDE	-	+	-	+	+	-
28	ABDE	+	+	-	+	+	-
29	CDE	-	-	+	+	+	-
30	ACDE	+	-	+	+	+	-
31	BCDE	-	+	+	+	+	-
32	ABCDE	+	+	+	+	+	-
33	F	-	-	-	-	-	+
34	AF	+	-	-	-	-	+
35	BF	-	+	-	-	-	+
36	ABF	+	+	-	-	-	+
37	CF	-	-	+	-	-	+
38	ACF	+	-	+	-	-	+
39	BCF	-	+	+	-	-	+
40	ABCF	+	+	+	-	-	+
41	DF	-	-	-	+	-	+
42	ADF	+	-	-	+	-	+
43	BDF	-	+	-	+	-	+
44	ABDF	+	+	-	+	-	+
45	CDF	-	-	+	+	-	+
46	ACDF	+	-	+	+	-	+
47	BCDF	-	+	+	+	-	+
48	ABCDF	+	+	+	+	-	+
49	EF	-	-	-	-	+	+
50	AEF	+	-	-	-	+	+
51	BEF	-	+	-	-	+	+
52	ABEF	+	+	-	-	+	+
53	CEF	-	-	+	-	+	+
54	ACEF	+	-	+	-	+	+
55	BCEF	-	+	+	-	+	+
56	ABCEF	+	+	+	-	+	+
57	DEF	-	-	-	+	+	+
58	ADEF	+	-	-	+	+	+
59	BDEF	-	+	-	+	+	+
60	ABDEF	+	+	-	+	+	+
61	CDEF	-	-	+	+	+	+
62	ACDEF	+	-	+	+	+	+
63	BCDEF	-	+	+	+	+	+
64	ABCDEF	+	+	+	+	+	+

Tabela 3: Experimento 2^6 utilizado

A outra parte da etapa de planejamento dos experimentos é a definição do número de experimentos necessários para que haja uma confiabilidade estatística satisfatória.

De acordo com Dally *et al*. (1993), o uso da distribuição *t* de Student é a mais apropriada quando se tem pequenas amostras. Como a distribuição *t* depende do tamanho da amostra *n*, o valor de *t* pode ser usado para estimar *n* de tal forma que se obtenha uma estimativa da média da amostra para uma dada confiança. Para determinar o tamanho da amostra para o primeiro objetivo, que é o de analisar a influência das variáveis controláveis na qualidade das soluções, utilizou-se a seguinte equação:

$$n = \left(t \frac{S}{p * \mu} \right)^2 \tag{4.1}$$

Onde:

t: valor obtido pela distribuição t de Student para a confiança desejada;

S: desvio padrão;

p: precisão;

μ: média;

n: tamanho da amostra.

O procedimento para encontrar o tamanho da amostra é interativo, onde primeiramente determina-se um valor para n e verifica-se se t atende a confiança desejada. Se não atender, realizam-se novos ensaios, recalcula-se a média e o desvio padrão repetindo-se o procedimento até a convergência de n.

Para os cenários estudados estabeleceu-se a confiança de 95% e precisão de mais ou menos 1%. Para tanto, obteve-se através do procedimento interativo o número de 16 amostras.

As configurações adotadas para os níveis baixo e alto dos parâmetros de configuração do *ACO*, bem como cenários referentes ao número de ordens a serem

programadas e o número de recursos disponíveis estão descritos na Tabela 4. Tais configurações foram adotadas para a análise da influência desses parâmetros na qualidade da solução para o problema.

			Cenários															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	**VMR**	Mínimo	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
		Máximo	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
F	**NF**	Mínimo	20	20	20	20	10	10	10	10	10	10	10	10	10	10	10	10
		Máximo	50	50	50	50	20	20	20	20	20	20	20	20	20	20	20	20
D	**QPI**	Mínimo	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		Máximo	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
E	**QAP**	Mínimo	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
		Máximo	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
C	**NVF**	Mínimo	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
		Máximo	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
B	**EV**	Mínimo	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		Máximo	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	Número de máquinas		3	3	3	3	6	6	6	6	10	10	10	10	10	10	10	10
	Número de ordens		15	20	30	45	15	20	30	45	15	20	30	45	15	20	30	45

Tabela 4: Dados para análise do experimento 2^k

Após o término do delineamento dos experimentos, os testes foram realizados. Para a análise dos resultados obtidos com estes experimentos, foi utilizada a técnica estatística Análise de Variância para verificar se valores diferentes para as variáveis de configuração do sistema trariam alguma alteração para a variável que indica a qualidade da resposta.

A combinação de todos os parâmetros de configuração do sistema apresenta 64 resultados para cada cenário de produção. A análise é feita então com os resultados dos 16 cenários, ou seja, com as 1.024 (64x16) respostas obtidas. A Tabela 5 corresponde ao quadro ANOVA, que é o quadro padrão para que se faça a análise de variância, para os resultados dos cenários mostrados na Tabela 4.

ANOVA					
parâmetro	SS	GL	MS	Fo	Fcritico
A	315,30	1	315,30	96,51	4
B	19,74	1	19,74	6,04	4
C	72,01	1	72,01	22,04	4
D	35,27	1	35,27	10,79	4
E	17,39	1	17,39	5,32	4
F	9,03	1	9,03	2,76	4
ERRO	186,22	57	3,27		
T	533,96	63			

Tabela 5: Tabela ANOVA para os experimentos 2^6

Pela análise da Tabela 5, e com base na teoria sobre análise de variância, pode-se concluir que as variáveis (parâmetros) A, B, C, D e E tem influência sobre os resultados do problema.

O parâmetro A corresponde a variável valorização da melhor resposta, que pode influenciar na qualidade da resposta, pois se seu valor for muito pequeno, não será depositado uma quantidade suficiente de feromônio nos melhores caminhos, o que não aumentaria a probabilidade de outras formigas seguirem por este mesmo caminho. Da mesma forma, se este parâmetro receber um valor muito alto, pode fazer com que todas as formigas apresentem convergência para um determinado caminho muito cedo, impossibilitando o sistema de encontrar melhores soluções.

