UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Um Método de Busca usando Algoritmo Genético para Programação Reativa da Produção de Sistemas de Manufatura com Recursos Compartilhados

Ana Claudia Deriz

São Carlos - SP
2007

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Um Método de Busca usando Algoritmo Genético para Programação Reativa da Produção de Sistemas de Manufatura com Recursos Compartilhados

ANA CLAUDIA DERIZ

Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: **PROF. DR. ORIDES MORANDIN JUNIOR**
Co-Orientador: **PROF. DR. EDÍLSON REIS RODRIGUES KATO**

São Carlos - SP
2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

D433mb

Deriz, Ana Claudia.
Um método de busca usando algoritmo genético para programação reativa da produção de sistemas de manufatura com recursos compartilhados / Ana Claudia Deriz. -- São Carlos : UFSCar, 2011.
118 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Ciência da computação. 2. Programação da produção. 3. Sistemas de manufatura. 4. Compartilhamento de recursos. 5. Algoritmos genéticos. I. Título.

CDD: 004 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Um Método de Busca usando Algoritmo Genético para Programação Reativa da Produção de Sistemas de Manufatura com Recursos Compartilhados"

ANA CLAUDIA DERIZ

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Orides Morandin Junior
(Orientador - DC/UFSCar)

Prof. Dr. Edílson Reis Rodrigues Kato
(Co-Orientador – DC/UFSCar)

Prof. Dr. André Carlos P. L. Ferreira de Carvalho
(ICMC/USP)

Prof. Dr. Stephane Julia
(Faculdade de Computação/UFU)

São Carlos
Dezembro/2007

A meus pais, José e Neusa, pelo amor, apoio e dedicação todos os dias.

Agradecimentos

Primeiro e principalmente a Deus, por ter me dado tudo o que tenho. A minha família, por me amar, ajudar e apoiar em tudo sempre. Ao Léo, pela paciência. Ao meu orientador, Prof. Dr. Orides Morandin Junior e co-orientador, Prof. Dr. Edílson Reis Rodrigues Kato, pela dedicação constante. Aos professores do Departamento de Computação da Universidade Federal de São Carlos pelo apoio constante. Aos colegas de estudo, por toda a ajuda. A todas as pessoas que direta ou indiretamente contribuíram para a realização deste trabalho.

Resumo

DERIZ, Ana C. *Um Método de Busca usando Algoritmo Genético para Programação Reativa da Produção de Sistemas de Manufatura com Recursos Compartilhados. Dissertação de Mestrado*; Departamento de Computação, Universidade Federal de São Carlos; Outubro de 2007.

Várias estratégias de busca têm sido usadas para resolver o problema da programação da produção de sistemas de manufatura com recursos compartilhados. Entretanto, dependendo do tamanho e complexidade do sistema de manufatura, o tempo de resposta da busca torna-se crítico. Várias pesquisas apontam para o uso de Algoritmos Genéticos como método de busca para resolver o problema da programação da produção. O presente trabalho propõe uma modelagem de Algoritmo Genético para resolver tal problema de maneira eficiente, tendo como critério de desempenho o mínimo *makespan* da programação e obtendo baixo tempo de resposta da busca.

Palavras-chave: Programação da Produção, Sistemas de Manufatura, Compartilhamento de Recursos, Algoritmos Genéticos.

Abstract

DERIZ, Ana C. *A search method using Genetic Algorithm for Reactive Scheduling of Sharing Resources Manufacturing Systems. Dissertação de Mestrado*; Departamento de Computação, Universidade Federal de São Carlos; Outubro de 2007.

Search strategies have been used to solve the problem of scheduling of Manufacturing Systems with Shared Resources. However, depending on the size and complexity of the manufacturing system, the response time of the search becomes critical. Reseaches aim to use of Genetic Algorithms as a search method to solve the scheduling problem. This work proposes a modeling of Genetic Algorithm to solve this problem having as performance criteria the minimum makespan of the scheduling and obtaining a low response time of the search.

Palavras-chave: Scheduling, Manufacturing Systems, Shared Resources, Genetic Algorithms.

Lista de Figuras:

Lista de Tabelas:

SUMÁRIO

1 INTRODUÇÃO

1.1 Caracterização do Problema e do Ambiente

Para sobreviver e obter sucesso no ambiente empresarial atual, as indústrias estão sempre buscando vantagem competitiva. Para tanto, entre outras características, necessitam de flexibilidade para se adaptarem às mudanças no sistema de produção. Em busca dessas características, as indústrias estão adotando, em seus processos produtivos, técnicas de automação e compartilhamento de recursos, como máquinas e sistema de transporte.

Quando se tem certa flexibilidade no sistema de manufatura, como a possibilidade de fazer um produto por várias rotas diferentes ou de uma máquina fazer várias operações diferentes, podendo ser utilizada para fazer produtos diferentes, espera-se obter uma utilização eficiente dos recursos, em termos de certos critérios de desempenho.

Para tanto, é necessário ter uma programação da produção e, conseqüentemente, um planejamento e controle da produção que possibilitem maximizar os benefícios que podem ser obtidos pelo sistema de manufatura. Obter tal programação da produção não é uma tarefa simples, especialmente em ambientes com compartilhamento de recursos.

O Planejamento e Controle da Produção (PCP) é responsável por decidir sobre o emprego dos recursos de produção de forma a assegurar a execução do que foi previsto pela empresa. O PCP se divide em diversas atividades, dentre as quais, uma das mais complexas é a Programação da Produção.

A Programação da Produção tem por objetivo determinar, da melhor forma possível, a ordem em que os produtos devem ser processados no sistema de produção, bem como o momento em que cada produto deve ser processado, de forma a atingir critérios determinados pela empresa.

Há alguns critérios que podem ser utilizados para se medir o desempenho de uma programação de produção. Geralmente, segundo Conway, Maxwell e Miller (1967), estas medidas estão relacionadas ao tempo médio de fluxo de peças no sistema, ao cumprimento de prazos e à adequada utilização dos recursos disponíveis.

Muitas das abordagens propostas para lidar com esse problema têm apontado dois elementos importantes para se determinar uma boa programação da produção: a modelagem do problema e a estratégia de solução (MAGGIO, 2005).

A modelagem do problema de programação da produção de um sistema de manufatura deve reproduzir as reais características do sistema. Várias pesquisas utilizam redes de Petri (REYES, 2000; LEE e DICESARE, 1994; MAGGIO, 2005), e suas variações, como redes de Petri Coloridas (HUANG *et al*, 2002; LIN *et al*, 2003), para modelar esse problema, bem como propõem várias estratégias para sua resolução, como métodos de busca heurística (Lee e Dicesare, 1992) e variações destes (LEE e DICESARE, 1994; REYES, YU e KELLEHER, 2000; REYES, YU e KELLEHER 2002; MAGGIO, 2005), métodos de busca usando algoritmos genéticos (TAKAHASHI, YAMAMURA e KOBAYASHI, 1996; YEUNG e MOORE, 2000; CHIU e FU, 1997; REYES, YU e LLOYD, 2001) e outros métodos de resolução como análise estrutural do modelo usando matrizes de incidência (GANG e WU, 2002). Também têm sido propostos métodos para solução deste problema usando algoritmos genéticos, não associados a um modelo de sistema de manufatura em PN (PONGCHAROEN, HICKS e BRAIDEN, 2004; CHAN, CHUNG e CHAN, 2005; JEONG, LIM e KIM, 2006).

Neste contexto, o grupo de pesquisa do TEAR, Laboratório de Pesquisa e Inovação em Tecnologias e Estratégias de Automação, investiga aplicações em indústrias de manufatura e de processos buscando a incorporação de questões estratégicas de produção e estratégias competitivas nos projetos. Os temas macro abordados pelo grupo são planejamento reativo de produção, automação da planta industrial e integração entre estratégias, planejamento e chão de fábrica. O grupo tem como ponto de partida das investigações os problemas, as oportunidades e as necessidades das manufaturas e tem por objetivo a busca por maior eficiência do sistema produtivo. As pesquisas de planejamento reativo da produção se dividem em duas linhas:

- Seqüenciamento Reativo da Produção com Técnicas de Inteligência Artificial e Simulação de Plantas Industriais;
- Programação Reativa da Produção com Técnicas de Inteligência Artificial.

Na linha de Programação Reativa da Produção com Técnicas de Inteligência Artificial, destaca-se o trabalho realizado por Maggio (2005), que usa uma heurística de busca aplicada sobre um modelo de Rede de Petri para gerar uma programação da produção. Neste trabalho são apontadas algumas restrições do sistema e o critério utilizado para medir a eficiência da programação gerada é o *makespan*, ou seja, o tempo total de processamento. O ambiente escolhido para a aplicação e teste do método de programação da produção foi um Sistema Flexível de Manufatura, mas a proposta pode ser aplicada a outros tipos de sistema de manufatura com compartilhamento de recursos. O próprio Maggio (2005), bem como outros

autores (TAKAHASHI, YAMAMURA e KOBAYASHI, 1996; YEUNG e MOORE, 2000; CHIU e FU, 1997; REYES, YU e LLOYD, 2001) apontam, em seus trabalhos, o uso de algoritmos genéticos para futuras investigações nesta área, entre outras razões, por apresentar uma metodologia 'robusta', que não necessariamente resulta em soluções ótimas, mas obtém um nível alto de resultados sobre uma gama imensa de problemas (YEUNG e MOORE, 2000).

Assim sendo e no intuito de dar continuidade aos trabalhos do grupo na linha de Programação Reativa da Produção com Técnicas de Inteligência Artificial, seguem os objetivos deste trabalho.

1.2 Objetivos

1.2.1 Objetivo Geral

Propor um método de programação da produção usando Algoritmos Genéticos que possa ser aplicado a sistemas de manufatura com recursos compartilhados, resultando em programações com baixo valor de *makespan*, em um tempo de resposta aplicável.

1.2.2 Objetivos Específicos

- Propor um método de busca usando Algoritmo Genético (AG) para programação da produção em sistemas de manufatura com recursos compartilhados;
- Desenvolver um *software* para testar e validar o método proposto;
- Comparar o desempenho do método e *software* desenvolvidos com outros métodos de programação da produção tendo como critérios de desempenho o mínimo *makespan* e o tempo de execução da busca.

1.3 Resultados Alcançados

Foi elaborado um método para gerar programação da produção em sistemas de manufatura com recursos compartilhados, utilizando algoritmos genéticos não diretamente

associado a um modelo do sistema em Redes de Petri.

Foi desenvolvido um *software* de programação da produção, baseado no método citado anteriormente.

O *software* foi testado para uma série de problemas e os resultados foram comparados com outros métodos propostos por Maggio(2005), Reyes *et al* (2002), Yu *et al* (2003a) e Yu *et al* (2003b), tendo como critérios o mínimo *makespan* e tempo de execução da busca.

1.4 Metodologia

Segundo Gil (*apud* SILVA e MENEZES, 2001, p.25), a investigação científica depende de um "conjunto de procedimentos intelectuais e técnicos" para a obtenção de seus objetivos.

Estes procedimentos são os métodos científicos, definidos a seguir conforme Gil e Lakatos; Marconi (*apud* SILVA e MENEZES, 2001, p.25):

> *Método científico é o conjunto de processos ou operações mentais que se devem empregar na investigação. É a linha de raciocínio adotada no processo de pesquisa. Os métodos que fornecem as bases lógicas à investigação são: dedutivo, indutivo, hipotético-dedutivo, dialético e fenomenológico.*

Conforme definição de cada um dos métodos científicos citados e considerando que "não há apenas uma maneira de raciocínio capaz de dar conta do complexo mundo das investigações científicas" (SILVA e MENEZES, 2001) e que "o ideal seria você empregar métodos, e não um método em particular, que ampliem as possibilidades de análise e obtenção de respostas para o problema proposto na pesquisa" (SILVA e MENEZES, 2001), este trabalho foi desenvolvido utilizando aspectos de três métodos científicos, a saber: indutivo, hipotético-dedutivo e dedutivo.

O método indutivo, segundo Gil e Lakatos; Marconi (*apud* SILVA e MENEZES, 2001, p.26) é definido como:

> *Proposto pelos empiristas Bacon, Hobbes, Locke e Hume. Considera que o conhecimento é fundamentado na experiência, não levando em conta princípios*

> *preestabelecidos. No raciocínio indutivo a generalização deriva de observações de casos da realidade concreta. As constatações particulares levam à elaboração de generalizações.*

O método hipotético-dedutivo foi definido por Gil (*apud* SILVA e MENEZES, 2001, p.27):

> *Proposto por Popper, consiste na adoção da seguinte linha de raciocínio: "quando os conhecimentos disponíveis sobre determinado assunto são insuficientes para a explicação de um fenômeno, surge o problema. Para tentar explicar as dificuldades expressas no problema, são formuladas conjecturas ou hipóteses. Das hipóteses formuladas, deduzem-se conseqüências que deverão ser testadas ou falseadas. Falsear significa tornar falsas as conseqüências deduzidas das hipóteses. Enquanto no método dedutivo se procura a todo custo confirmar a hipótese, no método hipotético-dedutivo, ao contrário, procuram-se evidências empíricas para derrubá-la".*

Gil, 1999; Lakatos; Marconi, 1993 (*apud* SILVA e MENEZES, 2001, p.25) também define o método dedutivo:

> *Método proposto pelos racionalistas Descartes, Spinoza e Leibniz que pressupõe que só a razão é capaz de levar ao conhecimento verdadeiro. O raciocínio dedutivo tem o objetivo de explicar o conteúdo das premissas. Por intermédio de uma cadeia de raciocínio em ordem descendente, de análise do geral para o particular, chega a uma conclusão. Usa o silogismo, construção lógica para, a partir de duas premissas, retirar uma terceira logicamente decorrente das duas primeiras, denominada de conclusão.*

Considerando tais características dos métodos científicos, este trabalho usa o

método indutivo, à medida que parte de observações de casos da realidade particular de sistemas de manufatura para desenvolver um produto que possa ser aplicado em sistemas de manufatura com recursos compartilhados em geral.

Usa também o método hipotético-dedutivo, pois, foi criada a hipótese de que uma busca utilizando algoritmos genéticos para a resolução do problema de programação da produção para ambientes de manufatura com recursos compartilhados pode gerar soluções mais eficientes do que outros métodos de programação da produção.

Porém, pretende-se confirmar a hipótese, e não torná-la falsa, aspecto que remete ao método dedutivo.

Segue, em ordem cronológica, uma lista das atividades que foram desenvolvidas ao longo deste trabalho para obter os resultados finais:

- **Levantamento bibliográfico**
  - Foi feito levantamento bibliográfico sobre o que está sendo desenvolvido na área acadêmica e comercial para solução do problema de programação da produção em sistemas de manufatura com recursos compartilhados.

- **Modelagem do sistema de manufatura**
  - Os sistemas de manufatura referidos neste trabalho podem ser modelados em Redes de Petri, representando suas restrições e características, porém o modelo em Rede de Petri não é diretamente utilizado para gerar a programação da produção.

- **Método de resolução do problema**
  - O método de busca utilizado para gerar a programação da produção foi desenvolvido usando Algoritmo Genético. Para maiores informações sobre Algoritmos Genéticos, consultar o Apêndice A.

- ***Software* desenvolvido**
  - O *software* de programação da Produção foi desenvolvido usando linguagem UML (do inglês Unified Modeling Language, ou Linguagem de Modelagem Unificada) e Processo Unificado.
    - O Processo Unificado (PU) é um método utilizado para

desenvolvimento de *software* orientado a objetos, que se baseia no princípio do desenvolvimento iterativo, isto é, o sistema é desenvolvido em ciclos que são considerados mini-projetos de duração fixa com o sistema integrado, testado e executável.

- O PU utiliza a UML, uma linguagem para documentação de projetos de *software* orientados a objetos, como notação para representar o processo de desenvolvimento.
- Cada iteração do PU é composta pela Análise de Requisitos, Projeto, Implementação e Testes.
- Para maiores informações sobre UML e Processo Unificado, consultar Apêndice B.

- O *software* foi implementado usando a linguagem de programação Java e o Ambiente Eclipse.

➢ **Testes**

- Os testes foram realizados em alguns cenários propostos, de tamanhos diferentes, no laboratório do TEAR, no Departamento de Computação da Universidade Federal de São Carlos.
- Foram realizados testes sobre alguns cenários ou problemas de programação da produção, considerando o caso de uso principal do sistema, ou seja, o caso de uso "Gerar Programação", que leva em conta alguns tipos de produtos para serem produzidos e o compartilhamento de um determinado número de máquinas, existentes na fábrica, para a produção destes produtos.
- Os cenários ou problemas testados diferem entre si nas quantidades de máquinas da fábrica, tipos de produtos e roteiros de fabricação para cada produto.

1.5 Estrutura do Trabalho

Com base na metodologia empregada e para melhor compreensão do processo de desenvolvimento, este trabalho está dividido da seguinte maneira:

A Revisão de Literatura, que norteou a escolha e definição das estratégias a

serem usadas para resolver o problema em questão, encontra-se no Capítulo 2. Este capítulo está dividido em duas partes: a primeira parte fala sobre como a área acadêmica tem abordado a resolução do problema da programação da produção em ambientes com recursos compartilhados; a segunda parte mostra as principais funcionalidades de um *software* utilizados no mercado para resolver este mesmo problema.

Para compreensão do ambiente, do problema e da solução desenvolvida, a caracterização do problema, as restrições do ambiente de manufatura e a modelagem do método proposto para sua solução são apresentados no Capítulo 3. Constam neste capítulo, detalhes sobre o algoritmo genético usado para a busca.

As etapas de desenvolvimento do *software* necessário para testar a solução proposta, organizadas em ordem cronológica, são mostradas no Capítulo 4. Como já citado anteriormente este desenvolvimento foi feito usando linguagem UML e Processo Unificado. Para compreensão do *software* proposto, podem ser observados, neste capítulo, detalhes dos módulos implementados no *software* e suas funcionalidades.

Para entendimento de como o método e o *software* foram testados e como foram comparados os resultados obtidos e usados na conclusão deste trabalho, o ambiente de FMS utilizado para testes, é mostrado no Capítulo 5, no qual também são descritas as situações testadas, os critérios usados para comparação e os métodos que foram comparados.

Levando em consideração o desenvolvimento deste trabalho e tendo os resultados obtidos dos testes e comparações, são apresentadas, no capítulo 6, as conclusões e propostas para trabalhos futuros em prosseguimento a este.

Em seguida, são apresentados os apêndices, que contêm conhecimento teórico necessário para o desenvolvimento e entendimento deste trabalho. No Apêndice A são apresentados conceitos básicos de Algoritmos Genéticos, necessários para o entendimento do método proposto. No Apêndice B é apresentada uma visão geral do Processo Unificado, usado para o desenvolvimento do *software* que implementa o método proposto. Destaca-se aqui que, muito embora os assuntos tratados nos apêndices citados sejam necessários para o entendimento do trabalho, a alocação destes para a condição de apêndices foi pertinente devido à revisão de literatura ter sido feita com foco nos trabalhos recentes desenvolvidos sobre o assunto tratado nesta pesquisa.

2 USO DE BUSCA HEURÍSTICA E ALGORITMO GENÉTICO NA PROGRAMAÇÃO DA PRODUÇÃO DE SISTEMAS DE MANUFATURA COM RECURSOS COMPARTILHADOS.

Este capítulo está dividido em duas partes:

- Na primeira parte, são mostradas algumas pesquisas que resolvem o problema da programação da produção de sistemas de manufatura com recursos compartilhados. Primeiramente são mostradas pesquisas que usam modelagem do sistema de manufatura em redes de Petri, aplicando uma busca heurística ao modelo para realizar a programação da produção. Em seguida são apresentadas algumas pesquisas que usam algoritmos genéticos associados ao modelo do sistema de manufatura em PN e, por último, pesquisas que usam algoritmos genéticos para resolver tal problema, sem o uso de PN para modelar o sistema.
- Na segunda parte, são apresentadas as principais funcionalidades de um *software* utilizado no mercado para resolver este mesmo problema.

Algumas abordagens para solução de problemas de programação de sistemas de manufatura com recursos compartilhados levam em consideração a modelagem do problema e a estratégia de solução utilizada.

O uso de Redes de Petri (PN) para modelar problemas de programação de tais sistemas tem se mostrado adequado porque permite modelar suas características dinâmicas e restrições (REYES, YU e KELLEHER, 2000) através de habilitação de transições e regras de disparo e também fornecem ferramentas para análise de seu comportamento. Isto torna PN ideal para modelar problemas com flexibilidade de rotas e recursos compartilhados (LEE e DICESARE, 1992).

A combinação de PN e estratégias de busca heurística têm se mostrado um caminho promissor para a resolução destes tipos de problemas (REYES et al, 2002). A busca através do espaço de estados tem sido amplamente utilizada (LEE e DICESARE, 1992) e gera questões importantes como a explosão do espaço de estados e a representação concisa dos estados.

Neste sentido, Lee e Dicesare (1992) apresentam uma modificação do

algoritmo de busca A*, chamada de L1, que adapta a busca A* para estruturas de Redes de Petri temporizadas, com um intervalo de tempo k_i associado a cada lugar p_i da PN. O objetivo é encontrar a melhor solução (melhor caminho no grafo de estados) buscando minimizar o *makespan* . Assume-se que tempos de *setup* e transporte são embutidos no tempo de operação das máquinas.

A busca L1 expande parte do grafo de alcançabilidade, desde a marcação inicial m_0 até a porção do grafo que toca a marcação final m_f, apresentando uma programação ótima em termos de seqüência de disparo de transições. A seleção da próxima marcação a ser explorada é baseada na função heurística $f(m)=g(m)+h(m)$, onde $g(m)$ representa o *makespan* de m_0 até m e $h(m)$ representa uma estimativa do custo (em termos de *makespan*) da marcação m até atingir m_f, por um caminho ótimo. O custo de um arco na árvore de alcançabilidade que leva uma marcação m a uma marcação m' através do disparo da transição t é dado pelo máximo tempo de operações remanescentes dos lugares de entrada de t, que pode ser entendido como o máximo tempo de contingência das marcas antes de serem consumidas por t.

Lee e Dicesare (1994) observam que, embora pareça razoável que o custo de um arco na árvore de alcançabilidade de uma rede de Petri temporizada seja dado pelo máximo atraso k_i dos lugares de entrada p_i da transição t, esse argumento não é inteiramente válido, pois quando uma marcação m é convertida para m', mesmo os lugares que não são entrada para t têm seus respectivos tempos alterados.

Várias funções heurísticas foram testadas, inclusive $h(m)=0$, que corresponde à busca de custo uniforme e garante encontrar a solução ótima global. Para garantir que a busca encontre a solução ótima é necessário que a função heurística seja admissível, ou seja, deve haver um limite inferior para o *makespan* (Reyes et al, 2002). Esta abordagem encontra a solução ótima ou quase ótima e pode ser aplicada a sistemas grandes com muitos recursos e tarefas. Resultados comparativos de suas aplicações são apresentados posteriormente (LEE e DICESARE, 1994).

Lee e Dicesare (1994) implementaram A* e resolveram intratabilidade propondo uma função heurística que faz o algoritmo preferir marcações que são mais profundas no grafo de alcançabilidade.

Um problema a ser considerado quando se usa busca heurística é a explosão do espaço de estados, com a geração de muitas marcas a serem exploradas, o que torna difícil o

controle do esforço de busca. Algumas abordagens têm considerado a limitação do espaço de *backtracking* uma maneira de controlar o esforço de busca (REYES et al, 2002).

Reyes Yu e Kelleher (2000) propõem um algoritmo de estágios de busca, baseado em A*, que utiliza dois métodos: o gerador inteligente de sucessores (IGS), que evita a geração de caminhos não promissores e a busca de janela dinâmica (DWS), um algoritmo baseado em estágios de busca.

O IGS tem por objetivo reduzir o esforço de busca evitando a geração de programações que não levarão à solução ótima, através do uso de uma Agenda, uma lista de operações que podem ou não ser aplicadas em um estado, isto é, Agenda(m) é uma lista de transições que podem ou não ser disparadas em uma marcação m. Se a transição t_i disparar em m, uma nova marcação m' é gerada e com ela, uma Agenda(m'), com todas as transições remanescentes de m mais as transições potencialmente disparáveis em m'. Se t_i não disparar em m, ela é removida da Agenda(m) e pode aguardar até que uma mudança significativa (em termos de disponibilidade de recursos) ocorra no sistema, para ser habilitada novamente. Foram consideradas restrições de recursos (máquinas, *buffers* etc) e de tempo (tempo mínimo e máximo de espera nos *buffers*).

O outro método, o DWS, segue o esquema básico do A* e estágios de busca, onde o espaço de busca é visto parcialmente através de uma janela de tamanho limitado, uma subárvore, onde o algoritmo é aplicado, que desliza sobre a árvore de busca. O limite inferir da janela apresenta as marcações para as quais ainda é possível aplicar o *backtracking* e, quando a janela desliza (em direção aos nós-folha), tornam-se decisões irrevogáveis, diminuindo o esforço de busca. Para evitar que as marcações da janela aumentem exponencialmente na largura, é determinado um limite máximo para o número de marcações em um certo nível (profundidade) da janela.

As marcações são avaliadas por uma função heurística f(m). Quando uma nova marcação é gerada, ela pode ser incluída no nível l até o nível atingir o limite máximo estipulado, caso isso ocorra, apenas marcações com f(m) melhor que alguma m' já incluída no nível serão incluídas. Neste caso, a marcação m'', com o pior f(m´´) do nível será descartada, o que permite a exploração apenas das marcações mais promissoras. A janela avança quando o número de marcações no limite superior atinge um certo número máximo, ou quando todas as marcações do limite inferior já foram exploradas. Três funções heurísticas foram testadas e os resultados foram melhores, em termos de eficiência que os de Lee e Dicesare (1994).

Esta abordagem é aprimorada por Reyes *et al* (2002), através da busca de adiantamento de estágio dinâmico (do inglês *dynamic look-ahead stage search* - DLSS*), que utiliza basicamente a mesma idéia, com a introdução de alguns novos conceitos e assumindo que *buffers* são de tamanho infinito. A nomenclatura é alterada: o IGS agora é chamado de gerador controlado de sucessores (CGS) e o DWS, de quadro de busca (SF). Foram feitos estudos de otimalidade e eficiência e os resultados, comparados com Lee e Dicesare (1994) demonstraram certo grau de otimalidade sem aumento exponencial de custo.

Maggio (2005) propõe uma heurística para a programação de FMS usando modelagem em PN virtuais com tempo associado a transições, para representar duração de operações.

Para tal modelagem, é assumido que os elementos no chão de fábrica (máquinas, AGV´s) nunca falham, ou seja, todos os recursos envolvidos na produção estão disponíveis durante todo o processo. Assume-se, também, que o tempo de transporte é considerado desprezível e a capacidade de armazenamento de peças em *buffers* é infinita. Assume-se, ainda, que um recurso realiza uma operação por vez e retorna um único lote e, uma vez iniciada uma operação, essa não é interrompida. Os tempos de *setup* (a demora na configuração de uma máquina para uma dada operação), são embutidos no tempo de operação. Os tempos de processamento de operações são previamente conhecidos.

A estratégia de busca proposta por Maggio (2005), é uma variante do A*, baseado na versão proposta por Lee e DiCesare (1994). A busca é feita baseando-se no custo total estimado f(n), optando-se pelos nós de exploração menos custosos. O custo, no caso, é o tempo decorrido referente ao sistema modelado. O universo de alcançabilidade descreve assim o espaço de estados do problema, e as marcações temporizadas passam a ser os nós da árvore de busca. A função de geração de sucessores utiliza a extensão da regra de disparo de transição, que considera o tempo, para gerar os nós sucessores, novas marcações temporizadas.

Além disso, todas as possibilidades de disparo das transições usadas no modelo serão investigadas como decisões relevantes para a programação da produção, e pertencerão à agenda de ações obtida.

Abordagens utilizando métodos evolutivos, como Algoritmos Genéticos (AG), também têm sido propostas, associadas ou não, a modelos de PN para solução de programação de sistema de manufatura com recursos compartilhados. Há alguns aspectos

importantes a considerar no uso de busca com AG para obter um bom desempenho e cobrir todo o espaço de busca do problema.

Um desses aspectos, segundo Lacerda e Carvalho (1999), é a codificação do cromossomo, já que, em geral, cada cromossomo representa uma solução para o problema e deve representar um conjunto de parâmetros da função objetivo.

Outros aspectos importantes são o tamanho da população, os operadores genéticos de cruzamento e mutação e suas respectivas taxas. Yeung e Moore (2000), depois de revisão sobre os métodos de aplicação de AG em problemas de programação de produção, concluíram que a escolha da representação do cromossomo e da operação de cruzamento é crucial para o sucesso do método.

A seguir, são citadas algumas propostas para solução da programação da produção usando algoritmos genéticos.

Takahashi, Yamamura e Kobayashi, (1996) propõem um método de busca estocástica, chamado de busca adiada, associado a AG, para resolver problemas de alcançabilidade em PN.

Na busca adiada, a seqüência de disparo de transições é codificada como uma *string* de *bits*, onde cada *bit* representa uma transição. É assumido existir um vetor x, de inteiros não negativos, que atende a condição de alcançabilidade necessária, ou seja, cujos elementos são maiores ou iguais a 1, onde o elemento x_i determina o número de vezes que a transição t_i aparece, aleatoriamente, na *string*. Assim, o tamanho da *string* é igual à soma dos elementos do vetor x.

As transições são analisadas, uma a uma, iniciando da t_i mais à esquerda na *string*, em uma marcação inicial m_0. Quando t_i é disparável, um parâmetro p é incrementado e a próxima transição é analisada. Quando t_i não é disparável, ela é realocada para o final da *string*. As outras transições, que ainda não foram analisadas, são deslocadas um *bit* à esquerda e um parâmetro q é incrementado. A busca pára quando o valor de p é igual ao tamanho da *string* (todas as transições foram disparadas) ou o valor de $p+q$ é igual ao tamanho da *string* (todas as transições já foram analisadas).

Para utilização do AG, cada *string* representa um cromossomo da população. A função de *fitness* é dada pelo número de transições sucessivamente disparadas no cromossomo, partindo de m_0 e é obtida através do parâmetro p da busca adiada aplicada a cada cromossomo. O cruzamento é feito através de troca de subseqüências de genes entre dois cromossomos escolhidos aleatoriamente, sendo que as subseqüências de ambos devem ter o

mesmo tamanho e conter as mesmas transições. Após o cruzamento são selecionados os dois cromossomos com melhor valor de *fitness* entre os cromossomos pais e filhos.

A busca acaba quando o tempo acaba ou quando uma das soluções que maximiza a função de *fitness* é encontrada. Em experimentos realizados com busca adiada isolada e com busca adiada associada ao algoritmo genético, constatou-se que para problemas com grande espaço de busca a busca associada ao algoritmo genético é mais eficiente e confiável.

Cavalieri (1998) propõe uma estratégia de otimização de desempenho que leva em conta a seqüência de programação para atividades desempenhadas por cada máquina do sistema e o *work in process* (WIP) em cada ciclo de produção, isto é, o número de peças em cada ciclo de produção. Para tanto, utiliza modelagem por meio de PN, com tempo de operação associado a transições. Os ciclos de produção são modelados, ligando a transição que corresponde ao último processo de cada ciclo de produção à transição que corresponde ao primeiro processo do mesmo ciclo de produção por meio de um lugar.

Nesta abordagem, a seqüência de lugares e transições que representam atividades desempenhadas por uma máquina forma o *circuito de comando* daquela máquina. O número de marcas presente em cada circuito de comando é igual a 1, pois uma máquina só processa um trabalho por vez. Desta forma, o *throughput* (número de peças processadas por unidade de tempo) do sistema é limitado pela máquina que tem maior tempo de processamento.

Cavalieri (1998) utiliza AG para explorar o espaço de seqüência de programação para as atividades desempenhadas por cada máquina e utiliza um algoritmo heurístico de ajustamento (AHA), que determina o mínimo WIP em cada ciclo de produção para cada cenário de produção e retorna este valor como valor de *fitness*. Os cromossomos são codificados por um vetor de tamanho n*m de números inteiros, agrupados em m subvetores, onde m é o número de máquinas do sistema (número de circuitos de comando) e n é o número máximo de atividades processadas por cada máquina. Cada subvetor diz respeito a uma máquina específica e cada inteiro está associado a transições contidas em cada circuito de comando.

A operação de cruzamento é feita selecionando dois cromossomos pais e, para um subvetor correspondente em ambos, forma-se o subvetor dos filhos, alternando entre elementos de ambos os pais. A operação de mutação é feita pela troca de dois elementos, aleatoriamente escolhidos em cada filho. A busca pára quando a população converge.

O autor apresenta um exemplo de aplicação do algoritmo no qual, iniciando com um grafo, o AG encontra uma solução ótima que apresenta menor valor de WIP, mas não apresenta resultados de desempenho em comparação com outros métodos.

Outra abordagem, proposta por Yeung e Moore (2000), é focada na flexibilidade operacional, isto é, na habilidade de uma peça ser produzida em diferentes caminhos. Esta proposta visa a obter uma programação que melhora o desempenho do sistema em termos de reduzir o *makespan* total das ordens e maximizar a média de utilização das máquinas, por meio de seqüenciamento de trabalhos e designação de operações.

