FERNANDO SIMEONE

ALGORITMOS GENÉTICOS APLICADOS AO PROBLEMA GERAL DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DE PRODUÇÃO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Cláudio Fabiano Motta Toledo

LAVRAS
MINAS GERAIS - BRASIL
2009

FERNANDO SIMEONE

ALGORITMOS GENÉTICOS APLICADOS AO PROBLEMA GERAL DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DE PRODUÇÃO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 15 de Junho de 2009

_____
Prof. Dra. Marluce Rodrigues Pereira

_____
Prof. Dr. Wilian Soares Lacerda

_____
Prof. Dr. Cláudio Fabiano Motta Toledo
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL
2009

Dedico este trabalho à minha família, em especial minha mãe, meu pai e meu irmão.

A todos os meus amigos e companheiros de trabalho.

Aos meus companheiros de república.

Agradecimentos

Agradeço especialmente a meu orientador, Prof. Cláudio, pelo apoio, por acreditar no meu potencial e por ser um amigo durante esta jornada. A todos aqueles que contribuíram para a realização deste trabalho. Aos professores e companheiros pelos conhecimentos transmitidos.

ALGORITMOS GENÉTICOS APLICADOS AO PROBLEMA GERAL DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DE PRODUÇÃO

RESUMO

O presente trabalho propõe algoritmos genéticos (AGs) para solucionar o Problema Geral de Dimensionamento de Lotes e Programação da Produção (PGDLPP) com e sem máquinas paralelas, e com penalização para demandas não atendidas. Um modelo matemático é apresentado para este caso do PGDLPP e conjuntos de instancias são gerados, baseados em parâmetros utilizados na literatura. Essas instancias são solucionadas usando uma ferramenta de modelagem matemática cujas soluções servem para avaliação do desempenho dos algoritmos genéticos. As metaheurísticas também são comparadas a outras heurísticas e a um método exato. Os resultados revelam o melhor desempenho obtido pelos algoritmos genéticos nas instancias mais complexas envolvendo o PGLDPP com máquinas paralelas.

Palavras Chave: Programação da produção. Dimensionamento de lotes. Algoritm genético.

GENETIC ALGORITHMS APPLIED TO THE GENERAL LOT-SIZING PROBLEM

ABSTRACT

The present paper proposes algorithm genetics (AGs) to solve the General Lot sizing and Scheduling Problem (GLSP) with and without parallel machines, and with penalties for demand shortcoming. A mathematical model is presented for this GLSP case and set of instances are generated using parameters found in the literature. These instances are solved using a mathematical modeling computational package whose solutions will be benchmarks to evaluate the GAs performance. These metaheuristics are also compared with other heuristics and with a exact method.. The results report the better performance found by the AGs in complex instances for the GLSP with parallel machines.

Keyword. Scheduling. Lot sizing. Genetic Algorithm.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

1.1 Contextualização e Motivação

O presente trabalho realiza pesquisa na área de problemas de produção e heurísticas utilizadas para resolvê-los. A principal proposta do trabalho é a utilização de uma abordagem baseada em algoritmo genético (AG) para a resolução de um problema de produção, o Problema Geral de Dimensionamento de Lotes e Programação da Produção (PGDLPP). Trata-se de um problema de programação de produção, onde o objetivo é organizar a produção de itens de forma a minimizar os custos.

Os problemas de produção têm considerável importância no contexto da produção industrial. Uma produção bem planejada pode ser crucial para seu sucesso numa indústria. O PGDLPP consiste no planejamento da produção de itens em um número limitado de máquinas ao longo de um horizonte de planejamento finito. O problema apresenta uma grande complexidade, visto que o planejamento da produção está sujeito a várias restrições, como demandas a serem atendidas e restrições de capacidade. Desta forma, costuma ser inviável a aplicação de métodos exatos, por não retornarem a solução dentro de um tempo razoável. O trabalho tem como motivação a abordagem diferenciada do problema, considerando cenários com a utilização de máquinas paralelas na produção dos itens e penalização para demandas não atendidas. Este enfoque aumenta consideravelmente a complexidade do problema, o que aumenta a relevância do estudo de heurísticas que possam resolvê-lo com eficiência.

Heurísticas são métodos exploratórios que buscam soluções para problemas baseando-se em aproximações e avaliações das circunstâncias em

cada etapa desta busca. As técnicas de computação evolutiva são heurísticas que vem sendo cada vez mais utilizadas na resolução de problemas de otimização. Elas são baseadas em processos naturais, como a seleção natural e a reprodução genética. O algoritmo genético está inserido neste contexto, e tem demonstrado considerável eficiência na resolução de vários problemas. Este algoritmo simula a reprodução genética e trabalha com várias possíveis soluções para o problema, modificando e combinando as mesmas de forma a convergir para uma solução de boa qualidade.

A avaliação do algoritmo requer testes perante diversos cenários do problema. Para a obtenção destes cenários, conhecidos como instâncias do problema, o trabalho contempla o estudo e desenvolvimento de um gerador de instâncias para o PGDLPP. Este gerador é capaz de criar instâncias em vários níveis de complexidade, possibilitando uma melhor avaliação dos métodos de resolução.

O algoritmo genético é comparado com outros métodos, heurísticos e exatos. Algumas configurações diferentes do algoritmo são testadas em busca daquela com melhor desempenho. A partir do resultado de testes realizados é possível avaliar criteriosamente seu desempenho e a viabilidade de sua aplicação para o PGDLPP. Assim, o objetivo deste trabalho é, utilizando o AG na resolução do PGDLPP, alcançar resultados relevantes quando comparados àqueles alcançados por métodos exatos e por outras heurísticas relatadas na literatura.

1.2 Metodologia Científica

O tipo de pesquisa deste trabalho é classificado quanto à natureza como tecnológica, pois visa aplicar conceitos pré-existentes em busca de resultados expressivos para a resolução de um problema de produção. Quanto aos

objetivos, a pesquisa pode ser classificada como exploratória, pois consiste na busca por um método mais eficiente que aqueles já existentes para a resolução do problema em questão. Quanto aos procedimentos, este trabalho é classificado como pesquisa operacional, pois visa otimizar resultados aplicando um método proposto. A pesquisa também é classificada como de laboratório, pois variáveis que possam vir a prejudicá-la podem ser controladas. O trabalho foi executado no laboratório de Pesquisa III e do Departamento de Ciência da Computação. Os equipamentos utilizados nos testes foram computadores com processador Core 2 Duo, 2,66 GHz e 2 GB RAM. Foi utilizada a tecnologia Java na implementação dos algoritmos, executada sobre o sistema operacional Microsoft Windows XP.

1.3 Objetivos do Trabalho

O presente trabalho tem como objetivo o estudo do Problema Geral de Dimensionamento de Lotes e Programação de Produção (PGDLPP) para uma única máquina (linha de produção) e, a partir deste, a proposta de um novo modelo matemático para este problema contemplando cenários com várias máquinas. O trabalho também visa o estudo do processo de geração de instâncias para o PGDLPP e a implementação de um gerador de instâncias que execute tal processo. Este gerador deve ser capaz de criar instâncias em diversos níveis de complexidade para o modelo com várias máquinas proposto. Outro objetivo é a proposta e implementação de uma abordagem de Algoritmo Genético Hierarquicamente Estruturado capaz de resolver o PGDLPP para instâncias com uma ou mais máquinas. A partir desta implementação, o trabalho visa à execução de testes e avaliações de desempenho do AG quando comparado a outros métodos heurísticos e exatos de resolução do problema.

Outros Objetivos:

- Implementação do modelo matemático do PGDLPP para uma e várias máquinas utilizando a linguagem de modelagem AMPL;
- Avaliação de desempenho de diferentes tipos de operadores de *crossover* para o AG ;
- Avaliação de desempenho de diferentes estruturas populacionais para o AG.

1.4 Estrutura do Trabalho

O capítulo 2 apresenta um embasamento teórico necessário para a compreensão do trabalho. O capítulo 3 apresenta o problema abordado neste trabalho. O capítulo 4 apresenta o método proposto para a resolução do problema. O capítulo 5 apresenta os resultados obtidos e uma análise sobre eles. O capítulo 6 apresenta as conclusões do trabalho.

Capítulo 2

Referencial Teórico

Este capítulo apresenta o embasamento teórico necessário para o completo entendimento deste trabalho. Primeiramente são apresentados conceitos relacionados com problemas de produção, e em seguida é apresentado o algoritmo genético.

2.1 Problemas de Produção

A produção industrial envolve fatores que podem ser cruciais para o sucesso de indústrias e empresas. As demandas do mercado variam constantemente, dificultando as tomadas de decisões, de forma que a empresa precisa se adaptar a estas condições. Para gerenciar todos os fatores envolvidos, como demandas, recursos, custos e prazos, é necessário um planejamento capaz de coordenar o processo produtivo. Este planejamento é conhecido como Planejamento e Controle da Produção (PCP).

Segundo Zacarelli (1979), o PCP é um conjunto de funções inter-relacionadas com o objetivo de coordenar o processo produtivo. Segundo Berreta (1997), o PCP é responsável pela a coordenação de todas as atividades referentes a um processo produtivo, desde a aquisição de matéria prima até a entrega de produtos acabados. Para Anthony (1965), o PCP pode ser dividido em três níveis hierárquicos: estratégico, tático e operacional. O nível estratégico trata de decisões de longo prazo, como a escolha e projeto do processo de produção, determinação das capacidades em função das demandas e determinação da quantidade de produção unitária. O nível tático abrange decisões de médio prazo, como planejamento agregado (ex.: níveis de mão de

obra e subcontratação) e planejamento das quantidades de produção. Para Berreta (1997), a principal função do nível tático é o planejamento da produção, alocando efetivamente os recursos para satisfazer as demandas, levando em consideração os custos envolvidos. O nível operacional aborda decisões de curto prazo com o objetivo de cumprir os planos definidos anteriormente, como atividades diárias, designação de tarefas e programação de tarefas em cada maquina. De acordo com Arenales (2007), um bom plano de produção é aquele que satisfaz as demandas sem atraso, respeita a capacidade dos recursos disponíveis e minimiza os custos de produção.

Os problemas de produção estão relacionados com o planejamento da produção, envolvendo decisões como escalonamento de tarefas, alocação de recursos, minimização de custos, levando em consideração as demandas e prazos a serem atendidos. Assim este trabalho visa os problemas relacionados à tomada de decisão no nível tático em um sistema de produção.

Uma revisão envolvendo métodos heurísticos na resolução de problemas de produção pode ser encontrada em Jans (2007).

2.2 Problemas de Dimensionamento de Lotes

Um problema de dimensionamento de lotes consiste em um ou vários itens (produtos), que devem ser produzidos em uma ou mais máquinas. Demandas devem ser atendidas para cada produto em um determinado prazo. Há custos para configurar uma máquina para um determinado produto (*setup*), custos de estoque para cada produto, restrições de capacidade, entre outros obstáculos que podem aparecer de acordo com o modelo do problema. O objetivo é organizar a produção de forma a minimizar os custos de produção (estoque, *setup*) respeitando as restrições existentes no problema.

2.2.1 Classificação e Características

Existe uma grande variedade de problemas de dimensionamento de lotes e uma classificação para este conjunto de problemas facilita o estudo. Segundo Karimi (2003), citado por Nascimento (2007), as características a seguir podem ser relevantes na classificação de problemas de dimensionamento de lotes.

- **Horizonte de planejamento –** É o tempo para o qual a produção será planejada. Pode ser finito ou infinito. Também pode ser dividido em períodos;
- **Número de Níveis ou Estágios –** Um produto a ser produzido pode depender de outros produtos, como se um fosse componente de outro. O problema pode ser monoestágio, quando nenhum produto depende de outro para ser produzido, ou de multiestágio, quando existe a dependência entre os produtos;
- **Número de produtos a serem produzidos –** O problema pode possuir apenas um ou múltiplos produtos a serem produzidos;
- **Capacidade de recursos –** O problema pode ser capacitado ou não capacitado, isto é, pode possuir ou não restrições de capacidade que limitam a produção;
- **Demanda –** A demanda dos produtos pode ser estática ao longo do tempo ou dinâmica, quando é alterada ao longo do tempo. Na maioria das abordagens, onde o horizonte de planejamento é segmentado em períodos, as demandas devem ser atendidas em cada período. Neste caso a demanda estática é aquela que é igual para todos os períodos do horizonte de planejamento;
- **Custos de configuração (*setup*) –** Os custos de *setup* podem ser dependentes dos itens produzidos anteriormente ou independentes destes, ou seja, podem ser dependentes ou não da seqüência de

produção. Nas abordagens mais próximas de sistemas de produção reais, os custos de *setup* são dependentes da seqüência de produção. Assim a troca de configuração entre dois produtos similares pode ser considerada mais dispendiosa que a troca entre produtos mais distintos;

- **Número de máquinas (linhas de produção) –** O número de máquinas disponíveis para a produção e atendimento das demandas pode ser maior que um. Com um maior número de máquinas o problema se torna mais complexo, visto que existem maiores possibilidades de organização da produção;
- **Número de micro-períodos –** Os períodos do horizonte de planejamento podem estar sujeitos a restrições relacionadas ao número de configurações que a máquina pode assumir nele. Este número máximo de configurações em um período são os micro-períodos. Eles não estão relacionados com a capacidade do período, apenas limitam o número de trocas de configuração nele.