O parâmetro B corresponde à variável evaporação, que determina qual a quantidade de feromônio que os caminhos perdem com o passar do tempo. Maior quantidade de substância evaporada em determinado tempo pode ser importante para que uma solução de mínimo local não seja a escolhida.

O parâmetro C corresponde ao número de viagens que cada formiga deve fazer. Esta influencia na qualidade da solução, pois o feromônio - seja ele multiplicado pelo valor de valorização caso o caminho percorrido seja o menor, ou uma quantidade padrão caso não seja - só é depositado quando as formigas fazem uma viagem completa.

A importância do parâmetro D, que se refere a quantidade de feromônio inicial do sistema na qualidade das respostas, é percebida ao compará-las com a quantidade de feromônio adicionado pelas formigas a cada final de viagem. Se a quantidade de

feromônio inicial é muito grande e a de feromônio adicionado for muito pequena, por mais que esta última tenha sua quantidade multiplicada por uma determinada valorização, em termos proporcionais este aumento vai ser pequeno, e pode não aumentar significativamente a probabilidade de que as outras formigas optem por este caminho.

De modo análogo ao parâmetro D, o parâmetro E, que corresponde a quantidade de feromônio adicionada pelas formigas no final de cada viagem, influência diretamente a qualidade das respostas, pois dependendo desta quantidade, a probabilidade de que outras formigas optem por este caminho pode variar drasticamente.

Finalmente, o parâmetro F, que corresponde ao número de formigas do sistema, não se apresentou significativa ao se analisar sua influência na qualidade das respostas do problema. Isto porque o feromônio é adicionado nos caminhos percorridos pelas formigas apenas no final de suas viagens. O que quer dizer que se houver um número muito grande de formigas mas com cada uma delas fazendo um pequeno número de viagens, a comunicação entre as formigas não vai ser significativa. A variável F é classificada então como não significante, isso quer dizer que não se pode afirmar, como o nível de significância adotado e com o número de experimentos realizados, que variações nesta variável trará alterações na variável de resposta.

Parâmetro		Fo	Fcritico	Avaliação
A	VMR	96,51	4	Significante
B	EV	6,04	4	Significante
C	NVF	22,04	4	Significante
D	QPI	10,79	4	Significante
E	QAP	5,32	4	Significante
F	NF	2,76	4	Não Significante

Tabela 6: Tabela resumo dos resultados do experimento 2^k

A Tabela 6 traz um resumo dessa análise, mostrando quais variáveis se mostraram significantes ou não ao ser realizado o experimento 2^k proposto.

A análise dos experimentos realizados nesta parte do trabalho serviu para verificar a influência dos parâmetros de configuração do sistema na qualidade das respostas para o problema proposto. Isto porque se verificou a lógica de inteligência da

técnica *ACO*, que tem como principal estratégia, o uso da informação entre os agentes, no caso as formigas, para que encontrem o menor caminho do ninho até a fonte de comida, utilizando para isto uma estratégia que consiste em adicionar certa quantidade de feromônio nos caminhos percorridos. Dessa forma, todas as variáveis, que de uma forma ou de outra influenciam nesta quantidade de feromônio adicionada em cada caminho, se mostram significativas na análise de sua influência na qualidade das respostas.

4.2 Análise da eficiência da metaheurística ACO

Com a análise referente à influência das variáveis de controle sobre a qualidade das respostas para o problema proposto, novos experimentos foram realizados para a verificação da eficiência da técnica proposta (*ACO*) na resolução de problemas de programação de produção adotados neste trabalho.

Para testar a eficiência do uso da metaheurística *ACO* na otimização da programação de produção em sistemas produtivos semelhantes ao adotado neste trabalho, foram realizados testes comparativos da solução gerada pela metaheurística *ACO* com soluções obtidas através do método de otimização *branch and bound*, através de um *software* já implementado por Vieira (2008).

Os parâmetros de configuração para do sistema utilizados para os testes comparativos foram determinados também conforme metodologia proposta por Montgomery & Runger (2004), e que consiste basicamente em escolher o nível da variável em que a soma das médias das respostas seja maior. Os parâmetros utilizados foram os seguintes:

VMR = 5

NF = 50

QPI = 10

QAP = 50

NVF = 100

EV = 5

Como mostrado na seção 4.1, o parâmetro referente ao número de formigas do sistema não se mostrou significante. Ainda segundo Montgomery & Runger (2004), nestes casos deve-se optar por um nível do parâmetro que otimize a economia, a operacionalidade ou algum outro fator técnico preponderante ao se executar o algoritmo. A única alteração realizada foi neste parâmetro, para o qual se utilizou como valor o número 10, o que diminui o tempo computacional necessário para se chegar a resposta, e não altera a qualidade da mesma, já que esta variável não se mostrou significante na realização do experimento 2^k.

Para a realização desta segunda fase dos experimentos, foi considerado um sistema produtivo que tivesse a possibilidade de produção de 20 ítens, ou seja, haviam cadastrado no *software* 20 produtos. Ao todo foram realizados os experimentos para 9 cenários diferentes, onde nos 3 primeiros cada produto tinha um conjunto de roteiros flexíveis composto de 5 roteiros (máquinas ou recursos), numerados de I a V. Nos outros 6 cenários, foi considerado que cada produto poderia passar por 10 roteiros (máquinas ou recursos) diferentes, numerados de I a X.

O sistema produtivo que está sendo utilizado nesta pesquisa, conforme já explicado, é formado por apenas um estágio de processamento (mono-estágio). Isto quer dizer que os produtos passam por apenas 1 máquina (recurso). Este sistema possui máquinas paralelas, ou roteiros flexíveis, o que quer dizer que um produto pode passar por qualquer uma das máquinas do sistema. Estas máquinas são diferentes, ou possuem velocidades de produção diferentes. As velocidades para os 20 produtos considerados em cada roteiro (máquina ou recurso) estão mostradas na Tabela 7, e foram atribuídos de forma intencional para que pudesse representar da melhor forma possível casos reais onde a velocidade de produção de diferentes produtos em diferentes máquinas são realmente diferentes.