Nesta proposta, existem vários trabalhos a serem processados. Cada trabalho possui vários WOP (planos de operação de trabalho), que são possíveis seqüências de operações para realizar aquele trabalho. Cada operação, por sua vez, pode ser realizada em mais de uma máquina.

Os autores apresentam várias razões para a escolha de AG como ferramenta para tal programação: por AG apresentar uma metodologia 'robusta', que não necessariamente resulta em soluções ótimas, mas obtém um nível alto de resultados sobre uma gama imensa de problemas. Por possibilitar facilidade de implementação de sistemas híbridos e fácil integração com várias técnicas de busca, simulação e técnicas matemáticas. Por terem encontrado na literatura muitos registros de aplicações de AG para várias classes de problemas de programação de produção que obtiveram sucesso. Por AG poder ser usado em ambiente distribuído, com memória compartilhada e outras arquiteturas de computação paralela.

Nesta proposta, cada cromossomo é representado por uma lista de inteiros positivos, onde cada inteiro representa um número de ordem de trabalho, na seqüência que será processado. Tabelas de consulta interna com os WOPs para cada trabalho e tabelas com as máquinas que podem ser designadas para cada operação são mantidas, para fins de seleção. A população inicial deve ter entre 20 e 30 indivíduos. Os cromossomos são gerados aleatoriamente, bem como o WOP para cada trabalho e a máquina designada para cada operação, entre as máquinas disponíveis, é selecionada aleatoriamente.

O cruzamento é feito entre dois cromossomos pais p_1 e p_2, gerando dois cromossomos filhos f_1 e f_2 e é determinado por um vetor de bits aleatório, com bits 0's e 1's. Os trabalhos do cromossomo pai p_1 cuja posição correspondente no vetor de bits é '1' serão mantidos no cromossomo filho f_1. Os outros trabalhos sofrerão cruzamento, baseado na ordem em que os trabalhos aparecem no cromossomo p_2, sem repetir o trabalho. Esse processo é

repetido para gerar os dois filhos. Depois de cruzados, um dos trabalhos de cada um dos cromossomos filhos é selecionado aleatoriamente e um novo WOP é selecionado também de modo aleatório e, por sua vez, as operações são designadas de forma aleatória para máquinas disponíveis.

A mutação é aplicada a cada filho, selecionando dois trabalhos e trocando suas posições. O valor de *fitness* de cada cromossomo é obtido utilizando simulação. Resultados experimentais foram obtidos e o melhor foi com população igual a 20, taxa de cruzamento igual a 80% e taxa de mutação igual a 2%, no qual o *makespan* total apresentou uma melhora de 50% em comparação com busca cega.

Os autores também apresentam um estudo de caso no qual um planejamento de processo flexível e o programador baseado em AG são integrados com um sistema de controle de células modelado em uma PN colorida, para implementar um sistema tolerante a falhas.

Uma abordagem chamada de programação adaptativa tem sido proposta para problemas com volume alto de produção. A idéia chave consiste em dividir o problema em partes, gerando sucessivas programações parciais, onde cada programação resolve totalmente uma parte do problema, e depois as partes são combinadas de alguma maneira para resolver o problema todo.

Partindo deste princípio, Chiu e Fu (1997) propõem uma busca com algoritmo genético embutido, que realiza uma programação parcial, um segmento de programação, que é traduzido em forma de uma seqüência de transições e aplicado ao modelo de PN.

Nesse método, um cromossomo c_i é gerado diretamente de um modelo de PN e é formado por uma coleção de listas $c_i = \{l_1, l_2, ..., l_n\}$, onde cada lista contém uma seqüência de transições, aleatoriamente ordenadas, cuja ordem representa a prioridade das transições naquela lista. Um lugar p que é lugar de entrada para mais de uma transição terá uma lista dessas transições associada a ele. O lugar p_i pode representar uma seleção de plano para o trabalho J_i, a designação de um recurso para o trabalho J_i ou a competição por um recurso R_i.

Um construtor de programação transforma um cromossomo em uma programação factível, através da identificação de transições habilitadas na marcação m e comparação com a respectiva lista no cromossomo c_i. A função de *fitness* é aplicada em duas etapas: a função de *fitness* bruta, aplicada a cada cromossomo, que é uma operação sobre os valores de *makespan* máximo, mínimo e do indivíduo na geração atual e atribui valores de *fitness* mais altos para indivíduos com *makespan* mais baixo. O valor de *fitness* obtido f_b é,

então aplicado à função de *fitness* escalada $f = a * f_b + b$. Os valores de a e b são escolhidos de forma a garantir que indivíduos com melhor valor de *fitness* sejam copiados mais vezes para a população intermediária, antes do cruzamento e mutação e que indivíduos com valores de *fitness* médios sejam copiados uma vez, já que a seleção utiliza o valor de f. A probabilidade de um cromossomo c_i ser selecionado é proporcional a seu valor de *fitness* escalado, dado por $prob_i = f_i / \sum f_i$ e o número de vezes que ele é selecionado para formar a população intermediária é a parte inteira do valor de $e_i = prob_i * tamanho\_da\_população$.

O cruzamento é feito entre dois cromossomos c_i e c_j escolhidos aleatoriamente da população intermediária, de maneira que, para cada lista l_i de c_i é decidido, por probabilidade se haverá cruzamento de um ponto com a respectiva lista l_j de c_j. Depois de cruzados, os novos cromossomos c_i' e c_j' serão avaliados e os valores de *fitness* comparados, respectivamente com c_i e c_j. Os cromossomos com maiores valores de *fitness* irão para a população intermediária. O operador de mutação é aplicado com probabilidade baixa, incorporado no procedimento de cruzamento, onde para cada elemento de cada lista l_i é aplicada a probabilidade e se tiver sucesso, um valor válido aleatório é atribuído ao elemento.

A busca com algoritmo genético embutido é aplicada a cada subproblema, que é formado por uma mistura de partes de trabalhos J_i´s, que têm prioridades uns sobre os outros definidas por alguns parâmetros. Cada programação parcial gerada é traduzida e aplicada no modelo de Rede de Petri e um novo subproblema é definido, até que todos os problemas J_i tenham sido solucionados.

Reyes, Yu e Lloyd (2001) propõem a divisão do problema total da programação em subproblemas, por tamanho de lote ou agrupamento de trabalhos, ou ambos. Cada subproblema é resolvido utilizando DLSS*, resultando em uma coleção de diferentes soluções quase ótimas para cada subproblema, que, juntas, formam uma população inicial de subprogramações, na qual cada membro tem informações sobre quanto do problema total resolve e como.

Cada subprogramação é codificada em um cromossomo, que tem associado a ele uma seqüência de disparo de transições, que representa a subprogramação sobre o modelo de PN. Um construtor de programação junta duas seqüências de transições, formando uma nova subprogramação (um novo indivíduo), próxima da que resolve o problema todo.

A função de *fitness* associada a cada indivíduo favorece indivíduos com menor *makespan* e é dada por: $f(e) = h(e) + g(e)$, onde, novamente, $h(e)$ é o *makespan* de m_0 até m_e e $h(m_e)$ é a estimativa do *makespan* de m_e até a marcação final m_f. Indivíduos com menor

valor para f(e) têm maior probabilidade de serem selecionados para a próxima geração. Testes realizados, comparados com A* obtiveram melhor desempenho, o que, indica o uso de algoritmos genéticos para trabalhos futuros, segundo Reyes Yu e Lloyd (2001).

Algumas abordagens têm tentado evitar *deadlocks* no sistema, através da análise estrutural do modelo de PN, como a abordagem proposta por Gang e Wu (2002), que, gera uma matriz de incidência de sinal da PN, que é uma variação da matriz de incidência, onde os elementos aij são 0, + , - ou ± e, com base nessa matriz e em informações da marcação inicial m_0, pode identificar marcações onde haverá *deadlocks*, pela presença de sifões vazios na marcação e descartar os cromossomos referentes a tais marcações, antes de aplicar quaisquer operadores (cruzamento e mutação).

O cromossomo é codificado, nesta abordagem, como um conjunto de máquinas colocadas em paralelo, onde cada máquina contém a designação de operações para esta máquina, através de alguns parâmetros. A população é inicializada e a função de *fitness* aplicada. A função de *fitness* é o mínimo *makespan*, onde, para cada cromossomo com M máquinas, é calculado o tempo que cada máquina leva para desempenhar as operações designadas e depois o tempo máximo entre eles. Esta abordagem não admite rotas alternativas de máquinas, mas o método pode ser usado para casos com muitas máquinas e trabalhos extensos.

Huang e Wu (2004) propõem um modelo de programação livre de *deadlocks* baseado em algoritmo genético e redes de Petri também com uso de matriz de sinal de incidência e detecção de sifões vazios. Uma política de detecção de *deadlocks* é adotada e os *deadlocks* são detectados antes do disparo das transições que levam a eles. Detectado um *deadlock*, a seqüência de transições pode ser reprogramada.

Abordagens utilizando PN coloridas são propostas especificamente para sistemas de manufatura de circuitos integrados (*wafers)*, mas que podem ser extendidos para resolver outros problemas de programação (Lin et al, 2003).

Uma ferramenta de modelagem proposta por Huang et al (2002), chamada QCPN (Filas de Redes de Petri Coloridas), que combina redes de Petri coloridas temporizadas (CTPN) com sistemas de filas. Esta modelagem é utilizada juntamente com um programador baseado em AG para encontrar a combinação ótima de um número de regras heurísticas. O

sistema é utilizado para prever datas de entrega.

No modelo de CTPN é introduzido um tipo especial de transição, chamado *transição de fila.* Este tipo de transição tem um modelo de fila embutido que calcula o tempo de espera associado a cada cor de marca que habilita a transição (prioridade das marcas). Após esse tempo, a transição pode disparar, consumindo aquela marca.

Os lotes de produtos são classificados por prioridade, segundo um método, sendo os mais urgentes com prioridades mais altas. Essas prioridades são utilizadas para calcular os tempos de espera. O programador baseado em AG tem por objetivo minimizar o tempo de ciclo de produto e o *work in process* e encontrar restrições de data devida. O programador utiliza AG para encontrar as regras apropriadas de liberação de lote, de seleção de máquinas e de despacho.

Os cromossomos são codificados em números inteiros, com três tipos de genes, que representam, respectivamente, política de liberação de lote, regra de seleção de máquina e regra de despacho, sendo a última uma combinação de valores de prioridade de várias regras, com um valor normalizado para cada regra. Os códigos dos genes são diretamente relacionados ao modelo QCPN, sendo que a regra de despacho é relacionada ao QCPN pela seleção de uma fila de prioridade apropriada para cada lote.

A função de *fitness* é formada pela soma de pesos de três funções: média de tempo de ciclo, grau do *throughput* avaliado e grau da perda devida ao número de ordens que falharam em encontrar a data devida. O desempenho para cada política de programação é avaliado através de análise matemática sobre o modelo QCPN. Em experimentos realizados, comparada com outras regras de prioridade, a abordagem proposta obteve melhor desempenho.

Nesta mesma linha, Lin et al (2003) usam CTPN (Redes de Petri temporizadas coloridas) para modelar sistemas de centro de sondagem de *wafer*. O objetivo do processo de sondagem é garantir que *wafers* são livres de defeitos, usando equipamentos de teste automáticos para examinar cada circuito e determinar se está operando na especificação correta.

Na fase de programação, são utilizados dois métodos: selecionar máquinas mais adequadas, baseadas em sua situação corrente e designá-las para lotes por meio de algumas *regras de seleção de equipamento* (ESR) e selecionar lotes adequados e direcioná-los para máquinas de acordo com algumas *regras de seleção de lotes* (LSR). Em cada método, AG é utilizado para buscar uma combinação ótima das regras. O Programador tem

por objetivo minimizar o tempo de ciclo e o WIP e maximizar a utilização de equipamento.

Os cromossomos são representados por inteiros positivos, com dois tipos de genes: ge e gl, que representam, respectivamente, ESR e LSR. Cada gene é formado por uma combinação de regras, com valores inteiros normalizados que representam a prioridade de cada regra. A função de *fitness* é formada pela soma de pesos de cinco funções: média de tempo de ciclo, avaliação do objetivo encontrado, grau do *throughput* avaliado, grau da perda devida ao número de ordens que falharam em encontrar a data devida e utilização pura.

O desempenho é medido através de simulação no sistema. Foram feitos experimentos e comparados com regras LSR, ESR e a abordagem proposta obteve melhor desempenho.

Há algumas pesquisas que abordam o uso de AG para resolver o problema da programação da produção de sistemas de manufatura com recursos compartilhados não associados a modelo em Redes de Petri.

Pongcharoen, Hicks e Braiden (2004) propõem um algoritmo genético modificado em relação ao funcionamento geral dos AGs, para programação da produção de produtos complexos com estrutura de produto de múltiplos níveis, considerando restrições de recursos e relações de precedência entre seqüências de operações e montagens.

Neste algoritmo, após a aplicação dos operadores genéticos, há um processo de reparo nos cromossomos, que corrige programações não factíveis que possam ser geradas pela aplicação do cruzamento e mutação. Os cromossomos são codificados como *strings* alfanuméricas, onde cada gene tem três partes: um identificador da estrutura do produto, um identificador da instância do produto e o número da operação. Os genes são aleatoriamente seqüenciados para formar cada cromossomo da população. Cada cromossomo é subdividido em n subcromossomos, que representam seqüências de operações para n recursos.

As operações genéticas são aplicadas em cada subcromossomo para gerar seqüência de operações para cada máquina. Nos cromossomos selecionados aleatoriamente, é aplicado cruzamento de um ponto, que consiste em selecionar de maneira aleatória um ponto que divide os cromossomos pais em duas partes. A primeira parte do primeiro pai é diretamente copiada para o primeiro filho. Os genes remanescentes são obtidos do segundo pai. O processo é repetido em ordem reversa para produzir o segundo filho. Mutação inversa é aplicada dentro de cada subcromossomo, respeitada a probabilidade, que consiste em selecionar aleatoriamente dois pontos no subcromossomo e realocar os genes localizados

entre esses dois pontos em ordem reversa.

Após as operações genéticas, é aplicado um processo de reparo, pois os cromossomos gerados podem representar programações não factíveis. O processo de reparo tem quatro estágios: ajustamento da operação precedente, ajustamento da precedência das peças, temporização da tarefa e considerações de capacidade finita e, por último, ajustamento de *deadlock*.

A função de *fitness* considera os custos de multas, tanto por adiantamento, que tem custo de armazenamento (de matéria prima, de produtos em processo e de produtos acabados) associado, como por atraso nas entregas (somente de produtos acabados), que gera multas. Em uma programação ideal, estes custos, ou seja, o valor de *fitness* deve ser igual a zero. A função de *fitness* considera o custo total como sendo a soma do custo de adiantamento para todos os componentes, do custo de adiantamento para todos os produtos e do custo de atraso para todos os produtos.

A seleção dos indivíduos para a próxima geração é feita pelo método da roleta, onde a probabilidade de sobrevivência e o número de réplicas de um indivíduo na próxima geração é proporcional ao seu valor de *fitness*. Em experimentos aplicados a três problemas com tamanhos diferentes com relação à quantidade de operações de usinagem e montagem e à quantidade de recursos, verificou-se que o número de cromossomos da população e o número de gerações determinam o tempo de execução, além disso, juntamente com as probabilidades de cruzamento e de mutação, influenciam na convergência e na qualidade das soluções encontradas.

O fator principal que influencia nos custos de multas é o tamanho do problema. A qualidade da solução é inversamente proporcional ao tamanho do problema. Obtiveram-se melhores resultados com tamanho de população igual a 40 cromossomos do que com 20 cromossomos e com 40 gerações do que com 20 gerações. Outro fator importante foi a probabilidade de mutação, os melhores resultados foram obtidos com valores entre 0,06 e 0,1. A probabilidade de cruzamento não foi estatisticamente significante para os testes realizados, com valores entre 0,6 e 0,9.

O processo de reparo é um fator crítico para corrigir programações não factíveis. Todos os cromossomos em cada geração são corrigidos. As programações geradas pelo AG mostraram custos de multas muito mais baixos do que os métodos tradicionais usados nas companhias, mostrando mais de 80% de redução nos custos.

Pongcharoen *et al* (2002) propõem um experimento deste AG aplicado a uma

manufatura de bens de capital, para identificar valores dos parâmetros do AG, que produzem os melhores resultados com um número total de cromossomos gerados fixo. O objetivo é identificar os valores dos parâmetros do AG mais eficientes para resolver um problema grande e computacionalmente custoso, com tempo de execução específico.

Foram testadas três combinações de valores de tamanho da população e número de gerações, restritos a um número total de 1200 cromossomos gerados. Os valores de tamanho da população e número de gerações testados foram, respectivamente: 20 e 60; 40 e 30; 60 e 20. Os melhores resultados foram obtidos com tamanho da população igual a 60 cromossomos e 20 gerações. O processo de reparo dos cromossomos teve impacto crítico na geração de programações factíveis. Os resultados das programações geradas pelo AG e passadas pelo processo de reparo foram comparados com programações geradas aleatoriamente e passadas pelo processo de reparo. As programações geradas pelo AG foram melhores que as geradas aleatoriamente.

O AG foi aplicado a um problema grande e custoso computacionalmente, com população de 60 cromossomos, 20 gerações, taxa de mutação de 18% e probabilidade de cruzamento de 0,6 e 0,9 e os resultados foram comparados com a programação da companhia. A programação da companhia resultou em atraso na entrega e custos de multa por atraso. A programação resultante do AG gerou entrega pontualmente, mas gerou multa de menor valor, por adiantamento, relacionada a custos de armazenamento. No problema testado, o melhor resultado foi obtido com probabilidade de cruzamento de 0,6, não gerando atraso na entrega e atingindo uma redução de custos de 63% em relação à programação feita pela companhia. A probabilidade de cruzamento, embora tenha sido estatisticamente insignificante anteriormente, teve impacto neste caso particular, o que indica o estudo, para trabalhos futuros, da relação entre parâmetros do AG e o tamanho do problema.

Jeong, Lim e Kim (2006) propõem um método híbrido, combinando AG e simulação para resolver o problema da programação da produção, sendo o AG configurado para decidir programações ótimas para minimizar o *makespan* e o modelo de simulação utilizado para encontrar o tempo de execução das programações fixadas pelo AG. O ambiente é composto de m máquinas e n trabalhos, com rotinas de processamento específicas (operações) a serem seguidas. Cada operação de cada trabalho tem uma precedência e leva um tempo determinístico em uma determinada máquina. O tempo de início de operação de cada trabalho está sujeito ao tempo disponível e à data devida do trabalho.

Uma solução factível é uma programação que satisfaz às restrições de

precedência de operações em um mesmo trabalho e à restrição de que uma máquina só pode processar uma operação de cada vez, ou seja, processar um trabalho de cada vez, dentre os trabalhos que estão na fila de espera. Para um problema com n trabalhos e m máquinas, um cromossomo é do tamanho nxm.

A representação do cromossomo é baseada em operações, sendo uma programação codificada como uma seqüência de operações, uma seqüência de números naturais, de 1 até nxm, onde cada gene representa uma operação. A operação de cruzamento é feita selecionando uma *substring* aleatoriamente de um pai, encontrando sua posição no outro pai e copiando os itens restantes para o segundo pai, na ordem em que eles aparecem no primeiro pai. A mutação é feita selecionando duas posições em um cromossomo e trocando seus conteúdos.

A seleção de indivíduos para cruzamento usa um mecanismo baseado em categoria, no qual a probabilidade de seleção não está diretamente ligada ao valor de *fitness* do indivíduo, mas é uniformemente curva, evitando que bons indivíduos dominem a evolução precocemente. A função objetivo é o mínimo *makespan*. O *makespan* de cada cromossomo é produzido pelo processo que atribui operações para as máquinas, pela seqüência de cada trabalho, examinando o cromossomo da esquerda para a direita. O *makespan* objetivo é selecionado por comparação dos desempenhos das programações.

Como o *makespan* obtido pelo AG é baseado em tempos fixos de operações, o que não acontece em sistemas reais, no modelo de simulação são incorporadas características dinâmicas do sistema, como quebra e reparo de máquinas, enfileiramento e espera de produtos. São distribuídos tempos médios de quebra e reparos a máquinas aleatoriamente. O tempo de conclusão é o tempo de processamento no modelo de simulação, ou seja, o tempo total de simulação gasto no sistema para processar todas as operações, baseado na programação da produção do AG.

Em seguida, o tempo de operação do AG é ajustado pelos resultados da simulação e o modelo AG gera novas programações pelo tempo de operação ajustado. Este processo segue até que a taxa de diferença entre o tempo da simulação precedente e o tempo da simulação corrente seja pequeno o suficiente para ser aceitável.

Chan, Chung e Chan (2005) propõem uma idéia chamada de Genes Dominantes (DGs) em AG para resolver problemas de programação da produção de FMS com rotas de produção alternativas. Neste problema, um trabalho tem várias operações e cada operação pode ser processada em uma máquina, entre as máquinas adequadas. A variável de

decisão é o tempo de determinada operação, de determinado trabalho, em uma máquina.

Há duas funções objetivos possíveis: minimizar o *makespan* e minimizar o atraso dos trabalhos, onde cada trabalho tem um peso que representa sua importância. Quanto mais o atraso de um trabalho aumenta, este trabalho se tornará mais crítico.

Há algumas restrições definidas para este modelo: o tempo de início de cada operação só pode ser processado depois de sua operação precedente. Uma vez que uma operação começa, ela será concluída sem interrupção. Cada operação é processada em apenas uma máquina. Cada operação é processada em apenas uma máquina a cada unidade de tempo. Cada máquina processa apenas uma operação a cada unidade de tempo.

O cromossomo é codificado da seguinte maneira: cada gene consiste de quatro números inteiros que representam, respectivamente, máquina, trabalho, operação e dominação (1 para dominante e 0 para normal). Um cromossomo contém vários genes referentes a várias operações de vários trabalhos. A prioridade de programação dos trabalhos nas máquinas é definida pela ordem dos genes no cromossomo: maior prioridade à esquerda, menor prioridade à direita.

A idéia de Genes Dominantes é proposta para melhorar o desempenho da busca e sua função é representar os genes fortes no cromossomo. Na população inicial, alguns genes são aleatoriamente designados como DGs. Cada cromossomo pode conter mais de um DG. Durante evoluções, apenas aqueles DGs sofrem cruzamento em cada par de pais para gerar um par de filhos. Cada filho preserva a maior parte dos genes de um dos pais e herda apenas dos DGs do outro pai. Se estes DGs herdados tornam o filho mais forte do que o pai (com melhor valor de *fitness*), eles se tornarão DGs no filho, senão, eles se tornarão genes normais. Esta idéia assegura que os melhores genes serão passados ao filho.

Há dois operadores de mutação: no primeiro, um par de genes no mesmo cromossomo é aleatoriamente selecionado e trocado, alterando, assim, as prioridades de programação das operações dos trabalhos. No segundo tipo de mutação, alguns genes são aleatoriamente selecionados e mutados, aleatoriamente no parâmetro “máquina”, com o propósito de aumentar a diversidade genética. Em ambos os tipos de mutação, se o cromossomo, após a mutação, é mais forte que antes, o gene mutado torna-se DG.

É utilizada uma técnica de elitismo para impedir que o melhor cromossomo se perca. O melhor cromossomo é identificado e gravado. Caso ele se perca ou se torne fraco, depois da evolução, ele será inserido na população intermediária para a próxima evolução.

Foram modelados dois exemplos de problemas de programação de FMS com rotas alternativas de produção. O primeiro exemplo, proposto originalmente por Lee e

Dicesare (1994) tem 5 trabalhos, 3 máquinas e 4 operações em cada trabalho. Algumas operações podem ser feitas em mais de uma das três máquinas, devendo ser selecionada uma. A função objetivo é minimizar o *makespan*. Lee e Dicesare (1994) alcançaram *makespan* de 439, enquanto o *makespan* alcançado aplicando o método proposto por Chan, Chung e Chan (2005) foi 360.

No segundo exemplo, num cenário com 127 trabalhos, 33 máquinas e número máximo de 7 operações por trabalho, tendo como objetivo minimizar o atraso, o método proposto também mostrou melhores resultados que outras pesquisas que utilizaram AG e AG adaptativo.

Por ser um problema real das manufaturas, há algumas soluções de *software* disponíveis no mercado, para realizar a programação da produção de sistemas de manufatura com recursos compartilhados. Algumas características foram observadas em um destes *softwares*, que serão citadas a seguir.

Com relação às funcionalidades, tal *software* permite fazer seqüenciamento das operações manualmente (utilizando o mouse) ou automaticamente, seguindo um dos critérios: nível de prioridade, data de entrega ou ordem de chegada. O *software* oferece três diferentes métodos de programação automática: para frente (tão logo o recurso esteja disponível, antecipando a produção), para trás (a partir da última operação, de acordo com a data de entrega, informa a data limite para cada operação), bi-direcional (a partir de uma operação intermediária previamente selecionada).

Pode-se cadastrar horários de folga do calendário (almoço, noite, etc.) e tratar imprevistos como quebra de maquinas (editar no calendário). Com relação às informações disponibilizadas, o *software* indica operações que serão concluídas com atraso em relação à data final desejada, mostra o nível de ocupação de recursos de produção para um determinado período e permite visualizar e ajustar utilização de recursos secundários.

A interface do *software* é amigável, possuindo os seguintes menus: Ajuda, Editar ordens de produção, Gerar programação (principal), Visualizar gráfico de Gantt, Visualizar gráficos de ordem de produção (as operações são agrupadas por ordem de produção), Manutenção de turnos, Manutenção de bancos de dados e Opções adicionais. A tela principal do *software* é Gerar Programação, e está disposta da seguinte maneira: no eixo vertical são mostrados os recursos de produção, no eixo horizontal é mostrada a escala de tempo (em dias e horas), há uma janela com as ordens de produção ainda não programadas,

onde cada operação de fabricação é representada por um ícone. As operações (ícones) podem ser manipuladas com o mouse.

Comentários finais

Neste capítulo foi apresentado um levantamento de trabalhos propostos para a Programação da Produção de sistemas de manufatura com compartilhamento de recursos. Foram apresentadas, também, algumas características de *software* disponível no mercado para solucionar este problema.

No próximo capítulo será apresentado o método de busca com AG proposto para a programação da produção de sistemas de manufatura com recursos compartilhados e seu funcionamento.

3 UMA BUSCA BASEADA EM ALGORITMOS GENÉTICOS PARA PROGRAMAÇÃO DA PRODUÇÃO DE SISTEMAS DE MANUFATURA COM RECURSOS COMPARTILHADOS

Sistemas de manufatura com compartilhamento de recursos, tais como máquinas e AGVs têm sido utilizados para automação da produção permitindo flexibilidade condizente com as necessidades de uma boa parcela de manufaturas do mercado. Esta flexibilidade se dá porque, nestes sistemas, máquinas podem ser utilizadas para várias operações diferentes e produtos, às vezes podem ser processados por várias máquinas diferentes. O problema é que esta flexibilidade toda exige um esforço considerável para definir a programação da produção usando os recursos disponíveis ao longo do tempo, com o objetivo de satisfazer certos critérios de desempenho.

Nesse sentido, várias abordagens têm sido propostas, enfocando dois aspectos, como já comentado: a modelagem do problema e a estratégia de solução. Para o primeiro aspecto, Redes de Petri têm se mostrado uma ferramenta conveniente, pois permite modelar as características do comportamento dinâmico de um FMS e suas restrições. Além disso, PN permitem que algumas técnicas de análise e busca para a resolução do problema sejam associadas a elas.

No grupo de trabalho do TEAR, uma heurística de busca, variante da A*, para a programação de FMS usando modelagem em PN virtuais temporizadas foi proposta por Maggio (2005). A busca é feita baseando-se no custo total estimado, optando-se pelos nós de exploração menos custosos em relação ao tempo decorrido referente ao sistema modelado. O critério de desempenho escolhido é o *makespan*. O próprio autor sinaliza que, para trabalhos futuros poderá ser utilizado algoritmos genéticos para melhorar o desempenho e a rapidez na busca pela solução, especialmente tratando-se de problemas com grande espaço de busca, nos quais a aplicação de algoritmos derivados do A* não se mostra eficiente por causa da explosão do espaço de estados.

Nesta linha, este trabalho propõe uma abordagem de programação da produção para sistemas de manufatura com compartilhamento de recursos, que podem ser modelados em Redes de Petri, pela aplicação de uma estratégia de busca utilizando algoritmos genéticos (AG) para a solução do problema, porém, a estratégia de busca proposta não está diretamente associada ao modelo do sistema de manufatura em PN.

Neste capítulo serão definidas as características e restrições do sistema e, em

seguida, proposta a modelagem do algoritmo genético para a solução do problema de programação da produção.

3.1 Caracterização e restrições do sistema

Inicialmente, é necessário definir as características e restrições do problema a ser tratado. Algumas considerações que implicarão na complexidade da modelagem do problema e no desempenho da estratégia de solução serão feitas a seguir.

O ambiente no qual é aplicada a proposta é um ambiente de manufatura com compartilhamento de recursos, no qual há várias máquinas e vários tipos de produtos que podem ser produzidos, devendo, para isso, seguir um roteiro de fabricação. Um roteiro de fabricação é uma seqüência de operações que deve ser seguida na ordem em que se encontra, onde cada operação é uma etapa da fabricação do produto e, ao final da seqüência o produto estará pronto. Pode haver mais de uma máquina possível para executar uma mesma operação, o que leva um produto a poder ter mais de um roteiro de fabricação diferentes. Assim, os roteiros de fabricação de cada produto são representados pelas máquinas através das quais ele deve passar para ser produzido.

Além disso, assume-se que:

(i) Um tipo de produto será produzido seguindo um único roteiro de fabricação, ou seja, se houver vários roteiros possíveis, deverá ser selecionado apenas um;

(ii) Um determinado produto deverá passar por todas as máquinas do roteiro de fabricação escolhido, sem alteração da ordem;

(iii) Uma operação será executada em apenas uma máquina, em um determinado tempo, ou seja, um produto passará por apenas uma máquina em determinado tempo;

(iv) Uma máquina processará apenas um produto por vez;

(v) Tempos de operação dos produtos nas máquinas são determinísticos e previamente conhecidos;

(vi) Máquinas nunca falham;

(vii) Uma vez que uma determinada operação é iniciada (que um determinado produto passa por uma máquina), ela não é interrompida até que seja concluída;

(viii) Tempos de transporte são considerados como parte do tempo de operação;

(ix) Existem áreas de armazenamento intermediário, com capacidade infinita entre as máquinas;

(x) Tempos de *setup* de máquinas estão incluídos nos tempos de operação.

Neste trabalho, os critérios adotados para definir o desempenho da produção são baseados na obtenção do mínimo *makespan*, ou seja, o mínimo tempo total de processamento de todos os produtos e no tempo de execução da busca.

3.2 Modelagem do Algoritmo Genético

No presente trabalho, um cromossomo é codificado de maneira a indicar uma programação da produção, uma solução completa para o problema, A solução encontrada é 'traduzida' para uma programação da produção através da leitura dos genes do cromossomo. Alguns parâmetros do AG são variáveis e influenciam no comportamento e desempenho da busca. Tais parâmetros são: tamanho da população, taxa de cruzamento e taxa de mutação.

O Algoritmo Genético funcionará da seguinte maneira: a população inicial será gerada, obedecendo aos produtos a serem produzidos e possíveis roteiros de fabricação para cada produto. Cada cromossomo, ou seja, cada possível solução para a programação da produção, será avaliado de acordo com o seu *makespan* e um valor de *fitness*, que representa a qualidade daquela solução, será atribuído a ele.

Em seguida serão selecionados cromossomos para aplicar-se cruzamento e mutação, para gerar novos indivíduos para a próxima população. O melhor cromossomo de cada geração será selecionado para a população intermediária, para, de acordo com a probabilidade, sofrer cruzamento e mutação e gerar descendentes para a próxima geração. Este tipo de elitismo foi aplicado para que o melhor cromossomo tenha mais chances de ir para a população da próxima geração sem que a busca leve a soluções ótimas locais.

O AG pára quando um determinado número de gerações é atingido ou quando o algoritmo converge, ou seja, a média das últimas três populações é igual.

3.2.1 Codificação do cromossomo

Seja n o número de produtos a serem produzidos, s o número máximo de

roteiros possíveis para cada produto, e t o número máximo de operações em todos os possíveis roteiros de fabricação para os n produtos. Um cromossomo é representado por um vetor de inteiros de tamanho n+n*t, subdividido em n subvetores de tamanho t+1.

Cada subvetor representa um produto a ser processado, da seguinte forma: o primeiro gene do subvetor é o índice que identifica o produto representado pelo subvetor, e os t genes seguintes representam as máquinas do roteiro de fabricação escolhido para aquele produto.