2.2.2 Variações do problema

Nesta seção são apresentadas algumas variações de problemas de dimensionamento de lote. Algumas características comuns dos problemas apresentados a seguir são: o horizonte de planejamento é dividido em períodos e as demandas podem variar de um período para o outro. Estas variações do problema são baseadas naquelas apresentadas por Toledo (1998).

- **Um produto sem restrição de capacidade**

A solução para este problema visa minimizar os custos de configuração da máquina e de estoque do produto para cada período. A cada período existe

uma demanda do produto que deve ser atendida sem atraso, isto é, a soma entre quantidade produzida e a quantidade já existente em estoque do item deve ser maior ou igual à demanda em cada período. Uma quantidade produzida maior que a demanda acarreta na acumulação de estoque, resultando em custos adicionais. As decisões referentes ao planejamento da produção a serem tomadas para este problema são as quantidades do item produzidas a cada período.

Pesquisadores vêm desenvolvendo algoritmos para a resolução deste problema, pois, apesar de simples, ele é relevante já que se encaixa como um subproblema de problemas mais complexos. Alguns métodos ótimos foram apresentados, como o de Warner e Whitin (1958), que é baseado na busca de caminhos mínimos em grafos direcionados.

- **Um produto com restrição de capacidade**

O problema capacitado com um único produto a ser produzido é similar ao não capacitado, tendo como adicional apenas a restrição de capacidade, limitando a quantidade produzida a cada período. Restrições de capacidade contribuem para a produção antecipada e estocagem, visto que a capacidade de um período pode não comportar a produção de toda a demanda do produto neste período.

- **Múltiplos itens com restrições de capacidade**

Este problema se caracteriza por possuir uma única máquina, na qual devem ser produzidos vários produtos diferentes. O objetivo é minimizar a soma dos custos de configuração (*setup*) e estoque. Cada produto possui uma demanda particular em cada período, e os custos de *setup* da máquina também são específicos dos produtos. O custo de configuração depende do produto que está

atualmente na máquina e daquele para o qual a maquina será configurada, de modo que, quanto maior a similaridade entre estes dois produtos, menor será o custo de configuração. A restrição de capacidade consiste em um tempo máximo que a máquina pode produzir por período, de forma que o total de tempo de produção de um período deve ser menor ou igual a este tempo máximo. O tempo gasto para produzir uma unidade de um produto é particular deste produto.

Algumas abordagens consideram, além do custo de configuração, o tempo de configuração. Isto acaba restringindo mais ainda a produção, pois a soma dos tempos gastos com produção e os tempos gastos com configuração deve ser menor que o tempo máximo. Este problema apresenta-se um pouco mais próximo dos problemas encontrados no planejamento de linhas de produção reais, sendo também mais complexo que os apresentados anteriormente. Resultados a respeito de sua complexidade podem ser verificados em Bitran e Yanasse (1982).

- **Máquinas paralelas e múltiplos produtos**

Este problema não é muito citado na literatura. Ele é semelhante àquele que possui múltiplos itens e apenas uma máquina, porém apresenta mais de uma máquina, e a produção dos itens deverá ser distribuída entre essas máquinas. Esta abordagem apresenta-se bastante interessante pois as máquinas envolvidas não são necessariamente iguais. As máquinas podem possuir capacidades, custos de configuração, tempos de produção, entre outras características, diferentes entre si. Desta forma, o problema se torna ainda mais complexo, visto que o número de possibilidades de planejamento aumenta consideravelmente com um maior número de máquinas. Outra característica importante desta variação do problema é que as máquinas trabalham simultaneamente, em um mesmo horizonte de planejamento e as demandas a serem atendidas são únicas. Ou seja,

o somatório das produções efetuadas nas máquinas será utilizado para atender esta demanda única.

- **Múltiplos níveis ou estágios**

A principal característica deste problema é a interdependência entre os produtos. Para produzir um item pode ser necessária a produção de certa quantidade de outros itens. Esta característica torna mais complexa a programação da produção, havendo mais restrições para a ordem em que os itens serão produzidos.

2.3 O Algoritmo Genético

O algoritmo genético está inserido na área de Computação Evolutiva. Esta área visa o desenvolvimento de metodologias para resolução de problemas inspiradas nos processos biológicos de seleção natural. Estas metodologias vêm sendo cada vez mais utilizadas em problemas de otimização por demonstrarem grande eficiência. Esta seção apresenta os conceitos de biologia necessários para o entendimento do funcionamento do algoritmo genético. Em seguida o AG é apresentado e seus aspectos mais relevantes são descritos.

2.3.1 Conceitos de Biologia

A hereditariedade consiste no conjunto de processos naturais pelo qual ocorre a transmissão de características de um indivíduo para seus descendentes através da herança genética. De acordo com Von Zuben (2007), apesar das primeiras idéias sobre hereditariedade surgirem a 6000 anos atrás, apenas por volta de 500 a.C. filósofos gregos propuseram idéias do que realmente viria a ser

a hereditariedade. Aristóteles (filósofo grego, 384 a.C. - 322 a.C) acreditava que no sêmen masculino existia algum tipo de substância responsável pela herança de características entre os indivíduos e seus descendentes. Acreditava-se que a mulher teria apenas o papel de incubadeira na reprodução. Mesmo com a descoberta do óvulo 1672, pelo holandês Graaf, e do espermatozóide em 1675, pelo holandês Von Leeuwenhoeck, apenas em 1866 surgiram as idéias fundamentadas sobre a hereditariedade, com Gregor Mendel. Mendel fez experimentos com o cruzamento de pés de ervilha de diferentes linhagens, e a partir deste conjunto de experimentos escreveu as leis de Mendel com suas conclusões a respeito da hereditariedade. Apenas 30 anos depois, a comunidade científica veio a reconhecer a real importância destes experimentos e seus resultados.

A idéia da seleção natural, proposta por Darwin (1859), consiste em um processo de seleção de indivíduos de uma determinada espécie, onde aqueles que possuem as características mais favoráveis e uma maior capacidade de adaptação possuem maiores chances de sobrevivência. O processo de seleção natural faz com que as características favoráveis a sobrevivência tendam a aparecer com maior freqüência nas gerações posteriores do que as características não favoráveis. Para Mayr (1988), esta predominância de características mais favoráveis em gerações futuras é considerada como uma evolução da espécie, pois ao longo das gerações a espécie como um todo será beneficiada por tal processo de seleção.

O fenótipo e o genótipo são dois conceitos que estão relacionados entre si. Segundo Ramalho (2004), o fenótipo é um conjunto de formas alternativas de expressão de uma característica e essa expressão depende do genótipo e do ambiente. O genótipo é a constituição genética de um indivíduo. Em outras palavras, o genótipo contém a codificação de características (cor dos olhos, por

exemplo) que, juntamente com a influência do ambiente resultarão na expressão desta característica (fenótipo). De acordo com Von Zuben (2007), um elemento do genótipo pode afetar diversas características do fenótipo, e uma característica do fenótipo pode ser determinada por vários elementos do genótipo.

São apresentados abaixo outros conceitos biológicos relevantes:

- **Gene:** segmento do cromossomo, unidade fundamental da hereditariedade;
- **Cromossomo:** seqüência de DNA que contém vários genes;
- ***Crossover*:** recombinação realizada entre dois cromossomos através da troca de material genético;
- **Mutação:** processo responsável pela mudança na seqüência dos genes.

2.3.2 O Algoritmo

Os algoritmos genéticos, introduzidos por Holland (1975), são metodologias baseadas no processo de seleção natural e evolução biológica. Segundo Von Zuben (2007), esta maneira de evolução realizada no algoritmo genético trata-se de um processo de busca no espaço de soluções do problema. Este algoritmo é comumente aplicado na resolução de problemas de busca e de otimização.

Basicamente o algoritmo cria uma população inicial de indivíduos, que são possíveis soluções para o problema. O método trabalha sobre esses indivíduos de forma que, a cada iteração (geração), os indivíduos considerados mais aptos prevalecerão, e os demais serão modificados. Este processo é repetido até que a solução ótima seja atingida, ou até que a convergência do algoritmo se estabilize.

De acordo com Mendes e Gonçalves (2003), o algoritmo genético pode ser representado pelo pseudocódigo apresentado na figura 2.1.

```
Início
        Gerar a população inicial St
        Avaliar a população St

        Enquanto o critério de parada não for satisfeito, repetir
        Inicio
                Selecionar os elementos de St e colocar em St+1
                Cruzar os elementos de St e colocar em St+1
                Aplicar a mutação de novos elementos e colocar em St+1
                Avaliar a nova população St+1
                St = St+1
        Fim

Fim
```

Figura 2.1 – Pseudocódigo do AG segundo Mendes e Gonçalves (2003).

Para Davis e Steenstrup (1987), são necessários cinco itens para construir um algoritmo genético capaz de resolver um problema:

- Uma forma de representar uma possível solução para o problema como um cromossomo.
- Uma forma de criar uma população inicial (conjunto de possíveis soluções).
- Uma função para determinar a aptidão (*fitness*) dos indivíduos (possíveis soluções).
- Um conjunto de operadores genéticos, capaz de alterar os indivíduos de uma população.
- Um conjunto de parâmetros como tamanho da população, probabilidade de aplicar cada um dos operadores genéticos, entre outros.

Uma visão mais detalhada sobre algoritmos genéticos pode ser encontrada em Goldberg (1989), Davis (1990), Michalewicz (1996) e Bäck *et al.* (2000).

Os indivíduos ou cromossomos representam as possíveis soluções para o problema. Segundo Toledo (2005), os diferentes atributos que caracterizam cada indivíduo são chamados genes e as informações que descrevem uma solução do problema estão codificadas nos genes de cada indivíduo. O indivíduo é normalmente codificado na forma de vetor ou listas. Sua representação numérica pode utilizar o padrão binário, real ou inteiro. A codificação deve ser o mais simples possível, facilitando o tratamento e a aplicação dos operadores genéticos.

A representação binária, segundo Holland (1992), possui vantagens de desempenho em várias abordagens. Porém, a codificação binária pode ter um desempenho insatisfatório em relação às representações inteira e real em problemas de otimização com parâmetros reais. De acordo com Michalewicz (1996), o desempenho de um algoritmo genético utilizando codificação binária é baixo quando o espaço de busca apresenta dimensão elevada. Além disso, segundo Fogel (1994), o aumento do paralelismo implícito na representação binária nem sempre significa que o desempenho será ótimo.

A população inicial normalmente é construída pela geração aleatória dos indivíduos. Caso exista algum conhecimento que possa contribuir na convergência do algoritmo este pode ser aplicado na geração da população inicial (Von Zuben, 2007). Uma motivação para a geração aleatória é a possibilidade de executar várias vezes o mesmo algoritmo e obter resultados distintos, podendo estes ser comparados para a avaliação da implementação.

A função de avaliação é responsável por avaliar o indivíduo, retornando um valor capaz de ser comparado com a avaliação de outros indivíduos. A

função fitness pode representar o custo de uma possível solução, ou quão boa é aquela solução. Ela está altamente relacionada com a forma de codificação dos indivíduos, visto que deve ser capaz de avaliar as restrições e a vantagens implícitas nesta codificação.

Os operadores genéticos são responsáveis por alterar a população, com o objetivo de obter soluções diferentes das atuais. Isso faz com que o algoritmo atinja diferentes regiões no espaço de busca. Os operadores mais comuns são *Crossover* (recombinação) e Mutação.

O *crossover* atua sobre dois indivíduos combinando suas cadeias numéricas, gerando novos indivíduos. Normalmente, utilizando a representação binária, são escolhidos um ou mais pontos de corte, na maioria das vezes de forma aleatória, e esses pontos delimitam os trechos de bits que serão trocados entre os indivíduos pais, e os cromossomos resultantes desta troca serão os indivíduos filhos repassados à próxima geração. Outra abordagem para o *crossover* é chamada *crossover* uniforme que, segundo Linden (2006), é capaz de combinar qualquer esquema de representação do indivíduo. Para cada posição do cromossomo filho, é sorteado de qual dos dois cromossomos pai será herdada a posição em questão.