Quando considerado um cenário com 5 máquinas e um determinado produto A, o mesmo poderia passar pelas máquinas I, II, III, IV ou V. As velocidades de produção em cada máquina são 5, 6, 7, 8 e 9 unidades/hora respectivamente. Quando considerado um cenário com 10 máquinas, o produto A poderia passar pelas máquinas I, II, III, IV, V, VI, VII, VII, IX ou X. As velocidades de produção são respectivamente

para cada máquina 5, 6, 7, 8, 9, 10, 11, 12, 13 e 14 unidades/hora. Isto tudo está definido na Tabela 7.

	Velocidade de Produção (unidades/hora) Roteiros (máquinas)									
	I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
Produto A	5	6	7	8	9	10	11	12	13	14
Produto B	7	6	5	6	7	8	9	10	11	12
Produto C	17	15	13	11	9	7	5	7	9	11
Produto D	9	8	7	6	5	6	7	8	9	10
Produto E	7	5	7	9	11	13	15	17	19	21
Produto F	10	9	8	7	6	5	6	7	8	9
Produto G	8	7	6	5	7	9	11	13	15	17
Produto H	12	11	10	9	8	7	6	5	6	7
Produto I	21	19	17	15	13	11	9	7	5	7
Produto J	27	25	23	21	19	17	15	13	11	9
Produto K	14	13	12	11	10	9	8	7	6	5
Produto L	19	18	17	16	15	14	13	12	11	10
Produto M	29	27	25	23	21	19	17	15	13	11
Produto N	15	14	13	12	11	10	9	8	7	6
Produto O	18	17	16	15	14	13	12	11	10	9
Produto P	6	5	6	7	8	9	10	11	12	13
Produto Q	11	9	7	5	7	9	11	13	15	17
Produto R	5	7	9	11	13	15	17	19	21	23
Produto S	7	6	5	6	7	8	9	10	11	12
Produto T	9	8	7	6	5	6	7	8	9	10

Tabela 7: Velocidade de produção para cada produto em cada roteiro

Sobre as características dos cenários utilizados para a análise quanto à eficiência do *ACO* para o problema proposto, já foi mostrado então que o mesmo consiste de um sistema que há 20 produtos cadastrados. Existem 9 cenários diferentes, nos quais 3 primeiros cada produto pode passar por 5 máquinas diferentes, e nos outros 6 cenários, cada produto pode passar por 10 máquinas diferentes.

Cada um dos 9 cenários possui uma quantidade de OPs (ordens de produção) a serem programadas. Aqui, cada OP corresponde a um *Job* que determina certa quantidade de um produto a ser produzido. Nesta parte do trabalho foi considerado que cada cenário seria caracterizado por possuir 20, 40, 60, 80, 100 ou 120 OPs.

A Tabela 8 mostra a quantidade de cada produto a ser produzido em cada OP para cada cenário.

120 OPs *(Jobs)*

100 OPs (*Jobs*)

80 OPs (*Jobs*)

60 OPs (*Jobs*)

40 OPs (*Jobs*)

20 OPs (*Jobs*)

	Job	Quant.	Job	Quant.	Job	Quant.	Job	Quant.	Job	Quant.	Job	Quant.
Produto A	1	220	21	730	41	110	61	110	81	700	101	700
Produto B	2	150	22	560	42	300	62	75	82	530	102	530
Produto C	3	450	23	580	43	225	63	225	83	550	103	550
Produto D	4	370	24	370	44	740	64	185	84	340	104	340
Produto E	5	890	25	960	45	445	65	445	85	930	105	930
Produto F	6	450	26	790	46	900	66	225	86	760	106	760
Produto G	7	670	27	450	47	335	67	335	87	420	107	420
Produto H	8	1.350	28	645	48	675	68	675	88	615	108	615
Produto I	9	850	29	375	49	425	69	425	89	345	109	345
Produto J	10	760	30	490	50	1.520	70	380	90	460	110	460
Produto K	11	490	31	760	51	245	71	245	91	730	111	730
Produto L	12	375	32	850	52	750	72	188	92	820	112	820
Produto M	13	645	33	1.350	53	323	73	323	93	1.320	113	1.320
Produto N	14	450	34	670	54	900	74	225	94	640	114	640
Produto O	15	790	35	450	55	395	75	395	95	420	115	420
Produto P	16	960	36	890	56	480	76	480	96	860	116	860
Produto Q	17	370	37	370	57	740	77	185	97	340	117	340
Produto R	18	580	38	450	58	290	78	290	98	420	118	420
Produto S	19	560	39	150	59	280	79	280	99	120	119	120
Produto T	20	730	40	220	60	365	80	365	100	190	120	190

Tabela 8: Quantidade de cada produto em cada ordem de produção

De forma resumida, para a análise da eficiência da técnica *ACO*, os cenários adotados são os mostrados na Tabela 9.

Cenário	Número de Máquinas	Número de *Jobs* OPs
1	5	20
2	5	40
3	5	60
4	10	20
5	10	40
6	10	60
7	10	80
8	10	100
9	10	120

Tabela 9: Cenários para realização dos testes de eficiência do *ACO*

Os indicadores de desempenho da qualidade das soluções geradas foram o mínimo *makespan* e o tempo computacional necessário para se chegar a solução. A metodologia utilizada para a análise comparativa das duas técnicas foi o Teste de Hipótese com duas amostras, utilizando o teste de hipótese para diferença entre duas médias e teste de hipótese para diferença entre duas variâncias.

As Tabelas 10 e 11 mostram os resultados obtidos com os experimentos dos cenários mostrados na Tabela 9, onde na coluna intitulada BB estão os resultados para a técnica *branch-and-bound* e na coluna intitulada *ACO* estão os resultados para a técnica implementada neste trabalho.