Defina-se P_{i_p} como o produto a ser processado e $M_{i_p,K_{ip}}$ como uma máquina do roteiro de fabricação deste produto, onde K_{ip} representa a ordem da máquina no roteiro do produto, ou seja, a K_{ip}-ésima máquina pela qual o produto P_{i_p} deverá passar. Assim, temos uma representação completa do cromossomo da seguinte maneira:

P_{i_1}	$M_{i_1,1}$	$M_{i_1,2}$	$M_{i_1,K_{i1}}$	P_{i_2}	$M_{i_2,1}$	$M_{i_2,2}$	$M_{i_2,3}$	$M_{i_2,K_{i2}}$	...	P_{i_p}	$M_{i_p,K_{ip}}$

Figura 1 – Representação completa do cromossomo

Onde:

$i_1, i_2, i_3, ..., i_p \in \{1, 2, ... n\}$ & $i_g \neq i_j$, $g \neq j$.

$K_{ip} \in \{1, 2, ... t\}$ & $K_{ip_g} \neq K_{ip_j}$, $g \neq j$.

Para que cada subvetor tenha tamanho fixo t+1, caso o subvetor represente um produto com $K_{ip} < t$, há a necessidade de inserção de genes vazios, sendo preenchidos com o valor 0 entre o gene da última máquina do roteiro e o indicador do próximo produto no cromossomo.

Para fins de implementação, são mantidas tabelas internas com índices dos roteiros possíveis de cada produto e a seqüência de máquinas pertencentes a cada roteiro, sendo a estrutura do cromossomo simplificada, contendo, em cada subvetor, o índice do produto e o índice do roteiro escolhido, conforme mostrado na Figura 2, em que P_{i_p} representa o produto a ser processado e $R_{i_p,j}$ representa o roteiro escolhido para o produto P_{i_p}, entre os roteiros possíveis.

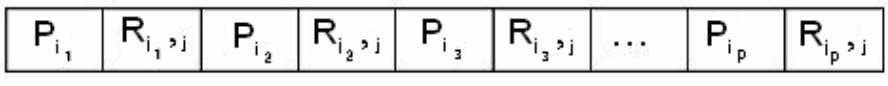

Figura 2 – Representação simplificada do cromossomo

Onde:

$i_1, i_2, i_3, ..., i_p \in \{ 1, 2, ... n\}$ & $i_g \neq i_j$, $g \neq j$.

$j \in \{ 1, 2, ... s\}$.

Vejamos um exemplo de cromossomo, considerando uma fábrica com 6 máquinas M_1, M_2, M_3, M_4, M_5, M_6 e 3 tipos de produtos P_1, P_2, P_3.

Na Tabela 1 estão relacionados os roteiros de fabricação possíveis para os produtos.

Tabela 1 – Produtos e Roteiros de Fabricação

Produto	**Roteiros de Fabricação**
P_1	R_{11} (M_1, M_2, M_6)
	R_{12} (M_4, M_5, M_6)
P_2	R_{21} (M_1, M_2, M_5, M_6)
	R_{22} (M_3, M_4, M_5, M_6)
P_3	R_{31} (M_4, M_3, M_2)
	R_{32} (M_4, M_1, M_5)

Na Figura 3 é mostrada a codificação completa de um cromossomo C_1, gerado a partir dos produtos e roteiros de fabricação relacionados na Tabela 1. O primeiro gene indica o produto 1, e os 4 genes seguintes indicam as máquinas para o produto 1. O sexto gene indica o produto 3 e o produto 2 é indicado no décimo primeiro gene. Na aplicação, os genes 'vazios' são codificados com o valor 0 (zero).

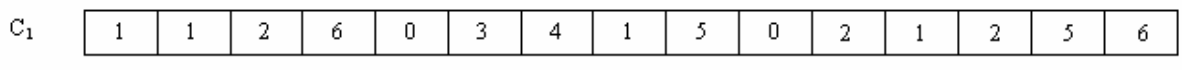

Figura 3 – Cromossomo

Na Figura 4 é mostrado o mesmo cromossomo C_1, com a codificação simplificada, onde o primeiro e o segundo genes indicam que o produto P_1 será produzido pelo roteiro R_{11}, o terceiro e o quarto genes indicam que o produto P_3 será produzido pelo roteiro R_{32} e quinto e o sexto genes indicam que o produto P_2 será produzido pelo roteiro R_{21}. A codificação simplificada facilita a implementação dos operadores de cruzamento e mutação. A codificação completa é usada no cálculo do *makespan* de cada cromossomo.

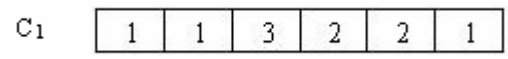

Figura 4 – Cromossomo simplificado

3.2.2. Cruzamento

O cruzamento é feito entre dois cromossomos pais C_1 e C_2, trocando o subvetor referente ao mesmo produto P_{i_p} em cada um deles, gerando dois cromossomos filhos F_1 e F_2, que irão para a nova população, como é mostrado na Figura 5, onde é aplicado o cruzamento em relação ao produto P_1.

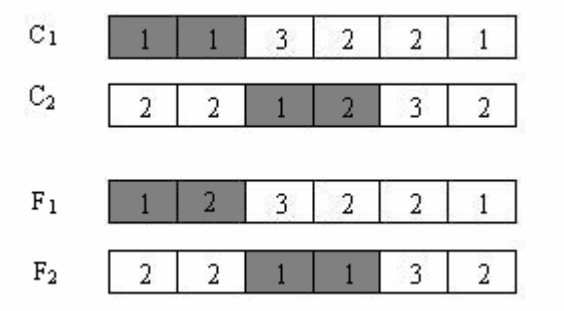

Figura 5: Cruzamento

3.2.3. Mutação

A mutação é realizada em um cromossomo, obtido aleatoriamente da população, trocando o roteiro de fabricação de um produto P_{i_p} também aleatoriamente escolhido, por um dos possíveis roteiros para P_{i_p}, como pode ser visto na Figura 6.

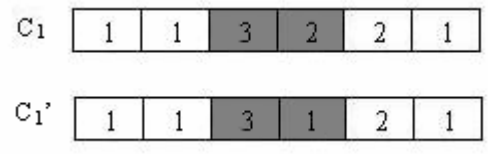

Figura 6: Mutação

3.2.4. Função de *fitness*

A função de *fitness* usada para avaliar as soluções para programação da produção considera o mínimo *makespan,* ou seja, cromossomos com menor valor de *makespan* são considerados mais aptos do que os cromossomos com maior valor de *makespan*. Cada cromossomo indica uma programação da produção, com produtos e roteiros de fabricação.

Para avaliar cada programação, são computados os tempos de operações dos produtos nas máquinas do roteiro de fabricação, considerando o paralelismo entre as máquinas. O tempo total de processamento computado para determinado cromossomo é seu valor de *makespan*. Para obter o valor de *fitness f(i)* de um cromossomo C_i, é realizada uma operação sobre os valores de *makespan* da geração corrente, dada por:

$$f(i) = max - mkp(i) + min$$

Onde, *max* é o maior valor de *makespan* da geração, *min* é o menor valor de *makespan* da geração e *mkp(i)* é o valor de *makespan* do indivíduo C_i na geração. Esta operação atribui valores de *fitness* mais altos a indivíduos com valores de *makespan* mais baixos na geração, conforme proposto por Chiu e Fu (1997). Essa expressão é usada para calcular o valor de *fitness* de um cromossomo, pois faz com que um cromossomo com valor menor de *makespan*, ou seja, um cromossomo mais adaptado, obtenha valor maior de *fitness*, tendo probabilidade maior de ser selecionado para gerar descendentes para a próxima geração, já que os cromossomos são escolhidos para a população intermediária pelo método da roleta, onde a probabilidade de um cromossomo ser selecionado é $f(i) / \sum f(i)$, sendo, assim, proporcional ao seu valor de *fitness*.

Neste trabalho, é usado um tipo de elitismo, no qual o melhor cromossomo de cada geração sempre é selecionado para a população intermediária, para sofrer cruzamento e mutação, de acordo com a probabilidade. Este tipo de elitismo permite aumentar a chance do melhor cromossomo de cada geração ser selecionado para a população da próxima geração, com a preocupação de não favorecer a convergência para resultados ótimos locais.

Após a seleção, os cromossomos da população intermediária sofrem cruzamento e mutação, respeitando a probabilidade de cada operador.

3.2.5. Programação da produção a partir de um cromossomo

Um cromossomo é 'traduzido' em uma programação da produção alocando-se as máquinas indicadas nos genes de cada subvetor, considerando a modelagem completa do cromossomo, para o produto referente àquele subvetor, a partir de um tempo inicial. Os subvetores (produtos), são tratados paralelamente, priorizando a ordem em que aparecem no cromossomo.

Tomando como exemplo o cromossomo da Figura 1, primeiro é alocada a máquina M_{i_1} , $_1$ para P_{i_1} ,em seguida é alocada M_{i_2} ,$_1$ para P_{i_2} , assim sucessivamente até todas as máquinas ou todos os produtos terem sido alocados, sempre respeitando a disponibilidade das máquinas. Quando uma máquina se torna disponível, o cromossomo é percorrido de maneira a alocá-la para o produto com maior prioridade. Este procedimento é repetido até que todas as máquinas de todos os roteiros contidos no cromossomo tenham sido alocadas para os respectivos produtos.

O algoritmo usado para a "tradução" de um cromossomo para uma programação da produção pode ser visto na figura 7. Ao iniciar a execução do algoritmo mostrado na figura 7, existe um "tempo corrente", cujo valor inicial é zero e será aumentado com o decorrer do tempo de produção. Ao final da execução do algoritmo, o *makespan* referente ao cromossomo é o valor do "tempo corrente" do algoritmo, ou seja, o tempo total decorrido no sistema produtivo para a fabricação de todos os produtos representados no cromossomo.

O resultado deste procedimento de "tradução" aplicado ao cromossomo obtido como solução para o problema de programação da solução gera informações como tempo inicial e final do processamento de cada produto em cada máquina e é posteriormente visualizado em forma de um gráfico de Gantt.

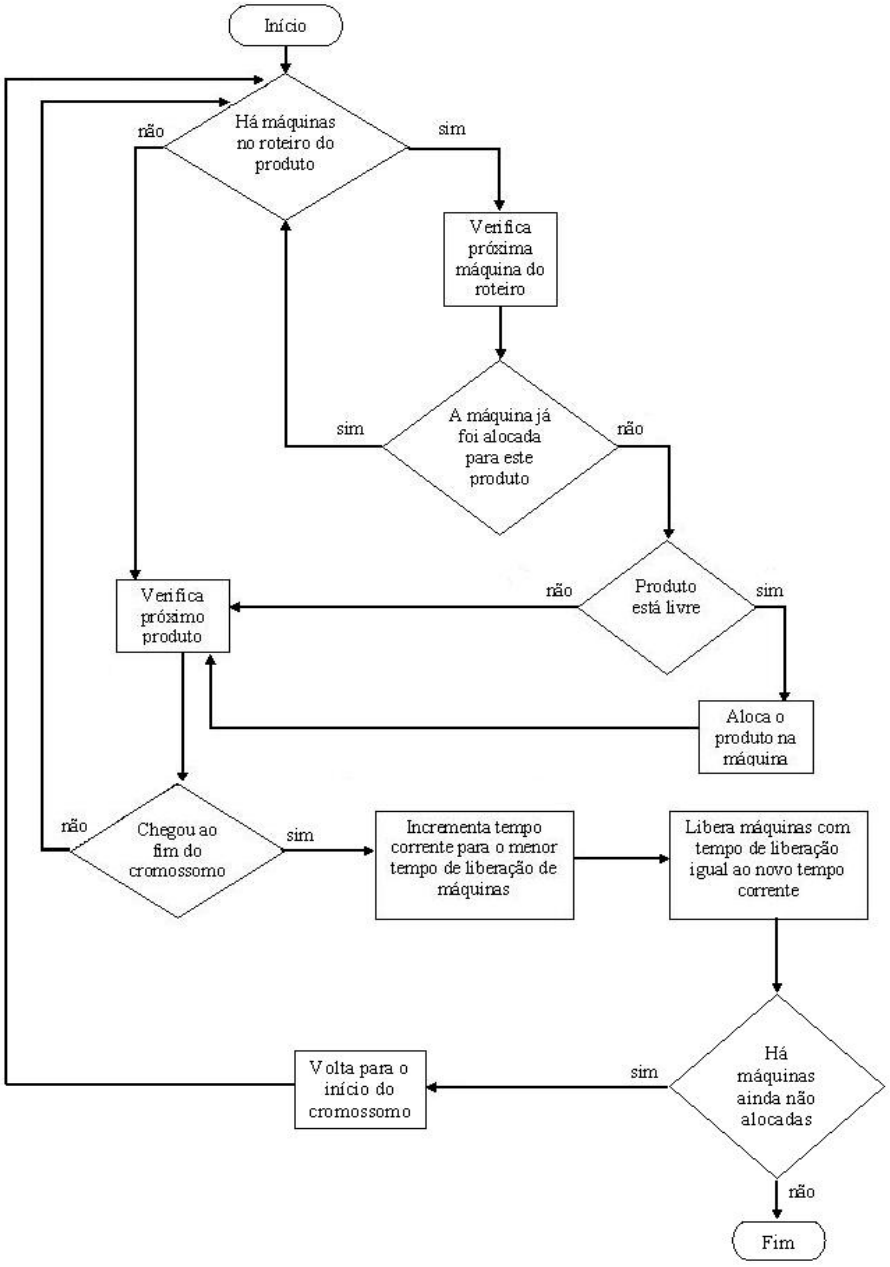

Figura 7 – "tradução" de um cromossomo em uma programação

Comentários finais

Neste capítulo foi apresentado o método de busca com AG proposto para a programação da produção de sistemas de manufatura com recursos compartilhados e seu funcionamento.

No capítulo 4 serão mostradas as etapas de desenvolvimento do *software*

Programador da Produção baseado em AG, que implementa o método proposto neste capítulo. Serão mostrados, também, os principais documentos gerados durante o planejamento e desenvolvimento do *software*.

4 DESENVOLVIMENTO DO *SOFTWARE* PROGRAMADOR DA PRODUÇÃO BASEADO EM AG PARA SISTEMAS DE MANUFATURA COM RECURSOS COMPARTILHADOS

Com o objetivo de avaliar o método de busca proposto neste trabalho, foi elaborado um Programador da Produção baseado em AG, um *software* desenvolvido para implementar o modelo proposto. O *software* Programador da Produção baseado em AG foi planejado e desenvolvido usando linguagem UML, Processo Unificado e a linguagem de programação Java.

O Processo Unificado (PU) é um método utilizado para desenvolvimento de *software* orientado a objetos, que se baseia no princípio do desenvolvimento iterativo, isto é, o sistema é desenvolvido em ciclos que são considerados mini-projetos de duração fixa, com o sistema integrado, testado e executável. O PU utiliza a UML, uma linguagem para documentação de projetos de *software* orientados a objetos, como notação para representar o processo de desenvolvimento. Em cada etapa do desenvolvimento, alguns diagramas UML e documentos são elaborados com a finalidade de descrever o sistema e seu comportamento. Detalhes sobre o desenvolvimento de *software* usando PU podem ser obtidos no apêndice C.

O *software* foi implementado usando a linguagem de programação Java e o Ambiente de programação Eclipse 3.2.

A seguir serão mostrados alguns dos principais documentos gerados no processo de desenvolvimento do Programador da Produção baseado em AG. Na Figura 8, pode ser vista a arquitetura geral do sistema, com seus principais componentes. O método de busca com AG proposto é implementado no módulo AG. No módulo programador, a solução é “traduzida” para uma programação da produção, que é, posteriormente, exibida como um gráfico de Gantt, o que é feito no módulo gerador de Gantt. O banco de dados armazena os dados necessários para o funcionamento do AG e para a “tradução” da solução para uma programação da produção e geração do gráfico de Gantt.

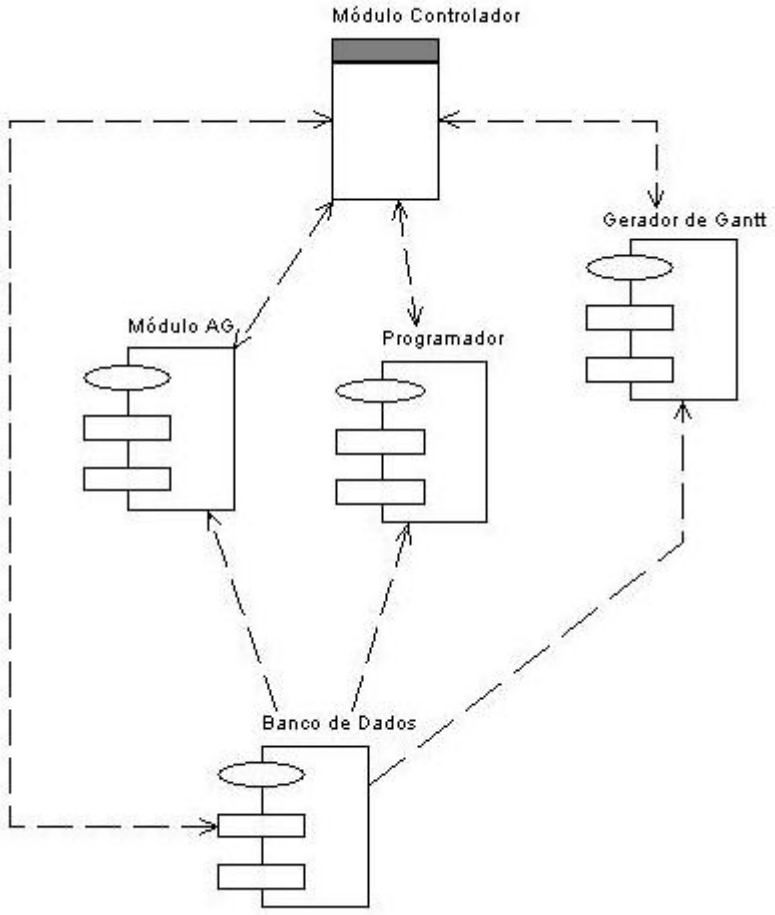

Figura 8 – Arquitetura geral do Programador da Produção baseado em AG

O Programador da Produção baseado em AG foi planejado e implementado para ser flexível para futuras alterações, com o objetivo de adicionar novas funcionalidades e recursos que serão discutidos no capítulo 6.

Inicialmente, foi observada a necessidade de três tipos de usuário, com tipos de utilização do sistema e funções diferentes na resolução do problema de programação da produção. O primeiro tipo de usuário é o Usuário Programador da Produção, que usará o sistema freqüentemente, para realizar a programação da produção da manufatura. O segundo tipo de usuário é o Usuário Configurador da Manufatura, que será responsável pelos cadastros de produtos, máquinas e roteiros de fabricação e o terceiro tipo de usuário, foi chamado de Usuário Administrador do Sistema, que poderá alterar parâmetros do AG e gerenciar os usuários do sistema. Os três tipos de usuário e os principais casos de uso inicialmente levantados para cada um deles podem ser vistos na Tabela 2.

Tabela 2: Usuários e Casos de uso

Ator Principal (Usuário)	Objetivo
Usuário Programador da Produção	Gerar programação Iniciar Encerrar
Usuário Configurador da Manufatura	Cadastrar/alterar/excluir produtos Cadastrar/alterar/excluir máquinas Cadastrar/alterar/excluir roteiros Iniciar Autenticar Encerrar
Administrador do Sistema	Alterar parâmetros AG Adicionar usuários Modificar usuários Excluir usuários Iniciar Autenticar Encerrar

O caso de uso "Gerar Programação" é onde está implementado o método de busca proposto e provavelmente será o caso de uso usado com mais freqüência no Programador da Produção baseado em AG. Este caso de uso, resumidamente, consiste no seguinte:

- O usuário programador da produção entra com a quantidade de peças de cada tipo de produto a ser produzido, um tipo de cada vez e envia estes dados para o sistema.
- Depois de processados, o usuário pode ver o resultado, ou seja, a programação da produção para os produtos solicitados através de um Gráfico de Gantt.

Graficamente, este caso de uso pode ser visto no diagrama de caso de uso mostrado na Figura 9. Um diagrama de caso de uso não leva em conta a seqüência temporal, mas sim as atividades do caso de uso.

No caso de uso da Figura 9, pode-se ver que o Usuário Programador da Produção entra com os dados referentes aos produtos que deseja produzir (tipos de produto e quantidades). Em seguida, estes dados são enviados ao módulo AG, que inicia o caso de uso "encontrar solução", ou seja, inicia a busca para encontrar a melhor programação da produção

para os produtos solicitados. A cada geração do AG, os cromossomos são avaliados, como pode ser visto, pelo acionamento do caso de uso "avaliar", que é executado no módulo "Programador". Após o término da busca com AG, uma solução é retornada e é iniciado o caso de uso "Programar Produção", que traduz o cromossomo em uma programação da produção, no módulo "Programador". Depois, a programação é enviada ao módulo "Gerador de Gantt", onde o gráfico de Gantt é gerado para ser visualizado pelo usuário.

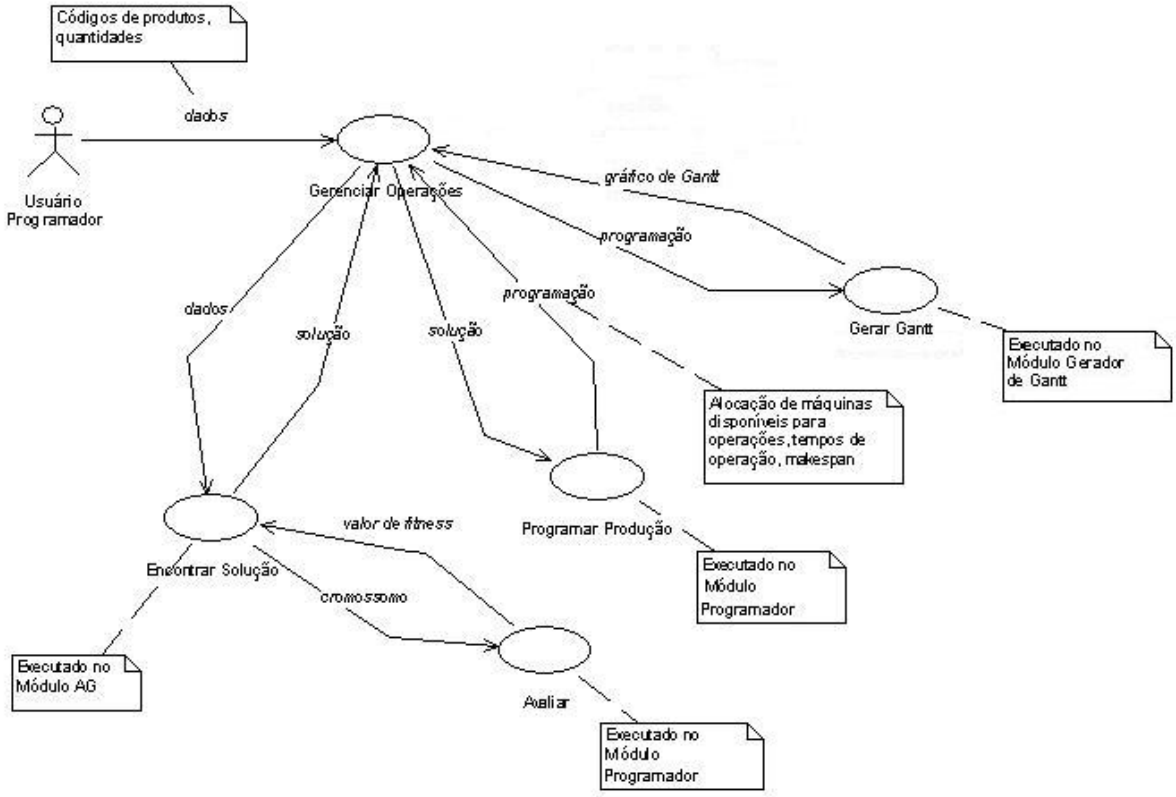

Figura 9 – Diagrama de Caso de Uso "Gerar Programação"

O funcionamento do caso de uso "Gerar Programação" no *software* Programador da Produção baseado em AG, tal como descrito anteriormente, pode ser visto no Diagrama de Seqüência do Sistema (DSS), levando em conta a seqüência temporal de cada atividade do sistema, que é mostrado na Figura 10.

O módulo "Controlador Principal" representa a interface entre o Usuário Programador da Produção e o Programador da Produção Baseado em AG, ou seja, é este módulo que aciona os outros módulos do sistema, de acordo com as solicitações do usuário. Pode-se ver, na Figura 10, a seqüência temporal das atividades do sistema e quais os módulos envolvidos em sua realização. As caixas de comentário são usadas para indicar os dados

necessários para a realização de cada ação do sistema e para descrever quais são as informações geradas pelo sistema, por exemplo, o que é a "programação" gerada pelo módulo "Programador" e armazenada no banco de dados.

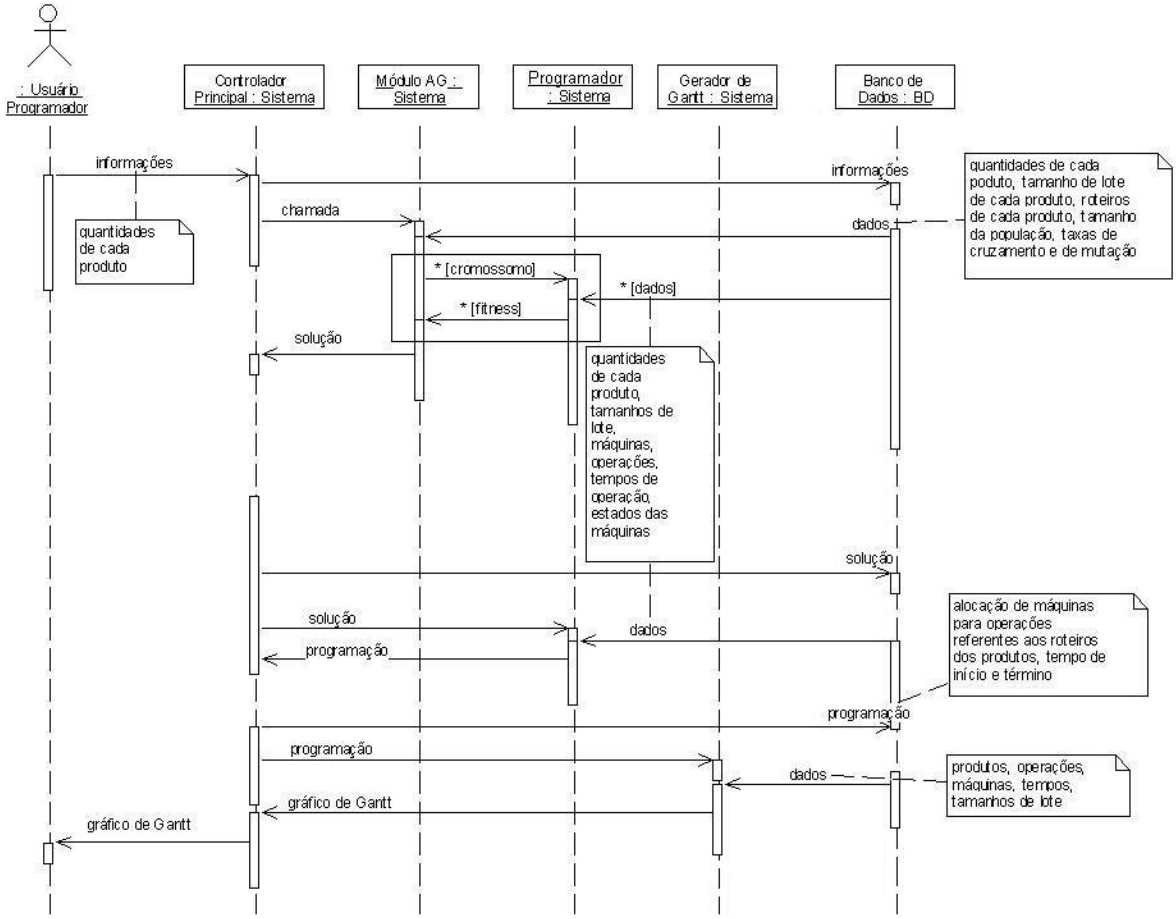

Figura 10 – Diagrama de Seqüência do Sistema

O método de busca com AG proposto para a solução do problema da programação da produção é acionado pelo caso de uso Gerar Programação, ou seja, é acionado para gerar a programação da produção, de acordo com os dados referentes ao ambiente de manufatura constantes no sistema.

O caso de uso Gerar Programação é subdividido em dois subcasos de uso: o subcaso de uso Encontrar Solução – AG, que se refere à busca com AG, modelada e descrita no capítulo 3 e o subcaso de uso Programar Produção, que se refere ao processo de transformar um cromossomo em uma programação da produção, que depois será mostrada em um gráfico de Gantt. As etapas de funcionamento do AG, descritas no capítulo 3 podem ser visualizadas mais detalhadamente no diagrama de casos de uso mostrado na Figura 11 e no diagrama de seqüência mostrado na Figura 12, que se referem, ambos, ao subcaso de uso

Encontrar Solução – AG.

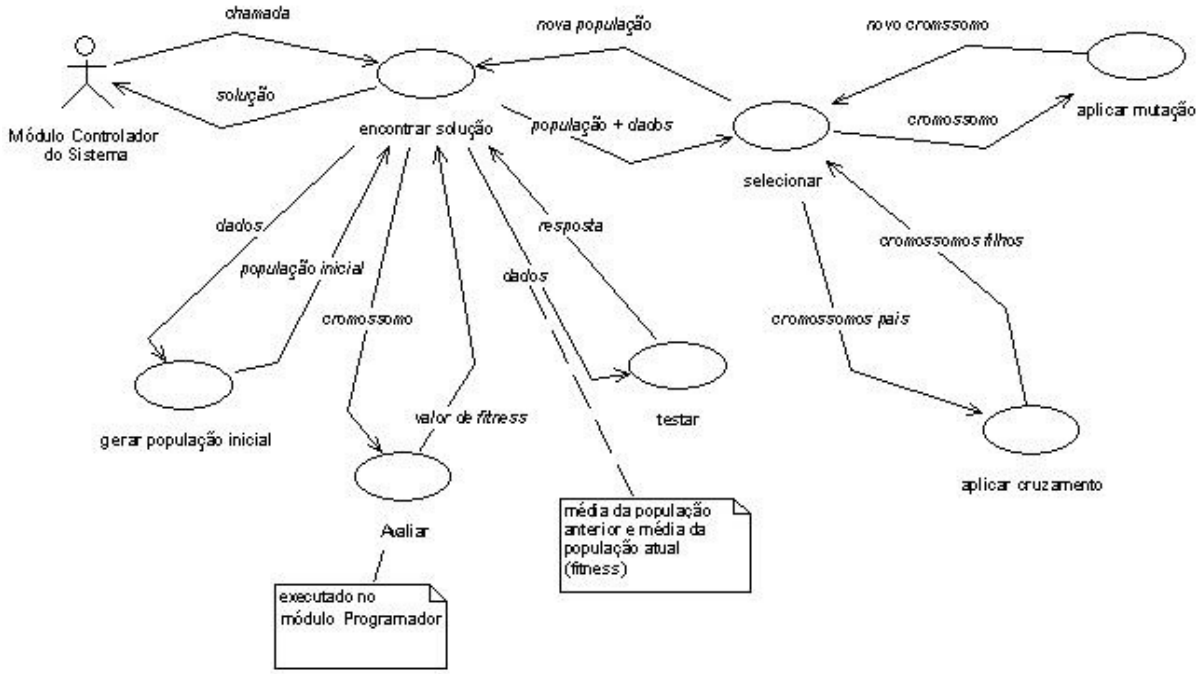

Figura 11 – Diagrama Caso de Uso do subcaso de uso Encontrar Solução – AG.