O operador de mutação visa modificar os indivíduos da população contribuindo na variabilidade da mesma. Considerando a notação binária, isto é feito modificando aleatoriamente o valor de um ou mais bits de um indivíduo. A freqüência em que ocorrem as mutações é denominada taxa de mutação. Normalmente é atribuído a esta taxa um valor pequeno, de modo que contribua na diversidade dos indivíduos, mas não atrapalhe a convergência do algoritmo (Von Zuben, 2007).

Capítulo 3

Problema Geral de Dimensionamento de Lotes e Programação de Produção

3.1 Descrição

O problema atacado neste trabalho é um problema de produção, conhecido como Problema Geral de Dimensionamento de Lotes e Programação da Produção (PGDLPP). Este pode ser descrito da seguinte forma: um determinado número de produtos deve ser produzido por uma máquina em um horizonte de planejamento finito. Este horizonte de planejamento é segmentado em períodos, e em cada um destes existem demandas de cada produto, que devem ser completamente atendidas pela produção, não sendo permitido atraso. A quantidade total produzida em cada período é limitada pela capacidade da máquina, e esta capacidade pode diferir entre os períodos. As quantidades produzidas podem exceder às demandas, possibilitando a geração de estoques para um período seguinte. Este estoque resulta em um custo que é dependente da quantidade estocada e do custo unitário de estoque do produto. Existem custos de troca (*setup*) entre os produtos e este custo é estático, sendo o mesmo para todos os períodos. Cada período é segmentado em um ou mais micro-períodos, que representam o número de configurações que podem ser atribuídas à máquina em cada período. Isto significa que em um período podem ser produzidos diferentes produtos, porém o número de trocas entre produtos é limitado. O objetivo é organizar a produção de forma a minimizar a soma dos custos de

estoque e de trocas de configuração (*setup*), atendendo às demandas, respeitando as restrições de capacidades e limites de trocas de configuração.

3.2. Modelo Matemático

A modelagem do problema é feita considerando um horizonte de planejamento segmentado em períodos $t = 1,..., T$, e produtos $j = 1,..., J$ que devem ser produzidos ao longo destes períodos. O tamanho de um período t é determinado pela capacidade de produção da máquina, representada por K_t. Um período é dividido em um ou mais micro-períodos de tamanho variável. O conjunto de micro-períodos ao longo do período t é representado por S_t, e o tamanho de cada micro-período é determinado pelo tempo gasto na produção efetuada nele. Os micro-períodos de todos os períodos são ordenados de forma unificada $s = 1,..., S$. Micro-períodos consecutivos onde a máquina é alocada para o mesmo produto formam um mesmo lote de produção deste produto. Todo lote de um produto j deve ser maior ou igual ao tamanho mínimo do lote deste produto (m_j), isto é, em toda produção de um lote do produto j se deve produzir no mínimo m_j unidades.

A figura 3.1 ilustra um exemplo de estrutura dos períodos de um horizonte de planejamento, onde T = {p1, p2, p3} são os períodos, S = {s1, s2, s3, s4, s5, s6, s7, s8, s9} são os micro-períodos e J = {j1, j2, j3, j4} são os produtos a serem produzidos. Pode-se observar no exemplo que o número de micro-períodos pode diferir entre os períodos, assim como os tamanhos dos micro-períodos e dos períodos também podem ser diferentes. Os micro-períodos s3 e s4 são um exemplo de dois micro-períodos que formam apenas um lote de produção. O mesmo pode ser observado com os micro-períodos s6 e s7.

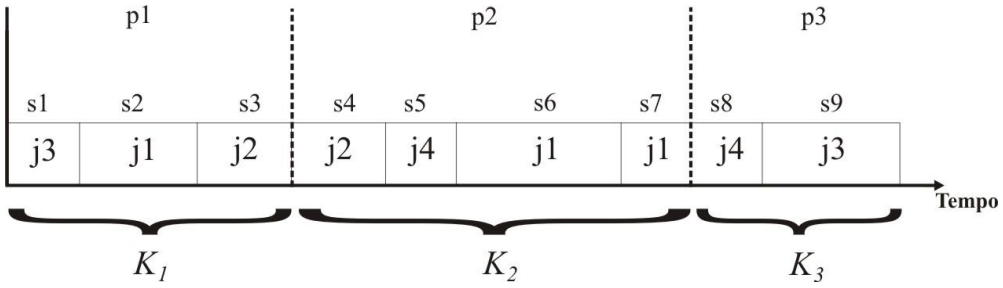

Figura 3.1 – Estrutura dos períodos do horizonte de planejamento.

O modelo aqui utilizado também foi apresentado por Fleischmann (1997). Seguem abaixo os dados passados como parâmetro e as variáveis do problema.

Dados:

J número de produtos a serem produzidos.

T número de períodos do horizonte de planejamento.

S_t conjunto de micro-períodos pertencentes ao período t.

K_t capacidade de produção (tempo) disponível no período t.

a_j tempo gasto para produzir uma unidade do produto j.

m_j tamanho mínimo do lote do produto j (unidades).

h_j custo de estoque por unidade do produto j.

s_{ij} custo de troca (*setup*) do produto i para o produto j.

d_{jt} demanda do produto j no período t (unidades).

I_{j0} estoque inicial do produto j no início do horizonte de planejamento (unidades).

y_{j0} configuração inicial da máquina: igual a 1, se no início do horizonte de planejamento a máquina está configurada para o produto *j*, caso contrário, é igual a 0.

Variáveis:

$I_{jt} \geq 0$ estoque do produto *j* no final do período *t* (unidades).

$x_{js} \geq 0$ quantidade do item *j* produzido no micro-período *s* (unidades).

$y_{js} \in \{0, 1\}$ configuração da máquina: igual a 1 se a máquina está configurada para o produto *j* no micro-período *s*, caso contrário é igual a 0.

$z_{ijs} \geq 0$ trocas de configuração: igual a 1 se houve uma troca de configuração da máquina do produto *i* para o produto *j* no início do micro-período *s*.

Seguem abaixo as equações do problema:

$$\text{minimize} \sum_{j,t} h_j I_{jt} + \sum_{i,j,s} s_{ij} z_{ijs} \tag{3.1}$$

Sujeito a

$$I_{jt} = I_{j,t-1} + \sum_{s \in S_t} x_{js} - d_{jt} \qquad \forall t, j \tag{3.2}$$

$$\sum_{j, s \in S_t} a_j x_{js} \leq K_t \qquad \forall t \tag{3.3}$$

$$x_{js} \leq \frac{K_t}{a_j} y_{js} \qquad \forall t, s, j \tag{3.4}$$

$$x_{js} \geq m_j (y_{js} - y_{j,s-1}) \qquad \forall s, j \tag{3.5}$$

$$\sum_j y_{js} = 1 \qquad \forall s \tag{3.6}$$

$$z_{ijs} \geq y_{i,s-1} + y_{js} - 1 \qquad \forall s, i, j \tag{3.7}$$

A equação (3.1) é a função objetivo do modelo. Ela representa a minimização da soma dos custos de estoque e de troca ao longo de todo o horizonte de planejamento. A equação (3.2) é a restrição de balanço de estoque. Ela faz com que o estoque do produto *j* ao final do período *t* (I_{jt}) seja a soma entre o estoque de *j* remanescente do período anterior *t – 1* e toda a quantidade de *j* produzida ao longo do período *t*, menos a demanda deste produto no período. A restrição de capacidade (3.3) faz com que a soma dos tempos gastos nas produções do período não excedam sua capacidade. A restrição (3.4) mostra que a quantidade produzida de um produto em um micro-período só pode ser maior que zero, se a máquina estiver alocada para este produto neste micro-período. Esta restrição também não permite que a produção em um micro-período exceda a capacidade total do período. A restrição (3.5) faz com que a produção de um lote, caso não seja zero, seja maior ou igual ao tamanho mínimo do lote, fazendo com que este valor mínimo já seja produzido no primeiro micro-período do lote. A restrição (3.6) permite que a máquina seja alocada para apenas um produto por micro-período. A restrição (3.7) estabelece a relação entre a variável de troca de setup *z* e de configuração da maquina *y*, para que *z* possa ser utilizada corretamente na função objetivo. De acordo com as restrições (3.4) e (3.5) lotes vazios não são proibidos.

3.3 Modelo Matemático com Várias Máquinas

Um dos objetivos deste trabalho é propor uma extensão do modelo do Problema Capacitado de Dimensionamento de Lotes para suportar várias

máquinas (ou linhas de produção). Situações com máquinas operando em paralelo tornam o problema mais complexo, aumentando seu espaço de busca já que um maior número de combinações de seqüência de produção pode ser criado.

Apesar das modificações, este modelo também suporta o problema com uma única máquina, exatamente da mesma forma que o modelo anterior. Assim, este é um modelo mais abrangente, visto que o conjunto de situações (instâncias) contempladas pelo modelo anterior é um subconjunto das contempladas por este.

A figura 3.2 ilustra um exemplo de estrutura dos períodos em um horizonte de planejamento para o modelo de várias maquinas, onde L = {l1, l2} são as máquinas, T = {p1, p2, p3} são os períodos, S_1 = {s11, s12, s13, s14, s15, s16, s17} são os micro-períodos da máquina 1 e S_2 = {s21, s22, s23, s24, s25} são os micro-períodos da máquina 2.

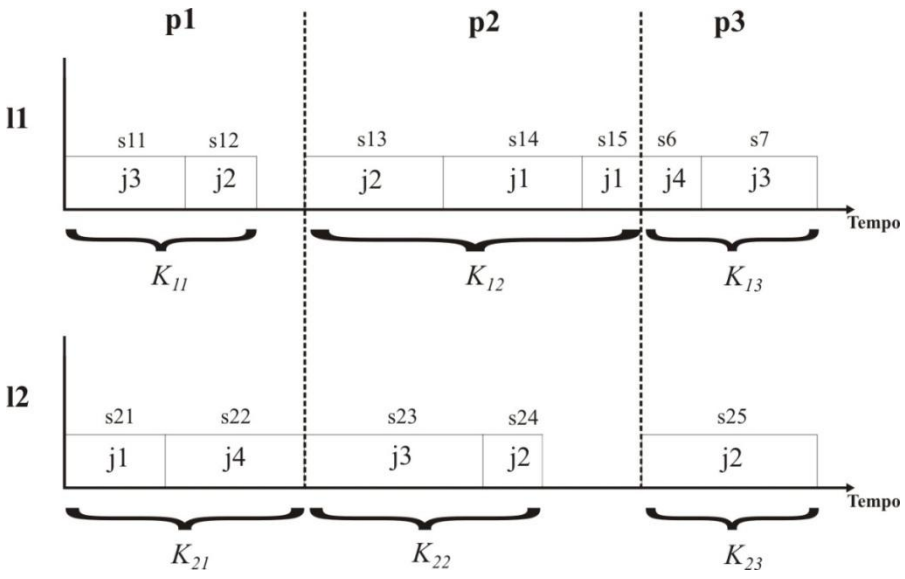

Figura 3.2 – Estrutura dos períodos do horizonte de planejamento para várias máquinas.

Pode-se observar no exemplo que a capacidade e o número de micro-períodos podem ser diferentes para cada máquina. Além disso, apesar das capacidades diferirem, os períodos são os mesmos para todas as máquinas. Uma máquina deve "esperar" até que todas as outras terminem suas produções em um período para iniciar a produção no próximo. Ao término de um período, é feito o cálculo do estoque que envolve o somatório das produções de todas as máquinas.

Seguem abaixo os dados passados como parâmetros e as variáveis do problema, seguidos de uma explicação das alterações realizadas.

Dados:

L número de máquinas.

J número de produtos a serem produzidos.

T número de períodos do horizonte de planejamento.

S_{lt} conjunto de micro-períodos pertencentes ao período t na máquina l.

K_{lt} capacidade de produção (tempo) disponível da máquina l no período t.

a_{lj} tempo gasto para produzir uma unidade do produto j na máquina l.

m_{lj} tamanho mínimo do lote do produto j (unidades) na máquina l.

h_j custo de estoque por unidade do produto j.

s_{lij} custo de troca (*setup*) do produto i para o produto j na máquina l.

d_{jt} demanda do produto j no período t (unidades).

I_{j0} estoque inicial do produto j no início do horizonte de planejamento (unidades).

y_{lj0} configuração inicial da máquina l: igual a 1, se no início do horizonte de planejamento a máquina l está configurada para o produto j, caso contrário, é igual a 0.