Cenário	Função Objetivo *(Makespan)*	
	Métodos	
	BB	ACO
1	14,01	13,29
2	25,51	26,33
3	38,24	38,26
4	10,84	11,30
5	18,26	19,05
6	24,37	29,20
7	36,41	38,36
8	40,72	43,81
9	56,89	62,39

Tabela 10: Resultados do *makespan* para os cenários estudados

Cenário	Tempo Computacional (segundos)	
	Métodos	
	BB	ACO
1	1	14
2	3	56
3	14	93
4	547	28
5	1.298	112
6	3.254	186
7	8.620	357
8	19.865	800
9	23.838	2.867

Tabela 11: Tempo computacional para obtenção dos resultados da Tabela 9

Primeiramente, foi aplicado o *teste F* para os resultados das duas variáveis de resposta estudadas, função objetivo (*makespan*) e tempo computacional. Com isto, pode-se verificar se as variâncias entre as duas técnicas, levando-se em consideração as duas variáveis, eram significativamente diferentes, para daí sim aplicar o *teste t*, para verificar se a diferença entre as médias dos resultados das duas variáveis são diferentes, podendo-se então tirar conclusões sobre a eficiência da técnica *ACO* na otimização da programação de produção para o caso estudado.

4.2.1 O *teste F* para as 2 variáveis de resposta

O *teste F* foi executado para verificação quanto à diferença das variâncias das duas amostras, tanto nas respostas da função objetivo (*makespan*), quanto ao tempo computacional necessário para obter estas respostas. Para isto, utilizou-se a ferramenta Análise de dados do Microsoft Excel 2007, mais especificamente a função Teste F: duas amostras para variâncias, com um nível de significância α=5%.

A Tabela 12 representa o resultado para o *teste F* relativo a análise das variâncias dos resultados do *makespan* das duas técnicas a serem comparadas.

Teste-F: duas amostras para variâncias do *makespan*		
	BB	*ACO*
Média	29,47	31,33
Variância	219,61	265,60
Observações	9	9
gl	8	8
F	0,83	
P(F<=f) uni-caudal	0,40	
F crítico uni-caudal	0,29	

Tabela 12: *Teste F* para variâncias do *makespan*

Como o $F_o > F_c$, (0,83 > 0,29) a hipótese nula, que pressupõe que as variâncias das duas amostras são iguais, não deve ser aceita, ou seja, há evidência de que há diferença entre as variâncias é significativa.

A Tabela 13 representa o resultado para o *teste F* relativo a análise das variâncias do tempo computacional gasto para a obtenção dos resultados para cada cenário e para cada técnica.

Teste-F: duas amostras para variâncias do tempo computacional		
	BB	*ACO*
Média	6.382	501
Variância	85.393.845	848.108
Observações	9	9
gl	8	8
F	100,69	
P(F<=f) uni-caudal	0,00	
F crítico uni-caudal	3,44	

Tabela 13: *Teste F* para variâncias do tempo computacional

Como o $F_o > F_c$, (100,69 > 3,44) a hipótese nula, que pressupõe que as variâncias das duas amostras são iguais, não deve ser aceita, ou seja, há evidência de que a diferença entre as variâncias é significativa relativa ao tempo computacional.

4.2.2 O *teste t* para as 2 variáveis de resposta

Os resultados do *teste F* para as duas variáveis de resposta mostraram que nos dois casos as variâncias entre as amostras são diferentes. Isto indica que o *teste t* para análise da diferença entre as médias dos resultados das duas variáveis estudadas deve ser o *teste t* para duas amostras presumindo variâncias diferentes.

O *teste t* foi executado para verificação quanto à diferença entre as médias das duas amostras, tanto nas respostas da função objetivo (*Makespan*), quanto ao tempo computacional necessário para obter estas respostas. Para isto, utilizou-se a ferramenta Análise de dados do Microsoft Excel 2007, mais especificamente a função Teste t: duas amostras presumindo variâncias diferentes, com um nível de significância α=5%.

A Tabela 14 representa os resultados do *testes t* para os dados referentes ao *makespan* das duas técnicas comparadas.

Teste-t: duas amostras presumindo variâncias diferentes para o *makespan*

	BB	*ACO*
Média	29,47	31,33
Variância	219,61	265,60
Observações	9	9
Hipótese da diferença de média	0	
gl	16	
Stat t	-0,25	
P(T<=t) uni-caudal	0,40	
t crítico uni-caudal	1,75	
P(T<=t) bi-caudal	0,80	
t crítico bi-caudal	2,12	

Tabela 14: *Testes t* para o *makespan*

No caso da comparação das médias dos resultados relativos ao *makespan* nas duas técnicas, a hipótese nula deve ser aceita, pois o *p-value* = 40%, que é maior do que o nível de significância adotado de 5%. Isto quer dizer que a diferença entre as médias não é significante.

Os resultados do *testes t* para os dados referentes ao tempo computacional das duas técnicas comparadas é apresentado na Tabela 15.

Teste-t: duas amostras presumindo variâncias diferentes para o tempo computacional

	BB	*ACO*
Média	6.382	501
Variância	85.393.845	848.108
Observações	9	9
Hipótese da diferença de média	0	
gl	8	
Stat t	1,900	
P(T<=t) uni-caudal	0,047	
t crítico uni-caudal	1,860	
P(T<=t) bi-caudal	0,094	
t crítico bi-caudal	2,306	

Tabela 15: *Teste t* para o tempo computacional

No caso da comparação das médias dos resultados relativos ao tempo computacional nas duas técnicas, a hipótese nula deve ser rejeitada, pois o *p-value* = 4,7%, que é menor do que o nível de significância adotado de 5%. Isto quer dizer que a diferença entre as médias é significante.

	Makespan		Tempo Computacional (s)	
Cenário	*BB*	*ACO*	*BB*	*ACO*
1	14,01	13,29	1	14
2	25,51	26,33	3	56
3	38,24	38,26	14	93
4	10,84	11,30	547	28
5	18,26	19,05	1.298	112
6	24,37	29,20	3.254	186
7	36,41	38,36	8.620	357
8	40,72	43,81	19.865	800
9	56,89	62,39	23.838	2.867
	Makespan		Tempo Computacional (s)	
	Fo	*Fc*	*Fo*	*Fc*
Teste F	0,83	0,29	100,69	3,44
	Makespan		Tempo Computacional (s)	
	t	*α*	*t*	*α*
Teste t	40,0%	5,0%	4,7%	5,0%

Tabela 16: Resumo dos resultados da análise da eficiência do *ACO*

Com base em todos os experimentos realizados nesta parte do trabalho, a Tabela 16, que é um resumo de todas as tabelas apresentadas nesta seção, traz um resumo dos resultados encontrados o que é necessário para as principais conclusões do trabalho.