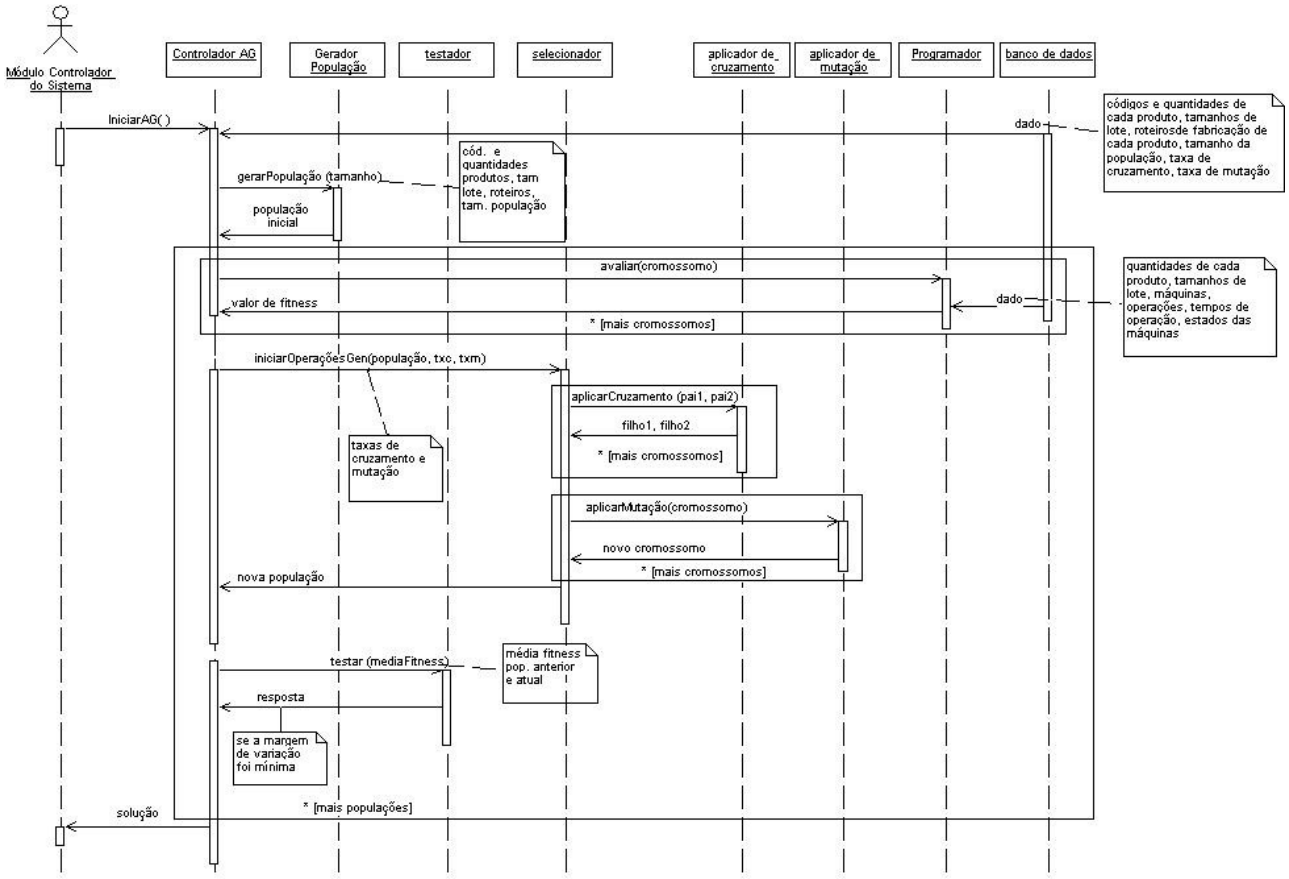

Figura 12 – Diagrama de Seqüência do subcaso de uso Encontrar Solução – AG.

O Modelo de Domínio do sistema é mostrado na Figura 13. Um modelo de domínio é uma descrição de coisas no mundo real do domínio do problema, usando diagramas de classe da UML, porém sem os métodos, sendo diferente de um modelo de classes de *software*. Um modelo de domínio é usado para compreensão do sistema e das relações entre as classes conceituais envolvidas no problema. Outras informações sobre os diagramas mostrados neste capítulo e sobre o Processo Unificado estão disponíveis no Apêndice C.

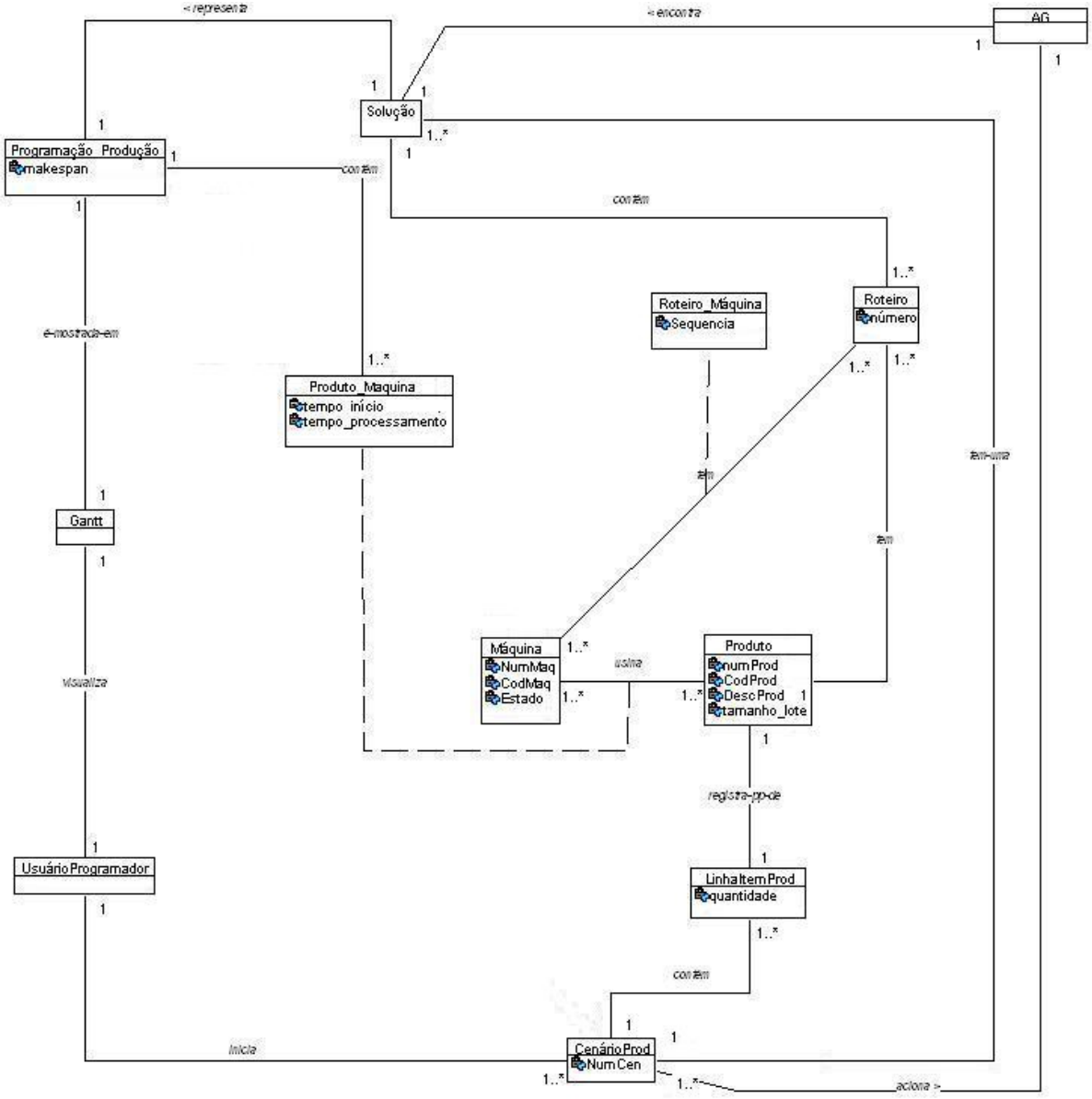

Figura 13 – Modelo de Domínio do sistema

O Diagrama de Classes de Projeto (DCP) do sistema pode ser visto na Figura 14 e uma breve descrição de suas classes e responsabilidades é mostrada na Tabela 3.

O DCP mostra as classes do *software* a ser implementado, com seus atributos e métodos. As classes rotuladas como <<persistente>> deverão ter uma tabela do banco de dados para armazenar suas instâncias. O diagrama de classes de *software* é criado a partir da identificação das classes do Modelo de Domínio que farão parte da solução de *software* e de suas responsabilidades, isto é, seus métodos.

Na Tabela 4 pode ser vista uma breve descrição das classes e seus atributos, a fim de cumprir as responsabilidades descritas na Tabela 3. As classes constantes na Tabela 3 e não citadas na Tabela 4 não contêm atributos.

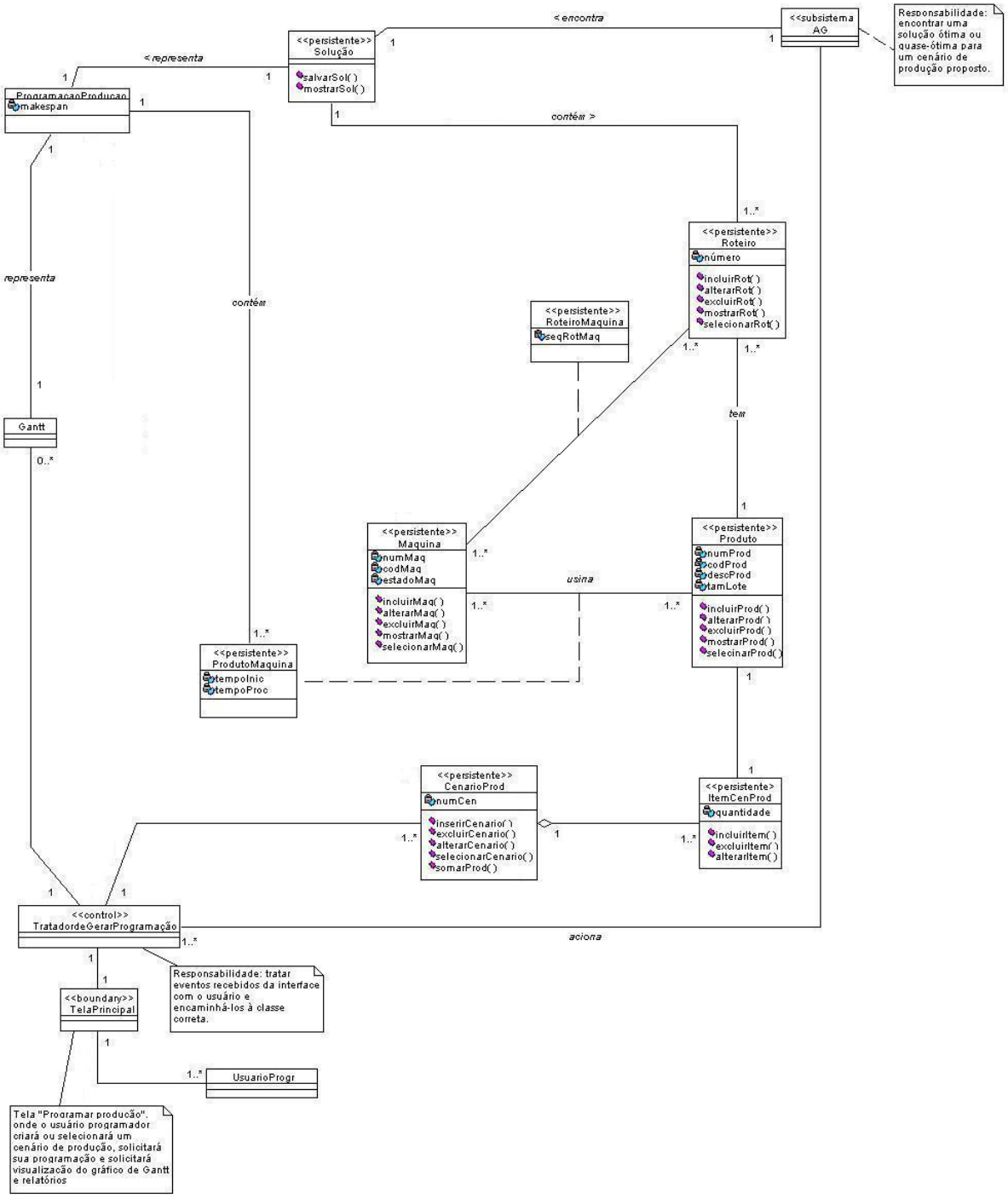

Figura 14 – Diagrama de classes de projeto (DCP)

Tabela 3 – descrição das classes e responsabilidades do DCP do sistema

Classes	**Responsabilidades**
Produto	Inserir produtos Alterar produtos Excluir produtos Mostrar características de produtos
Maquina	Inserir máquinas;

	Alterar máquinas; Excluir máquinas; Mostrar características de máquinas.
ProdutoMaquina	Determinar tempos de inicio e processamento de cada produto em cada máquina.
Roteiro	Manter os roteiros de fabricação de cada produto.
RoteiroMaquina	Determinar a seqüência de cada máquina em um roteiro de fabricação.
CenarioProd	Determinar os produtos e quantidades que deverão ser produzidos.
ItemCenProd	Descrever um produto e quantidade que deverá ser produzido.
UsuárioProgramador	Inserir cenários de produção; Solicitar programação da produção (acionar AG); Visualizar relatórios e Gantt.
AG (subsistema)	Encontrar uma solução ótima ou quase-ótima para um cenário de produção.
Solução	Armazenar e exibir a solução encontrada para o problema.
ProgramaçãodaProdução	Transformar uma solução em uma programação da produção factível.
Gantt	Exibir uma programação da produção.
TelaPrincipal	Interface entre o usuário programador e o sistema. Refere-se à tela "Programar Produção".
TratadordeGerarProgramacao	Responsável por tratar eventos recebidos da interface com o usuário e encaminhá-los à classe correta.

Tabela 4 – descrição das classes e atributos do DCP do sistema

Classes	**Atributos**	**Tipo**	**Descrição**
Produtos	**numProd**	Inteiro	Identificador do produto para uso interno do sistema. O usuário não tem conhecimento deste número.
	codProd	Texto [10]	Código do produto na fábrica. O sistema não usa este código, mas ele é exibido sempre para o usuário.
	descProd	Texto	Descrição do produto.
	tamLote	inteiro	Tamanho de lote padrão do produto
Maquinas	**numMaq**	Inteiro	Identificador da máquina para uso interno do sistema. O usuário não tem conhecimento deste número.
	codMaq	Texto [10]	Código da máquina na fábrica. O sistema não usa este código, mas ele é exibido sempre para o usuário.
	descMaq	Texto	Descrição da máquina.
	estadoMaq	booleano	Estado da máquina: true = disponível, false = indisponível para processamento. Uma máquina se torna indisponível quando está processando um produto ou quando está em manutenção.
Roteiro	**numRot**	Inteiro	Identificador do roteiro para uso interno do sistema.

ProdutoMaquina	**tempoInic**	Inteiro	Tempo do início do processamento de um determinado produto em uma máquina. Este valor será usado na programação da produção para determinado cenário.
	tempoProc	Inteiro	Tempo de processamento de um tipo de produto em uma determinada máquina. Este valor é fixo. Cada produto tem um tempo de processamento em cada máquina de seus roteiros de fabricação.
RoteiroMaquina	**seqRotMaq**	Inteiro	Seqüência (ordem) de uma máquina em um determinado roteiro de fabricação de um produto.
CenarioProd	**numCen**	Inteiro	Identificador do cenário de produção para uso interno e para o usuário.
ItemCenProd	**quantidadeIC**	Inteiro	Quantidade a ser produzida do produto a que se refere o item do cenário de produção
ProgramacaoProducao	**makespan**	Inteiro	Tempo total de processamento de uma solução encontrada para um cenário de produção

O diagrama de classes de *software* da busca implementada usando AG é mostrado na Figura 15. A partir das classes, atributos e métodos que podem ser vistos nesse diagrama, o *software* foi implementado. As caixas de comentário descrevem as responsabilidades de cada classe, que são cumpridas através de seus métodos.

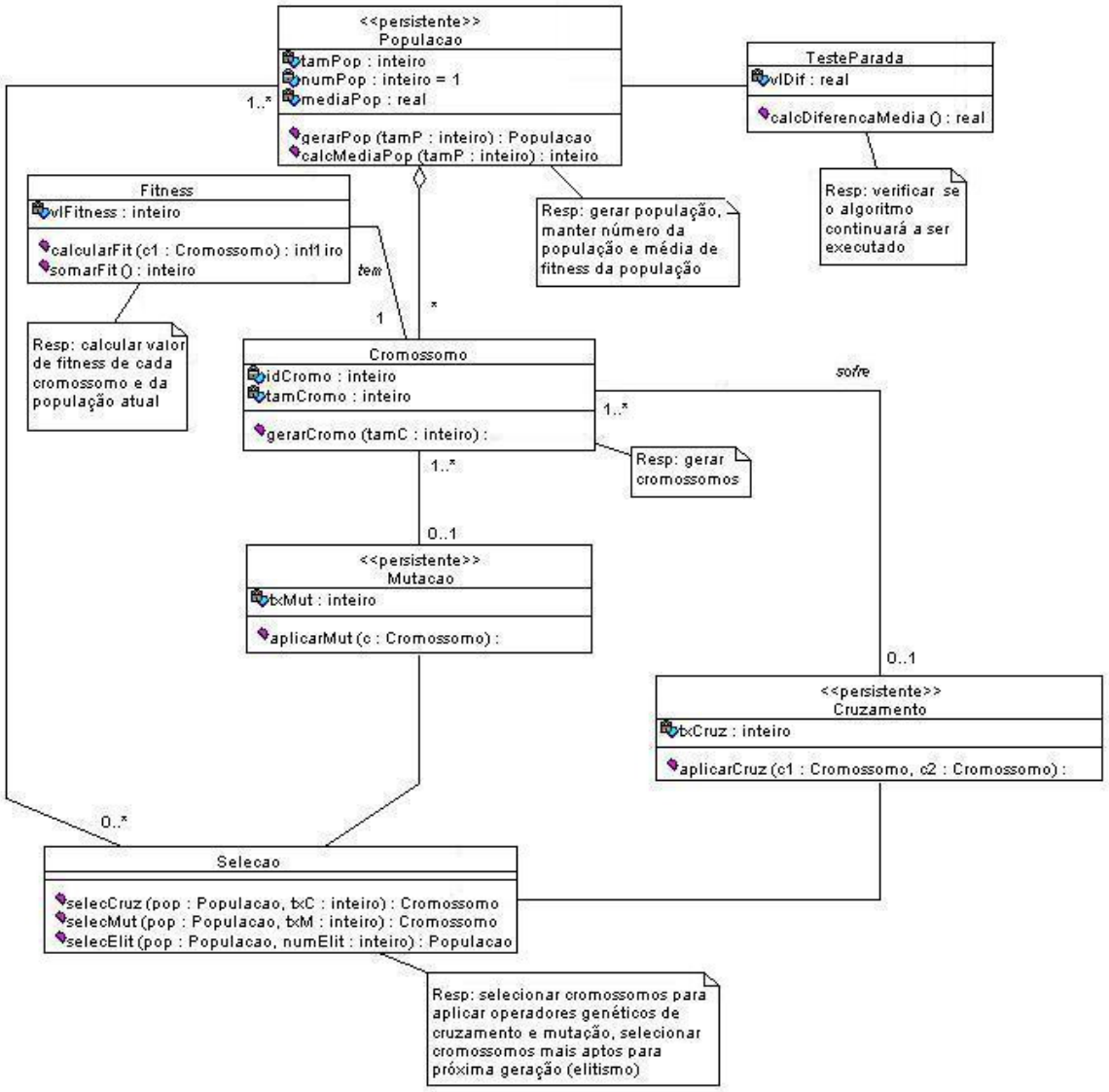

Figura 15 – Diagrama de Classes de *software* da busca com AG

Foram desenvolvidas também algumas telas para o sistema, que servem como interface entre os usuários e o Programador da Produção baseado em AG. Três das principais telas do sistema implementado podem ser vistas nas Figuras 16, 17 e 18. Na Figura 16, é mostrada a primeira tela do sistema que contém o menu principal. No submenu Cadastro é possível cadastrar tempos de operação, produtos, máquinas e roteiros. Este submenu será usado pelo Usuário Configurador da Manufatura, conforme os casos de uso mostrados na Tabela 2.

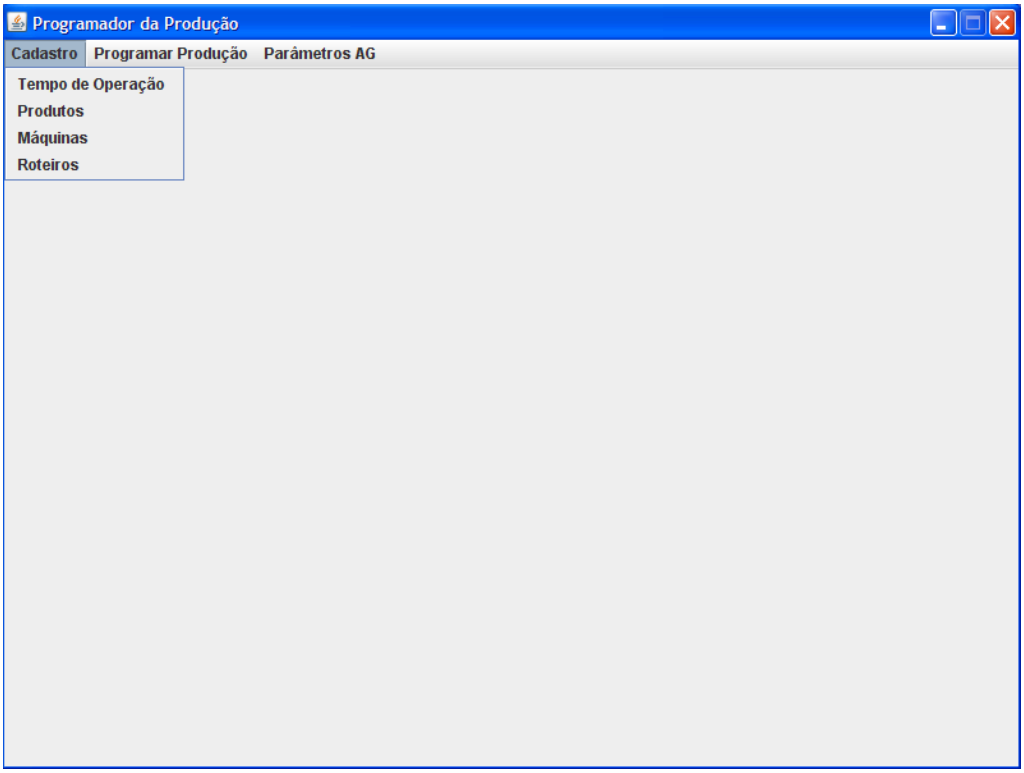

Figura 16 – Tela Inicial do sistema

Na figura 17 pode ser vista a tela "Cadastrar Produtos", onde o Usuário Configurador da Manufatura poderá cadastrar novos produtos, com seus códigos, descrições e tamanhos de lote padrão.

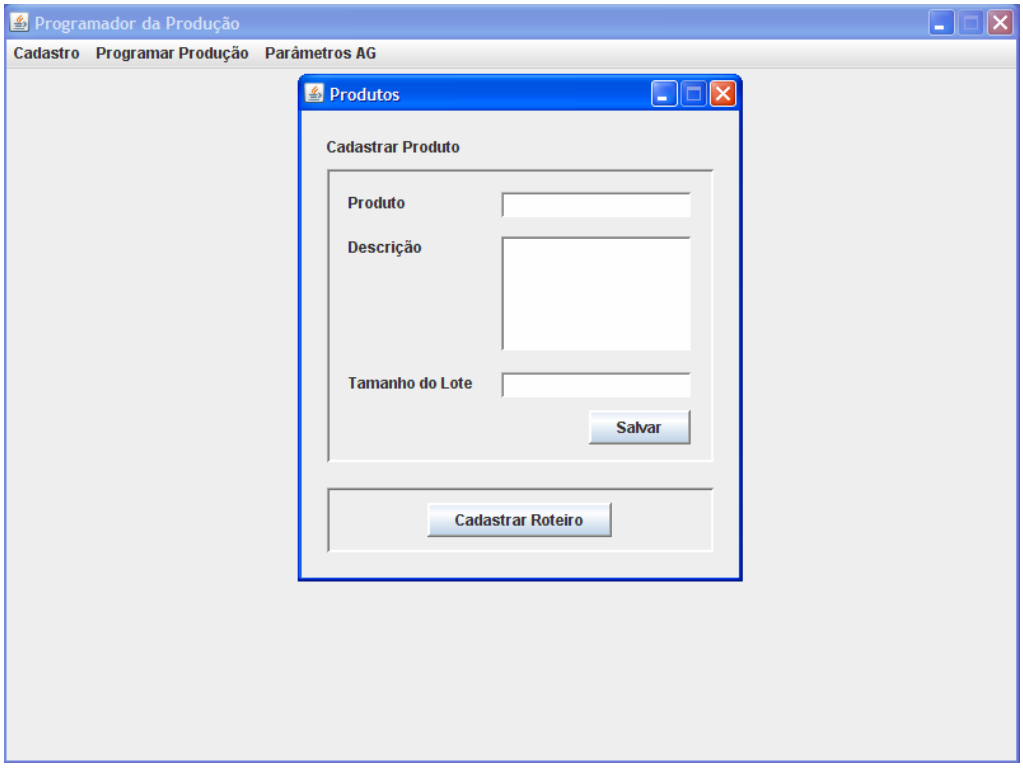

Figura 17 – Tela "Cadastrar Produtos"

Para os testes os cenários de produção foram previamente cadastrados no sistema. O submenu Programar Produção levará à tela mostrada na Figura 18, onde a programação da produção será gerada, ou seja, a busca com AG será acionada e retornará a melhor solução encontrada e o valor do *makespan* para ela. Nesta tela também é possível visualizar o gráfico de Gantt para a programação.

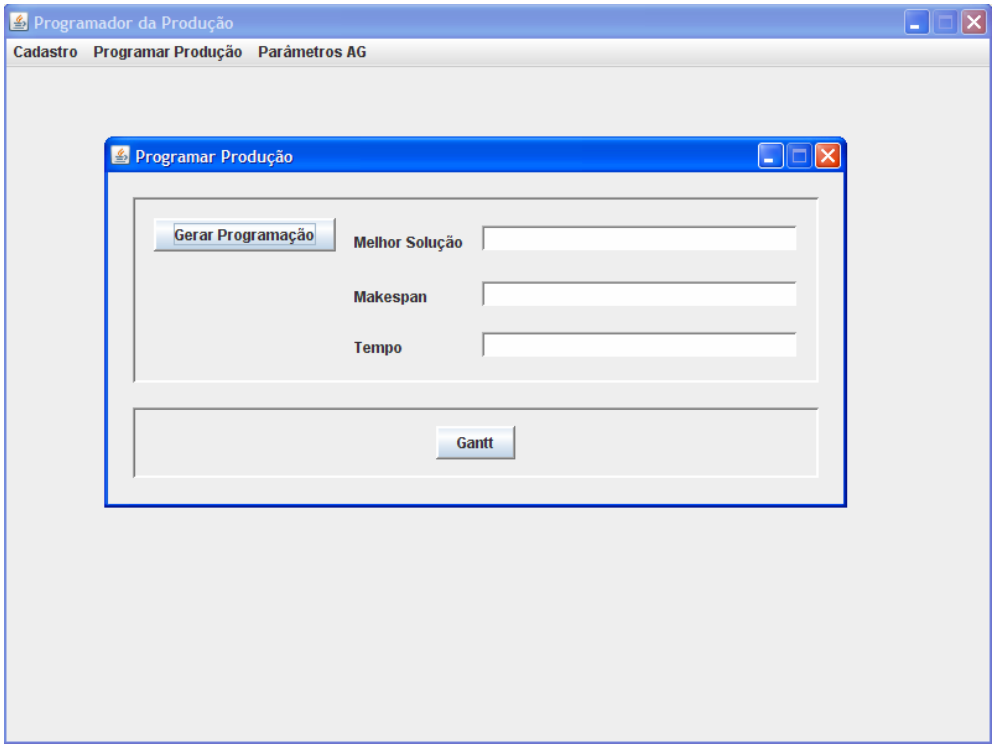

Figura 18 – Tela “Programar Produção”

A solução encontrada pelo sistema é mostrada como um gráfico de Gantt, conforme pode ser visto um exemplo na Figura 19, onde o eixo vertical representa os recursos do sistema de manufatura, neste caso, as máquinas e o eixo horizontal representa o tempo, dividido em unidades. Neste gráfico é possível ver em que momento cada produto passa por cada máquina. Cada produto é representado por uma cor diferente.

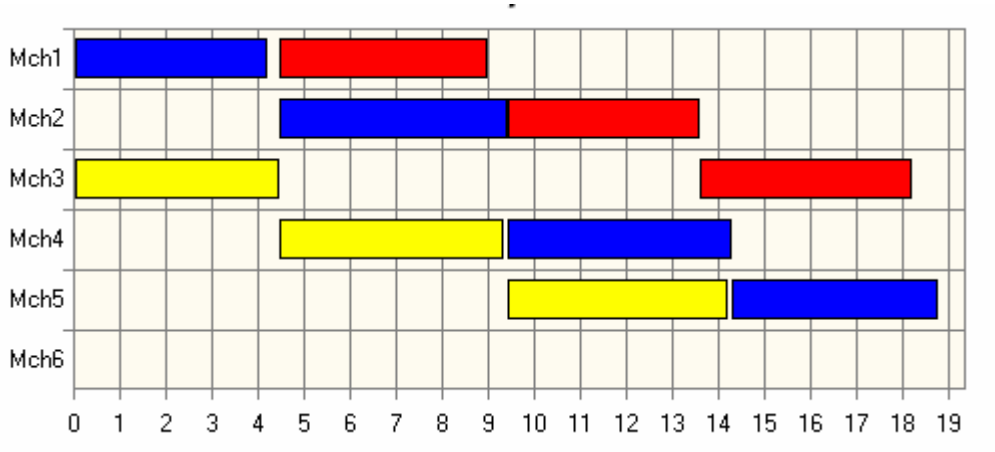

Figura 19 – gráfico de Gantt

O modelo entidade-relacionamento do banco de dados pode ser visto na Figura 20, onde retângulos representam entidades, círculos representam atributos e losangos representam relacionamentos.

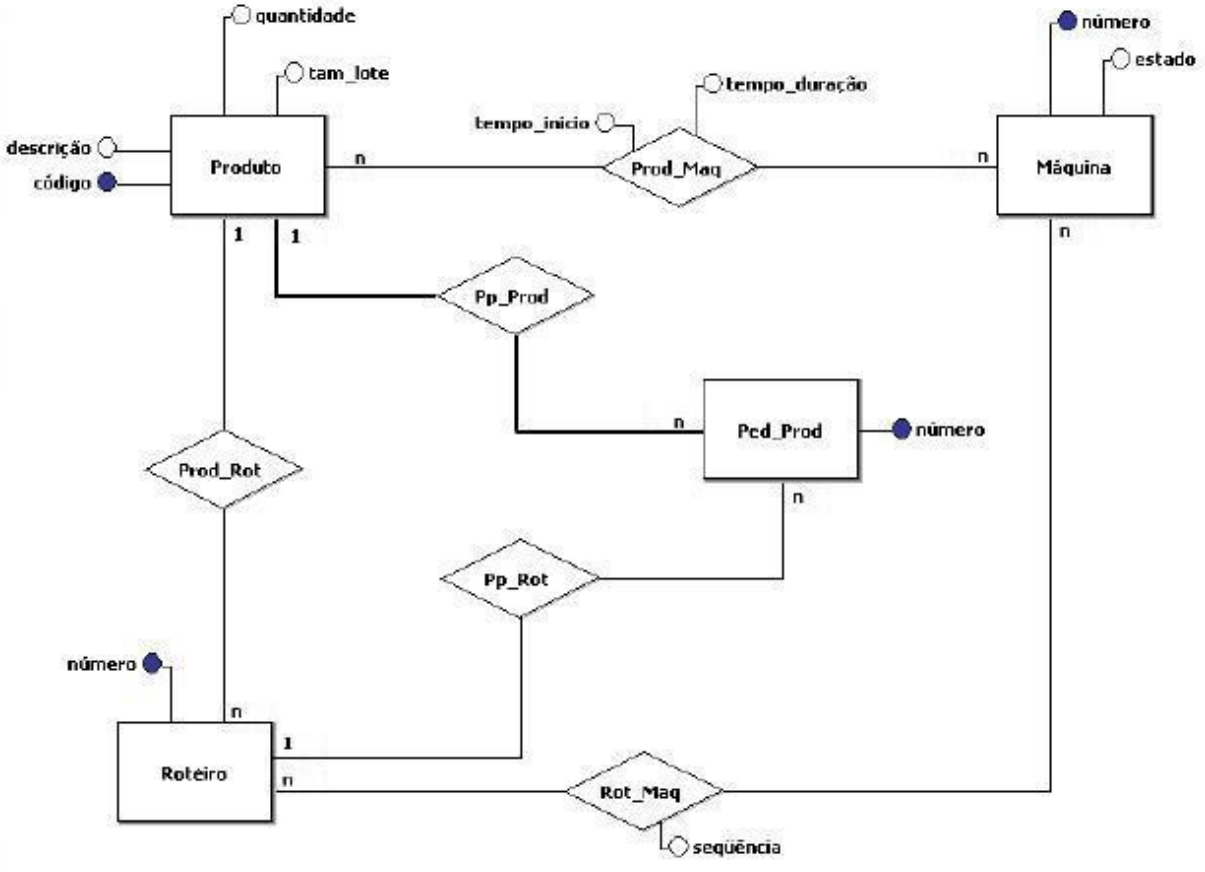

Figura 20 – DER do sistema

Comentários finais

Neste capítulo foram mostradas as principais etapas de desenvolvimento do Programador da Produção baseado em AG e os principais documentos gerados em cada etapa, seguindo o Processo Unificado.

No capítulo seguinte serão definidos os problemas (cenários) que foram testados para avaliação do método proposto e os resultados mostrados e comparados com outros métodos de busca, propostos por Maggio (2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), tendo como critérios de desempenho o mínimo *makespan* e o mínimo tempo de execução da busca.