M penalização por unidade de demanda não atendida.

Variáveis:

$I_{jt} \geq 0$ estoque do produto j no final do período t (unidades).

$X_{ljs} \geq 0$ quantidade do item j produzido no micro-período s na máquina l (unidades).

$Y_{lsj} \in \{0, 1\}$ configuração da máquina l: igual a 1 se a máquina l está configurada para o produto j no micro-período s, caso contrário é igual a 0.

$z_{lijs} \geq 0$ trocas de configuração: igual a 1 se houve uma troca de configuração da máquina l do produto i para o produto j no início do micro-período s.

$q_j \geq 0$ quantidade da demanda do produto j que não foi atendida.

Um novo parâmetro foi adicionado para representar o numero de máquinas. Assim as máquinas podem ser representadas por l = 1,..., L. Vários dados passaram a ser indexado por máquina, pois podem diferir de uma máquina para outra. São eles: capacidade K_{lt}, tempo unitário de produção a_{lj}, tamanho mínimo de lote m_{lj}, custo de troca s_{lij}, configuração inicial da maquina y_{lj0} e conjunto de micro-períodos S_{lt}. A respeito deste último, ele é indexado por máquina pois elas podem possuir um número diferente de micro-períodos, e seus micro-períodos são contados e ordenados separadamente (por exemplo,

S_1 = {1, 2, 3, 4} e S_2 = {1, 2, 3}). As variáveis *x* (quantidades produzidas), *y* (produtos alocados nos micro-períodos) e *z* (trocas de configuração) passam a ser indexadas por máquina pois elas se referem à organização da produção (seqüência e quantidades), que é independente para cada máquina. Dados e variáveis relacionados a demandas e estoque não foram indexados por máquina. As demandas são únicas, pois o somatório das produções de todas as máquinas será utilizado para satisfazê-las, isto é, para cada produto e cada período, existe apenas uma demanda a ser atendida por todas as máquinas em conjunto. O estoque não é indexado por máquina porque, para cada produto e em cada período, ele é único e seu cálculo envolve o estoque já existente, a demanda, e a produção total, onde esta produção total é o somatório das produções de todas as máquinas.

Este novo modelo possibilita o não atendimento de demandas, que tem como conseqüência uma penalização no valor da função objetivo. Para isto foi adicionado um novo parâmetro *M*, que representa a penalização sobre unidade de demanda não atendida (independente do produto), e a variável q_j, utilizada para armazenar as quantidades de demandas não atendidas.

Segue o modelo matemático do Problema Capacitado de Dimensionamento de Lotes para várias máquinas:

$$\text{minimize} \sum_{j,t} h_j I_{jt} + \sum_{l,i,j,s} s_{lij} z_{lijs} + M \sum_{j} q_j \qquad (3.8)$$

Sujeito a

$$I_{j1} = I_{j0} + q_j + \sum_{l,s \in S_1} x_{ljs} - d_{j1} \qquad \forall t, j \qquad (3.9a)$$

$$I_{jt} = I_{j,t-1} + \sum_{l,s \in S_{lt}} x_{js} - d_{jt} \qquad \forall t, j \tag{3.9b}$$

$$\sum_{j,s \in S_{lt}} a_{lj} x_{ljs} \leq K_{lt} \qquad \forall l, t \tag{3.10}$$

$$x_{ljs} \leq \frac{K_{lt}}{a_{lj}} y_{ljs} \qquad \forall l, t, s, j \tag{3.11}$$

$$x_{ljs} \geq m_{lj}(y_{ljs} - y_{lj,s-1}) \qquad \forall l, s, j \tag{3.12}$$

$$\sum_{j} y_{ljs} = 1 \qquad \forall l, s \tag{3.13}$$

$$z_{lijs} \geq y_{li,s-1} + y_{ljs} - 1 \qquad \forall l, s, i, j \tag{3.14}$$

A função objetivo (3.8) corresponde à equação (3.1) alterada para que os custos de troca (*setup*) de todas as máquinas fossem somados e para contabilizar a penalização (M) sobre demanda não atendida (q_j). Também foi adicionada uma restrição de balanço de estoque para o primeiro período (3.9a). Ela faz com que as demandas não atendidas para um produto j sejam acumuladas na variável q_j. A restrição de balanço de estoque (3.2) do modelo anterior foi modificada, resultando na equação (3.9b) deste modelo, para que a quantidade total produzida, envolvida no cálculo do estoque, contemple a produção de todas as máquinas. As equações (3.3), (3.4), (3.5), (3.6) e (3.7) do modelo anterior foram alteradas para suas correspondentes neste modelo (3.10), (3.11), (3.12), (3.13) e (3.14). Estas adaptações foram realizadas apenas para que estas restrições possam ser aplicadas para todas as máquinas, separadamente.

3.4 Os Métodos Threshold Accepting e Greedy-Mod

3.4.1 Threshold Accepting

O método *Threshold Accepting*, apresentado por Fleischmann (1997), é uma heurística de busca local aplicada a muitos problemas combinatórios, onde apresenta bons resultados quando comparados a outras técnicas. O método consiste em, a partir de uma configuração inicial (solução atual), gerar um conjunto de configurações candidatas, chamado de vizinhança, e verificar se algum desses candidatos possui qualidade superior à configuração atual. Se isso ocorrer, este candidato é aceito como a nova configuração e uma nova vizinhança é gerada para ele. A geração desta vizinhança é realizada a partir da aplicação dos "operadores de vizinhança" sobre a configuração atual. Estes operadores causam pequenas perturbações na configuração, gerando variações dela, que são as configurações candidatas (ou vizinhança). Para que o método não fique preso em ótimos locais e sua convergência não seja prejudicada, um candidato de qualidade inferior à configuração atual pode ser aceito como a nova configuração. Nesse caso, a diferença de qualidade entre ele e a configuração atual deve ser inferior a um determinado valor, chamado *threshold*. Fleischmann (1997) apresenta o método em uma versão auto-adaptativa, em que o valor *threshold* é calculado de acordo com as configurações aceitas.

Na aplicação do *Threshold Accepting* ao PGDLPP, uma configuração equivale a um padrão de *setup* da produção que representa a alocação dos produtos nas máquinas em cada micro-período, com o tamanho dos lotes atribuídos. A qualidade de uma configuração corresponde ao custo da função objetivo do problema. Para reduzir o tempo de processamento do método, Fleischmann (1997) utiliza os operadores de vizinhança apenas para gerar novos padrões de *setup*, sendo o dimensionamento dos lotes (atribuição dos valores de

x_{js}) feito de forma determinística por "procedimentos orientados a retrocesso", entre eles o *Greedy-Mod*.

Com o objetivo de facilitar a aplicação dos operadores de vizinhança, uma outra estrutura é utilizada pelo *Threshold Accepting*, além daquelas já presentes no modelo do problema. No lugar da variável y_{js}, utilizada para representar o padrão de *setup* ao longo do horizonte de planejamento, é utilizada a variável $u_t = (u_t^1, u_t^2, ..., u_t^{nt})$ que representa os produtos processados na máquina no período *t*. Nesta representação, u_t^k indica qual produto foi alocado no *k*-ésimo micro-período do período *t*. O último produto produzido no período é representado por u_t^{nt}, onde $0 \leq nt \leq |S_t|$. Isto significa que, nessa nova estrutura, nem todos os micro-períodos dentro de um período precisam estar alocado a produtos. Outra característica importante da variável u_t é que ela não admite duas alocações consecutivas de um mesmo produto em um mesmo período.

A busca por configurações candidatas se utiliza de três operadores de vizinhança: inserção, deleção e troca. O operador de inserção consiste em inserir um novo produto u_t^k na produção, sendo que o período *t* e o micro-período *k*, onde o novo produto será alocado, são selecionados aleatoriamente, assim como o produto a ser alocado (valor de u_t^k). Quando um produto u_t^k for inserido, todos os produtos $u_t^{k'}$ onde $k' \geq k$ são realocados, cada um para a posição seguinte ao seu respectivo micro-período. Eles são deslocados "para frente" de forma a deixar uma posição vaga para o novo produto a ser inserido. O operador de deleção consiste em excluir um produto existente na produção, sendo este selecionado da mesma forma que no operador de inserção. Ao excluir um produto u_t^k, todos os produtos alocados posteriormente a ele e dentro do mesmo período, ou seja, todos os produtos $u_t^{k'}$ em que $k' > k$, são realocados cada um para a posição anterior ao seu respectivo micro-período. O operador de troca

seleciona aleatoriamente dois produtos e troca suas posições. Estes dois produtos não precisam estar alocados no mesmo período.

A seleção de qual operador de vizinhança será utilizado é feita de forma determinística. O método analisa qual foi o último operador aplicado, e se ele gerou uma solução melhor que a anterior. Caso isto aconteça, o operador é repetido, senão, as chances de repetir este operador diminuem, mas não acabam. Um operador que obteve sucesso várias vezes consecutivas terá mais chances de ser aplicado novamente, mesmo não obtendo uma configuração melhor em uma ultima aplicação. Porém, se este operador continuar a falhar, os outros operadores serão selecionados. Esta forma de seleção dos operadores faz com que o método busque utilizar aqueles operadores que estiverem contribuindo mais para a sua convergência.

O método *Threshold Accepting* depende de uma solução inicial como ponto de partida para gerar outras soluções. A geração de uma solução inicial factível é um problema de alta complexidade. Para contornar este problema, Fleischmann (1997) opta por gerar uma solução inicial infactível e penalizá-la. Assim, a própria convergência do método poderá levá-lo a soluções factíveis. Um período fictício 0, sem restrições de capacidade, é utilizado para alocar as demandas que não foram atendidas ao longo do horizonte de planejamento. Uma produção de quantidade x_j^0 neste período resulta em uma penalização de $h_j^0 x_j^0$, onde h_j^0 é a penalização por unidade de demanda do produto *j* não atendida. Assim, a solução inicial é gerada atribuindo a produção de todas as demandas de todos os produtos no período fictício 0.

3.4.2 Greedy-Mod

O *Greedy-Mod*, apresentado por Fleischmann (1997), é um algoritmo determinístico que, a partir de um padrão de *setup* de uma solução, das capacidades dos períodos e das demandas dos produtos, consegue determinar o tamanho dos lotes para esta configuração minimizando seu custo. O método é dividido em várias etapas. Na primeira, ele passa por todos os períodos atribuindo aos seus lotes os valores de lote mínimo dos produtos. A segunda etapa consiste em percorrer todos os períodos, partindo do último para o primeiro, alocando o restante da capacidade de cada período aos lotes.

Para cada período *t*, dois passos são executados para determinar os lotes. Cada produto *j* que possui ao menos um lote alocado em *t*, onde D_j é sua demanda remanescente, é feita uma verificação da possibilidade de se alocar toda a demanda restante de *j* nos períodos *t'* precedentes a *t* (*t'* < *t*). Isto é feito calculando a soma das capacidades dos períodos *t'* < *t* em que existe algum lote do produto *j* alocado, e verificando se esta capacidade total é capaz de comportar a produção de toda a demanda restante de *j*.

O cálculo referente a esta soma das capacidades é apresentado na equação (3.15), onde K'_{jt} é a soma das capacidades dos períodos precedentes a *t* onde está alocado pelo menos um lote do produto *j* , e $y'_{j\tau}$ representa o número de lotes do produto *j* alocados no período τ.

$$K'_{jt} = \sum_{\tau=1}^{t-1} \min\{1, y'_{j\tau}\} K_\tau \qquad (3.15)$$

Caso não seja possível alocar toda a demanda restante de *j* em K'_{jt}, isto é, se $K'_{jt} < D_j a_j$, a parcela da produção de D_j que não "couber" nos períodos precedentes a *t* será alocada em *t* em um dos lotes de *j* existentes. Tal alocação deve ser feita respeitando a capacidade de *t*. O cálculo da quantidade a ser

alocada é apresentado na equação (3.16), onde esta quantidade é representada por *diff*, considerando que a capacidade do período (K_t) já foi diminuída, conforme o tamanho dos lotes alocados até então.

$$diff = \min\{D_j - \frac{K'_{jt}}{a_j};\quad \frac{K_t}{a_j};\quad D_j - \sum_{\tau=1}^{t-1} d_{j\tau}\} \qquad (3.16)$$

Este primeiro passo contribui para evitar que, ao chegar à iteração referente ao primeiro período, o algoritmo não consiga alocar toda a demanda restante, pois parte desta deveria ter sido alocada em um período posterior. No segundo passo da alocação da capacidade entre os lotes do período *t* é necessário percorrer todos eles. Isso é executado na ordem decrescente dos custos de estoque relativo de seus produtos, alocando a demanda restante do produto, menos as demandas referentes a períodos precedentes a *t*, sem exceder a capacidade de K_t. O cálculo da quantidade a ser alocada é apresentado na equação (3.17), onde esta quantidade é representada por *diff* e *j* é o produto cujo lote foi selecionado. Neste caso, é considerado que a capacidade do período (K_t) já foi diminuída conforme o tamanho dos lotes alocados ate então.

$$diff = \min\{D_j - \sum_{\tau=1}^{t-1} d_{j\tau};\quad \frac{K_t}{a_j}\} \qquad (3.17)$$

Até esta etapa do algoritmo, apenas a quantidade total de cada produto que é produzido em cada período é calculada. Em uma última etapa do algoritmo, chamada de desagregação, para cada período *t*, a quantidade total de um produto a ser produzida em *t* é dividida igualmente e alocada para os lotes de *t* referentes a este produto. Um pseudocódigo para o algoritmo *Greedy-Mod* é apresentado na figura 3.3, onde x'_{jt} representa a quantidade total do item *j* produzida no período *t*, e y'_{jt} representa o número de lotes do produto *j* presentes no período *t*.