Os objetivos que nortearam esta pesquisa estavam suportados por duas questões básicas, que foi primeiramente verificar se a metaheurística *ACO* é uma técnica viável para a solução de problemas de otimização da programação de produção para trás em sistemas produtivos mono-estágio, com recursos paralelos e velocidades de produção diferentes, e com roteiros flexíveis para os produtos. A outra questão básica da pesquisa, e que foi necessária também para responder a primeira questão, buscou verificar através de uma comparação direta com a técnica *branch and bound*, se a metaheurística *ACO* é eficiente ao levar-se em consideração a qualidade do programa de produção gerado, medido pelo *makespan*, e também em relação ao tempo computacional necessário para a geração destes programas de produção.

As questões centrais da pesquisa são então respondidas na medida em que os resultados dos testes realizados demonstram que a metaheuística *ACO* é uma técnica viável para a programação de produção em ambientes similares ao proposto, através da demonstração de forma comparativa que a meaheurística *ACO* é eficiente ao ser comparada com a técnica *branch and bound* no que diz respeito à qualidade do programa gerado, concluindo que estatisticamente não pode-se afirmar que os programas tem *makespans* diferentes, e demonstrando também que em termos de custo computacional, a metaheurística *ACO* é superior se comparada com a técnica *branch and bound*.

5 CONCLUSÕES

O objetivo principal deste trabalho foi verificar se a técnica conhecida como *Ant Colony Optimization (ACO)* é eficiente na otimização da programação de produção. Para isso, uma análise das variáveis de configuração da metaheurística quanto a sua influência na variação da qualidade da resposta do problema foi necessária e também foi objetivo de estudo deste trabalho.

Para a realização destas análises, um aprofundamento teórico sobre os assuntos ligados direta ou indiretamente ao tema foram feitos através da revisão da literatura sobre o assunto, no qual se discutiu desde a classificação dos sistemas de produção, a problemática da programação de produção passando pela sua definição formal, algumas técnicas utilizadas na resolução de problemas de programação de produção, e uma discussão mais aprofundada no que diz respeito a metaheurística propriamente dita.

Realizou-se também uma discussão de como a técnica *ACO* tem sido aplicada na resolução de problemas de programação de produção e uma modelagem para o problema proposto, que consiste em programar a produção em um sistema produtiva mono-estágio, com máquinas paralelas e com velocidades diferentes, e roteiros flexíveis para dos os *jobs*.

Para a verificação da influência das variáveis controláveis do processo na qualidade da resposta do problema, foi realizado um experimento 2^k, e logo após uma análise de variância com os resultados obtidos. Na análise da eficiência da técnica *ACO* para resolver problemas de programação de produção em um sistema de produção equivalente ao estudo neste trabalho, foi realizado o teste de hipótese, *teste t.*

Com a análise de variância dos resultados a partir do experimento 2^k, foi possível concluir que as seguintes variáveis se mostraram significantes quanto à influência na qualidade das respostas para o problema tratado: valorização da melhor resposta (VMR), evaporação (EV), quantidade de feromônio inicial (QPI), quantidade de feromônio adicionado (QAP) e número de viagens das formigas (NVF). Isto corresponde em termos estatísticos a ter um *F* observado maior do que o *F* crítico com um grau de

confiança de 95%. A única variável, entre as estudadas, que não se mostrou significante foi a referente ao número de formigas do sistema (NF), pois apresentou um *F* observado menor do que o *F* crítico com um grau de confiança de 95%.

Esta primeira parte dos experimentos foi necessária primeiramente para verificar quais variáveis de configuração do sistema influenciam na qualidade da resposta do algoritmo; também para a definição de qual nível seria o ideal para ser utilizada por cada variável na segunda fase dos experimentos, já que nesta o objetivo foi analisar a eficiência da metaheurística *ACO*. Dessa forma, quanto melhor fosse o resultado, mais eficiente se mostraria a técnica, e para isso, a escolha do nível ideal para as variáveis de configuração é fundamental.

Para análise quanto à eficiência da técnica *ACO* foi feita, tanto para a qualidade da resposta para o problema, quanto ao tempo computacional gasto para se chegar a esta resposta, uma comparação com a técnica *branch and bound,* através de um *software* já implementado.

Com este direcionamento, foi realizado o *teste t*, que é um teste de hipótese para verificar se a diferença entre as médias dos resultados das duas técnicas, tanto no que diz respeito a qualidade da resposta (*makespan*) quanto ao tempo computacional necessário para obter esta resposta. Após a realização do *teste t*, pode-se concluir que a diferença entre as médias das respostas para o problema proposto não eram significativas, ou seja, com os testes realizados, não se pode afirmar que há diferença entre a média dos resultados das duas técnicas.

Isso permite concluir que, levando-se em consideração a forma de implementação utilizada no trabalho, as variáveis de configuração bem como seu níveis da metaheurística *ACO* e, ainda, o cenário de produção em que a esta foi aplicada, a técnica *ACO* é eficiente na otimização da programação de produção se comparada com a técnica *branch and bound*.

Quanto ao tempo computacional, o *teste t* indicou ainda que a técnica *ACO* é mais adequada para a resolução do problema de otimização da programação de produção para trás, em sistemas produtivos compostos de apenas um estágio de processamento, com recursos paralelos e roteiros flexíveis, principalmente nos casos

em que o número de recursos e o número de *jobs* é grande. O teste de hipótese mostrou que há diferença entre as médias das respostas das duas técnicas e numa comparação direta pode-se verificar claramente que o tempo necessário para se chegar a resposta para o problema aumenta em proporção muito maior na técnica *branch and bound* do que na técnica *ACO* na medida que a complexidade do problema também aumenta.

5.1 Limitações da pesquisa e sugestões para trabalhos futuros

As conclusões desta pesquisa limitam-se apenas ao problema nela abordado, que é o de programar a produção em um sistema de produção mono-estágio, com máquinas paralelas e com velocidades diferentes, e roteiros flexíveis para todos os *jobs*. Do mesmo modo, há que se destacar que a técnica *ACO* tem sido implementada de diferentes formas para a resolução de diferentes problemas, o que limita as conclusões deste trabalho apenas para implementações equivalentes as que aqui foram feitas, principalmente no que diz respeito aos parâmetros controláveis adotados, seus níveis e a lógica de funcionamento do algoritmo.