5 EXPERIMENTOS COMPUTACIONAIS

Foram realizados testes computacionais com os problemas descritos abaixo e os resultados encontrados foram comparados com outras abordagens propostas, variantes do método de busca A*, tendo como critério de comparação o *makespan* obtido e o tempo de execução da busca. Os mesmos problemas foram resolvidos usando o método de busca proposto neste trabalho e os métodos de busca propostos por Maggio(2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), que foram chamados, respectivamente, de AG, Maggio, Reyes, Yu e RCR Cost e os resultados foram comparados e analisados.

Para definir o número de testes a serem executados, a fim de obter certo grau de confiabilidade na análise estatística dos resultados, usou-se como base o Teorema do Limite Central ou Teorema Central do Limite (MARTINS, 2005), que afirma que a distribuição da média amostral x, de uma amostra aleatória de tamanho n extraída de uma população não-normal, com média $\mu(x)$, desvio padrão $\sigma(x)$ é aproximadamente normal com a média $\mu(x)$ e o desvio padrão $\sigma(x)/\sqrt{n}$.

O Teorema Central do Limite tem validade para $n > 30$ e, quanto maior o tamanho da amostra, mais a forma da distribuição amostral da média aproxima-se da forma normal (MARTINS, 2005).

Considerando esse comportamento da distribuição amostral semelhante a uma curva normal, podem-se aplicar alguns testes estatísticos para análise dos resultados obtidos. O teste de Wilcoxon (MARTINS, 2005) foi aplicado para comparar os resultados obtidos pelo método proposto por este trabalho e pelos outros métodos testado, em relação ao *makespan* obtido.

No teste de Wilcoxon, uma amostra A1 é submetida a um tratamento T1 e tem seu efeito medido. Posteriormente, essa mesma amostra, chamada agora de A2 é submetida a um segundo tratamento T2, e tem seu efeito medido pela mesma variável usada no primeiro tratamento. Comparando-se o efeito dos dois tratamentos em cada elemento da amostra, pode-se verificar se o efeito aumentou, diminuiu ou permaneceu o mesmo. Através do teste de Wilcoxon, pode-se analisar se os resultados obtidos pelos dois tratamentos aplicados à mesma amostra têm diferença estatisticamente relevante, levando-se em conta a magnitude do aumento ou da diminuição dos resultados e não apenas a direção da variação para mais ou para menos (CAMPOS, 2007).

Para cada problema de cada tamanho foram realizados 50 testes, ou seja, o

sistema proposto foi acionado 50 vezes para um mesmo problema, assim como as outras propostas. Devido às características não determinísticas da busca com AG, os resultados variaram em cada teste, já os resultados obtidos pelos outros métodos propostos foram os mesmos em todos os testes. Como foram realizados no mesmo sistema computacional, os tempos de execução também foram considerados sem variação.

Após obtidos os resultados, foram realizadas comparações estatísticas entre os resultados obtidos pelo método de busca proposto neste trabalho e os resultados obtidos pelos outros métodos de busca.

5.1 Caracterização do ambiente de testes

Foram testados problemas (cenários), gerados aleatoriamente, de três tamanhos diferentes em relação a quantidades de produtos e máquinas. As características e restrições do sistema de manufatura são as mesmas já definidas no capítulo 3.

Os roteiros de fabricação dos produtos foram gerados aleatoriamente, assim como os tempos de operação, que, como dito anteriormente, são determinísticos, uma vez gerados, permanecem fixos. Os tamanhos de cenários testados podem ser vistos na Tabela 5, onde as colunas representam, respectivamente, tamanho do problema, identificador dos produtos, identificador das máquinas, identificador dos roteiros, quantidade de roteiros para cada produto (a quantidade de roteiros pode variar de um produto para outro), tamanho dos roteiros (quantidade de máquinas em cada roteiro, pode variar de um roteiro para outro)

Para cada problema testado, foi considerada a programação de um produto de cada tipo, o que pode corresponder também a um lote de produtos de tamanho padrão, se considerarmos que os tempos de operação de cada produto em cada máquina também correspondem aos tempos de um lote de tamanho padrão do produto.

Tabela 5: tamanhos de problemas

Tamanho	Produtos	Máquinas	Roteiros	Qtd Roteiros	Tamanho Roteiro
Tamanho 1	P_i, 1<=i<=3	M_j, 1<=j<=6	R_{ik}, 1<=k<=4	[2,4] para cada P_i	[3,5] máquinas
Tamanho 2	P_i, 1<=i<=5	M_j, 1<=j<=8	R_{ik}, 1<=k<=6	[2,6] para cada P_i	[4,7] máquinas
Tamanho 3	P_i, 1<=i<=9	M_j, 1<=j<=9	R_{ik}, 1<=k<=6	2 para cada P_i	[5,7] máquinas

5.2 Testes realizados

Foram testados 3 problemas diferentes do Tamanho 1, um problema do Tamanho 2 e 1 problema do Tamanho 3. Os problemas testados, de cada tamanho, foram gerados aleatoriamente e podem ser visualizados na Tabela 6. As características de cada problema testado podem ser vistas nas Tabelas 7 a 16, onde P_i representa os produtos, M_j representa as máquinas e R_{ik} representa os roteiros de fabricação para cada produto P_i.

Tabela 6: problemas testados e seus tamanhos

Problema	Tamanho
Problema 1	1
Problema 2	1
Problema 3	1
Problema 4	2
Problema 5	3

Os Roteiros de fabricação possíveis para cada produto do Problema 1 podem ser vistos na Tabela 7 e os tempos de operação de cada produto em cada máquina, deste problema, são mostrados na Tabela 8.

Tabela 7: Roteiros do Problema 1

Produtos		Roteiros
P_1	R_{11}	M_1 M_2 M_3 M_4 M_5
	R_{12}	M_1 M_2 M_3 M_6
P_2	R_{21}	M_1 M_4 M_5 M6
	R_{22}	M_2 M_4 M_5 M_6
	R_{23}	M_3 M_4 M_5 M_6
P_3	R_{31}	M_1 M_5 M_6
	R_{32}	M_2 M_5 M_6
	R_{33}	M_3 M_4 M_5 M_6

Tabela 8: Tempos de Operação do Problema 1

Máquinas /Produtos	P_1	P_2	P_3
M_1	434	458	472
M_2	452	443	465
M_3	400	405	469
M_4	472	485	459
M_5	460	402	432
M_6	421	435	445

Os Roteiros de fabricação possíveis para cada produto do Problema 2 são mostrados na Tabela 9 e os tempos de operação de cada produto em cada máquina, deste problema, podem ser vistos na Tabela 10.

Tabela 9: Roteiros do Problema 2

Produtos	Roteiros	
P_1	R_{11}	M_1 M_3 M_4 M_5 M_6
	R_{12}	M_2 M_3 M_4 M_5 M_6
P_2	R_{21}	M_1 M_2 M_5 M6
	R_{22}	M_3 M_5 M_6
	R_{23}	M_4 M_5 M_6
P_3	R_{31}	M_1 M_2 M_3 M_4
	R_{32}	M_1 M_2 M_5
	R_{33}	M_1 M_2 M_6

Tabela 10: Tempos de Operação do Problema 2

Máquinas /Produtos	P_1	P_2	P_3
M_1	427	429	413
M_2	429	480	494
M_3	401	430	422
M_4	434	481	447
M_5	496	471	456
M_6	467	442	422

Na Tabela 11, podem ser vistos os Roteiros de fabricação possíveis para cada produto do Problema 3 e os tempos de operação de cada produto em cada máquina, deste problema, podem ser vistos na Tabela 12.

Tabela 11: Roteiros do Problema 3

Produtos	Roteiros	
P_1	R_{11}	M_1 M_4 M_5 M_6
	R_{12}	M_2 M_4 M_5 M_6
	R_{13}	M_3 M_4 M_5 M_6
P_2	R_{21}	M_1 M_2 M_3 M_4
	R_{22}	M_1 M_2 M_3 M_5 M_6
P_3	R_{31}	M_1 M_2 M_4 M_6
	R_{32}	M_1 M_2 M_5 M_6
	R_{33}	M_3 M_4 M_6
	R_{34}	M_3 M_5 M_6

Tabela 12: Tempos de Operação do Problema 3

Máquinas /Produtos	P_1	P_2	P_3
M_1	456	438	414
M_2	484	458	411
M_3	423	439	424
M_4	494	468	465
M_5	467	498	436
M_6	416	466	402

Os Roteiros de fabricação possíveis para cada produto do Problema 4 são mostrados na Tabela 13 e os tempos de operação de cada produto em cada máquina são mostrados na Tabela 14.

Tabela 13: Roteiros do Problema 4

Produtos	Roteiros	
P_1	R_{11}	M_1 M_2 M_3 M_5 M_6 M_7 M_8
	R_{12}	M_4 M_5 M_6 M_7 M_8
P_2	R_{21}	M_1 M_2 M_3 M_4 M_5 M_6 M_7
	R_{22}	M_1 M_2 M_3 M_4 M_5 M_8

Tabela 14: Tempos de Operação do Problema 4

Máquinas /Produtos	P_1	P_2	P_3	P_4	P_5
M_1	465	430	492	474	466
M_2	491	480	477	480	491

P_3	R_{31}	$M_1\ M_2\ M_3\ M_4\ M_5\ M_7$
	R_{32}	$M_1\ M_2\ M_3\ M_4\ M_5\ M_8$
	R_{33}	$M_1\ M_2\ M_3\ M_4\ M_6\ M_7$
	R_{34}	$M_1\ M_2\ M_3\ M_4\ M_6\ M_8$
P_4	R_{41}	$M_1\ M_2\ M_3\ M_4\ M_8$
	R_{42}	$M_5\ M_6\ M_7\ M_8$
P_5	R_{51}	$M_1\ M_2\ M_4\ M_8$
	R_{52}	$M_1\ M_2\ M_5\ M_8$
	R_{53}	$M_1\ M_2\ M_6\ M_7\ M_8$
	R_{54}	$M_1\ M_3\ M_4\ M_8$
	R_{55}	$M_1\ M_3\ M_5\ M_8$
	R_{56}	$M_1\ M_3\ M_6\ M_7\ M_8$

M_3	447	424	470	406	477
M_4	479	488	439	452	494
M_5	492	414	412	411	470
M_6	410	426	487	472	460
M_7	440	456	415	417	417
M_8	469	408	483	467	453

Na Tabela 15, podem ser visualizados os Roteiros de fabricação possíveis para cada produto do Problema 5 e na Tabela 16 são mostrados os tempos de operação de cada produto em cada máquina.

Tabela 15: Roteiros do Problema 5

Produtos		Roteiros
P_1	R_{11}	$M_1\ M_2\ M_4\ M_5\ M_7\ M_9$
	R_{12}	$M_3\ M_4\ M_5\ M_6\ M_8\ M_9$
P_2	R_{21}	$M_1\ M_2\ M_3\ M_4\ M_5\ M_6\ M_7$
	R_{22}	$M_2\ M_3\ M_5\ M_7\ M_8\ M_9$
P_3	R_{31}	$M_4\ M_5\ M_6\ M_7\ M_8$
	R_{32}	$M_2\ M_3\ M_7\ M_8\ M_9$
P_4	R_{41}	$M_2\ M_3\ M_4\ M_6\ M_7$
	R_{42}	$M_1\ M_5\ M_6\ M_8\ M_9$
P_5	R_{51}	$M_4\ M_5\ M_7\ M_8\ M_9$
	R_{52}	$M_1\ M_2\ M_3\ M_5\ M_6$
P_6	R_{61}	$M_2\ M_4\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{62}	$M_1\ M_3\ M_6\ M_7\ M_8\ M_9$
P_7	R_{71}	$M_1\ M_2\ M_4\ M_5\ M_6\ M_9$
	R_{72}	$M_1\ M_2\ M_3\ M_7\ M_8\ M_9$
P_8	R_{81}	$M_4\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{82}	$M_3\ M_4\ M_5\ M_7\ M_8\ M_9$
P_9	R_{91}	$M_3\ M_5\ M_6\ M_7\ M_8\ M_9$
	R_{92}	$M_2\ M_4\ M_6\ M_7\ M_8\ M_9$

Tabela 16: Tempos de Operação do Problema 5

Máquinas /Produtos	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
M_1	428	439	453	403	481	446	414	491	458

M_2	423	433	474	436	440	495	457	419	486
M_3	459	487	417	410	477	474	452	435	416
M_4	433	405	447	410	442	448	426	491	454
M_5	467	447	486	400	450	469	493	495	452
M_6	461	497	496	468	468	408	408	452	438
M_7	464	495	459	489	436	454	457	477	484
M_8	455	469	489	439	486	424	497	452	435
M_9	418	439	480	457	435	482	445	408	416

5.3 Resultados obtidos

Os problemas descritos anteriormente foram resolvidos com o método Programador da Produção Baseado em AG proposto neste trabalho e com os métodos propostos por Maggio(2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), que foram chamados, respectivamente, de AG, Maggio, Reyes, Yu e RCR Cost.

Para realização dos testes com o método proposto, há três parâmetros do algoritmo genético cujos valores podem ser alterados: tamanho da população, taxa de cruzamento e taxa de mutação. Inicialmente, os testes foram realizados com tamanho da população entre 30 e 40, taxa de cruzamento = 0,8 (80%) e taxa de mutação = 0,05 (5%). Tais valores foram escolhidos por estarem entre as faixas de valores comumente utilizadas para estes parâmetros e que apresentaram bons resultados na literatura consultada (Yeung e Moore, 2000; Pongcharoen et al, 2004; Pongcharoen et al, 2004).

Nas Tabelas 17, 18, 19, 20 E 21 podem ser vistos os resultados dos testes para cada problema. As colunas indicam o número do teste (de 1 a 50), o *makespan* obtido pelo método proposto, o tempo de execução do *software* proposto e as gerações do AG necessárias para a obtenção da resposta, respectivamente. As próximas colunas representam os valores de *makespan* obtidos pelos outros métodos testados e seus tempos de execução, respectivamente. Os valores de *makespan* foram contados em unidades de tempo (u.t.) e os tempos de execução, em segundos.

Na Tabela 17 são mostrados os resultados dos testes realizados para o Problema 1, com tamanho da população = 30, taxa de cruzamento = 0,8 e taxa de mutação = 0,05.

Tabela 17: Resultados obtidos para o Problema 1.

	AG	t AG	Ger	Maggio	t Maggio	Yu	t Yu	RCR Cost	t RCR	Reyes	t Reyes
1	2214	0,2	44	2251	0,031	2284	17,4	2239	8,469	2239	9,062
2	2198	0,2	73	2251	0,031	2284	17,4	2239	8,469	2239	9,062
3	2198	0,3	88	2251	0,031	2284	17,4	2239	8,469	2239	9,062
4	2587	0,1	39	2251	0,031	2284	17,4	2239	8,469	2239	9,062
5	2587	0,2	43	2251	0,031	2284	17,4	2239	8,469	2239	9,062
6	2281	0,2	53	2251	0,031	2284	17,4	2239	8,469	2239	9,062
7	2214	0,1	33	2251	0,031	2284	17,4	2239	8,469	2239	9,062
8	2587	0,2	48	2251	0,031	2284	17,4	2239	8,469	2239	9,062
9	2214	0,2	46	2251	0,031	2284	17,4	2239	8,469	2239	9,062
10	2214	0,3	106	2251	0,031	2284	17,4	2239	8,469	2239	9,062
11	2198	0,3	99	2251	0,031	2284	17,4	2239	8,469	2239	9,062
12	2252	0,5	179	2251	0,031	2284	17,4	2239	8,469	2239	9,062
13	2226	0,4	148	2251	0,031	2284	17,4	2239	8,469	2239	9,062
14	2309	0,1	38	2251	0,031	2284	17,4	2239	8,469	2239	9,062
15	2218	0,2	43	2251	0,031	2284	17,4	2239	8,469	2239	9,062
16	2161	0,5	180	2251	0,031	2284	17,4	2239	8,469	2239	9,062
17	2198	0,4	145	2251	0,031	2284	17,4	2239	8,469	2239	9,062
18	2252	0,1	31	2251	0,031	2284	17,4	2239	8,469	2239	9,062
19	2249	0,1	18	2251	0,031	2284	17,4	2239	8,469	2239	9,062
20	2198	0,2	77	2251	0,031	2284	17,4	2239	8,469	2239	9,062
21	2201	0,2	76	2251	0,031	2284	17,4	2239	8,469	2239	9,062
22	2201	0,2	46	2251	0,031	2284	17,4	2239	8,469	2239	9,062
23	2596	0,1	28	2251	0,031	2284	17,4	2239	8,469	2239	9,062
24	2214	0,4	147	2251	0,031	2284	17,4	2239	8,469	2239	9,062
25	2676	0,3	90	2251	0,031	2284	17,4	2239	8,469	2239	9,062
26	2205	0,1	26	2251	0,031	2284	17,4	2239	8,469	2239	9,062
27	2781	0,4	155	2251	0,031	2284	17,4	2239	8,469	2239	9,062
28	2227	0,2	63	2251	0,031	2284	17,4	2239	8,469	2239	9,062
29	2198	0,1	38	2251	0,031	2284	17,4	2239	8,469	2239	9,062
30	2161	0,4	166	2251	0,031	2284	17,4	2239	8,469	2239	9,062
31	2205	0,2	67	2251	0,031	2284	17,4	2239	8,469	2239	9,062
32	2198	0,3	87	2251	0,031	2284	17,4	2239	8,469	2239	9,062
33	2596	0,2	54	2251	0,031	2284	17,4	2239	8,469	2239	9,062
34	2161	0,1	31	2251	0,031	2284	17,4	2239	8,469	2239	9,062
35	2249	0,2	76	2251	0,031	2284	17,4	2239	8,469	2239	9,062
36	2587	0,2	55	2251	0,031	2284	17,4	2239	8,469	2239	9,062

37	2256	0,2	60	2251	0,031	2284	17,4	2239	8,469	2239	9,062
38	2587	0,1	40	2251	0,031	2284	17,4	2239	8,469	2239	9,062
39	2249	0,2	77	2251	0,031	2284	17,4	2239	8,469	2239	9,062
40	2249	0,2	59	2251	0,031	2284	17,4	2239	8,469	2239	9,062
41	2256	0,3	92	2251	0,031	2284	17,4	2239	8,469	2239	9,062
42	2198	0,2	79	2251	0,031	2284	17,4	2239	8,469	2239	9,062
43	2252	0,1	26	2251	0,031	2284	17,4	2239	8,469	2239	9,062
44	2587	0,2	66	2251	0,031	2284	17,4	2239	8,469	2239	9,062
45	2587	0,1	37	2251	0,031	2284	17,4	2239	8,469	2239	9,062
46	2161	0,3	85	2251	0,031	2284	17,4	2239	8,469	2239	9,062
47	2345	0,2	53	2251	0,031	2284	17,4	2239	8,469	2239	9,062
48	2249	0,3	99	2251	0,031	2284	17,4	2239	8,469	2239	9,062
49	2205	0,1	36	2251	0,031	2284	17,4	2239	8,469	2239	9,062
50	2198	0,2	61	2251	0,031	2284	17,4	2239	8,469	2239	9,062

A média do *makespan* encontrada para o método proposto foi 2307,8 u.t., com desvio padrão de 170,23. O valor mínimo encontrado foi 2161 u.t. e o valor máximo encontrado foi 2781. Observou-se que, para este problema 74% dos resultados obtidos foram menores que o valor da média, houve tendência de melhora em 62% dos resultados obtidos com relação aos resultados obtidos por Maggio, houve tendência de melhora em 74% dos resultados em relação aos resultados obtidos por Yu e houve tendência de melhora em 52% dos resultados obtidos em relação aos resultados obtidos por Reyes e RCR Cost.

Com relação ao tempo de execução do método proposto, sua média foi 0,222 segundos, que é aproximadamente 7 vezes maior que o tempo de execução de Maggio, 78 vezes menor que o tempo de execução de Yu, 38 vezes menor que o tempo de execução de RCR Cost e 40 vezes menor que o tempo de execução de Reyes.

Através da comparação dos valores de *makespan* mostrados na Tabela 17, conclui-se, com 95% de confiabilidade, que os métodos comparados terão resultados sem diferença estatisticamente significativa para este problema.

Na tabela 18 são mostrados os resultados dos testes realizados para o Problema 2, com tamanho da população = 30, taxa de cruzamento = 0,8 e taxa de mutação = 0,05.

Tabela 18: Resultados obtidos para o Problema 2.

	AG	t AG	Ger	Maggio	t Maggio	Yu	t Yu	RCR Cost	t RCR	Reyes	t Reyes
1	2225	0,2	56	2343	0	2255	7,547	2255	7,125	2255	7,905
2	2228	0,1	23	2343	0	2255	7,547	2255	7,125	2255	7,905

3	2371	0,2	46	2343	0	2255	7,547	2255	7,125	2255	7,905
4	2227	0,2	40	2343	0	2255	7,547	2255	7,125	2255	7,905
5	2228	0,2	47	2343	0	2255	7,547	2255	7,125	2255	7,905
6	2371	0,2	53	2343	0	2255	7,547	2255	7,125	2255	7,905
7	2654	0,3	100	2343	0	2255	7,547	2255	7,125	2255	7,905
8	2225	0,1	23	2343	0	2255	7,547	2255	7,125	2255	7,905
9	2228	0,1	40	2343	0	2255	7,547	2255	7,125	2255	7,905
10	2227	0,2	46	2343	0	2255	7,547	2255	7,125	2255	7,905
11	2343	0,2	75	2343	0	2255	7,547	2255	7,125	2255	7,905
12	2228	0,1	26	2343	0	2255	7,547	2255	7,125	2255	7,905
13	2225	0,4	133	2343	0	2255	7,547	2255	7,125	2255	7,905
14	2699	0,1	17	2343	0	2255	7,547	2255	7,125	2255	7,905
15	2228	0,3	100	2343	0	2255	7,547	2255	7,125	2255	7,905
16	2343	0,1	23	2343	0	2255	7,547	2255	7,125	2255	7,905
17	2228	0,2	54	2343	0	2255	7,547	2255	7,125	2255	7,905
18	2227	0,3	84	2343	0	2255	7,547	2255	7,125	2255	7,905
19	2227	0,1	24	2343	0	2255	7,547	2255	7,125	2255	7,905
20	2342	0,2	49	2343	0	2255	7,547	2255	7,125	2255	7,905
21	2254	0,1	29	2343	0	2255	7,547	2255	7,125	2255	7,905
22	2228	0,2	61	2343	0	2255	7,547	2255	7,125	2255	7,905
23	2228	0,3	102	2343	0	2255	7,547	2255	7,125	2255	7,905
24	2654	0,3	81	2343	0	2255	7,547	2255	7,125	2255	7,905
25	2227	0,5	164	2343	0	2255	7,547	2255	7,125	2255	7,905
26	2342	0,1	38	2343	0	2255	7,547	2255	7,125	2255	7,905
27	2228	0,1	30	2343	0	2255	7,547	2255	7,125	2255	7,905
28	2227	0,1	36	2343	0	2255	7,547	2255	7,125	2255	7,905
29	2227	0,3	80	2343	0	2255	7,547	2255	7,125	2255	7,905
30	2225	0,2	73	2343	0	2255	7,547	2255	7,125	2255	7,905
31	2227	0,5	166	2343	0	2255	7,547	2255	7,125	2255	7,905
32	2227	0,2	48	2343	0	2255	7,547	2255	7,125	2255	7,905
33	2654	0,2	54	2343	0	2255	7,547	2255	7,125	2255	7,905
34	2232	0,1	32	2343	0	2255	7,547	2255	7,125	2255	7,905
35	2228	0,1	17	2343	0	2255	7,547	2255	7,125	2255	7,905
36	2225	0,1	39	2343	0	2255	7,547	2255	7,125	2255	7,905
37	2225	0,2	45	2343	0	2255	7,547	2255	7,125	2255	7,905
38	2342	0,3	83	2343	0	2255	7,547	2255	7,125	2255	7,905
39	2232	0,3	93	2343	0	2255	7,547	2255	7,125	2255	7,905
40	2228	0,1	39	2343	0	2255	7,547	2255	7,125	2255	7,905
41	2225	0,3	91	2343	0	2255	7,547	2255	7,125	2255	7,905
42	2225	0,2	62	2343	0	2255	7,547	2255	7,125	2255	7,905
43	2228	0,2	52	2343	0	2255	7,547	2255	7,125	2255	7,905
44	2227	0,3	88	2343	0	2255	7,547	2255	7,125	2255	7,905
45	2228	0,1	18	2343	0	2255	7,547	2255	7,125	2255	7,905
46	2227	0,2	73	2343	0	2255	7,547	2255	7,125	2255	7,905

47	2227	0,3	95	2343	0	2255	7,547	2255	7,125	2255	7,905
48	2228	0,2	41	2343	0	2255	7,547	2255	7,125	2255	7,905
49	2232	0,3	120	2343	0	2255	7,547	2255	7,125	2255	7,905
50	2228	0,1	23	2343	0	2255	7,547	2255	7,125	2255	7,905

A média do *makespan* encontrada para o método proposto foi 2280,18 u.t., com desvio padrão de 122,66. O valor mínimo encontrado foi 2225 u.t. e o valor máximo encontrado foi 2689. Observou-se que, para este problema, 78% dos resultados obtidos foram iguais ou menores que o valor da média, houve tendência de melhora em 88% dos resultados obtidos com relação aos resultados obtidos por Maggio e em 78% dos resultados em relação aos resultados obtidos por Yu, Reyes e RCR Cost.

Com relação ao tempo de execução do método proposto, sua média foi 0,32 segundos, que é maior que o tempo de execução de Maggio, e aproximadamente 22 vezes menor que o tempo de execução dos outros métodos testados.

Através da comparação dos valores de *makespan* mostrados na tabela 18, conclui-se, com 99,9% de confiabilidade, que o método proposto terá resultados diferentes estatisticamente e melhores que os resultados obtidos por Maggio e, com 90% de confiabilidade, conclui-se que o método proposto terá resultados diferentes estatisticamente e melhores que os resultados obtidos pelos outros métodos testados para este problema. A média dos *makespans* encontrados pelo método proposto foi 2,68% menor do que o *makespan* encontrado pelo método proposto por Maggio e 1,12% maior do que o *makespan* encontrado pelos outros métodos testados.

Na tabela 19 são mostrados os resultados dos testes realizados para o Problema 3, com tamanho da população = 30, taxa de cruzamento = 0,8 e taxa de mutação = 0,05.

Tabela 19: Resultados obtidos para o Problema 3

	AG	t AG	Ger	Maggio	t Maggio	Yu	t Yu	RCR Cost	t RCR	Reyes	t Reyes
1	2762	0,3	71	2202	0	1913	0,281	1913	0,203	1913	0,203
2	2200	0,2	28	2202	0	1913	0,281	1913	0,203	1913	0,203
3	1861	0,4	110	2202	0	1913	0,281	1913	0,203	1913	0,203
4	2261	0,4	115	2202	0	1913	0,281	1913	0,203	1913	0,203
5	2666	0,4	128	2202	0	1913	0,281	1913	0,203	1913	0,203
6	3117	0,2	73	2202	0	1913	0,281	1913	0,203	1913	0,203
7	2750	0,4	137	2202	0	1913	0,281	1913	0,203	1913	0,203
8	2261	0,2	71	2202	0	1913	0,281	1913	0,203	1913	0,203
9	2266	0,2	52	2202	0	1913	0,281	1913	0,203	1913	0,203
10	2166	0,3	119	2202	0	1913	0,281	1913	0,203	1913	0,203

11	2283	0,1	17	2202	0	1913	0,281	1913	0,203	1913	0,203
12	2286	0,4	159	2202	0	1913	0,281	1913	0,203	1913	0,203
13	2762	0,2	77	2202	0	1913	0,281	1913	0,203	1913	0,203
14	2409	0,2	39	2202	0	1913	0,281	1913	0,203	1913	0,203
15	2716	0,2	42	2202	0	1913	0,281	1913	0,203	1913	0,203
16	2217	0,2	45	2202	0	1913	0,281	1913	0,203	1913	0,203
17	1861	0,2	58	2202	0	1913	0,281	1913	0,203	1913	0,203
18	2261	0,2	41	2202	0	1913	0,281	1913	0,203	1913	0,203
19	2601	0,2	40	2202	0	1913	0,281	1913	0,203	1913	0,203
20	1861	0,2	69	2202	0	1913	0,281	1913	0,203	1913	0,203
21	2266	0,1	24	2202	0	1913	0,281	1913	0,203	1913	0,203
22	3163	0,2	55	2202	0	1913	0,281	1913	0,203	1913	0,203
23	2166	0,1	42	2202	0	1913	0,281	1913	0,203	1913	0,203
24	1861	0,1	12	2202	0	1913	0,281	1913	0,203	1913	0,203
25	2716	0,2	65	2202	0	1913	0,281	1913	0,203	1913	0,203
26	2762	0,2	51	2202	0	1913	0,281	1913	0,203	1913	0,203
27	2217	0,3	112	2202	0	1913	0,281	1913	0,203	1913	0,203
28	2668	0,2	55	2202	0	1913	0,281	1913	0,203	1913	0,203
29	2266	0,1	38	2202	0	1913	0,281	1913	0,203	1913	0,203
30	1861	0,2	54	2202	0	1913	0,281	1913	0,203	1913	0,203
31	2271	0,3	87	2202	0	1913	0,281	1913	0,203	1913	0,203
32	2266	0,1	21	2202	0	1913	0,281	1913	0,203	1913	0,203
33	1861	0,1	26	2202	0	1913	0,281	1913	0,203	1913	0,203
34	1861	0,2	67	2202	0	1913	0,281	1913	0,203	1913	0,203
35	2261	0,3	91	2202	0	1913	0,281	1913	0,203	1913	0,203
36	2668	0,2	53	2202	0	1913	0,281	1913	0,203	1913	0,203
37	2266	0,1	19	2202	0	1913	0,281	1913	0,203	1913	0,203
38	2217	0,2	50	2202	0	1913	0,281	1913	0,203	1913	0,203
39	2224	0,3	111	2202	0	1913	0,281	1913	0,203	1913	0,203
40	2259	0,2	70	2202	0	1913	0,281	1913	0,203	1913	0,203
41	2286	0,1	22	2202	0	1913	0,281	1913	0,203	1913	0,203
42	1861	0,1	36	2202	0	1913	0,281	1913	0,203	1913	0,203
43	2217	0,2	77	2202	0	1913	0,281	1913	0,203	1913	0,203
44	2240	0,1	20	2202	0	1913	0,281	1913	0,203	1913	0,203
45	2271	0,2	70	2202	0	1913	0,281	1913	0,203	1913	0,203
46	2271	0,5	199	2202	0	1913	0,281	1913	0,203	1913	0,203
47	2266	0,2	45	2202	0	1913	0,281	1913	0,203	1913	0,203
48	2261	0,4	125	2202	0	1913	0,281	1913	0,203	1913	0,203
49	2271	0,2	81	2202	0	1913	0,281	1913	0,203	1913	0,203
50	2716	0,4	144	2202	0	1913	0,281	1913	0,203	1913	0,203

A média do *makespan* encontrada para o método proposto foi 2326,54 u.t., com desvio padrão de 315,94. O valor mínimo encontrado foi 1861 u.t. e o valor máximo encontrado foi 3163.

Observou-se que, para este problema, 72% dos resultados obtidos foram iguais ou menores que o valor da média, houve tendência de melhora em 22% dos resultados obtidos com relação aos resultados obtidos por Maggio e em 16% dos resultados em relação aos resultados obtidos por Yu, Reyes e RCR Cost.

Com relação ao tempo de execução do método proposto, sua média foi 0,224 segundos, que é maior que o tempo de execução de Maggio, e próximo ao tempo de execução dos outros métodos testados.

Através da comparação dos valores de *makespan* mostrados na tabela 19, conclui-se, com 95% de confiabilidade, que o método proposto terá resultados diferentes estatisticamente e piores que os resultados obtidos por Maggio e pelos outros métodos testados para este problema. A média dos *makespans* encontrados pelo método proposto foi 5,66% maior do que o *makespan* encontrado pelo método proposto por Maggio e 21,62% maior do que o *makespan* encontrado pelos outros métodos testados.

Na tabela 20 são mostrados os resultados dos testes realizados para o Problema 4, com tamanho da população = 40, taxa de cruzamento = 0,8 e taxa de mutação = 0,05. Nota-se que o Problema 4, por ser maior que os anteriores com relação à quantidade de produtos e máquinas, só pôde ser testado com o método proposto por este trabalho e com o método proposto por Maggio(2005), pois, a execução da busca para este problema com os métodos propostos por Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b) foi inviável devido ao alto tempo de execução e travamento do sistema, condição indesejável para o sistema quando usado para programação reativa da produção, com a fábrica em operação, o que implica a necessidade de obtenção do resultado no menor tempo possível.

Tabela 20: Resultados obtidos para o Problema 4.