Agregar para todo *j, t:*
$x'_{jt} := \sum_{s \in S_t} x_{jt}$;
inicio
$x'_{jt} := 0$; ∀ *j,t*
$y'_{jt} := y_{jf_t} + \sum_{s \in S_t \{f_t\}} \max\{0; y_{js} - y_{j,s-1}\}$; ∀ j,t
$D_j := \sum_{t=1}^{T} d_{jt}$; ∀ *j*
para *t*:=*T* **até 1 faça**
para todo *j* **com** ($y'_{jt} > 0$) **faça**
$x'_{jt} := m_j y'_{jt;}$
$K_t := K_t - a_j x'_{jt;}$
$D_j := D_j - \min\{ x'_{jt}; D_j - \sum_{\tau=1}^{t-1} d_{j\tau}\}$;
se ($K_t < 0$) **então pare;**
/* qualidade(nova configuração) := ∞; */
fim do para;

para todo *j* **com** ($y'_{jt} > 0$) **faça**
se ($D_j > \frac{K'_{jt}}{a_j}$) **então**
$\text{diff} := \min\{D_j - \frac{K'_{jt}}{a_j}; \frac{K_t}{a_j}; D_j - \sum_{\tau=1}^{t-1} d_{j\tau}\}$;
$x'_{jt} := x'_{jt} + \text{diff}$;
$K_t := K_t - \text{diff} \cdot a_j$;
$D_j := D_j - \text{diff}$;
fim do se;
fim do para;

para todo *j* **com** *($y'_{jt} > 0$)* **em ordem decrescente** $\frac{hj}{a_j}$ **faça**
$\text{diff} := \min\{D_j - \sum_{\tau=1}^{t-1} d_{j\tau}; \frac{K_t}{a_j}\}$;
$x'_{jt} := x'_{jt} + \text{diff}$;
$D_j := D_j - \text{diff}$;
$K_t := K_t - \text{diff} \cdot a_j$;
fim do para;
fim do para;

$x_j^0 := D_j$; ∀ j
fim;

Desagregar para todo *j, t* **e** $s \in S_t$**:**
se ($y'_{jt} > 0$ **e** houve troca para *j* no microperíodo s) **então**
$x_{js} := \frac{x'_{jt}}{y'_{jt}}$;
senão
$x_{js} := 0$;

Figura 3.3 – Algoritmo Greedy-Mod em pseudocódigo.

3.5 Modelagem Matemática Utilizando AMPL

AMPL é uma linguagem de modelagem algébrica para problemas de otimização lineares e não lineares, com variáveis discretas e contínuas. O AMPL trabalha da seguinte forma: recebe um arquivo com o modelo matemático do problema e uma instância deste problema (arquivo de dados), ambos implementados seguindo a sintaxe própria do AMPL. As informações são então transformadas em um formato capaz de ser interpretado pelo *solver*. *Solvers* são implementações de algoritmos utilizados na resolução de problemas de otimização. O *solver* utilizado resolve o problema retornando uma solução para o AMPL. Diversas funcionalidades existentes no AMPL permitem ao usuário interpretar a solução obtida (Figura 3.4). Mais informação sobre a linguagem AMPL podem ser encontradas em AMPL (2009).

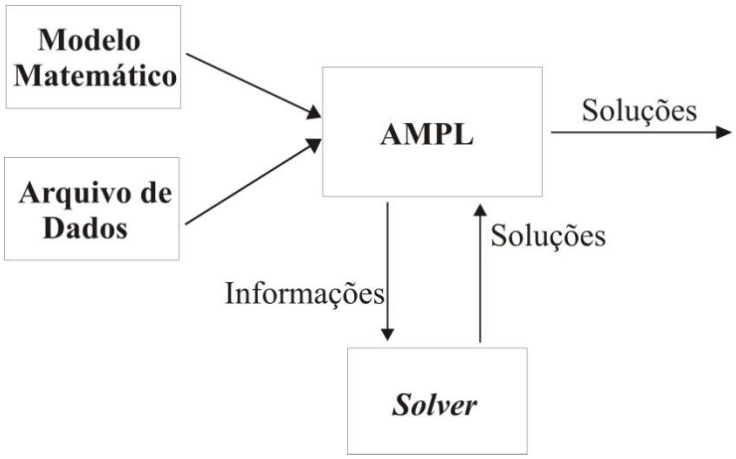

Figura 3.4 – Funcionamento do AMPL.

O modelo matemático do PGDLPP para uma ou várias máquinas usando AMPL é apresentado na figura 3.5.

```
set T ordered;          # Periodos.
set L ordered;          # Maquinas (linhas de producao).
set MP {L} ordered;     # Conjunto de todos os micro-periodos.
set J ordered;          # Itens a serem produzidos.
set S {L,T};            # Micro-periodos em cada periodo.
param K {L,T};          # Capacidades por periodo.
param a {L,J};          # Tempo de produção unitário.
param m {L,J};          # Tamanho minimo do lote para cada item.
param h {J};            # Custos de estoque para cada item.
param s {L,J,J};        # Custos de setup.
param d {J,T};          # Demanda de cada item em cada periodo.
param I0 {J};           # Estoque inicial de cada item.
param y0 {L,J};         # Setup inicial.
param M;                # Penalizacao por unidade de demanda nao antendida.

var I  {j in J, t in T} >= 0;    # Estoque no final de cada período.
var q  {j in J} >= 0;            # Demanda nao atendida para cada produto.
var x  {l in L, j in J, mp in MP[l]} >= 0;     # Quantidades produzidas.
var y  {l in L, j in J, mp in MP[l]} binary;   # Flag de produto setado.
var z  {l in L, i in J, j in J, k in MP[l]} >= 0;  # Flag de troca.

minimize Cost:                                          # Funcao Objetivo (1).
    ( sum{j in J, t in T} h[j] * I[j,t] ) + (M * ( sum{j in J} q[j] )) +
        ( sum{l in L, i in J, j in J, k in MP[l]} s[l,i,j] * z[l,i,j,k] );

subject to remaningInventory {j in J, t in T}: # Balanço estoque(2a e 2b).
    I[j,t] =
        if t > 1 then
            I[j, t - 1]  + ( sum{l in L, k in S[l,t]} x[l,j,k] ) - d[j,t]
        else
            I0[j] + q[j] + ( sum{l in L, k in S[l,t]} x[l,j,k] ) - d[j,t];

subject to capacityConstraint {l in L, t in T}:          # Capacidade (3).
    ( sum{j in J, k in S[l,t]} a[l,j] * x[l,j,k] ) <= K[l,t];

# Restricao que obriga a alocar o produto na maquina para produzi-lo (4).
subject to setupToProduce {l in L, t in T, j in J, k in S[l,t]}:
    x[l,j,k] <= ( K[l,t] / a[l,j] ) * y[l,j,k];

subject to lotsizeConstraint{l in L, j in J, k in MP[l]}: #Lote mínimo(5).
    x[l, j,k] >=
        if k > 1 then
            m[l,j] * ( y[l,j,k] - y[l,j,k - 1] )
        else
            m[l,j] * ( y[l,j,1] - y0[l,j] );

#--Restricao que limita setups por micro-periodo a um (6).
subject to setupByMicroperiod {l in L, k in MP[l]}:
    sum{j in J} y[l,j,k] = 1;

#--Restricao de mudanca de setup (7).
subject to setupChangeConstraint {l in L, i in J, j in J, k in MP[l]}:
    z[l,i,j,k] >=
        if k > 1 then
            y[l,i,k - 1] + y[l,j,k] - 1
        else
            y0[l,i] +      y[l,j,k] -1;
```

Figura 3.5 – Modelo matemático usando AMPL.

As Figura 3.6 e 3.7 compõem uma instância do problema utilizando AMPL.

```
set L := 1 2;
set T := 1 2 3 4 5;
set J := 1 2 3;

set MP[1] := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
set MP[2] := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;

set S[1,1] := 1 2 3;
set S[1,2] := 4 5 6;
set S[1,3] := 7 8 9;
set S[1,4] := 10 11 12;
set S[1,5] := 13 14 15;
set S[2,1] := 1 2 3;
set S[2,2] := 4 5 6;
set S[2,3] := 7 8 9;
set S[2,4] := 10 11 12;
set S[2,5] := 13 14 15;

param K : 1   2   3   4   5   :=
1         150 150 150 150 150
2         150 150 150 150 150;

param a : 1    2    3   :=
1         2.63 2.43 1.72
2         1.32 2.64 1.15  ;

param m : 1   2   3   :=
1          3   2   2
2          3   2   2  ;

param h :=
1 1
2 1
3 1;

param s :=
[1,1,1] 0
[1,1,2] 185
[1,1,3] 129
[1,2,1] 130
[1,2,2] 0
[1,2,3] 117
[1,3,1] 136
[1,3,2] 142
[1,3,3] 0
[2,1,1] 0
[2,1,2] 185
[2,1,3] 129
[2,2,1] 130
[2,2,2] 0
[2,2,3] 117
[2,3,1] 136
[2,3,2] 142
[2,3,3] 0;
```

Figura 3.6 – Arquivo de dados de uma instância do problema (parte 1).

```
param d : 1   2   3   4   5   :=
1         52  74  38  52  44
2         50  28  57  54  34
3         34  33  47  25  81 ;

param I0 :=
1 0
2 0
3 0;

param y0 : 1   2   3   :=
1          1   0   0
2         1   0   0  ;

param M = 10000;
```

Figura 3.7 – Arquivo de dados de uma instância do problema (parte 2).

3.6 Gerador de Instâncias

Uma instância contém todos os dados necessários para que o problema possa ser solucionado. Assim, todos os parâmetros envolvidos no problema precisam ser fornecidos. Os parâmetros contidos em uma instância do PGDLPP são:

- Número de máquinas;
- Número de períodos;
- Micro-períodos contidos em cada período;
- Número de produtos a serem produzidos;
- Capacidade de cada máquina em cada período;
- Tempo gasto para produzir uma unidade de cada produto;
- Tamanho mínimo do lote de cada produto;
- Custos de estoque;
- Custos de *setup*;

- Demandas dos produtos nos períodos;
- Estoque inicial de cada produto;
- Configuração inicial de cada máquina;
- Valor da penalização por unidade de demanda não atendida.

A criação de instância é feita ao se determinar valores para esses parâmetros a partir do domínio de cada um deles. Esse domínio é estabelecido baseado em valores que permitam simular situações envolvendo o problema estudado. Neste trabalho é proposto e implementado um gerador de instâncias que determina valores para todos os parâmetros envolvidos no problema. O gerador estabelece conjuntos de instâncias cuja resolução permite avaliar os métodos de resolução abordados neste trabalho.

As seguintes considerações, apresentadas por Haase (1995), são seguidas na implementação do gerador:

- Capacidade constante em todos os períodos;
- Tempo de produção constante ao longo dos períodos, para um mesmo produto em uma mesma máquina;
- A demanda de cada item para cada período deve ser um valor entre 0 e 100, escolhido aleatoriamente. Esses valores também devem respeitar as restrições de taxa de utilização da capacidade;
- Os custos de estoque devem ser constantes e iguais para todos os itens;
- Todas as máquinas devem estar configuradas para o mesmo item inicialmente. É utilizado para esta configuração inicial o primeiro produto da lista de produtos envolvidos;

- O custo de *setup* de um produto *i* para outro produto *j* deve ser um número entre 100 e 200 escolhido aleatoriamente, desde que *i* seja diferente de *j*. Caso sejam iguais, o custo deve ser nulo.