Para pesquisas futuras, sistemas produtivos com as características iguais ao estudado aqui, porém com mais de um estágio de processamento, poderiam ser colocados como problema para a técnica *ACO* pudesse resolvê-los. Isto poderia ser feito através primeiramente do cadastro de mais recursos, que seriam utilizados em fazes subseqüentes, uma definição no algoritmo que indique que cada produto deve passa por mais de um recurso, em quais recursos deve passar e finalmente em qual seqüência deve passar. Um ambiente de programação para frente também poderia ser estudado, o que implicaria apenas em não definir uma data para que as ordens estejam prontas, mas sim uma data inicial para que todas as ordens possam começar a ser produzidas. Implementações diferentes do *ACO* para o mesmo cenário também poderiam ser testadas e comparadas, podendo ser alterado o fato, por exemplo, de a evaporação poder ocorrer em cada movimento das formigas, e não somente quanto chegam a fonte de comida, bem como diferentes variáveis da metaheurística poderiam

ser adicionadas e testadas para diferentes níveis de valores. Além disso, um estudo que levasse em consideração uma função multiobjetivo, ou seja, outros objetivos a serem maximizados além do *makespan*, tais como atraso médio na entrega dos pedidos, tempo médio de fluxo das ordens e nível de ocupação dos recursos também trariam avanços em relação ao presente estudo.

REFERÊNCIAS

AARTS, E.H.L., KORST, J. H.M., **Simulated annealing and Boltzmann machines:** a stochastic approach to combinatotial optimization and neural computing. New York: Wiley Interscience, 1989.

AGRAWAL, S.; TIWARI, M. K. **A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem**. International Journal of Production Research, vol. 46, n· 6, pp. 1405 – 1429. India: Ranchi, 2008.

ALLE, A. **Técnicas de programação matemática aplicadas ao scheduling de processos químicos contínuos.** Tese de Doutorado - Escola Politécnica da Universidade de São Paulo. São Paulo, SP, 2003.

ARROYO, J. E. C. **Heurísticas e metaheurísticas para otimização combinatória multiobjetivo**. Tese de Doutorado – Universidade Estadual de Campinas. Campinas, SP, 2002.

BAUER, A., BULLNHEIMER, B., HARTL, R.F., STRAUSS, C. **An ant colony optimizations approach for the single machine total tardiness problem**. Universit of Vienna. Vienna, Áustria, 1999.

BLAZEWICZ, J., ECKER, K. H., PESCH, E., SCHMIDT, G., WEGLARZ, J. **Scheduling computer and manufacturing process**. Berlim, Germany, 1996.

BUSTAMANTE, L. M. **Minimização do custo de antecipação e atraso para o problema de seqüenciamento de uma máquina com tempo de preparação dependente da seqüência:** aplicação em uma usina siderúrgica. Dissertação de Mestrado – Universidade Federal de Minas Gerais. Belo Horizonte, MG, 2007.

BUZZO, W. R., MOCCELLIN, J. V. **Programação da produção em sistemas flow shop utilizando um método heurístico híbrido algoritmo genético-simulated annealing**. Revista Gestão & Produção, vol. 7, n. 3, pp. 364 – 377. São Paulo, SP, 2000.

CALEGARE, A. J. A. **Introdução ao delineamento de experimentos**. São Paulo: Edgard Blücher, 2001.

CAMPOS JR., A. L. **Escalonamento da produção:** uma proposta de abordagem. Dissertação de Mestrado – Universidade Federal de Minas Gerais, Belo Horizonte, MG, 2000.

CARVALHO, V. O. **Um modelo de seqüenciamento da produção para um sistema de apoio à decisão**. Dissertação de Mestrado – Universidade Federal de São Carlos. São Carlos, SP, 2003.

CASTRO, H. P. **Utilização de algoritmos genéticos para solução de problema de programação de produção de uma refinaria de petróleo**. Dissertação de Mestrado – Universidade Federal de Santa Catarina. Florianópolis, SC, 2001.

CHIU, CHUI-YO, WANG, CHANG-RUEI. **A two phase ant colony optimization method for flexible job shop scheduling problems**. International Journal of Production Research, 2008.

COELHO, L. S. **Fundamentos**, **potencialidade e aplicações de algoritmos evolutivos**. Notas em Matemática Aplicada. Sociedade Brasileira de Matemática Aplicada e Computacional. São Carlos, SP, 2003.

COLIN, E. C.; SHIMIZU, T. **Algoritmo de programação de máquinas individuais com penalidades distintas de adiantamento e atraso**. Escola Politécnica da USP. Departamento de Engenharia de produção. Pesquisa Operacional, vol. 20, n. 1. São Paulo, SP, 2000.

CORMEN, T. H,; LEISERSON, C. E.; RIVEST, R. L.;STEIN, C. **Algoritmos:** teoria e prática. Tradução de Vandenberd D. de Souza, 2. ed. Rio de Janeiro: Campus, 2002.

CORRÊA, H. L.; CORRÊA, C. A. **Administração de produção e operações – manufatura e serviços:** uma abordagem estratégica. São Paulo: Atlas, 2004.

DALLY, J.W., RILEY, W.F., McCONNELL, K.G. **Instrumentation for engineering measurements**. 2. ed. New York: John Wiley & Sons, 1993.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **The ant system:** an autocatalytic optimizing process. Technical Report N° 91-016 Revised, Politécnico di Milano. Italy, 1991.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **The ant system:** optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, n. 1, pp 1 – 13, 1996.

DORIGO, M.; DI CARO, G.; CAMBARDELLA, L.. M. **Ant algorithms for discrete optimization**. Artificial Life, Université Libre de Bruxelles, vol. 5, n. 2, pp. 137 – 172. Belgium, 1999.

DORN, J.; SLANY, W. A flow shop with compatibility constraints in a stellmaking plant, in: **Intelligent Scheduling**, Zweben, Monte; Fox, Mark S. (eds.), cap. 22. San Francisco: Morgan Kaufman, 1994.