	AG	t AG	Ger	Maggio	t Maggio
1	3614	0,7	56	3291	291,36
2	3676	0,4	35	3291	291,36
3	4045	0,9	90	3291	291,36
4	3731	1,6	165	3291	291,36
5	3601	0,8	74	3291	291,36
6	3738	0,6	60	3291	291,36
7	3689	0,4	35	3291	291,36
8	3693	0,9	90	3291	291,36
9	3601	1,1	102	3291	291,36
10	3656	0,8	78	3291	291,36
11	3751	0,7	68	3291	291,36

12	3210	0,6	64	3291	291,36
13	3210	0,5	44	3291	291,36
14	4190	0,5	44	3291	291,36
15	4125	2,5	254	3291	291,36
16	3669	0,7	68	3291	291,36
17	3642	0,3	29	3291	291,36
18	4121	0,9	89	3291	291,36
19	4530	0,7	64	3291	291,36
20	3252	0,9	84	3291	291,36
21	4140	0,6	52	3291	291,36
22	3676	1,1	121	3291	291,36
23	3650	0,5	42	3291	291,36
24	3676	0,7	67	3291	291,36
25	3768	1,5	160	3291	291,36
26	4168	0,8	74	3291	291,36
27	3642	0,3	24	3291	291,36
28	3638	0,5	46	3291	291,36
29	3533	1	97	3291	291,36
30	4060	0,4	39	3291	291,36
31	3601	0,8	78	3291	291,36
32	4558	1	95	3291	291,36
33	4160	0,7	66	3291	291,36
34	4120	1	95	3291	291,36
35	4521	0,7	67	3291	291,36
36	3642	0,3	32	3291	291,36
37	4099	0,6	49	3291	291,36
38	4596	0,8	80	3291	291,36
39	3255	1,2	126	3291	291,36
40	3682	0,5	41	3291	291,36
41	4580	1,1	105	3291	291,36
42	4128	1,6	158	3291	291,36
43	3533	0,4	38	3291	291,36
44	4623	2,7	275	3291	291,36
45	3313	0,5	46	3291	291,36
46	4030	0,5	46	3291	291,36
47	4588	1,7	162	3291	291,36
48	3676	2,2	229	3291	291,36
49	4563	0,9	90	3291	291,36
50	3693	0,5	49	3291	291,36

A média do *makespan* encontrada para o método proposto foi 3873,12 u.t., com desvio padrão de 399,35. O valor mínimo encontrado foi 3210 u.t. e o valor máximo encontrado foi 4623 u.t. Observou-se que, para este problema 60% dos resultados obtidos

foram iguais ou menores que o valor da média, e houve tendência de piora em 92% dos resultados obtidos com relação aos resultados obtidos por Maggio, embora não tenha sido feita análise estatística sobre estas observações.

Destaca-se, porém, que a média do tempo de execução do método proposto foi 0,87 segundos, com desvio padrão de 0,53 segundos, que é 549 vezes menor que o tempo de execução de Maggio e que os métodos propostos por Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b) não solucionaram o problema em tempo de execução viável, o que indica que para cenários com maiores números de produtos e máquinas, que implicam em maior espaço de busca, o método proposto é viável de ser aplicado com tempo de execução baixo.

Através da comparação dos valores de *makespan* mostrados na tabela 20, conclui-se, com 99,9% de confiabilidade, que o método proposto terá resultados diferentes estatisticamente e piores que os resultados obtidos por Maggio, sendo a média dos *makespans* encontrados pelo método proposto 17,69% maior do que o *makespan* encontrado pelo método proposto por Maggio. Isto indica resultados piores mas com valores não significativamente piores. Porém, com relação ao tempo de execução, com 99,9% de confiabilidade, conclui-se que o sistema terá resultados diferentes estatisticamente e melhores que os resultados obtidos por Maggio.

Na tabela 21 podem ser vistos os resultados dos testes realizados para o Problema 5, com tamanho da população = 40, taxa de cruzamento = 0,8 e taxa de mutação = 0,05. Nota-se que o Problema 5 só pôde ser testado com o método proposto por este trabalho, pois, a execução da busca para este problema com os métodos propostos por Maggio(2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b) foi inviável devido ao alto tempo de execução e travamento do sistema.

Tabela 21: Resultados obtidos para o Problema 5.

	AG	t AG	Ger
1	6068	2,3	87
2	5474	1,7	64
3	5397	2,5	92
4	5468	2,5	95
5	5891	0,9	31
6	5386	1,4	53
7	4914	1,1	43
8	5408	5,1	216
9	5412	1,4	60
10	6295	1,9	78
11	5907	1,1	43

12	5929	0,9	36
13	5539	2,2	90
14	5944	1,5	59
15	5188	1,1	44
16	5428	1,7	68
17	5573	1,7	73
18	5406	7	300
19	5023	0,9	37
20	5207	1	40
21	5239	1,4	54
22	5685	3,7	160
23	5517	1,4	59
24	5826	1,8	74
25	5175	1	39
26	6073	1	38
27	4970	1,3	51
28	5997	1,1	45
29	5757	4,7	204
30	5916	1,1	42
31	5086	2,3	99
32	5862	1	39
33	4988	1,4	56
34	6006	3,1	134
35	5494	2,8	118
36	5600	1,3	52
37	5477	1,5	64
38	5507	3,3	143
39	4562	1,9	83
40	5418	2	82
41	5774	2,4	101
42	5524	2,7	115
43	5524	2,5	104
44	5340	2	83
45	5280	3	127
46	5117	1,6	67
47	5415	1,5	62
48	5136	4,7	196
49	5099	1,7	68
50	5429	1,3	51

A média do *makespan* encontrada para o método proposto foi 5493 u.t., com desvio padrão de 355,67. O valor mínimo encontrado foi 4562 u.t. e o valor máximo encontrado foi 6295 u.t. Observou-se que, para este problema 52% dos resultados obtidos

foram iguais ou menores que o valor da média, embora não tenha sido feita análise estatística sobre esta observação.

Destaca-se, porém, que a média do tempo de execução do método proposto foi 2,05 segundos, com desvio padrão de 1,23 segundos, tempo máximo de 7 segundos e tempo mínimo de 0,9 segundos. Destaca-se, ainda, que os métodos propostos por Maggio (2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b) não solucionaram o problema em tempo de execução viável. Os tempos de execução observados nos testes viabilizam a utilização do método proposto para problemas com maior número de produtos e máquinas, que implicam em maior espaço de busca.

Observou-se, pelos testes realizados que, para alguns problemas o Programador da Produção baseado em AG apresenta resultados melhores que os outros métodos comparados e para outros problemas ocorre o contrário. Observou-se também que os resultados encontrados para cada vez que Programador da Produção baseado em AG é acionado são bem diferentes, devido ao caráter aleatório do AG.

A busca com AG pode convergir para soluções ótimas ou quase-ótimas mais rápida ou mais vagarosamente, e isso pode ser dirigido pela pressão seletiva do AG. A pressão seletiva leva os indivíduos a locais que já se mostraram promissores e é controlada pela função de *fitness*, fator de escalonamento e método de seleção (Barcellos, 2000). Além disso, técnicas de elitismo, operadores de cruzamento e mutação e suas taxas influenciam no desempenho da busca com AG.

Comentários finais

Neste capítulo foram definidos os problemas (cenários) que foram testados para avaliação do método proposto e os resultados mostrados e comparados com outros métodos de busca, propostos por Maggio (2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), tendo como critérios de desempenho o mínimo *makespan* e o mínimo tempo de execução da busca.

As conclusões sobre o método proposto, os testes realizados e possíveis alterações no AG proposto, com o objetivo melhorar seu desempenho e suas soluções, serão propostas e discutidas no capítulo 6, assim como as conclusões sobre o *software* desenvolvido e indicações para trabalhos futuros.

6 CONCLUSÃO

A fim de obter vantagem competitiva no cenário empresarial atual, as indústrias necessitam de flexibilidade para se adaptarem às mudanças no sistema de produção. Para tanto, as indústrias estão adotando, em seus processos produtivos, técnicas de automação e compartilhamento de recursos, que inserem certa flexibilidade ao sistema de manufatura, esperando obter, assim, uma utilização eficiente dos recursos, em termos de certos critérios de desempenho. Neste contexto, a programação da produção deve ser feita de maneira que possibilite maximizar os benefícios que podem ser obtidos pelo sistema de manufatura. Obter tal programação da produção não é uma tarefa simples, especialmente em ambientes com compartilhamento de recursos.

Várias abordagens têm sido propostas para lidar com esse problema, como modelagem do sistema de manufatura com Redes de Petri e suas variações, como Redes de Petri Virtuais e Coloridas vêm sendo investigadas, bem como várias estratégias solução vêm sendo propostas em conjunto com as Redes de Petri para sua resolução, como métodos de busca heurística, métodos de busca usando algoritmo genético e análise estrutural do modelo usando matrizes de incidência. Também têm sido propostos métodos para solução deste problema usando algoritmos genéticos, não associados ao modelo do sistema da manufatura em Redes de Petri.

Este trabalho propôs um método para gerar programação da produção em sistemas de manufatura com recursos compartilhados, utilizando algoritmo genético. Os sistemas de manufatura de que trata este trabalho podem ser modelados em Redes de Petri, porém, o AG proposto não é diretamente associado ao modelo em Redes de Petri.

Foi desenvolvido um *software* Programador da Produção Baseado em AG, que incorporou o método proposto. O método foi testado para uma série de problemas e os resultados foram comparados com outros métodos de busca propostos por Maggio(2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), tendo como critérios de desempenho o mínimo *makespan* obtido e o tempo de execução da busca. Os resultados obtidos foram mostrados no capítulo anterior.

6.1 Quanto aos testes realizados e resultados obtidos

Os resultados obtidos pelos testes realizados mostraram que, para alguns problemas testados, o Programador da Produção baseado em AG apresentou resultados melhores que os outros métodos comparados quanto ao *makespan*; para alguns problemas testados não houve diferença estatisticamente relevante entre os resultados obtidos pelos métodos testados e para outros problemas, ainda, o Programador da Produção baseado em AG apresentou resultados piores que os outros métodos testados. As diferenças entre os resultados foram observadas, em problemas dos dois tamanhos testados e comparados, embora não tenha sido feita análise estatística com relação a esta observação.

Um aspecto importante a ser destacado é que, com relação ao tempo de execução, os resultados foram melhores que os outros métodos propostos em problemas com maior quantidade de máquinas e produtos, como no caso do Problema 4, cuja média do tempo de execução do Programador da Produção baseado em AG foi aproximadamente 330 vezes menor que o tempo de execução do método proposto por Maggio (2005). Destaca-se também que, para o Problema 5, cuja média do tempo de execução do Programador da Produção baseado em AG foi 2,05 segundos, os outros métodos propostos por Maggio (2005), Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b), não apresentaram solução em tempo de execução viável (o *software* ficou buscando uma solução por aproximadamente 12 horas e a busca causou travamento no sistema computacional).

Com base nestes resultados, conclui-se que o Programador da Produção baseado em AG proposto neste trabalho pode ser aplicado para a solução de problemas de manufatura com compartilhamento de recursos de tamanho maior, com relação à quantidade de máquinas e produtos, do que os outros métodos testados, apresentando soluções com tempo de resposta aceitável, o que não ocorre com os outros métodos de busca testados.

Conclui-se, também, que, devido aos vários parâmetros, funções e métodos, tais como tamanho da população, taxas de cruzamento e de mutação, função de *fitness*, fator de escalonamento, método de seleção e operadores genéticos de cruzamento e de mutação, que influenciam o funcionamento de buscas com AG, faz-se necessário um estudo sobre possíveis alterações no método proposto, a fim de obter melhores soluções para os problemas em questão. Algumas possíveis alterações serão propostas na seção 6.2.

6.2 Quanto ao método de busca com AG

Quando se trata de buscas com AG, o comportamento da busca é influenciado ou mesmo controlado por alguns elementos, que podem tornar a busca mais ou menos eficiente. Alguns parâmetros como o tamanho da população, as taxas de cruzamento e de mutação, geralmente influenciam na obtenção das soluções desejadas (Yeung e Moore, 2000; Pongcharoen et al, 2002), assim como as taxas referentes aos operadores genéticos, os próprios operadores utilizados podem ser mais ou menos adequados a determinado tipo de problema.

Além disso, a função de *fitness* escolhida para avaliar a qualidade da solução representada pelo cromossomo para o problema em questão e o método de seleção, que, geralmente seleciona os cromossomos mais adaptados para sobreviverem e cruzarem e técnicas de elitismo influenciam no direcionamento da busca, diminuindo o seu aspecto aleatório.

6.2.1 Sugestões para melhoria do desempenho

Observou-se, no método de busca proposto que, em grande parte dos casos testados, cromossomos com melhores valores de *fitness* em uma geração acabam perdendo-se no processo evolutivo. Na tabela 22 podem ser vistos, novamente, os valores de *makespan* obtidos para o Problema 3, onde a primeira coluna representa o número do teste, a coluna "MKP solução" representa o *makespan* relativo à solução final encontrada pelo AG (as duas primeiras colunas são iguais às da tabela 19) e na coluna "menor MKP" é mostrado o *makespan* referente à melhor solução encontrada em todas as gerações, ou seja, o melhor cromossomo obtido durante o processo evolutivo.

Tabela 22: *Makespan* da solução apresentada e melhor *makespan* obtido para o Problema 3.

	MKP solução	menor MKP
1	2762	1861
2	2200	1861
3	1861	1861
4	2261	1861
5	2666	1861
6	3117	1861
7	2750	1861
8	2261	1861
9	2266	2200

10	2166	1861
11	2283	1861
12	2286	2200
13	2762	1861
14	2409	1861
15	2716	1861
16	2217	2200
17	1861	1861
18	2261	1861
19	2601	1861
20	1861	1861
21	2266	1861
22	3163	1861
23	2166	1861
24	1861	1861
25	2716	1861
26	2762	2200
27	2217	2200
28	2668	1861
29	2266	1861
30	1861	1861
31	2271	2200
32	2266	1861
33	1861	1861
34	1861	1861
35	2261	2217
36	2668	1861
37	2266	2200
38	2217	2200
39	2224	2200
40	2259	1861
41	2286	1861
42	1861	1861
43	2217	1861
44	2240	2200
45	2271	1861
46	2271	1861
47	2266	2200
48	2261	1861
49	2271	1861
50	2716	1861

A média dos *makespans* das soluções encontradas pelo AG é 2326 u.t. e a média dos menores *makespans* encontrados é 1942 u.t., o que mostra uma diferença de aproximadamente 16% entre o valor das médias dos *makespans* das duas colunas. Nota-se que, em 84% dos testes, uma solução melhor do que a solução final apresentada pelo AG foi encontrada anteriormente e perdeu-se durante o processo evolutivo.

Aplicando-se o teste de Wilcoxon, observou-se que, com 99,9% de confiabilidade, os resultados das colunas "MKP solução" e "menor MKP" são estatisticamente diferentes e que, os resultados da coluna "menor MKP" são melhores, ou seja, se o melhor cromossomo encontrado no processo de busca não se perdesse, com 99,9% de confiabilidade, os resultados obtidos pelo método proposto seriam melhores do que resultados apresentados como solução.

Pode-se concluir, também, novamente aplicando o teste de Wilcoxon, que os resultados da coluna "menor MKP" são estatisticamente diferentes e melhores que os resultado obtido por Maggio (2005), com 99,9% de confiabilidade e, pode-se concluir também com 95% de confiabilidade que os resultados da coluna "menor MKP" não têm diferença estatisticamente relevante dos resultados obtidos por Reyes et al (2002), Yu et al (2003a) e Yu et al (2003b).

Observando, ainda, a perda de soluções melhores no decorrer do processo evolutivo, na tabela 23 pode-se observar a mesma análise com relação a um problema maior, com maior quantidade de produtos e máquinas: o Problema 5.

Tabela 23: *Makespan* da solução apresentada e melhor *makespan* obtido para o Problema 5.

	MKP solução	menor MKP
1	6068	4962
2	5474	4945
3	5397	4865
4	5468	4933
5	5891	4628
6	5386	5043
7	4914	4914
8	5408	4932
9	5412	4952
10	6295	5089
11	5907	5017
12	5929	4928
13	5539	4987
14	5944	4973
15	5188	5003
16	5428	4929
17	5573	5017
18	5406	4929
19	5023	4898
20	5207	4735
21	5239	5015
22	5685	4961
23	5517	4753
24	5826	4952

25	5175	5024
26	6073	5018
27	4970	4970
28	5997	4903
29	5757	4926
30	5916	4931
31	5086	4823
32	5862	4975
33	4988	4988
34	6006	5064
35	5494	5049
36	5600	4901
37	5477	4687
38	5507	4670
39	4562	4562
40	5418	4983
41	5774	4875
42	5524	4923
43	5524	5017
44	5340	5029
45	5280	4907
46	5117	4670
47	5415	5025
48	5136	5041
49	5099	5026
50	5429	5007

A média dos *makespans* das soluções encontradas pelo AG é 5493 u.t. e a média dos menores *makespans* encontrados é 4927 u.t., o que mostra uma diferença de aproximadamente 10% entre o valor das médias dos *makespans* das duas colunas. Nota-se que, em 92% dos testes, uma solução melhor do que a solução final apresentada pelo AG foi encontrada e perdeu-se durante o processo evolutivo.

Através da aplicação teste de Wilcoxon, observou-se que, com 99,9% de confiabilidade, os resultados da coluna "menor MKP" são estatisticamente diferentes e melhores que os resultados da coluna "MKP solução".

Através desta análise, pode-se perceber que, mesmo tendo sido usada uma técnica de elitismo no AG proposto, tal técnica não foi suficiente para conservar os melhores cromossomos encontrados para as próximas gerações. A técnica de elitismo aplicada consiste em sempre selecionar o cromossomo com melhor valor de *fitness* da população da geração corrente para a população intermediária, para, depois, ser aplicada probabilidade de cruzamento e mutação. Esta técnica não garante que o cromossomo com melhor valor de *fitness* esteja presente na população da próxima geração, pois, se este sofrer cruzamento e/ou

mutação, o valor de *fitness* do novo cromossomo pode ser pior do que o valor de *fitness* do cromossomo original.

Com base nestas observações, a primeira sugestão de alteração no método proposto é implementar uma técnica de elitismo que selecione o melhor cromossomo da população da geração corrente e insira-o diretamente na população da próxima geração, sem sofrer qualquer operação genética, garantindo assim, que o melhor cromossomo da população corrente não se perca durante a evolução. Tal técnica aumentaria a pressão de seleção, tornando a busca menos aleatória, embora ela possa favorecer a convergência prematura, levando o algoritmo a encontrar soluções ótimas locais.

Outra sugestão para uma técnica diferente e, talvez, mais eficiente de elitismo é, após o cruzamento, selecionar para a população da próxima geração os cromossomos com maiores valores de *fitness* entre cada pai e seu respectivo filho, como proposto por Chiu e Fu (1997), ou os dois cromossomos com maiores valores de *fitness* entre os dois pais e os dois filhos, conforme proposto por Takahashi *et al* (1996).

Também se sugere que seja alterado o operador de mutação, de maneira que ele opere sobre a ordem de dois produtos, aleatoriamente escolhidos no cromossomo, alterando-a, além de alterar o roteiro de fabricação escolhido, ou seja, altere também as prioridades dos produtos na programação, inserindo mais diversidade genética na população, fazendo com que as soluções fiquem mais "espalhadas" pelo espaço de busca. Esta sugestão pode sem implementada no operador de mutação já usado ou pode ser criado um novo operador de mutação.

Como a combinação das taxas de cruzamento e mutação para a busca com AG pode ser crítica para a eficiência da busca, outra sugestão é alterar o método proposto para um AG adaptativo, que adapta tais taxas à medida que a população evolui, de modo a maximizar o valor da média de *fitness* da população (Barcellos, 2000).

Tais sugestões necessitam ser implementadas isoladamente, ou, talvez, a combinação de algumas delas, e testadas para os problemas em questão e, depois, verificados os resultados e comparados com os resultados obtidos e mostrados neste trabalho, a fim de verificar se houve melhora nas soluções encontradas.

6.2.2 Sugestões para trabalhos futuros

Na seção 6.2.1, foram sugeridas algumas possibilidades de alteração do método proposto, com a finalidade de melhorar o desempenho do AG em encontrar soluções ótimas

ou quase ótimas para o problema de programação da produção em sistemas de manufatura com recursos compartilhados.

Outras alterações também podem ser feitas, como alteração da função objetivo para outros parâmetros utilizados como medida de desempenho nas manufaturas, por exemplo: taxa de utilização de máquinas, cumprimento de data devida e outros. Pode-se também usar lógica nebulosa para combinar mais de um parâmetro como função objetivo.

Uma sugestão, com relação à realização dos testes é simular os resultados obtidos pelo Programador da Produção baseado em AG em um ambiente de manufatura modelado usando um *software* de simulação para verificar o quanto estes resultados se aproximam da realidade.

Quanto à modelagem e ao problema a ser resolvido, pode-se inserir também, além da programação da produção, a programação dos AGVs, ou seja, o tratamento do transporte de uma máquina para a outra, que não está previsto neste trabalho.

6.3 Quanto ao *software* desenvolvido

O *software* Programador da Produção baseado em AG, que implementa o método proposto neste trabalho, apresenta algumas funcionalidades verificadas em *software* para programação da produção disponível no mercado, tais como: cadastro de produtos, roteiros de fabricação e máquinas e visualização da programação encontrada por meio de um gráfico de Gantt.

O *software* foi implementado, conforme visto no capítulo 4, de maneira a ser flexível para a adição de novas funcionalidades e alteração de heurísticas de busca, parâmetros do AG e função de *fitness*.

Algumas sugestões para alterações futuras, com o objetivo de tornar o *software* Programador da produção Baseado em AG mais adequado às necessidades das manufaturas, como inclusão do tratamento de manutenção de máquinas e geração de relatórios adicionais, serão feitas nas seções 6.3.1 e 6.3.2.

6.3.1 Propostas de ampliações do *software* implementado

Com o objetivo de aumentar as funcionalidades do *software* Programador da Produção baseado em AG, de acordo com as necessidades das manufaturas, serão propostas e planejadas algumas ampliações no *software* já implementado e testado.

Dentre as propostas que serão apresentadas, pode ser vista na Figura 21, uma ampliação na a arquitetura geral do sistema implementado, mostrada anteriormente, na Figura 8. A ampliação consiste no acréscimo de um módulo gerador de relatórios, onde, serão implementados métodos para gerar e exibir relatórios sobre as características particulares do problema tratado, tais como, quantidades e tipos de produtos a serem produzidos, e informações sobre a solução encontrada, como roteiro escolhido para cada produto, tempos de operação em cada máquina do roteiro etc. Assim, o Usuário Programador poderá visualizar, para uma programação, além do gráfico de Gantt, seus relatórios complementares.

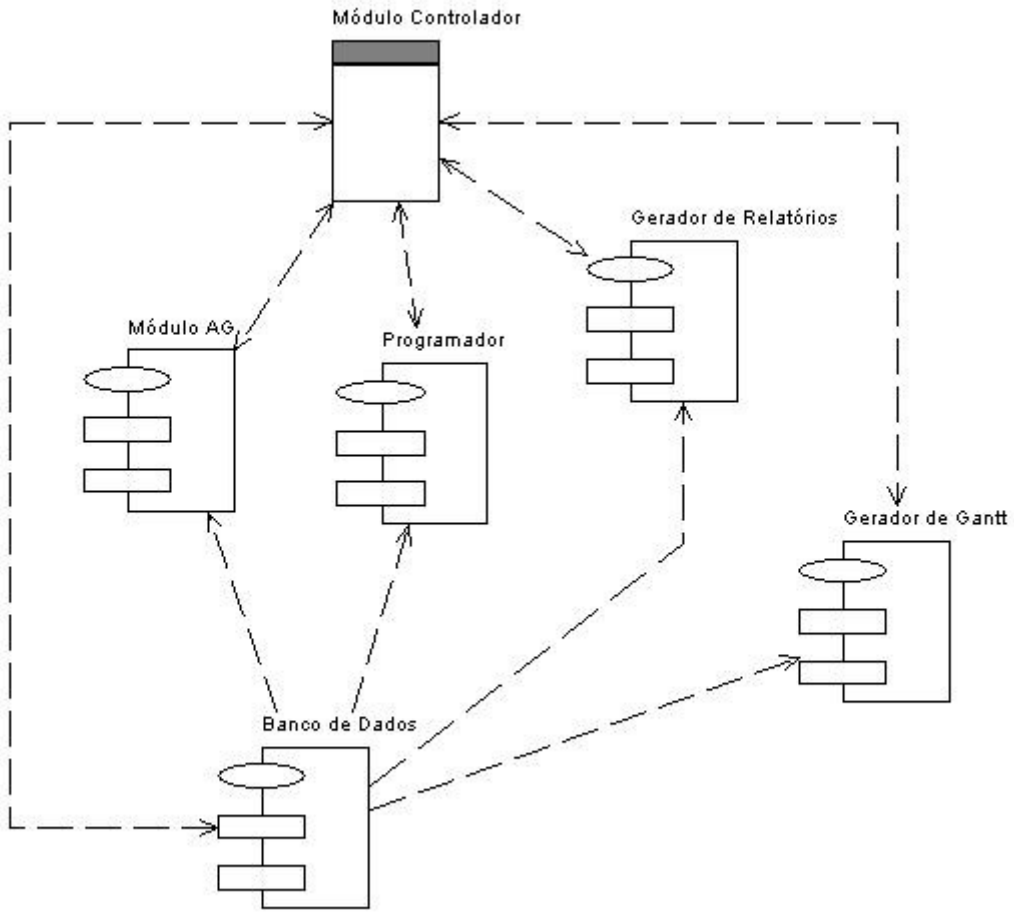

Figura 21 – Proposta de ampliação da arquitetura geral do sistema

Sobre os módulos existentes no sistema, foram implementados alguns casos de uso, citados na tabela 2, divididos em termos dos três tipos de usuários do sistema, de acordo com suas necessidades e funções. Além dos casos de uso mostrados na tabela 2, são propostos outros casos de uso, como pode ser visto na tabela 24.

Tabela 24: Novos casos de uso propostos

Ator Principal (Usuário)	Objetivo
Usuário Programador da Produção	Cadastrar manutenção de máquinas Reprogramar Produção
Administrador do Sistema	Alterar função de *fitness* Alterar heurística

O caso de uso "Cadastrar manutenção de máquinas", do Usuário Programador da Produção refere-se à manutenção preventiva, que pode ser cadastrada com antecedência e corretiva, em caso de falha ou quebra de máquinas, tornando possível a programação reativa tanto dos produtos já em andamento, ou seja, no meio do processo produtivo, quanto dos produtos ainda não iniciados, através do caso de uso "Reprogramar Produção".

Os casos de uso "Alterar função de *fitness*" e "Alterar heurística", do Usuário Administrador do Sistema, referem-se, respectivamente, às possíveis heurísticas de busca que poderão ser adicionadas ao sistema e às outras funções de *fitness* que poderão ser utilizadas como critério de desempenho das soluções.

Com a inclusão destes novos casos de uso e do módulo "Gerador de Relatórios", o novo Diagrama de Seqüência do Sistema é mostrado na Figura 22.

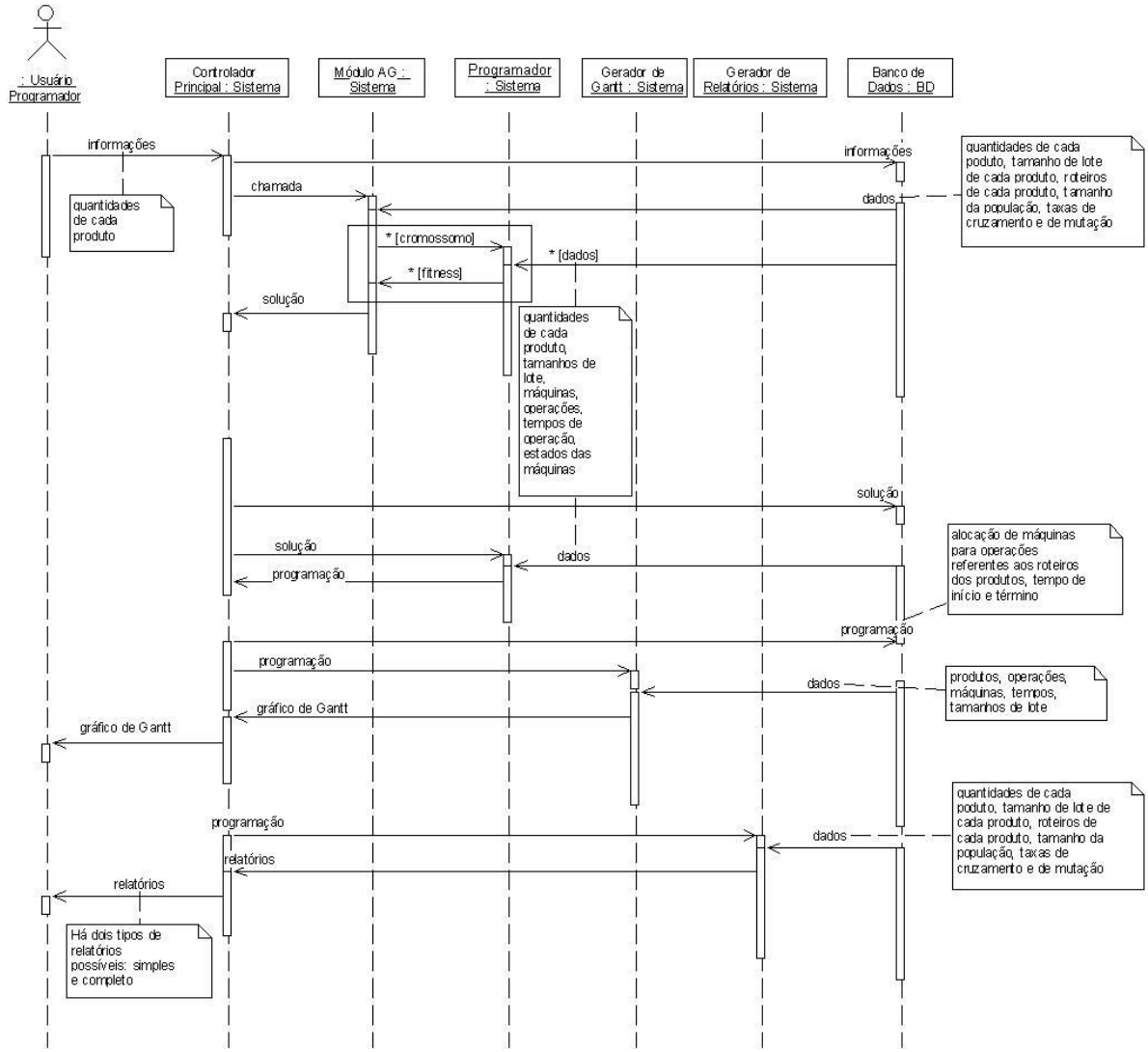

Figura 22 – Diagrama de Seqüência do Sistema com o módulo “Gerador de Relatórios”

O Modelo de Domínio do sistema também sofre modificações, com o acréscimo das classes “Manutenção”, relativa à manutenção das máquinas e da classe “Relatórios”, que se refere à geração dos relatórios sobre o cenário programado, conforme mostrado na Figura 23. Observa-se, também, o acréscimo de duas classes: “Ordem_Fabricação” e “Pedidos”, cujas propostas são armazenar informações sobre as ordens de fabricação a que se refere o cenário a ser programado e a quais pedidos se referem tais ordens de fabricação, respectivamente, considerando que um cenário (problema) pode conter uma ou várias ordens de fabricação e cada ordem de fabricação, por sua vez, pode ser referente a um ou mais pedidos.