O algoritmo recebe alguns dados de entrada, através dos quais é possível determinar o tipo de instância que será gerada e seu grau de dificuldade. São eles: número de máquinas (*L*), número de períodos (*T*), número de produtos (*J*), número máximo de micro-períodos por período (*mps*), tempo de processamento mínimo (*aMin*), tempo de processamento máximo (*aMax*) e taxa de utilização da capacidade (*U*). Todos estes parâmetros devem ser maiores que zero. A taxa de utilização da capacidade de ser maior que zero e menor ou igual a um ($0 < U \leq 1$).

A taxa de utilização da capacidade (*U*) representa a fração da capacidade das máquinas que se espera que seja utilizada. Assim, este valor é útil para a geração de instâncias, pois a produção de todas as demandas geradas para um período *t* deve ser determinada pelo somatório das capacidades das máquinas nesse período vezes a taxa de utilização da capacidade. Por exemplo, se a taxa de utilização da capacidade é 0,7, então, espera-se que 70% da capacidade total disponível em cada período sejam utilizados. Assim, as demandas dos produtos para cada período serão geradas de forma que a produção destas "caiba" nestes 70% da capacidade. Quanto menor a taxa de utilização da capacidade, maior será a capacidade excedente por período, possibilitando maior produção e, conseqüentemente, menor utilização das máquinas. A figura 3.8 apresenta um pseudocódigo para o algoritmo de geração de instâncias.

Procedimento geraInstancia (*L*, *T*, *J*, *mps*, *aMin*, *aMax*, *U*)
início

para l = 1 **até** L **faça**
se *mps* < J **então**
$mps := J$
fim do se

para t = 1 **até** T **faça**
S_{lt} := { (t - 1) mps + 1,..., (t - 1) *mps* + *mps* };
$K_{lt} := 50\ J$;
fim do para

para j = 1 **até** J **faça**
a_{lj} := sorteiaNumeroReal (*aMin*, *aMax*);
d'_{jt} := sorteiaNumeroInteiro (0, 100);
fim do para

$y_{lj0} := 0$; $\forall\ j$
$y_{l10} := 1$;
fim do para

$I_{j0} := 0$; $\forall\ j$

para j = 1 **até** J **faça**
para i = 1 **até** J **faça**
var custoDeSetup := sorteiaNumeroInteiro (100, 200);
s_{lji} := custoDeSetup; $\forall\ l$
fim do para
fim do para

$A_j := \max\{a_{1j},..., a_{Lj}\}$; $\forall\ j$

$$K'_t := \sum_{j=1}^{J} d'_{jt}\, a_j; \quad \forall t$$

$$d_{jt} := \frac{K_{0t}\, L\, d'_{jt}\, U}{K'_t}; \quad \forall j,t$$

para j = 1 **até** J **faça**
var dMin := min $\{d_{j1},..., d_{jT}\}$
$m_{lj} := \max\{1; 0.1\ \text{dMin}\}$; $\forall\ l$
fim do para

$M := 10000;$
fim;

Figura 3.8 – Pseudocódigo para a geração de instância do PGDLPP.

O primeiro passo do algoritmo consiste em garantir que o número de micro-períodos seja no mínimo o número de produtos. Esta verificação é feita para garantir que sempre possa existir pelo menos um lote de cada produto em cada período, diminuindo as chances de gerar instâncias sem soluções factíveis. Após isto, os micro-períodos serão distribuídos de forma igualitária para todos os períodos de todas as máquinas, isto é, todos os períodos terão o mesmo número de micro-períodos ($|S_{lt}| = mps, \ \forall\, l,t$). Em seguida, as capacidades das máquinas são atribuídas. Todas elas possuem a capacidade igual a cinqüenta vezes o número de produtos ($50J$) em todos os períodos. Os tempos de produção dos produtos são calculados em seguida. Esses tempos podem diferir entre as máquinas e são valores reais selecionados aleatoriamente entre o tempo de processamento mínimo e máximo ($aMin \leq \ a_{lj} \leq aMax, \ \forall\, l, j$).

O passo seguinte consiste em calcular uma demanda fictícia d'_{jt} para cada produto j em cada período t, sendo esta é um valor inteiro selecionada de forma aleatória no intervalo fechado de 0 a 100. Estes são valores temporários, úteis posteriormente para o cálculo das demandas definitivas. Em seguida são atribuídas as configurações iniciais das máquinas. Todas as máquinas são configuradas para o produto 1 ($y_{lj0} = 1, \ \forall$ l, j = 1 e $y_{lj0} = 0, \ \forall$ l, j $\neq$ 1). O próximo atributo da instância a ser calculado são os custos de troca (*setup*). O custo de troca entre dois produtos é um número inteiro gerado aleatoriamente no intervalo fechado de 100 a 200, sendo igual para todas as máquinas.

A variável temporária A_j é utilizada para armazenar o maior tempo de produção unitário de cada produto, considerando todas as máquinas. Neste ponto são calculadas as capacidades fictícias, representadas por K'_t. A capacidade fictícia de um período é a capacidade necessária para a produção de toda a demanda fictícia deste período, considerando o pior caso, em que cada produto seja produzido na máquina mais dispendiosa para ele (com tempo de

processamento A_j por unidade de j). Assim a capacidade fictícia de um produto t é calculada de acordo com a equação abaixo.

$$K'_t := \sum_{j=1}^{J} d'_{jt}\, a_j; \quad \forall t \qquad (3.18)$$

Após esta etapa já estão disponíveis todos os dados necessários para o cálculo das demandas definitivas da instância. Elas são calculadas através de um artifício matemático que é capaz de diminuir as demandas fictícias para que caibam na capacidade real do período, mantendo a proporção entre estas demandas. Manter a proporção entre as demandas ao diminuí-las é importante, visto que todas foram calculadas de forma semelhante em um mesmo intervalo. Outros métodos para realizar tal diminuição poderiam prejudicar a aleatoriedade do algoritmo. O cálculo das demandas definitivas é apresentado na equação a seguir.

$$d_{jt} := \frac{K_{0t}\, L\, d'_{jt}\, U}{K'_t}; \qquad \forall j,t \qquad (3.19)$$

O ultimo passo do algoritmo consiste em calcular os lotes mínimos dos produtos para cada máquina. Os lotes mínimos de um produto serão iguais em todas as máquinas. O seu valor é um décimo da menor demanda ao longo do horizonte de planejamento, considerando que seu valor mínimo deva ser um. Este algoritmo de geração de instâncias não é determinístico, ou seja, para as mesmas entradas, a cada execução podem ser geradas instâncias diferentes. Isto permite a geração de grupos de instâncias, contendo várias instâncias diferentes, à medida que os parâmetros de entrada são alterados.

Capítulo 4

Método de Resolução

4.1 Algoritmo Genético com Estrutura Hierárquica

O algoritmo genético proposto para resolver o Problema Capacitado de Dimensionamento de Lotes visa à busca por um indivíduo (solução) cuja representação (padrão de setup e dimensionamento dos lotes) resulte em um *fitness* (custo da solução) mínimo (Figura 4.1).

```
Procedure algoritmoGenético ( )
início
        repetir
                InicializarPopulação ();
                avaliarPopulação();
                repetir
                        recombinarPopulação ();
                        mutarPopulação ();
                        avaliarPopulação ();
                        reestruturarPopulação ();
                até (convergiu)
        até (tempo de execução acabou)
fim
```

Figura 4.1 – Pseudocódigo para o Algoritmo Genético proposto.

Em sua primeira etapa o algoritmo inicializa a população. São criados vários indivíduos de modo que eles estejam aptos a serem avaliados e também a sofrerem a ação dos operadores genéticos. Após ser inicializada, a população é avaliada, o que consiste em calcular o valor do *fitness* de seus indivíduos. Na segunda etapa do algoritmo, quatro passos são executados em busca da convergência. O primeiro deles é a recombinação da população. A aplicação do operador de recombinação (*crossover*) gera um novo indivíduo, alocado na população de acordo com seu valor de *fitness*. No segundo passo, os operadores

de mutação podem ser aplicados aos novos indivíduos gerados, modificando sua representação. Como a aplicação dos operadores genéticos resulta em modificações dos indivíduos, o terceiro passo consiste em reavaliar os indivíduos modificados. O quarto e último passo se resume em organizar a população de acordo com sua estrutura hierárquica, pois novos indivíduos podem ter sido inseridos. Caso novos indivíduos não tenham sido inseridos, o algoritmo voltará a sua primeira etapa reinicializando a população, mantendo apenas o melhor indivíduo. Se a convergência não for atingida, o algoritmo repetirá os quatro passos da segunda etapa continuando a evolução da população atual. A execução do algoritmo é interrompida quando o tempo para o qual ele foi programado termine. O pseudocódigo deste algoritmo é apresentado na figura 4.1.

4.2 Representação do Indivíduo

Uma representação do indivíduo é uma possível solução para o problema. Cada indivíduo será representado por linhas de produção, onde cada linha possui duas matrizes com informações sobre sua seqüência de produção e a demanda dos produtos a ser atendida. A primeira matriz é apresentada na figura 4.2.

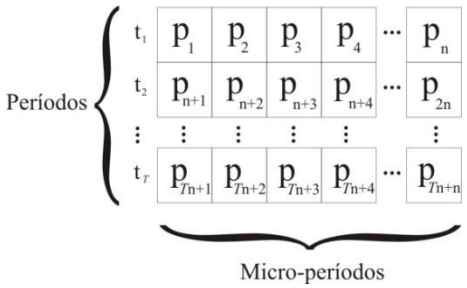

Figura 4.2 – Matriz de seqüência.

Esta matriz é do tipo T x n, onde T é o número de períodos e n é o número de micro-períodos. Cada entrada (t, n) corresponde ao produto que ocupa determinado micro-período dentro de um período. Duas entradas consecutivas em um mesmo período não podem ser ocupadas pelo mesmo produto. Também não pode haver uma entrada sem produto entre duas entradas ocupadas em um mesmo período. Essa representação foi adotada por Fleischmann (1997) para o PGDLPP com uma única linha.

A segunda matriz é a matriz de demandas do tipo T x J, onde T é o número de períodos e J é o número de produtos do problema (Figura 4.3).

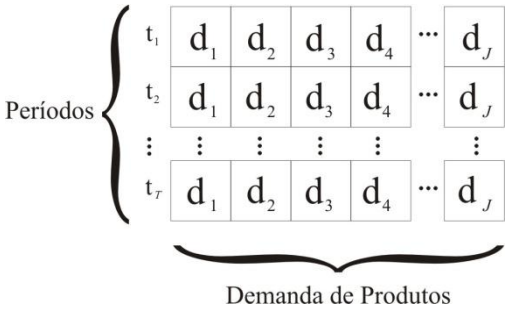

Figura 4.3 – Matriz de demandas.

Esta matriz representa as demandas dos produtos em cada período que serão atendidas pelas produções desta linha. Toda demanda do problema é distribuída ao longo das matrizes de demanda nas linhas. Se há uma única linha no problema, a matriz de demanda irá corresponder à demanda total do problema. Um exemplo de indivíduo é apresentado na figura 4.4. Neste exemplo, a instância possui duas linhas de produção (L = 2), três produtos (J = 3) e três períodos (T = 3), com um máximo de 4 micro-períodos cada um ($|S_t|$ = 4 $\forall$ t).

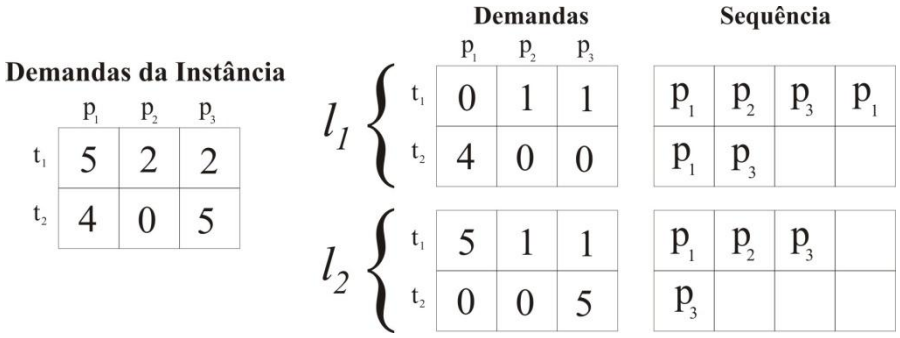

Figura 4.4 – Exemplo de indivíduo.

Pode-se observar no exemplo que não é obrigatória a utilização de todos os micro-períodos, e que o somatório das demandas das linhas coincide com a demanda total da instância. Outra característica exemplificada na figura é que, mesmo em uma linha não havendo demanda de determinado produto, nada impede que este produto seja produzido nesta linha.