DOWSLANDO, K. A., Simulated annealing, in: **Modern heuristic techniques for combinatorial problems.** Reeves, C.R. Blackwell Scientific Publications. New York, USA, 1993.

FERREIRA, A. B. H. **Novo dicionário da língua portuguesa**. Rio de Janeiro: Vozes 1972.

FIGLALI, N.; ÖZKALE, C.; ENGIN, O.; FIGLALI, A. **A investigation of ant system parameter interactions by using design of experiments for job-shop scheduling problems.** Computers & Industrial Engineering, Turkey (2007).

GAITHER**,** N.; FRAZIER**,** G. **Administração da produção e operações**. 8. ed. São Paulo: Pioneira, 2001.

GAREY, M. R., JOHNSON, D.S. **Computers and intractability:** a guide to NP-completeness**.** New York: W. H. Freemon and Company, 1979.

GIL, A. C. **Como elaborar projetos de pesquisa**. 3. ed. São Paulo: Atlas, 1991.

GLOVER, F. **Future paths for integer programming and links to artificial intelligence**. Computer & Operations Research, vol. 13, n. 5, 533 – 549. USA, 1986.

GLOVER, F. Tabu search, in: **Modern heuristic techniques for combinatorial problems.** Reeves, C.R. Blackwell Scientific Publications. New York, USA, 1993.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning.** Addison-Wesley, Reading. Massachussets, USA, 1989.

GOLDBARG, M. C.; LUNA H. P. L. **Otimização combinatória e programação linear** – modelos e algoritmos. Rio de Janeiro: Campus, 2000.

GAUTHIER, G. **Feeding ecology of nesting greater snow geese**. The Journal of wildlife management, vol. 57, n. 2, pp. 216 – 223. Canadá, 1993.

GRAVES, S. C. **A review of production scheduling**. Research 28, 646 - 675. Massachussets, USA, 1981.

HADDAD R. B. B.; CARVALHO, M. F. H. ***A case study of capacitated scheduling***. Third International Conference on Production Research. Curitiba, PR, 2006.

HO, J.C.; CHANG, Y.L. **A new heuristic for the n-job, m-machine flowshop problem**. European Journal of Operational Research. vol. 52, n. 2, pp. 194 – 202. Atlanta, USA, 1991.

HOLLAND, J. H. **Adaptation in natural and artificial systems**: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan. Michigan, USA, 1975.

KIRKPATRICK, S., GELLAT, C. D., VECCHI, M. P. **Optimization by simulated annealing**. Science, vol. 220, n. 4598, Caracas, Venezuela, 1983.

KRINK, T.; URSEM, R. K.; THOMSEN, R. **Ant systems and particle swarm optimization**. Topics of Evolutionary Computation, ALife Group, University of Aarhus EV, Denmark, 2002.

LIAO, C. J., LIAO, C.C. **An ant colony optimization algorithm for scheduling in agile manufacturing.** International Journal of Production Research, vol. 46, n. 7, pp 1831 – 1824. Taipei, Taiwan, 2008.

LAARHOVE, V. P. J. M.; AARTS, E.H.L., **Simulated annealing:** theory and applications. Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications, vol. 12, n. 1, pp 108 – 111. Norwel, USA, 1988.

LAPPONI, J.C. **Estatística usando Excel.** Rio de Janeiro: Campus, 2005.

LIN, B.M.T., LU, C.Y. **New features of ant colony optimization for flowshop scheduling.** National Chi-Nan University. Pu-Li, Taiwan, 2004.

LIU, J.; REEVES, C.R. **Constructive and composite heuristic solutions to the P//ΣC$_i$ scheduling problem.** European Journal of Operational Research, vol. 132, n. 2, pp, 439 – 452. Hong Kong, China, 2001

LUCHE, J. R. D., MORABITO, R. **Otimização na programação da produção de grãos eletrofundidos:** um estudo de caso. Revista Gestão & Produção. v. 12, n. 1, pp. 135 - 149. São Carlos, SP, 2004.

MAZZUCCO JR., J. **Uma abordagem híbrida do problema da Programação da produção através dos algoritmos simulated annealing e genético**. Tese de Doutorado – Universidade Federal de Santa Catarina. Florianópolis, SC, 1999.

MERKLE, D., MIDDENDORF, M. **An ant algorithm with a new pheromone evalution rule for total tardiness problems**. Institute for Applied Computer Science and Formal Description Methods. University of Karlsruhe. England, 2000.

MERKLE, D.; MIDDENDORF, M. **On solving permutation scheduling problems with ant colony optimization.** International journal of System Science, vol. 36, n. 5, pp 255 – 266. Leipzig, Alemanha, 2005.

MESGHOUNI, K.; HAMMADI, S.; BORNE, P. ***Evolutionaty algorithms for job-shop schduling***. Int. J. Appl. Math. Comput. Sci, vol. 14, n. 1, pp 91 – 103. França, 2004.

MIRANDA, J. M. **Programação de produção otimizada de misturas na indústria do petróleo: utilização de métodos de programação matemática.** Dissertação de Mestrado – Escola Federal de Engenharia de Itajubá. Itajubá, SP, 2001.

MONTEVECHI, J. B. A.; DUARTE, R.; NILSSON, G. V., **O uso da simulação para análise do layout de uma célula de manufatura**. Revista Pesquisa e Desenvolvimento Engenharia de Produção, n.1, pp 15 – 29. Itajubá, SP, 2003.

MONTGOMERY, D.C. **Design and analysis of experiments**. 3. ed. John Wiley and Sons. Arizona, USA, 1991.

MONTGOMERY, D.C.; RUNGER, G.C. **Estatística aplicada à engenharia**. Rio de janeiro: LTC, 2004.

MOREIRA, D. A. **Administração da produção e operações**. 3. ed. São Paulo: Pioneira, 1998.

MOURA JR., A. N. C. **Novas tecnologias e sistemas de administração da produção** – análise do grau de integração e informatização nas empresas catarinenses. Dissertação de Mestrado - Universidade Federal de Santa Catarina. Florianópolis, SC, 1996.

OGBU, F.A.; SMITH, D. K. **The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem**. Computers & Operations Research, vol. 17, n. 3, pp 243 – 253. Oxford, England, 1990.