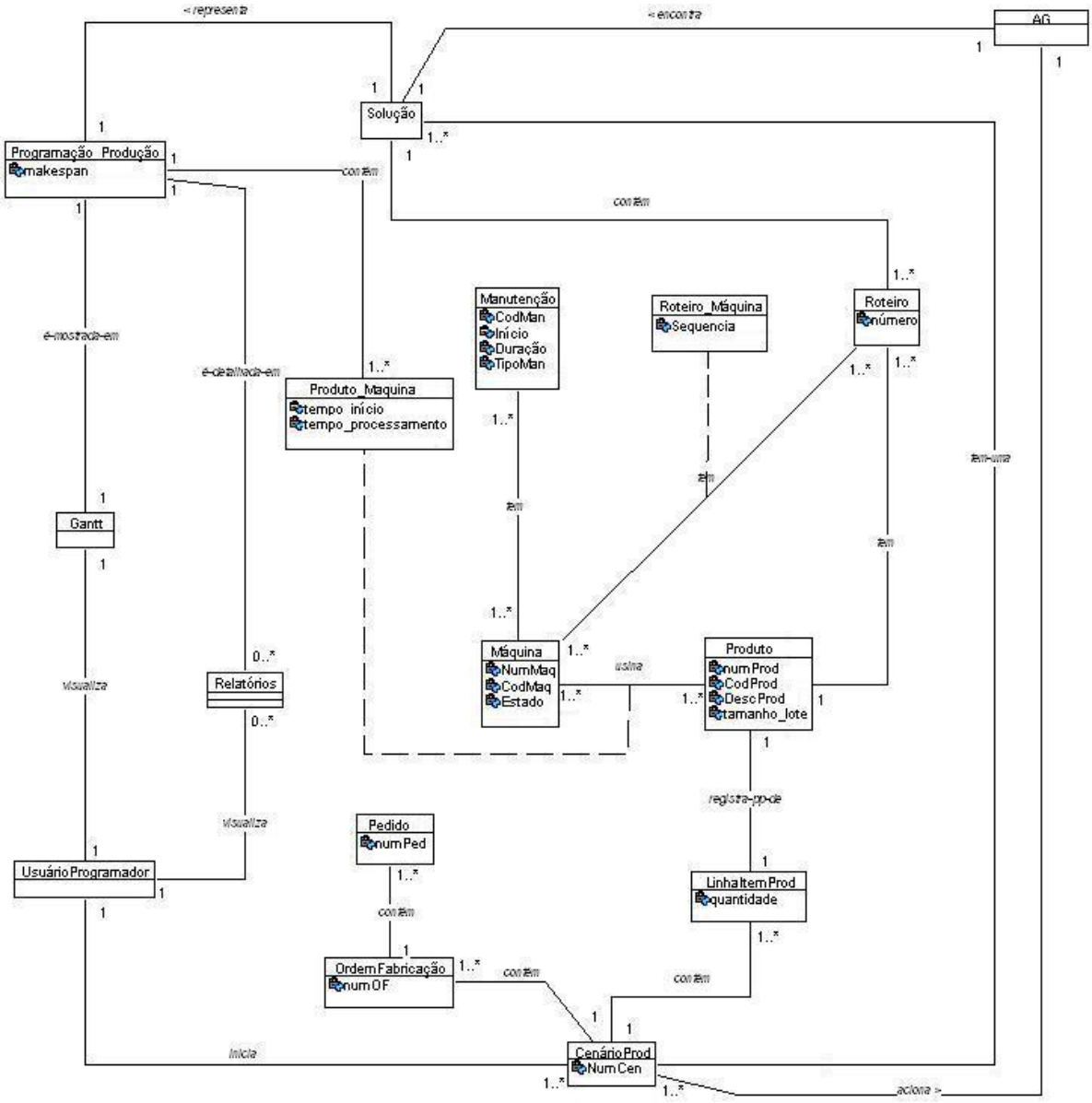

Figura 23 – Modelo de Domínio do Sistema alterado

O Diagrama de Classes de Projeto, com as ampliações propostas, pode ser visualizado na Figura 24, com a inclusão das classes “Relatórios”, “Manutenção”, suas especializações “ManCorretiva” e “ManPreventiva”, “Ordem_Fabricação” e “Pedido”. As responsabilidades de cada uma das classes acrescentadas podem ser vistas na tabela 25 e, na tabela 26, os atributos destas classes.

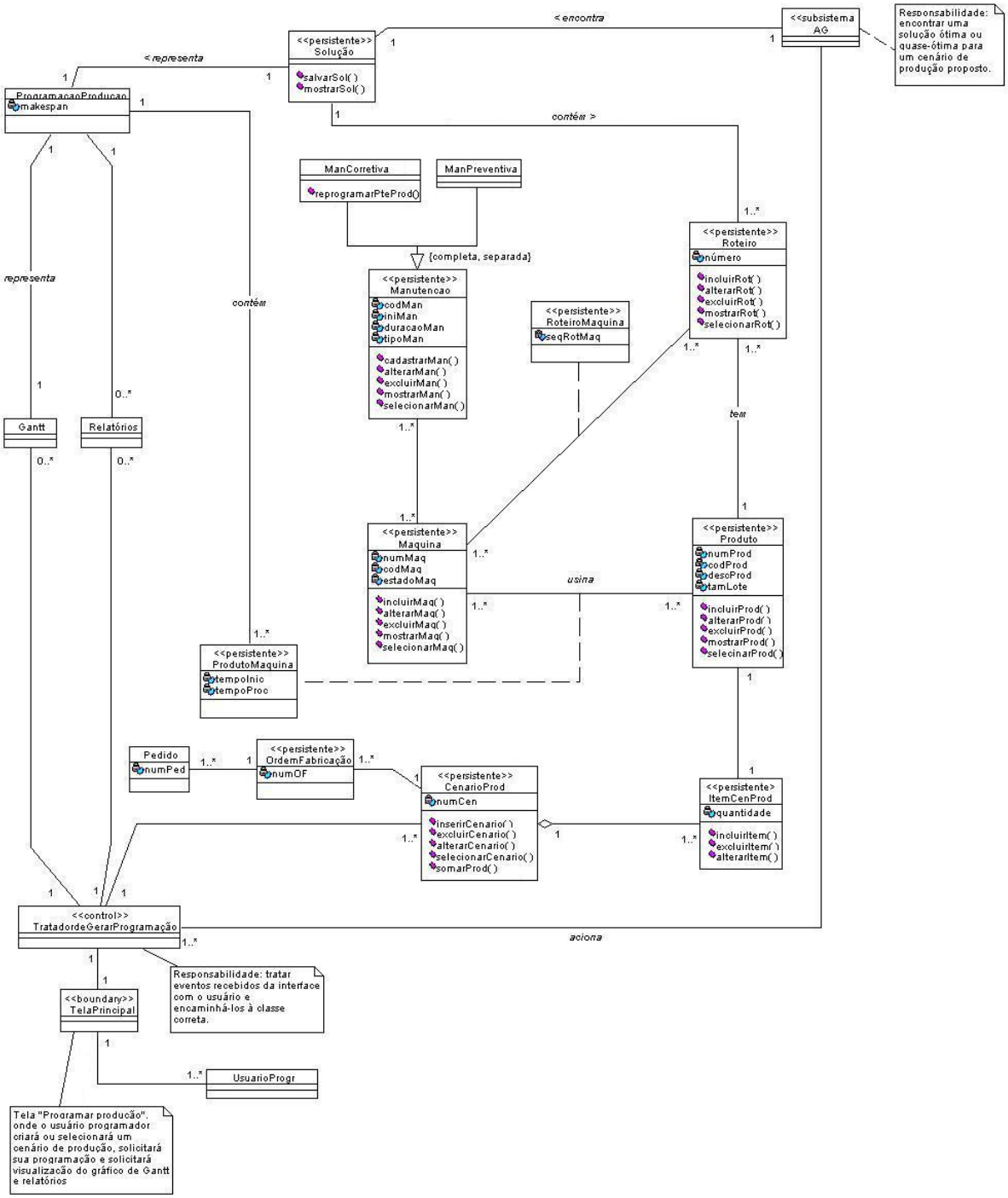

Figura 24 – Diagrama de Classes de Projeto alterado

Tabela 25 – descrição das novas classes e responsabilidades do DCP alterado

Classes	Responsabilidades
Manutenção	Agendar / exibir manutenção preventiva e corretiva de máquinas.
ManCorretiva	Especialização da Classe Manutenção. Quando a manutenção é corretiva, existe a necessidade de se realizar uma reprogramação da produção, pois significa que alguma máquina tornou-se indisponível no meio do processamento de uma programação da produção para um cenário de

	produção.
ManPreventiva	Especialização da Classe Manutenção. Quando a manutenção é preventiva, não existe necessidade de reprogramação da produção. Manutenção preventiva é uma manutenção planejada.
OrdemFabricação	Manter o(s) número(s) da(s) ordem(ns) de fabricação contempladas em um cenário de produção.
Pedido	Manter o(s) número(s) do(s) pedido(s) contemplado(s) em uma ordem de fabricação {não será executado neste sistema}.
Relatórios	Exibir detalhes de um cenário de produção e da programação para ele.

Tabela 26 – descrição das novas classes e atributos do DCP alterado

Classes	Atributos	Tipo	Descrição
Manutenção	**codMan**	Inteiro	Identificador da manutenção para uso interno do sistema. O usuário não tem conhecimento deste número.
	iniMan	Inteiro	Tempo de início da manutenção
	duracaoMan	Inteiro	Tempo de duração da manutenção
	tipoMan	Texto [1]	Tipo de manutenção: C = corretiva; P = preventiva.
OrdemFabricacao	**numOF**	Inteiro	Identificador da ordem de fabricação para uso interno e para o usuário
Pedido	**numPed**	Inteiro	Identificador do pedido para uso interno e para o usuário

6.3.2 Sugestões para trabalhos futuros

O *software* Programador da Produção baseado em AG foi implementado de forma a ser flexível para futuras alterações, algumas já previstas no Modelo de Casos de Uso, para serem implementadas futuramente.

Dentre as futuras alterações já previstas está a alteração da função de *fitness* do AG implementado, que poderia ser escolhida, de acordo com a necessidade da manufatura em determinado momento. Por exemplo, considerando as sugestões propostas na seção 6.2.2, para alterações da função objetivo, ou seja, do critério de desempenho do AG para outros parâmetros, como taxa de utilização de máquinas ou cumprimento de data devida, dependendo do momento e da situação da manufatura, pode ser interessante considerar um ou outro critério de desempenho para a programação da produção. Neste caso, tal critério poderia ser escolhido na hora de gerar a programação e remeteria à respectiva função de *fitness* no AG, gerando uma programação da produção adequada ao critério de desempenho escolhido. Podendo-se, também, usar lógica nebulosa para combinar mais de um parâmetro como função

objetivo, seria possível estabelecer a prioridade de cada parâmetro no critério de desempenho, de acordo com a necessidade da manufatura.

Outra possível alteração seria implementar outros métodos de busca, como busca heurística, por exemplo, com várias heurísticas possíveis e, na hora da programação, o Usuário Programador escolher o método de busca que deseja usar, ou até mesmo escolher mais de um método de busca e comparar os resultados de cada um deles, visualizando seus relatórios e gráficos de Gantt gerados.

Para a possível inclusão do tratamento de transporte, além da programação da produção, também sugerida na seção 6.2.2, podem, facilmente, ser inseridos geradores de relatórios específicos para visualização das informações resultantes desta programação.

Também poderão ser armazenados no banco de dados, quando este estiver implementado e integrado ao sistema, informações sobre os cenários programados, as ordens de produção a que se refere e cada cenário e os pedidos a que se refere cada ordem de produção, conforme já foi previsto e pode ser visualizado na Figura 13 que mostra o Modelo de Domínio do sistema.

Conclusões finais

Este trabalho propôs um método de busca baseado em AG e um *software*, que o implementou, para programação da produção de sistemas de manufatura com compartilhamento de recursos, cujos critérios de desempenho foram o *makespan* e o tempo de execução da busca. O método e o *software* propostos se mostraram promissores para problemas grandes, em relação à quantidade de máquinas e produtos, especialmente por obter soluções em tempo de processamento baixo, o que não ocorreu com outros métodos de busca testados para o mesmo problema. Com relação ao *makespan*, concluiu-se que as soluções podem ser melhoradas.

Assim, foram propostas possíveis alterações para melhorar os resultados encontrados pelo método de busca e aumentar as funcionalidades do *software*, a fim de adequá-lo às reais necessidades das manufaturas.

Referências

BARCELLOS, J. C. H. **Algoritmos genéticos adaptativos**: um estudo comparativo. Dissertação de Mestrado em Engenharia – Escola de Politécnica, Universidade de São Paulo. São Paulo, 2000.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML guia do usuário**. 1.Ed. Rio de Janeiro: Campus, 2000.

CAMPOS, M.C. **Estatística prática para docentes e pós graduados**. Disponível em: <http://www.forp.usp.br/restauradora/gmc/gmc_livro/gmc_livro.html>. Acesso em: 11 out. 2007.

CARVALHO, V. O. **Um modelo de seqüenciamento da produção para um sistema de apoio à decisão**. 109 f, Dissertação de Mestrado em Ciência da Computação – Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2003.

CASTRO, P. A. D. **Um paradigma baseado em algoritmos genéticos para o aprendizado de regras fuzzy**. 75 f, Dissertação de Mestrado em Ciência da Computação – Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2004.

CAVALIERI, S. Petri nets and genetic algorithms to increase productivity in FMS. In: **Second International Conference on Knowledge-Based Intelligent Electronic Systems**, Adelaide, Australia, 1998. p. 134-142.

CHAN, F. T. S.; CHUNG, S. H.; CHAN, P. L. Y. An introduction of dominant genes in genetic algorithm for scheduling of FMS. In **Proceedings of the IEEE International Symposium on Intelligent Control**. Limassol, Cyprus, Junho 27-29, 2005. p. 1429 – 1434.

CHIU, Y. F., FU, L. C. A. GA embedded dynamic search algorithm over a petri net model for an FMS scheduling. In: **Proceedings of the 1997 IEEE International Conference on Robotics and Automation**, Albuquerque, New Mexico, Abril, 1997. v. 1 p.513-518.

CONWAY, R. W.; MAXWELL, W. L.; MILLER, L. W. **Theory of scheduling.** Reading, Massachusetts: Addison-Wesley, 1967.

GANG, X., WU, Z. Deadlock-free scheduling method using petri net model analysis and GA search. In: **Proceedings of the 2002 IEEE International Conference Control Applications**, Glasgow, Scotland, U.K., Setembro, 2002. p. 1153-1158.

HUANG, Z., WU, Z. Deadlock-free scheduling method for automated manufacturing systems using genetic algorithm and petri nets. In: **Proceedings of the 2004 IEEE International Conference on Robotics and Automation**, New Orleans, LA, Abril, 2004. p. 66-571.

HUANG, A.C. et al. Modeling, scheduling and prediction for wafer fabrication: queueing colored petri net and GA based approach. In: **Proceedings of the 2002 IEEE**

International Conference on Robotics and Automation, Washington, D.C., Maio 2002. p. 3187-3192.

JEONG, S. J. ; LIM, S. J. ; ·KIM, K. S. Hybrid approach to production scheduling using genetic algorithm and simulation. **International Journal of Advanced Manufacturing Technology**, London, v. 28, p. 129 – 136, 2006.

LACERDA, E. G. M., CARVALHO, A. C. P. L. F., **Introdução aos algoritmos genéticos**. In: Anais XIX Congresso Nacional da Sociedade Brasileira de Computação, Rio de Janeiro, Julho 1999. p. 51-126.

LARMAN, C. **Utilizando UML e padrões**. 2. ed. Porto Alegre: Bookman, 2004.

LEE, D. Y. e DICESARE, F. FMS scheduling using petri nets and heuristic search. In: **Proceedings of the 1992 IEEE International Conference on Robotics and Automation**, **Anais.** Nice, Fr, Maio 1992. p. 1057-1062.

LEE, D. Y. e DICESARE, F. Scheduling flexible manufacturing systems using petri nets and heuristic search. **IEEE Transactions on Robotics and Automation**, v.10, n. 2, p. 123-133, 1994.

LIN, Y. L. et al. Colored timed petri net and GA based approach to modeling and scheduling for wafer probe center. In: **Proceedings of the 2003 IEEE International Conference on Robotics and Automation**, Taipei, Taiwan, Setembro 2003. p. 1434-1439.

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

MARTINS, G. A. **Estatística geral e aplicada**. 3. Ed. São Paulo: Atlas, 2005.

MORANDIN JR., O. **Metodologia de modelagem de sistemas flexíveis de manufatura utilizando rede de petri virtual**. Tese de Doutorado – Escola de Engenharia de São Carlos, Universidade de São Paulo. São Carlos, 1999.

MURATA, T. Petri Nets: Properties, analysis and applications. In: **Proceedings of the IEEE**. v. 77, n.4, p.541-580, 1989.

PONGCHAROEN et al. Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. **International Journal of Production Economics**. n. 78, p. 311–322, 2002.

PONGCHAROEN, P.; HICKS, C. & BRAIDEN, P.M. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure. **European Journal of Operational Research**. n. 152, p. 215–225, 2004.

REYES, A., YU, H. e KELLEHER, G. Advanced scheduling methodologies for flexible manufacturing systems using Petri nets and heuristic search. In: **ICRA 2000: IEEE**

International Conference on Robotics and Automation. **Anais.** San Francisco, CA, USA, 2000. p. 2398-2403

REYES, A., YU, H. e LLOYD, S. An evolutionary hybrid scheduler based in petri net structures for FMS scheduling. In: **IEEE International Conference on Systems Man and Cybernetics**, 2001, v. 4, p. 2516 – 2521.

REYES, A., YU, H. e KELLEHER, G. Hybrid heuristic search for the scheduling of flexible manufacturing systems using Petri nets. **IEEE Transactions on Robotics and Automation**, v.18, n. 2, p. 240-245, 2002.

REYES, A., et al. Integrating petri nets and hybrid heuristic search for the scheduling of FMS, **Computers in Industry**, v.47, n. 1, p. 123-138. 2002.

RUSSEL, S. J. & NORVIG, P. **Inteligência artificial: um enfoque moderno**. 2. Ed. São Paulo: Campus, 2004.

SILVA, E. L.; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 3 ed. Florianópolis: Laboratório de Ensino à Distância da UFSC, 2001

SLACK, N., et al. **Administração da Produção**. São Paulo: Atlas, 1999.

TAKAHASHI, K., YAMAMURA, M. e KOBAYASHI, S. A GA approach to solving reachability problems for petri nets. In: **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**. Institute of Electronics, Inf & Commun Engineers of Japan, v. 79-A, n 11, p 1774-1780, Novembro 1996.

TEMPELMEIER, H. e KUHN, H. **Flexible manufacturing systems –** decision support for design and operation. New York, NY: John Wiley & Sons, Inc. 1993.

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

VALETTE, R., PRANDIN-CHÉZAVIEL, B. e GIRAUT, F. **An introduction to petri net theory**. Physica-Verlag Fuzziness in Petri Nets – Studies in Fuzzyness and Soft Computing. Heidelberg, Germany: CARDOSO, J.; CAMARGO, H., v. 22, 3-24, 1999.

YEUNG, W. H. R., MOORE, P. R. Genetic algorithms and flexible process planning in the design of fault tolerant cell control for flexible assembly systems. **International Journal of Computer Integrated Manufacturing**, vol. 13, n 2, p. 157-168, 2000.

YU, H., et al. Combined petri net modeling and AI based heuristic hybrid search for flexible manufacturing systems **-** part I petri net modeling and heuristic search. **Computers and Industrial Engineering,** v. 44, n. 4, p. 527-543, 2003a.

YU, H., et al. Combined petri net modeling and AI-based heuristic hybrid search for flexible manufacturing systems **-** part II. heuristic hybrid search. **Computers and Industrial Engineering**, v. 44, n. 4, p. 545-566, 2003b.

APÊNDICE A

Algoritmos Genéticos

Algoritmos Genéticos (AG) são métodos de otimização e busca inspirados nos mecanismos de evolução dos seres vivos. Foram introduzidos por John Holland [Holland, 1975] e popularizados por um dos seus alunos, David Goldberg [Goldberg, 1989 apud Lacerda e Carvalho (1999)]. O princípio que norteia os AGs é o da seleção natural e sobrevivência do mais apto, declarado em 1859 pelo naturalista e fisiologista inglês Charles Darwin em seu livro "**A Origem das Espécies**", onde afirma: "Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes".

As técnicas de busca e otimização de problemas geralmente apresentam (Lacerda e Carvalho, 1999):

- Um *espaço de busca*, onde se encontram todas as possíveis soluções para o problema;
- Uma *função objetivo* (chamada de *função de aptidão* ou *função de fitness*, quando se trata de AGs), que é utilizada para avaliar as soluções produzidas, associando a cada uma delas um valor numérico.

1 Funcionamento

Primeiramente, um Algoritmo Genético gera uma população inicial de cromossomos, onde cada cromossomo representa uma possível solução do problema a ser resolvido. Durante o processo de evolução, a população é avaliada e cada cromossomo recebe uma *nota de aptidão*, que representa a qualidade de sua solução. Geralmente, os cromossomos mais aptos são selecionados e os menos aptos são descartados. Os membros selecionados podem sofrer modificações através dos operadores de cruzamento e mutação, gerando descendentes para a próxima geração. Este processo é repetido até que uma solução satisfatória seja encontrada, conforme ilustrado no Algoritmo 5.1 (Lacerda e Carvalho, 1999).

Algoritmo A.1 – um algoritmo genético típico (Lacerda e Carvalho, 1999)

Seja S(t) a população de cromossomos na geração t.
t ← 0
inicializar S(t)
avaliar S(t)

enquanto o critério de parada for não satisfeito **faça**
    t ← t+1
    selecionar S(t) a partir de S(t-1)
    aplicar cruzamento sobre S(t)
    aplicar mutação sobre S(t)
    avaliar S(t)
fim enquanto

Cada etapa dos AGs típicos será descrita com mais detalhes a seguir.

1.1 Representação dos Parâmetros

Um **cromossomo** é uma estrutura de dados que representa uma possível solução para o problema a ser resolvido. Geralmente, um cromossomo representa parâmetros da função objetivo. O conjunto de todas as configurações que o cromossomo pode assumir forma o seu **espaço de busca**.

O primeiro passo para se resolver um problema utilizando AG é escolher a representação do cromossomo mais adequada ao problema. Um cromossomo pode ser representado por uma cadeia de *bits* ou por um vetor numérico. No caso de a função objetivo ter vários parâmetros, a cadeia ou vetor deve representar todos eles, ou seja, cada parte do cromossomo representa um parâmetro. Cada posição do cromossomo recebe o nome de *gene*. Cada cromossomo é um *indivíduo* e um conjunto de cromossomos é uma *população*, conforme ilustrado na Figura A.1, com os cromossomos c1, c2, c3, c4 e c5 codificados com valores inteiros.

Escolhida a maneira de representar o cromossomo, é associada a cada cromossomo uma nota de aptidão que significa a qualidade daquela solução para o problema.

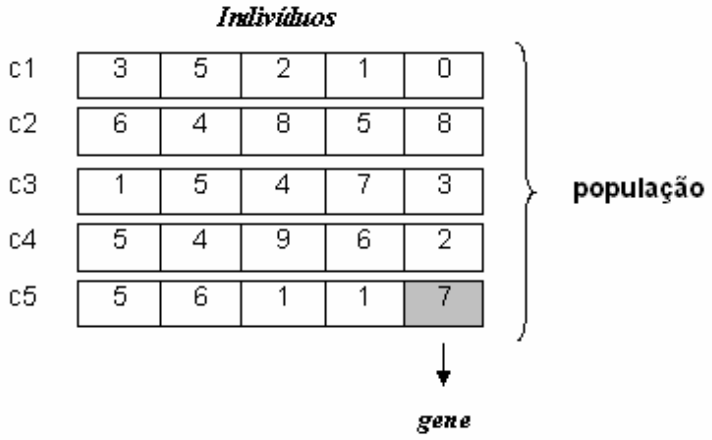

Figura A.1 – população de cromossomos

1.2 Seleção

Inicialmente é gerada uma população de N cromossomos. O valor de N pode ser aleatório e influencia diretamente o desempenho e a eficiência do algoritmo. Se a população for pequena, o algoritmo é mais rápido, mas pode não abranger todo o espaço de soluções. Se a população for grande, oferecerá maior cobertura representativa do domínio do problema, mas o algoritmo será mais lento. Os melhores cromossomos da população inicial (com maior valor de aptidão) deverão ser selecionados para gerar filhos através de cruzamento e mutação. A probabilidade de seleção de um cromossomo pai é, geralmente, proporcional ao seu valor de aptidão.

Os cromossomos pais selecionados são colocados em uma *população intermediária*, também chamada de *mating pool.*

Há vários métodos de seleção, dos quais, os mais conhecidos serão apresentados a seguir:

- **Roleta:** Neste método, os cromossomos podem ser representados como segmentos consecutivos dispostos em uma roleta. Cada cromossomo ocupa um espaço na roleta de tamanho proporcional a sua aptidão. A roleta será girada N vezes (N é o tamanho da população inicial). A cada vez que a roleta parar de girar, o cromossomo selecionado pelo marcador será copiado para a próxima geração. Cromossomos com maior espaço na roleta terão maior chance de serem selecionados. Se os cromossomos apresentarem valor de aptidão muito diferentes (exemplo, se um cromossomo ocupar 90% do espaço da roleta e os outros dividirem os 10% restantes), este método pode levar a perda de diversidade genética e à *convergência prematura* da população, ou seja, a solução converge para um máximo ou mínimo local, sem a exploração de todo o espaço de busca.

- **Amostragem Universal Estocástica:** É semelhante ao método da Roleta, mas com um número de marcadores igual ao número de indivíduos da população inicial, igualmente espaçados na roleta. Dessa forma, a roleta gira apenas uma vez e cada cromossomo apontado por um marcador é copiado para a próxima geração. Neste método, cromossomos que ocupam pequeno espaço na roleta têm mais chance de serem selecionados, garantindo a diversidade genética da população. A Figura A.2 ilustra este método.

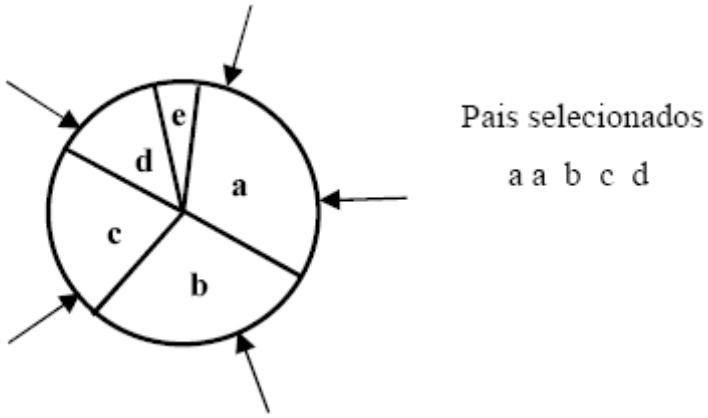

Figura A.2 – Amostragem Universal Estocástica

- **Torneio:** Um número qualquer, k, $2 \leq k \leq N$, de indivíduos é selecionado aleatoriamente e o que possuir melhor valor de aptidão é copiado para a próxima população. O processo é repetido até que a população esteja completa.

Mesmo que os melhores cromossomos tenham maior probabilidade de serem selecionados, nada impede que estes não sejam efetivamente escolhidos para a próxima geração. Para garantir que os melhores cromossomos da população permaneçam na próxima geração, pode-se adicionar ao método de seleção escolhido, uma técnica chamada *elitismo*, que consistem em substituir aleatoriamente n indivíduos da população atual pelos n melhores indivíduos da população anterior.

1.3 Cruzamento

O cruzamento consiste em escolher dois cromossomos pais para cruzar e gerar cromossomos filhos, que os substituirão na população. É necessário especificar uma taxa de cruzamento, que definirá a probabilidade com a qual indivíduos serão cruzados. Se esta taxa for alta, novos indivíduos serão inseridos na população mais rapidamente, mas pode-se perder indivíduos aptos. Se for muito baixa, o algoritmo pode se tornar lento.

Existem vários operadores de cruzamento, dos quais alguns serão especificados a seguir:

- **Cruzamento de um ponto:** um ponto é escolhido aleatoriamente, no qual os dois cromossomos são cortados, gerando duas cabeças e duas caudas. As cabeças e caudas são trocadas, de modo que cada cromossomo filho seja formado pela cabeça de um pai e a cauda do outro.

- **Cruzamento de n pontos:** são escolhidos n pontos de cruzamento, dividindo os cromossomos pais em n+1 partes, que podem ser trocadas entre eles para gerar os filhos. As Figuras A.3 e A.4 mostram, consecutivamente, uma operação de cruzamento de 2 pontos e uma operação de cruzamento de 4 pontos, ambas em cromossomos com codificação binária.

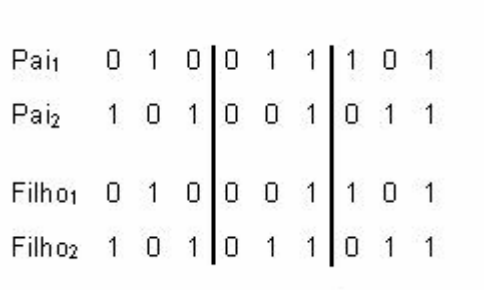

Figura A.3: cruzamento de 2 pontos

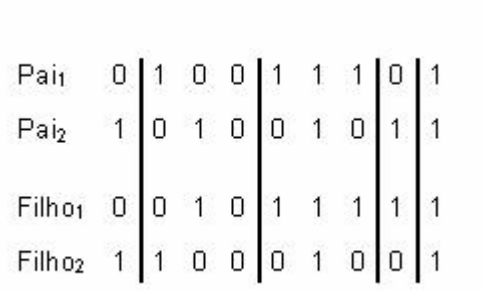

Figura A.4: cruzamento de 4 pontos

- **Cruzamento max-min aritmético:** este operador é aplicado a cromossomos com codificação real. Sejam os cromossomos pais $C_v = \{c_{v1}, ...,c_{vn}\}$ e $C_w = \{c_{w1}, ...,c_{wn}\}$, os filhos gerados são:

$$c_{1i} = a*c_{wi}+(1-a)*c_{vi}$$

$$c_{2i} = a*c_{vi}+(1-a)*c_{wi}$$

$$c_{3i} = \min \{c_{vi},c_{wi}\}$$

$$c_{41} = \max \{ c_{vi},c_{wi}\}$$

Onde a é um número real, $0 \leq a \leq 1$, que pode ser constante ou variar a cada geração e $i = 1,...,n$. Dos quatro filhos gerados, os dois melhores serão inseridos na população, conforme a Figura A.5, onde $a = 0,35$.

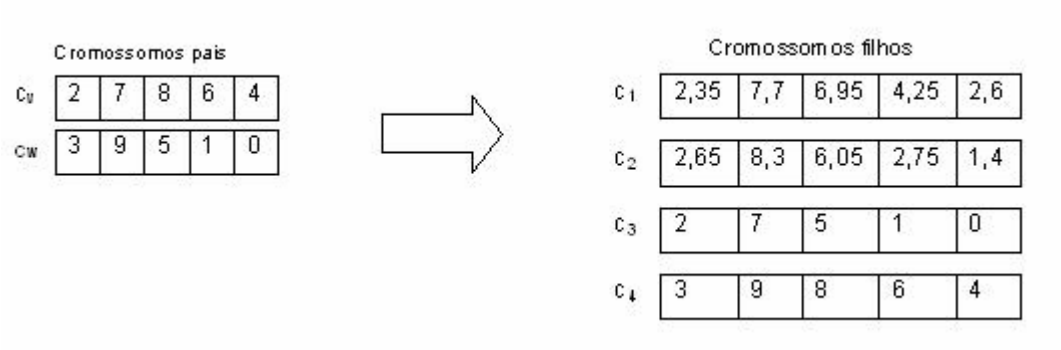

Figura A.5 – Cruzamento max-min aritmético

1.4 Mutação

A mutação é a operação de escolher aleatoriamente um gene do cromossomo e alterar seu valor. O objetivo da mutação é introduzir e manter diversidade genética na população. Como no cruzamento, existe uma taxa de mutação, que, se for alta demais, pode tornar o algoritmo desprovido de direção no espaço de busca e se for baixa demais pode tornar o processo de busca lento.

Os operadores de mutação mais conhecidos e utilizados são:

- **Mutação padrão:** Simplesmente altera-se o valor do gene escolhido. Para cromossomos de codificação binária, basta inverter o bit, ou seja, trocar 1 por 0 ou vice-versa. Para cromossomos de codificação real, troca-se o gene por um dos possíveis valores que ele pode assumir. A Figura A.6 ilustra uma mutação padrão em cromossomos com codificação binária.

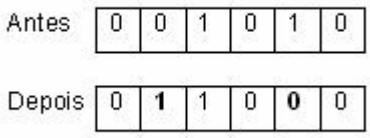

Figura A.6: mutação padrão

- **Mutação não uniforme:** Este operador é mais utilizado em cromossomos de codificação real. Seja $C = \{c_1,...,c_k,...,c_n\}$ um cromossomo que deverá sofrer mutação e c_k o gene escolhido para ser alterado. O cromossomo resultante será $C' = \{c_1,...,c_{k'},...,c_n\}$, onde:

$$ck' = \begin{cases} c_k + \Delta(t, LS(c_k) - c_k) \text{ se } a = 0 \\ c_k - \Delta(t, c_k - LI(c_k)) \text{ se } a = 1 \end{cases}$$

Onde, a é um dígito binário gerado aleatoriamente, LS é o limite superior e LI o limite inferior que c_k pode assumir e a função $\Delta(t,y)$ é uma função do tipo:

$$\Delta(t,y) = y\ (1-r^{(1-t/T)b})$$

Em que t é a geração atual, T é o número máximo de gerações, r é um valor no intervalo $[0,1]$ e b é um parâmetro que controla o grau de não uniformidade do operador.

5.2 – Algoritmos auto adaptativos

Para utilizar AGs, é necessário definir uma série de parâmetros que influenciam na eficiência e eficácia do algoritmo em explorar o espaço de busca. Estes parâmetros são: tamanho da população, taxa de mutação e taxa de cruzamento.

A escolha de valores inadequados para esses parâmetros pode causar uma série de problemas como a c*onvergência prematura,* causada pela perda de diversidade na população, que faz com que a solução convirja prematuramente para pontos de máximo ou mínimo locais.

Os valores escolhidos para estes parâmetros permanecem fixos durante toda a execução do algoritmo, e podem causar diferença de desempenho. Alguns valores podem ser eficientes para um problema especifico, mas para outros não.