4.3 Inicialização do Indivíduo

A inicialização do indivíduo é um ponto crítico do algoritmo genético, pois pode impactar diretamente em seu desempenho. Por exemplo, uma abordagem que induza a inicialização pode levar à geração de indivíduos similares, dificultando uma exploração ampla do espaço de busca. Na abordagem deste trabalho, a inicialização do indivíduo consiste em gerar uma seqüência de produção e uma matriz de demanda para cada linha de produção do indivíduo.

A geração da seqüência de produção é efetuada separadamente para cada linha, onde um número de micro-períodos é selecionado aleatoriamente para cada período. Este número está entre zero e o número máximo de micro-períodos, representando o número de micro-períodos que serão ocupados neste

período. Após isso, para cada micro-período a ser ocupado, um produto é selecionado aleatoriamente e alocado no micro-período. Verificações são efetuadas para impedir a alocação de um mesmo produto em micro-períodos consecutivos.

A geração das demandas consiste na distribuição das demandas totais da instância entre as linhas de produção do indivíduo. Para cada elemento da matriz de demandas da instância, uma parcela do valor desta demanda é sorteada e atribuída a uma linha selecionada aleatoriamente. Este processo é repetido até que toda a demanda da instância seja distribuída entre as linhas.

4.4 Estruturas Populacionais

A população de um algoritmo genético consiste em um conjunto de indivíduos e pode ou não ser estruturada. Em populações não estruturadas a população é avaliada e o melhor indivíduo é, provavelmente, armazenado em uma posição de fácil acesso, porém não existe uma organização lógica entre os demais indivíduos. A figura 4.5 ilustra uma população não estruturada.

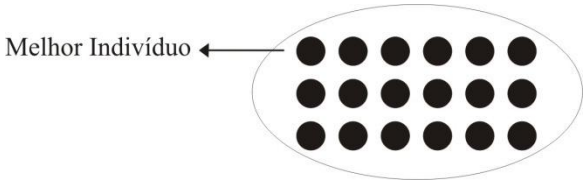

Figura 4.5 – População Não Estruturada.

Em busca de uma melhora no desempenho do algoritmo são utilizadas populações estruturadas. O diferencial para estas populações é que elas possuem uma organização lógica entre seus indivíduos, facilitando muitas vezes a aplicação dos operadores genéticos. A estrutura populacional abordada neste

trabalho é a estrutura hierárquica. Nesta, a população é estruturada em árvore e a hierarquia seguida está relacionada ao *fitness* dos indivíduos. O conjunto de um nó pai e seus filhos diretos é chamado de *cluster*. Todo nó pai possui o *fitness* de qualidade superior ao de seus nós filhos em um *cluster*. Mantendo a lógica desta estrutura, o melhor indivíduo sempre estará localizado na raiz da árvore. O nó pai de um *cluster* é denominado o líder deste *cluster* e seus filhos os seguidores. Em uma estrutura de árvore binária, onde cada nó possui dois filhos, cada cluster conterá três indivíduos. Similarmente, em uma estrutura de árvore ternária, onde cada nó possui três filhos, cada cluster conterá quatro indivíduos. A figura 4.6 ilustra as estruturas populacionais utilizando árvore binária e ternária.

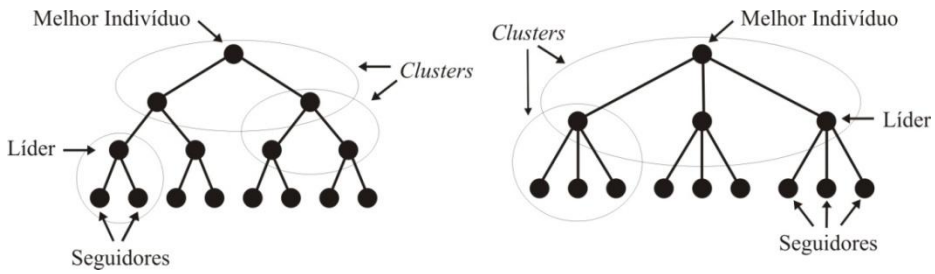

Figura 4.6 – Estrutura populacional em árvore binária e ternária.

Ao aplicar operadores de recombinação (*crossover*, a ser detalhado em seções posteriores), a estrutura hierárquica da árvore pode ser prejudicada. O operador de recombinação seleciona aleatoriamente um cluster, e a recombinação é aplicada ao líder juntamente com um de seus seguidores. O resultado deste operador é um novo indivíduo, que será inserido no lugar do seguidor envolvido, caso a qualidade de seu *fitness* seja superior. Caso o seguidor seja substituído pelo novo indivíduo gerado, pode ocorrer deste novo indivíduo também apresentar qualidade superior à do líder do cluster. Isso viola a estrutura hierárquica da população. Assim deve ser efetuada uma

reestruturação da árvore, que consiste em sucessivos movimentos de troca de posição entre os indivíduos, até que todo nó pai tenha qualidade superior aos seus filhos. A figura 4.7 exemplifica a reestruturação da população após a inserção de um novo indivíduo. Observe que o melhor indivíduo tem menor valor de *fitness* num problema de minimização, sendo posicionado como líder dos *clusters*.

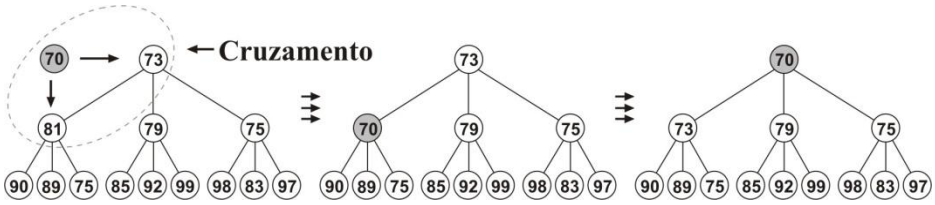

Figura 4.7 – Reestruturação da população com a inserção de novo indivíduo.

4.5 Decodificação e Avaliação do Indivíduo

A representação do indivíduo apresentada na seção 4.2 consiste em uma codificação da solução real. Esta codificação é utilizada como meio facilitador para a aplicação do algoritmo e seus operadores. Para que um indivíduo seja avaliado, ele deve ser decodificado em uma solução e o resultado do cálculo de custo desta corresponde ao *fitness* do indivíduo. Porém, após ser inicializado ou sofrer a ação dos operadores genéticos, um indivíduo ainda não está preparado para ser decodificado, pois seus lotes devem ser dimensionados.

O dimensionamento dos lotes é feito através do uso do algoritmo determinístico *Greedy-Mod* (seção 3.4.2). Este algoritmo foi utilizado por Fleischmann (1997) para o problema envolvendo uma só linha. Nesta abordagem do algoritmo genético, o *Greedy-Mod* foi adaptado para receber a

matriz de seqüência e a matriz de demanda de cada uma das linhas de produção do indivíduo. Quando o *Greedy-Mod* é aplicado a uma das linhas do indivíduo, ele atribui os tamanhos de lote sem considerar as demais linhas. Deste modo, o *Greedy-Mod* deve ser aplicado a cada uma das linhas separadamente.

Uma vez dimensionados os lotes, um indivíduo está pronto para ser decodificado. A decodificação consiste em, a partir da matriz de seqüência de cada linha, organizar a produção em uma seqüência linear de lotes. Desta forma, é possível identificar o início e o fim de cada período, ignorando os micro-períodos não utilizados. A figura 4.8 apresenta um exemplo de indivíduo e sua decodificação.

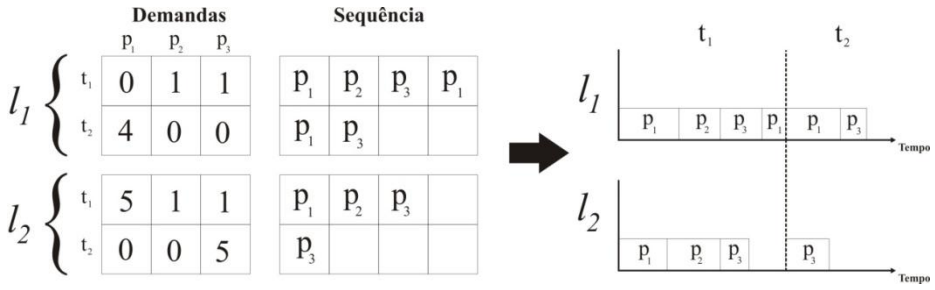

Figura 4.8 – Decodificação do Indivíduo.

A partir da solução decodificada é possível calcular o seu custo. Este cálculo é realizado de acordo com a função objetivo do problema, apresentada na seção 3.3. O valor do custo é atribuído ao *fitness* do indivíduo e a representação decodificada da solução é descartada, visto que as demais partes do algoritmo utilizam apenas a representação codificada (indivíduo).

4.6 Operadores Genéticos

Os operadores genéticos executam alterações na representação do indivíduo com intuito de modificá-lo visando à geração de novas soluções. Estes operadores são essenciais para a convergência do algoritmo e suas taxas de incidência são parametrizáveis. Na representação do indivíduo utilizada, os movimentos são aplicados nas matrizes de demandas e nas matrizes de seqüência.

Os operadores de recombinação envolvem dois indivíduos (pais) que são combinados a fim de gerar um novo indivíduo (filho). Neste trabalho foram avaliados dois operadores de recombinação, conhecidos como "*Crossover* Uniforme" e "*Crossover* de um ponto". Eles são aplicados de forma semelhante para as seqüência de produtos e para as demandas. Assim a explicação a seguir faz referência à matriz de seqüência, mas o raciocínio se estende à matriz de demanda.

O *crossover* uniforme é apresentado na figura 4.9.

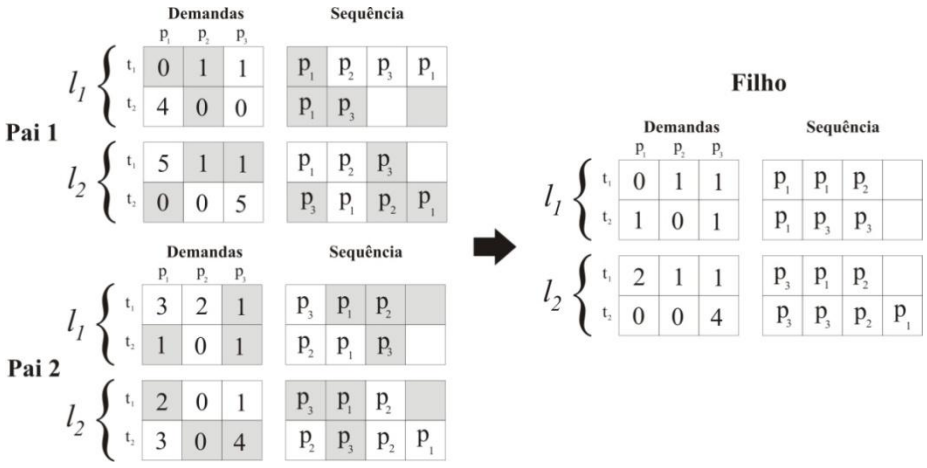

Figura 4.9 – *Crossover* Uniforme.

Este *crossover* consiste em, para cada linha, percorrer as linhas da matriz de seqüência (períodos) e, em cada célula da matriz (micro-período), selecionar aleatoriamente um dos pais, do qual o indivíduo filho herdará o conteúdo no micro-período (Figura 4.9).

O *crossover* de um ponto inicialmente seleciona aleatoriamente um ponto entre o início e o fim das colunas da matriz de seqüência. Selecionado o ponto, um dos indivíduos pais é escolhido aleatoriamente para ser aquele do qual o filho herdará as parcelas anteriores ao ponto de cada linha da matriz. As parcelas posteriores ao ponto serão herdadas do outro pai (Figura 4.10).

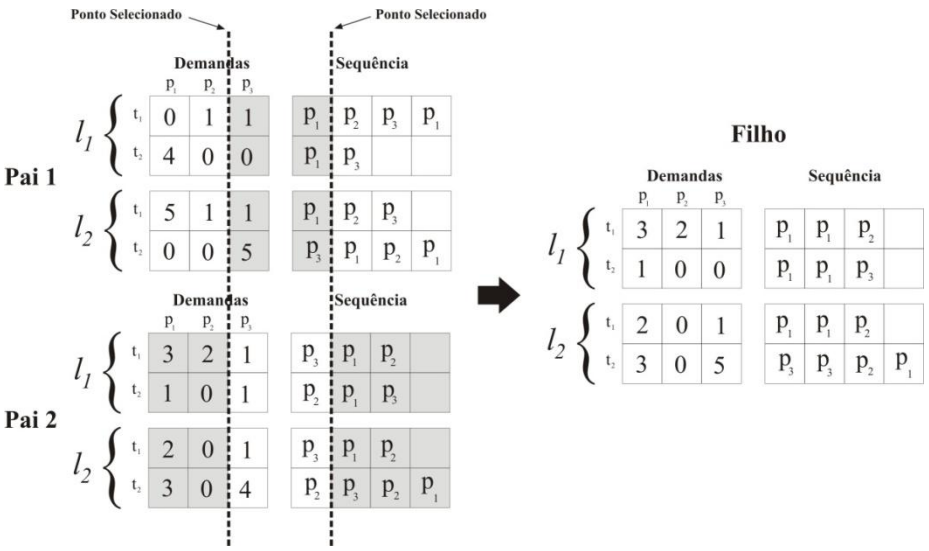

Figura 4.10 – *Crossover* de Um Ponto.