OGBU, F.A., SMITH, D. K. **Simulated annealing for the permutation flow-shop scheduling.** OMEGA, n. 19, pp 64 – 67. Oxford, England, 1991.

PALOMINO, R.C. **Um modelo baseado em redes de petri para o planejamento e a programação da produção em ambientes do tipo *job shop***. Tese de Doutorado – Universidade Federal de Santa Catarina. Florianópolis, SC, 2001.

PASSOS, C. A. S.; FONSECA, S. L. **A. uma arquitetura multiagente para a solução de problemas de seqüenciamento da produção**. Universidade de Campinas. Campinas, SP, 2005.

PEGDEN, D.; SHANNON, R.; SADOWSKI R., ***Introduction to simulation using* simam,** McGraw-Hill, Blacklike. USA, 1995.

PHANDNIS, S.; BREVICK, J.; IRANI, S. ***Development of a new heuristic for scheduling flow-shops with parallel machines by prioritizing bottleneck stages***.

Society for Design and Process Science, Transactions of the SDPS, vol. 7, n. 1, pp. 87 – 97. USA, 2003.

PRADHAN, S., LAM, S.S.Y. **Minimizing makespan during environmental stress screening using a genetic algorithm and na ant colony optimization**. Springer-Verlag London Limited. London, England, 2006.

RAJENDRAN, C.; ZIEGLER, H. **An efficient heuristic for scheduling in a flowshop scheduling to minimize makespan/total flowtime of jobs**. European Journal of Operational Research, vol. 155, pp 426 – 438. India / England, 1997.

RAJENDRAN, C.; ZIEGLER, H. **Two ant-colony algorithms for minimizing total flowtime in permutation flowshops**. Computers & Industrial Engineering vol. 48, pp 789 – 797. India / Inglaterra, 2004.

REBOUÇAS, O.; PINHO, D. **Estratégia empresarial**. São Paulo: Atlas, 1988.

REEVES, C. R.; BEASLEY, J. E. **Modern heuristic techniques for combinatorial problems**. Blackwell Scientific Publications. New York, USA, 1993.

REIS, J. **Uma introdução ao scheduling**. Relatório Interno, Departamento de Ciências e Tecnologias de Informação, (Instituto Superior de Ciências do Trabalho e da Empresa). Lisboa, Portugal, 1996.

RIBAS, P. C. **Análise do uso de tempera simulada na otimização do planejamento mestre da produção**. Dissertação de Mestrado – Pontifícia Universidade Federal do Paraná. Curitiba, PR,2003.

RODAMMER, F. A.; WHITE, K. P. ***A Recent Survey of Production Scheduling.*** IEEE Transactions on Systems, Man, and Cybernetics, vol. 118, n. 6, pp 841 – 851. Charlottesville, USA, 1988.

RODRIGUES, L. F. **Meta-heurísticas evolutivas para dimensionamento de lotes com restrições de capacidade em sistemas multiestágios**. Dissertação de Mestrado – Universidade de São Paulo. São Paulo, SP, 2000.

ROMERO, R.; MANTOVANI, J. R. S., **Introdução a metaheurísticas.** Anais do 3º Congresso Temático de Dinâmica e Controle da SBMAC. Ilha Solteira: DEE-FEIS-UNESP. São Paulo, SP, 2004.

RUSSOMANO, V. H. **Planejamento e controle da produção**. São Paulo: Pioneira, 1998.

SANTORO, M. C.; MORAES, L. H**. Simulação de uma linha de montagem de motores**. Revista Gestão & Produção, vol. 7, n. 3, pp 338 – 351. São Paulo, 2000.

SCOFIELD, W. C. L., **Aplicação de algoritmos genéticos ao problema job-shop**. Monografia de Graduação – Universidade Federal de Ouro Preto. Ouro Preto, MG, 2002.

SHYU, S.J.; LIN, B.M.T.; YIN, P.Y. **Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time**. Computers & Industrial Engineering vol. 47, pp 181-193. Taiwan, 2004.

SLACK, N.; CHAMBERS, S.; HARLAND, C.; HARRISON, A.; JOHNSTON, R. **Administração da produção**. São Paulo: Atlas, 1997.

SOARES, M. M**. Análise do uso de algoritmos genéticos na otimização do planejamento mestre da produção**. Dissertação de Mestrado – Pontifícia Universidade Católica do Paraná. Curitiba, PR, 2006.

TAILLARD, E. **Benchmarks for basic scheduling problems**. European Journal of Operational Research, vol. 64, n. 2, pp 278-285, Switzerland, 1993.

TANESE, R. **Distributed genetic algorithms**. Proceedings of the 3rd ICGA pp 434 - 439. USA, 1989.

TRIVIÑOS, Augusto N.S. **Introdução à pesquisa em ciências sociais**. São Saulo: Atlas, 1987.

TUBINO, D. F. **Manual de planejamento e controle da produção**. 2. ed. São Paulo: Atlas, 1997.

VACA, O. C. L. **Um algoritmo evolutivo para a programação de projetos multi-objetivos com nivelamento de recursos limitados**. Tese de Doutorado – Universidade Federal de Santa Catarina. Florianópolis, SC, 1995.

van der ZWAAN, S.; MARQUES, C. **Ant colony optimisation for job shop scheduling**. Instituto de Sistemas e Robótica, Instituto Superior Técnico (IST). Lisboa, Portugal, 1999.

VENTRESCA, M.; OMBUKI, B. **Ant colony optimization for job-shop scheduling problem**. Tecnical Report, Department of Computer Science. St. Catharines. Ontário, Canadá, 2004.

YAGMAHAN, B.; YENISEY, M.M. **Ant colony optimization for multi-objetive flow shop scheduling problem**. Computers & Industrial Engineering, vol. 54, n. 3, pp 411 – 420. Turkey, 2007.

YING, K.C.; LIAO, C. J. **An ant colony system approach for scheduling problems.** Production Planning & Controle, vol. 14, n. 1, pp 68 – 75. Taipei, Taiwan, 2003.

YING, K.C.; LIN, S. W**. Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach**. International Journal of Production Research, vol. 44, n. 19, pp 3161 – 3177. Taipei, Taiwan, 2006.