Uma maneira de melhorar a forma com que o algoritmo explora o espaço de busca é o ajuste dos valores desses parâmetros em tempo de execução.

Algoritmos auto-adaptativos têm a capacidade de alterar os valores iniciais de tais parâmetros durante o curso de evolução, com base no atual estado da população em termos de diversidade genética.

APÊNDICE B

PROCESSO UNIFICADO E LINGUAGEM DE MODELAGEM UNIFICADA

1 introdução

A Linguagem de Modelagem Unificada (do inglês *Unified Modeling Language* – UML) é uma linguagem que foi criada para documentação de projetos de *software* orientados a objetos, para especificar, visualizar, construir e documentar artefatos de *software*.

A UML é o resultado da unificação dos métodos Booch, OMT (*Object Modeling Technique*) e *Objectory*, originando uma linguagem padronizada para modelagem de *softwares*, sendo adotada pela indústria de *software* como linguagem-padrão e também por fornecedores de ferramentas CASE (*Computer Aided System Engineering* – Engenharia de Sistemas Auxiliada por Computador).

Um conceito importante na área de engenharia de *software*, os padrões, foi também adotado pela UML. Padrões são soluções clássicas, verificadas no processo de desenvolvimento de *software* aplicados na construção de novos sistemas.

A UML não é Análise e Programação Orientada a Objetos (A/POO) ou um método, é simplesmente uma notação e é independente de processo, ou seja, é possível usá-la com vários processos de engenharia de *software*.

Segundo Booch, Rumbaugh e Jacobson (2000) “um processo é um conjunto de passos parcialmente ordenados com a intenção de atingir uma meta” e “na engenharia de *software*, sua meta é entregar, de maneira eficiente e previsível, um produto de *software* capaz de atender às necessidades de seu negócio”.

Um dos processos usados para o desenvolvimento de *software* orientado a objetos é o Processo Unificado (PU). Ele se baseia no princípio do desenvolvimento iterativo, ou seja, o sistema é desenvolvido em ciclos ou iterações, uma série de miniprojetos de duração fixa, em que o produto de cada miniprojeto é um sistema testado, integrado ao sistema todo e executável. Cada iteração tem atividades de análise de requisitos, projeto, implementação e testes.

No PU, responsabilidades são atribuídas a classes, influenciando a robustez, a facilidade de manutenção e a reusabilidade de componentes de *software*. Uma classe é um

conjunto de objetos que compartilham os mesmos atributos, operações e relacionamentos.

Os artefatos UML auxiliam na representação de elementos do projeto, que é uma solução conceitual dos requisitos, e não sua implementação.

O objetivo deste apêndice é oferecer uma visão geral e resumida do Processo Unificado e citar alguns artefatos, dentre os muitos, que podem ser usados em algumas fases e iterações do PU, não sendo citadas, nem tampouco detalhadas, todas as fases, iterações ou etapas (que compõem cada iteração).

Os exemplos citados foram extraídos do livro "Utilizando UML e Padrões", (LARMAN, 2004), sendo a maioria deles baseada em um estudo de caso brevemente descrito a seguir:

Estudo de Caso: Sistema PDV ProxGer (Próxima Geração do sistema de Pontos de Venda)

Um sistema de Pontos-de-Venda é uma aplicação computadorizada usada (em parte) para registrar vendas e cuidar de pagamentos (muito utilizada por lojas de varejo). Inclui componentes de *hardware* (computador, leitor de códigos de barras) e um *software* para rodar o sistema e tem interfaces com aplicações de serviços (aplicação de cálculo de impostos, aplicação de controle de estoque). O sistema PDV deve suportar múltiplos e variados terminais e interfaces do lado cliente.

Detalhes sobre o estudo de caso e as fases de desenvolvimento do sistema podem ser encontrados em Larman (2004).

2 Características do Processo Unificado

O PU é um processo iterativo, ou seja, a cada iteração a compreensão do problema vai se expandindo e o sistema vai sendo refinado e incrementado até convergir para uma solução adequada. No decorrer do processo, pode haver mudanças nos requisitos e nos objetivos de negócio e, sendo iterativo, o PU têm uma grande flexibilidade para acomodar tais mudanças.

Em cada iteração do processo é importante escolher um pequeno subconjunto de requisitos, com enfoque maior naqueles que representam maiores riscos para a instituição nas etapas iniciais. É importante no PU que a equipe de desenvolvimento e as áreas envolvidas tenham como princípios a realimentação e a adaptação, para que possam acolher as mudanças constantemente.

Nas iterações iniciais o sistema tende a não ser exatamente o desejado. Obtendo a realimentação (*feedback*) de usuários, desenvolvedores e testes, a cada iteração o sistema é refinado e converge para o sistema esperado, progredindo por meio de ciclos estruturados em construção-realimentação-adaptação.

O PU busca resolver problemas onde os modelos de desenvolvimento tradicionais de *software*, tais como cascata e *Top-Down*, não oferecem soluções satisfatórias. Em muitas situações o sistema entregue não satisfaz as exigências de quem o encomendou, pois este só tem uma versão utilizável do sistema no final do processo, quando mudanças estruturais normalmente não são viáveis. No desenvolvimento iterativo do PU, a habilidade de progredir e descobrir o que é o certo para os interessados no sistema é uma característica inerente ao processo, fazendo com que os riscos sejam descobertos e tratados no início do processo. Normalmente nas iterações iniciais existe um desvio maior do caminho verdadeiro do que no fim. É natural e positivo no PU que se encontre o rumo certo ao longo do desenvolvimento.

O PU incentiva criação de modelos, que são simplificações da realidade, em vez de documentos impressos. Os modelos proporcionam representações ricas semanticamente e mais facilmente visíveis e compreensíveis do sistema proposto, além de proporcionar a maximização das informações relevantes ao desenvolvimento do sistema.

O desenvolvimento usando PU focaliza a criação de uma arquitetura robusta de *software*, que "facilita o desenvolvimento paralelo, minimiza a necessidade de refazer o trabalho e aumenta a reutilização de componentes e a capacidade de manutenção eventual do sistema" (BOOCH; RUMBAUGH; JACOBSON, 2000).

O desenvolvimento usando PU é orientado por casos de uso, ou seja, com base na compreensão completa de como o sistema será usado. Assim, os casos de uso guiam muitas atividades do processo: os requisitos são primeiramente registrados em casos de uso, que orientam o projeto e influenciam a organização dos manuais de usuário.

O PU suporta e incentiva técnicas orientadas a objetos, seus modelos são baseados nos conceitos de objetos, classes e relacionamentos e são usados os artefatos da UML para sua criação.

O PU pode ser personalizável para um determinado projeto, ou seja, pode ser configurado para as características do projeto no qual será usado, suportando, assim, projetos de vários tamanhos, com equipes e características diferentes.

Além disso, o PU permite o controle de qualidade e gerenciamento de riscos contínuos, através do processo iterativo e da análise de riscos feita no início do processo de

desenvolvimento, permitindo seu tratamento de forma adequada.

Alguns dos principais benefícios do PU são:

- Atenuação precoce de altos riscos;
- Progresso visível desde início;
- Realimentação, envolvimento do usuário e adaptação imediatos;
- A complexidade é administrada (equipe não é sobrecarregada pela "paralisia da análise" ou por passos muito longos e complexos);
- Aprendizado iterativo, melhorando o processo de desenvolvimento.

3 Fases e Iterações

De acordo com Booch, Rumbaugh e Jacobson (2000):

> "uma fase é o período de tempo entre dois importantes marcos de progresso do processo em que um conjunto bem definido de objetivos é alcançado, artefatos são concluídos e decisões são tomadas em relação à passagem para a fase seguinte".

O PU é dividido em quatro fases, conforme ilustrado na Figura B.1.

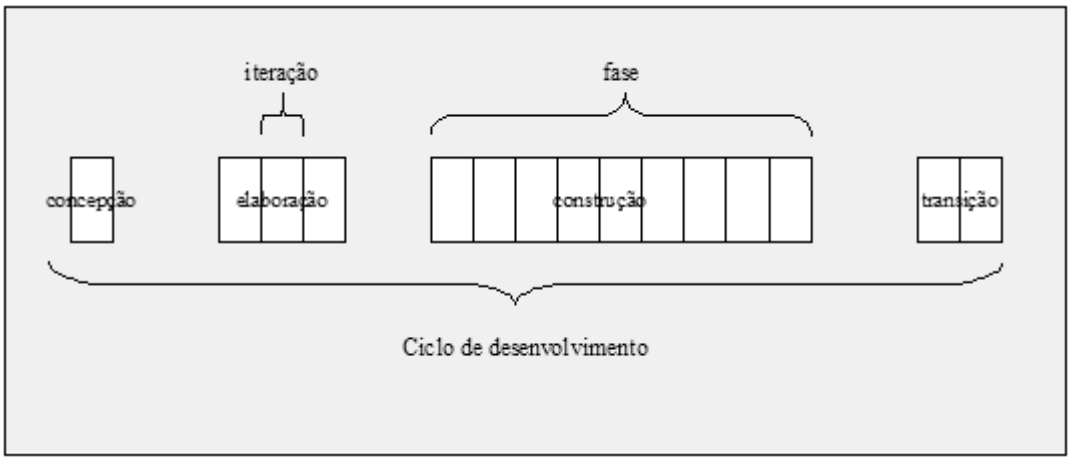

Figura B.1: fases do Processo Unificado
Fonte: Adaptado de Larman (2004)

Cada fase pode ter uma ou mais iterações. Ainda, segundo Booch, Rumbaugh e Jacobson (2000), "uma iteração representa um ciclo completo de desenvolvimento, desde a captação de requisitos na análise até a implementação e a realização de testes, resultando na versão de um projeto executável". Cada iteração tem atividades de análise de requisitos, projeto, implementação e testes.

A cada iteração de cada fase, são gerados artefatos. Booch, Rumbaugh e Jacobson (2000) definem um artefato como um "documento, relatório ou executável, que é produzido, manipulado ou consumido" e afirmam que no PU, os modelos são o tipo mais importante de artefato. Definem, também, modelo como "uma simplificação da realidade, criado para proporcionar uma melhor compreensão do sistema que está sendo criado" e afiram, ainda, que os modelos no PU, em conjunto, "abrangem todas as decisões importantes para a visualização, especificação, construção e documentação de um sistema complexo de *software*".

Na Tabela B.1 são mostrados os principais modelos do PU:

Tabela B.1 - Artefatos para a fase de Concepção

Modelo	**Comentário**
Modelo de negócio	Estabelece uma abstração da empresa.
Modelo de domínio	Estabelece o contexto do sistema.
Modelo de caso de uso	Estabelece os requisitos funcionais do sistema.
Modelo de projeto	Estabelece o vocabulário do problema e de sua solução.
Modelo de implantação	Estabelece a topologia do *hardware* em que o sistema é executado.
Modelo de implementação	Estabelece as partes usadas para montar e liberar o sistema físico.
Modelo de teste	Estabelece os caminhos pelos quais o sistema é validado e verificado.

Fonte: adaptado de Booch, Rumbaugh e Jacobson (2000)

3.1 Concepção

Nesta fase, são estabelecidos a visão aproximada do sistema, o caso de negócio, o escopo do projeto, as estimativas de recursos para o projeto, os principais marcos de progresso e um plano para a fase. Na fase de concepção deve-se fazer uma investigação sobre os requisitos para ter uma opinião racional e justificável sobre a finalidade geral e a viabilidade do sistema e decidir se vale a pena investir nele, ou seja, passar para a fase de elaboração.

Pode-se usar alguns artefatos, mostrados na Tabela B.2, como auxílio na fase de concepção. Tais artefatos são parcialmente completados nesta fase e têm conteúdo de

investigação leve, pouco detalhado, sendo refinados em fases posteriores.

Observe que a utilização de artefatos é opcional. Devem ser utilizados somente aqueles que acrescentem valor ao projeto, auxiliando o raciocínio, análise e presteza.

Tabela B.2 - Artefatos para a fase de Concepção

Artefato	**Comentário**
Visão e caso de negócio	Descreve os objetivos e as restrições de alto nível, o caso de negócio, além de um resumo para executivos.
Modelo de casos de uso	Descreve os requisitos funcionais e requisitos não funcionais relacionados.
Especificações suplementares	Descrevem outros requisitos.
Glossário	Contém a terminologia-chave do domínio.
Lista de riscos e plano de gerenciamento de riscos	Descrevem os riscos do negócio, técnicos, de recursos e de cronograma e as idéias para sua minimização ou solução.
Protótipos e provas de conceitos	Visam a esclarecer a visão e validar as idéias técnicas.
Plano de iteração	Descreve o que fazer na primeira iteração da elaboração.
Plano da fase e plano de desenvolvimento de *software*	Estimativa de baixa precisão para a duração e esforço da fase de elaboração. Ferramentas, pessoal, treinamento e outros recursos.
Pasta de desenvolvimento	Uma descrição dos passos e artefatos do PU personalizados para este projeto. O PU é sempre personalizado para um projeto.

Fonte: Larman (2004)

Diagramas de Casos de Uso

É uma notação fornecida pela UML para ilustrar os nomes dos casos de uso, dos atores e os relacionamentos entre eles, bem como a fronteira do sistema e como ele é usado. Os diagramas de casos de uso fazem parte do Modelo de Caso de Uso. Como dito anteriormente, o PU é guiado pelos casos de uso, sendo este modelo de extrema importância. Na Figura B.2 é mostrada uma sugestão de notação para Diagramas de Casos de Uso e na Figura B.3 pode ser visto um exemplo de Diagrama de Caso de Uso.

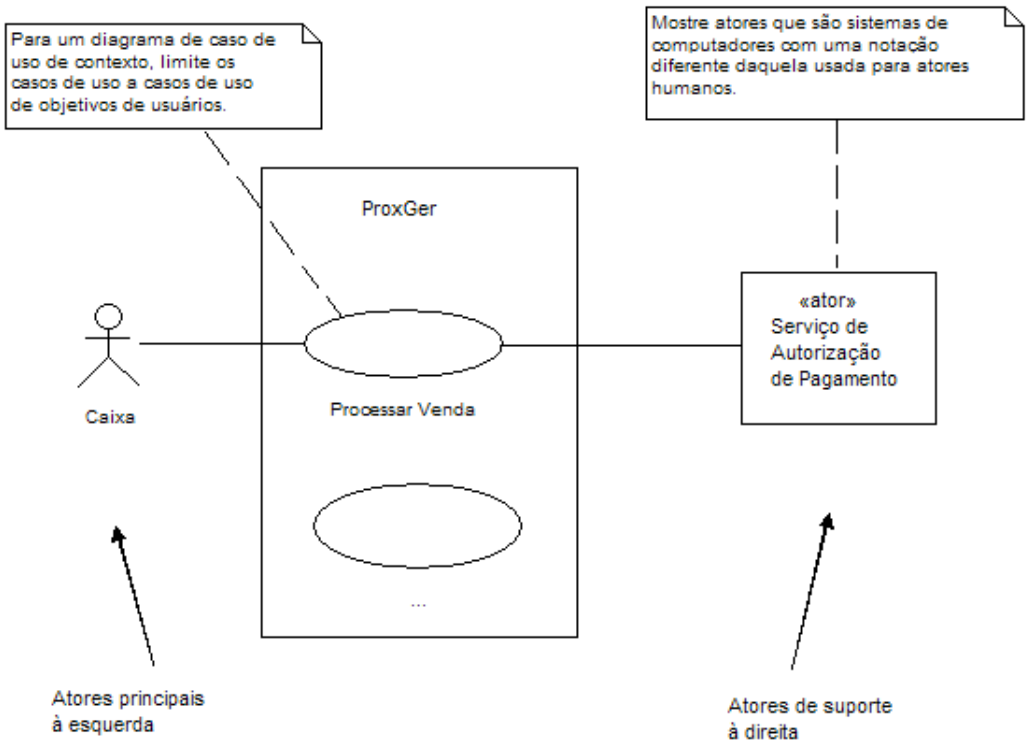

Figura B.2 – Sugestão de notação para Diagramas de Caso de Uso

Fonte: Larman (2004)

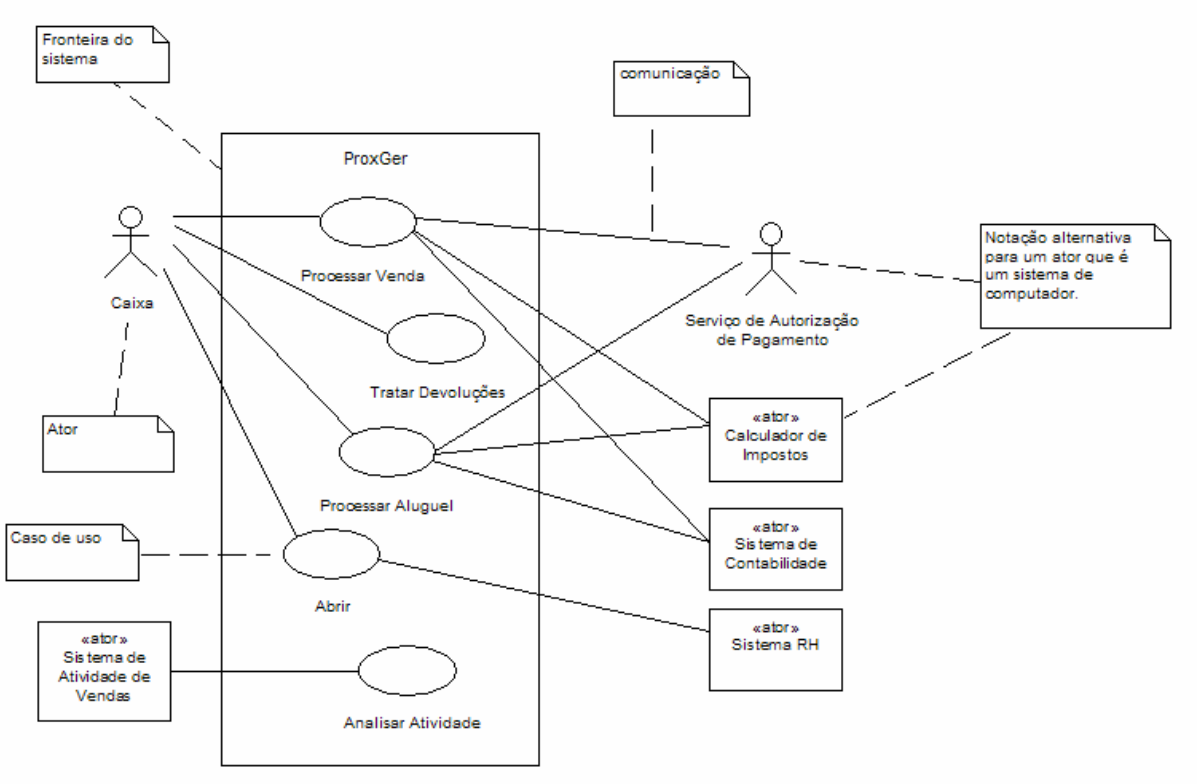

Figura B.3 – exemplo de Diagrama parcial de Caso de Uso

Fonte: adaptado de Larman (2004)

3.2 Elaboração

Na fase de Elaboração, são estabelecidas a visão refinada do projeto, a análise do domínio do problema, a implementação iterativa da arquitetura central, a resolução dos altos riscos, a identificação da maioria dos requisitos e estimativas de recursos necessários mais realistas e o desenvolvimento do plano do projeto.

É nesta fase que se enfatiza a introdução à Análise e Projeto Orientado a Objeto, aplicando a UML, os padrões e a arquitetura.

A elaboração freqüentemente tem de duas a quatro iterações, com duração entre duas a seis semanas cada iteração, com data de término fixa. Cada iteração deverá terminar no prazo, com uma versão estável e testada. Os "protótipos" criados na fase de elaboração são parte do sistema final, não são descartados em etapas futuras.

Aconselha-se começar a implementar e testar cedo, para obter a realimentação e adaptação. As interfaces dos cenários implementados também podem passar por este processo de adaptação.

Na Tabela B.3 são mostrados alguns artefatos que podem ser usados na fase de elaboração, excluídos aqueles que iniciaram na fase de concepção e deverão ser refinados nesta fase. Os artefatos constantes na Tabela B.3 deverão ser refinados ao longo de uma série de iterações da fase de elaboração.

Tabela B.3: amostras de artefatos da elaboração, excluídos aqueles iniciados na concepção.

Artefato	**Comentário**
Modelo de Domínio	Visualização dos conceitos do domínio; é similar a um modelo estático de informação das entidades do domínio.
Modelo de Projeto	Conjunto de diagramas que descreve o projeto lógico. Inclui, dentre outros, diagramas de classes de *software*, de iterações entre objetos e de pacotes e outros.
Documento de Arquitetura de *Software*	Um auxílio para o aprendizado que resume os problemas-chave de arquitetura e sua solução no projeto. É um resumo das idéias de projeto mais notáveis e do motivo da sua adoção no sistema.
Modelo de Dados	Este modelo inclui os esquemas de bancos de dados e as estratégias de mapeamento entre as representações de objetos e não objetos.
Modelo de Teste	Uma descrição do que será testado e como.

Modelo de Implementação	Implementação real – o código-fonte, o código executável, banco de dados etc.
Storyboards de Casos de Uso e Protótipo da Interface com o Usuário	Uma descrição da interface do usuário, de trajetórias de navegação, modelos de utilização etc.

Fonte: Larman (2004)

Modelo de Domínio

O **Modelo de Domínio** ilustra classes conceituais do mundo real, é o artefato mais importante a ser criado durante a Análise Orientada a Objetos e é requerido por vários artefatos subseqüentes.

Embora os Modelos de Domínio estejam relacionados a modelos conceituais entidade-relacionamento, utilizados para modelagem de dados, eles não são modelos de dados, mas são a representação visual de classes conceituais ou objetos do mundo real em um domínio de problema.

O Modelo de Domínio é um conjunto de **Diagramas de Classes**, sem operações e com associações e atributos. Em PU, um Modelo de Domínio usa notação UML de diagramas de classes e pode mostrar:

- Objetos do domínio ou classes conceituais;
- Associações entre classes conceituais;
- Atributos de classes conceituais.

Na Figura B.4 é mostrado um exemplo de modelo de domínio.

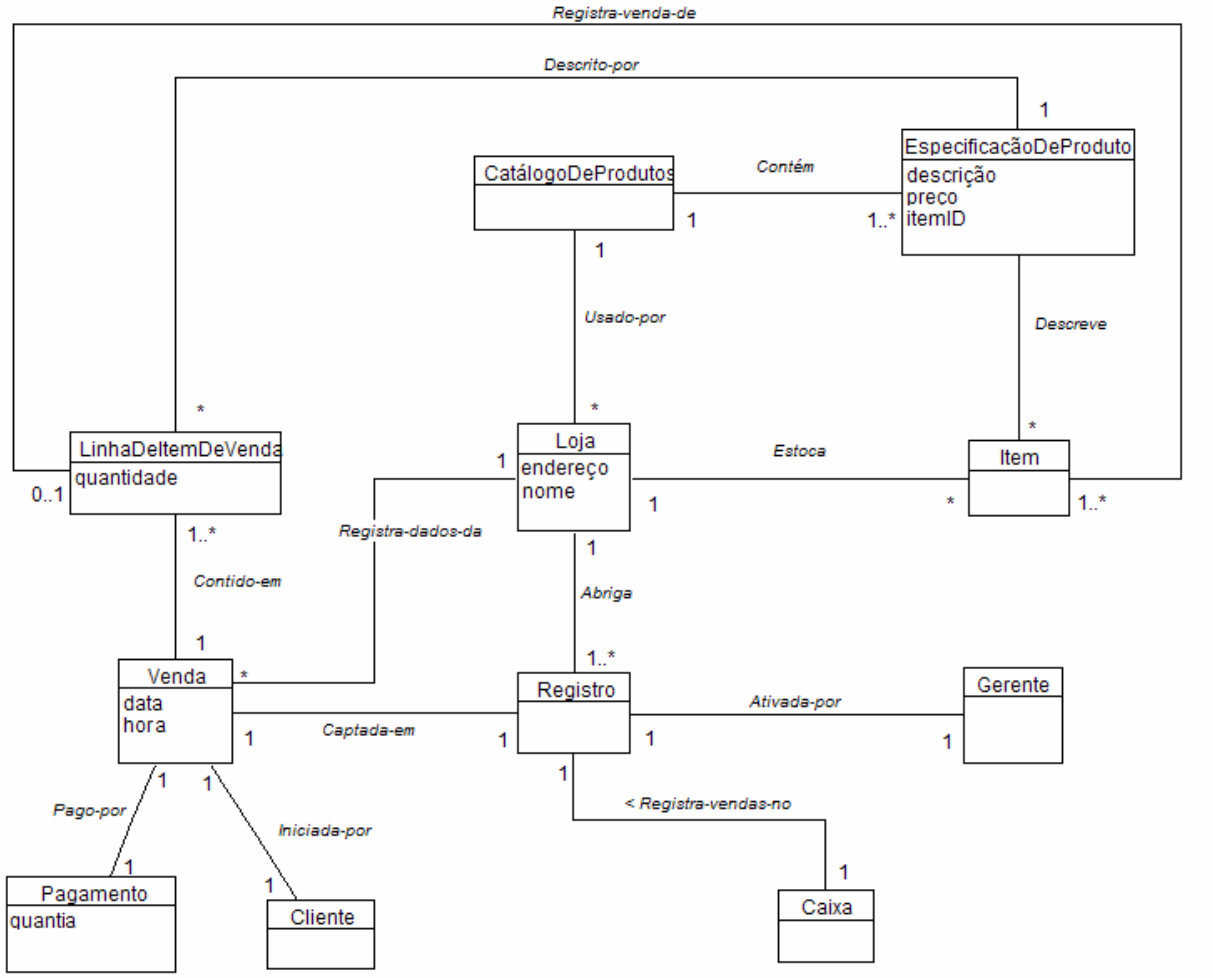

Figura B.4: um modelo de domínio parcial

Fonte: Larman (2004)

A partir do Modelo de Domínio, identificam-se as classes que fazem parte da solução de *software* e seus atributos e responsabilidades, que formarão os Diagramas de Classes de Projeto (DCP). Informações sobre tipos de atributos, parâmetros de métodos e valores de retorno de métodos podem ou não ser mostradas. Informações de navegabilidade devem ser acrescentadas aos DCPs.

Na Figura B.5 é mostrado um diagrama de classe no Modelo de Domínio e um diagrama de classe no Modelo de Projeto.

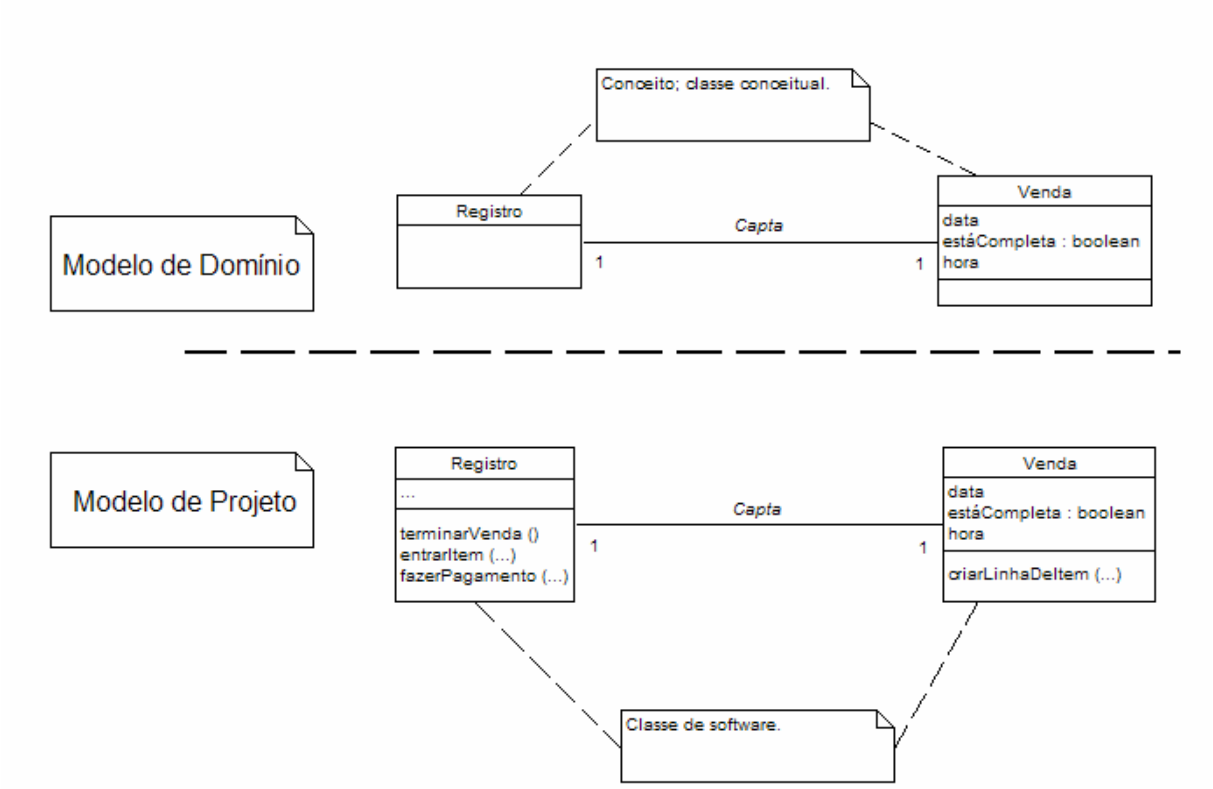

Figura B.5 – Modelo do Domínio *versus* classes de Modelo de Projeto.

Fonte: adaptado de Larman (2004)

Na Figura B.6 pode-se ver um exemplo de DCP.

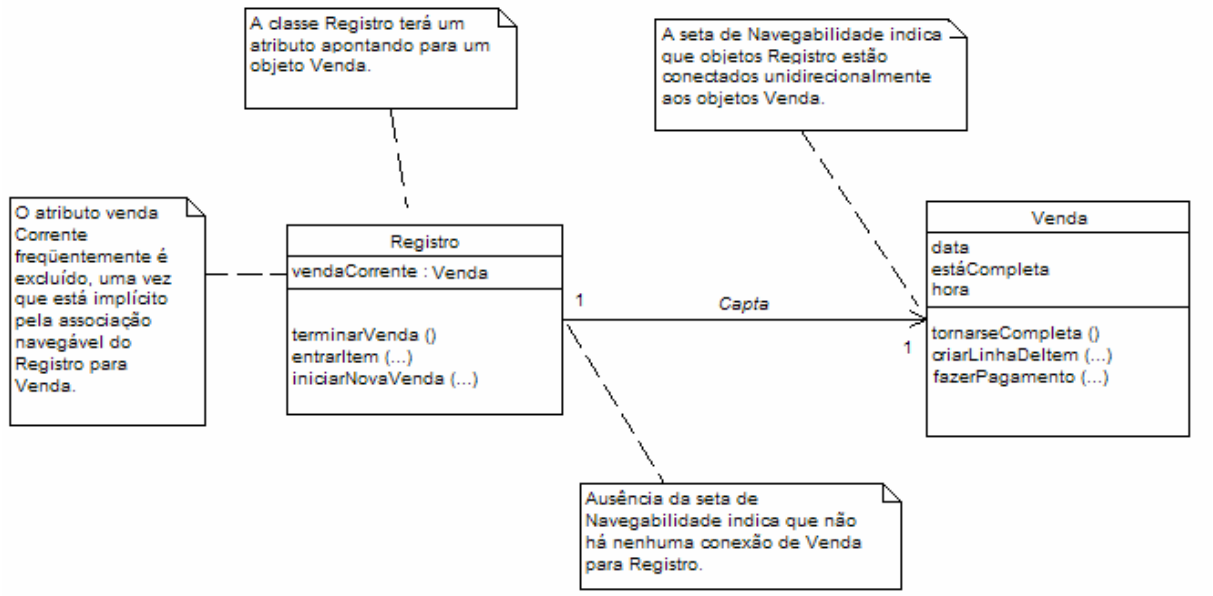

Figura B.6 – Mostrar a navegabilidade ou a visibilidade do atributo.

Fonte: adaptado de Larman (2004)

3.3 Construção

Nesta fase é realizada a implementação iterativa dos elementos restantes, resultando no sistema completo, pronto para a implantação. Envolve a descrição dos requisitos restantes e dos critérios de aceitação e os testes do *software*.

Na fase de Construção deve-se terminar a aplicação, visto que na fase da Elaboração o "esqueleto" do sistema ficou pronto, na construção deve-se terminar de fazer o que falta para o sistema ser implantado, fazer o teste alfa, preparar o sistema para a implantação, o que inclui também escrever guias do usuário, ajuda *on-line* e outros materiais de apoio necessários.

Recomenda-se uma equipe de trabalho maior nessa fase, que provavelmente trabalhará em paralelo. A fase de construção também prosseguirá via uma série de iterações de tempo fixo.

3.4 Transição

Na fase de Transição, são executadas as seguintes tarefas:

- Colocar o sistema em produção;
- Fazer teste beta e sua reação (realimentação);
- Conversão de dados;
- Treinamento;
- Extensão para o mercado;
- Operação paralela do sistema antigo e o novo.