Os operadores de mutação são aplicados para cada indivíduo separadamente alterando sua representação. Vários operadores de mutação foram utilizados nesta aplicação do algoritmo genético, sendo que eles são divididos em operadores de demanda e de seqüência. Ao aplicar uma mutação

em um indivíduo um dos operadores de mutação é selecionado aleatoriamente O primeiro operador de mutação para a matriz de seqüência é o de inserção (figura 4.11).

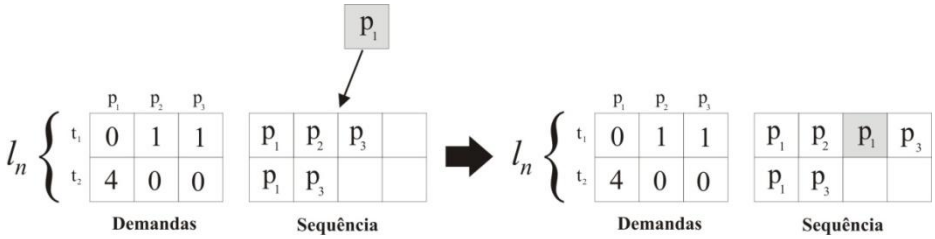

Figura 4.11 – Mutação de Inserção para Seqüência.

Este operador seleciona um produto e uma linha aleatoriamente e insere este produto em uma posição aleatória da seqüência de produção da linha. Esta inserção será realizada somente se o período selecionado não estiver totalmente ocupado, ou seja, apenas se existirem micro-período não utilizados. O segundo operador de seqüência é o de troca (figura 4.12).

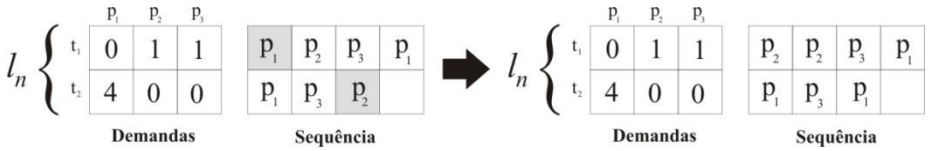

Figura 4.12 – Mutação de Troca para Seqüência.

Este operador troca dois produtos aleatoriamente selecionados em dois micro-períodos distintos. O terceiro operador de mutação para matriz de seqüência é o de remoção (figura 4.13).

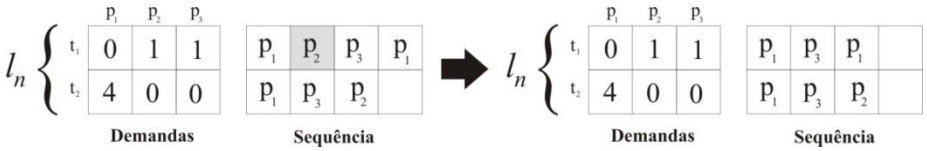

Figura 4.13 – Mutação de Remoção para Seqüência.

Um lote selecionado aleatoriamente é removido. Todos os lotes posteriores a ele no período são deslocados um micro-período para trás, preenchendo o micro-período vago. O quarto operador de mutação para seqüência é o operador de troca de períodos (figura 4.14).

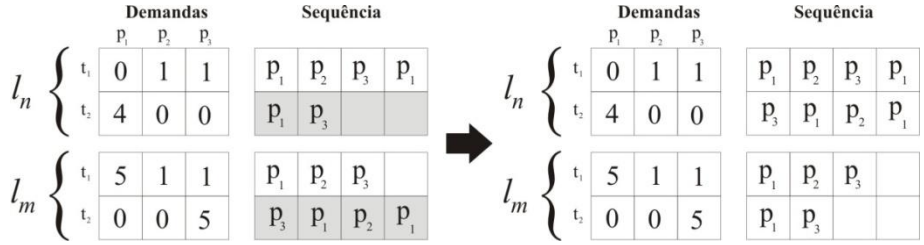

Figura 4.14 – Mutação de Troca de Períodos para Seqüência.

Este operador consiste em selecionar aleatoriamente um período e duas linhas de produção, trocando todas as produções deste período entre as duas linhas selecionadas. O último operador aplicado à matriz de seqüência de produção é o de inversão (figura 4.15).

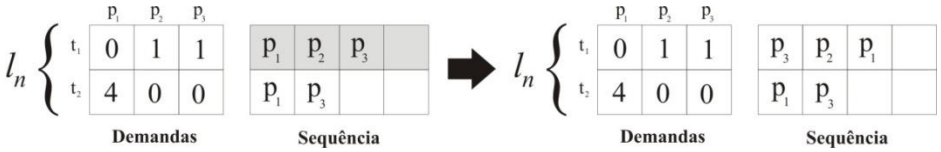

Figura 4.15 – Mutação de Inversão para Seqüência.

Ele consiste em selecionar aleatoriamente uma linha de produção e um período, e inverter a seqüência de produção desta linha neste período. Os operadores de mutação aplicados a demandas são aqueles que atuam sobre a matriz de demanda das linhas de produção de um indivíduo. O primeiro deles é o operador de troca (figura 4.16).

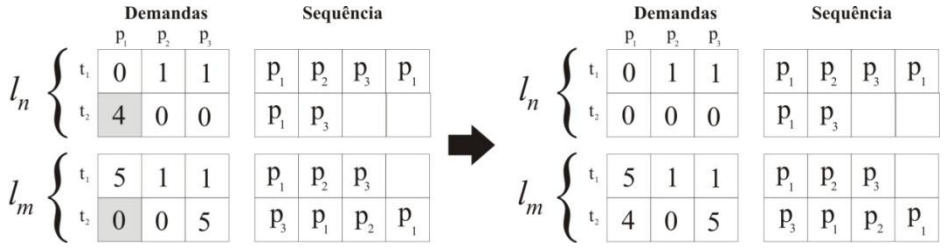

Figura 4.16 – Mutação de Troca para Demanda.

Este operador seleciona aleatoriamente duas linhas, um produto e um período, trocando as demandas deste produto no mesmo período entre as duas linhas. O segundo operador de mutação para matriz de demandas é o de inserção (figura 4.17).

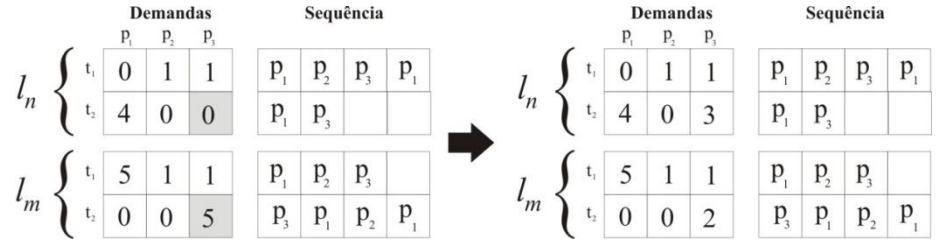

Figura 4.17 – Mutação de Inserção para Demanda.

Este operador seleciona aleatoriamente duas linhas de produção, um produto e um período. Em seguida, subtrai uma parcela da demanda deste produto neste período da linha, e adiciona a parcela subtraída em outra linha.

Esta parcela da demanda está entre zero e o valor total da demanda do produto naquele período. O ultimo operador de mutação para demandas é o operador de remoção. Este operador é similar ao de inserção, porém a parcela de demanda selecionada é sempre o valor máximo. Logo, toda a demanda selecionada é transferida de uma linha para outra (Figura 3.20).

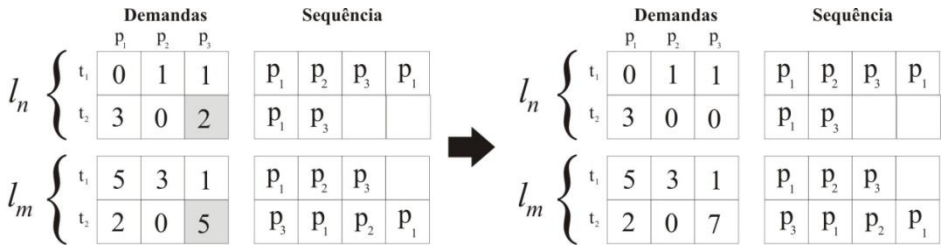

Figura 4.18 – Mutação de Remoção para Demanda.

Alguns operadores, tanto de mutação quanto de *crossover*, podem gerar irregularidades na representação do indivíduo. Duas produções consecutivas do mesmo produto em um mesmo período de uma linha não devem ocorrer em uma matriz de seqüência de produção. Para demandas, a irregularidade ocorre quando o somatório das demandas de cada produto em cada período das linhas na matriz de demanda não corresponde à demanda total da instância. Assim, uma rotina de reparo é executada sobre o indivíduo após a aplicação de um operador. Esta rotina corrige a matriz de seqüência de produção removendo uma das duas alocações consecutivas do produto. Para demandas, a correção é feita somando (ou subtraindo) demandas de uma linha selecionada aleatoriamente. Se esse procedimento não for suficiente, outra linha é selecionada e sofre a adição (ou subtração) de demanda necessária. A rotina se repete até que o somatório das demandas das linhas coincida com a demanda total da instância. A tabela 4.1 apresenta todos os operadores genéticos aplicados a demandas e seqüência de produção.

Seqüência	Demanda
Crossover uniforme	*Crossover* uniforme
Crossover de um ponto	*Crossover* de um ponto
Mutação de troca	Mutação de troca
Mutação de inserção	Mutação de inserção
Mutação de remoção	Mutação de remoção
Mutação de troca de períodos	
Mutação de inversão	

Tabela 4.1 – Operadores Genéticos.

Capítulo 5

Resultados Computacionais

Este capítulo apresenta os resultados obtidos nos testes realizados na resolução do PGDLPP. São definidas e geradas instâncias do problema, a partir das quais são realizados testes para avaliar configurações para o AG. Também é feita uma avaliação de desempenho do AG, comparando-o a outros métodos de resolução.

5.1 Definição das Instâncias

A primeira etapa do processo de avaliação de desempenho dos métodos propostos consiste em estabelecer um conjunto de instâncias a partir das quais serão executados os testes. As instâncias utilizadas neste trabalho estão divididas em dois grupos: instâncias com uma única máquina e instâncias com mais de uma máquina. Todas as instâncias utilizadas foram geradas a partir do algoritmo de geração de instâncias para o PGDLPP, apresentado na seção 3.6.

O conjunto de instâncias com uma única máquina está subdividido em quatro subconjuntos (S1, S2, S3 e S4), contendo dez instâncias cada um deles. Os parâmetros utilizados na geração destas instâncias são os mesmos apresentados por Haase (1996). A tabela 5.1 apresenta os parâmetros utilizados. A coluna "*U* (%)" representa a taxa de utilização da capacidade.

O conjunto de instâncias com máquinas paralelas é dividido em cinco subconjuntos (P1, P2, P3, P4 e P5), contendo cinco instâncias cada um deles. A tabela 5.2 apresenta os parâmetros utilizados para a geração destas instâncias.

	Máquinas	Períodos	Micro-períodos	Produtos	Capacidade	U(%)
S1	1	6	4	4	200	80
S2	1	6	4	4	200	90
S3	1	5	4	4	200	90
S4	1	5	5	5	250	90

Tabela 5.1 – Parâmetros para a geração de instâncias com uma única máquina.

	Máquinas	Períodos	Micro-períodos	Produtos	Capacidade	U(%)
P1	2	6	4	4	200	80
P2	3	6	4	4	200	80
P3	2	6	8	8	400	80
P4	3	6	8	8	400	80
P5	4	6	8	8	400	80

Tabela 5.2 – Parâmetros para a geração de instâncias com uma única máquina.

As instâncias foram divididas em subconjuntos a fim de gerar cenários com diferentes graus de complexidade, avaliando assim o comportamento das heurísticas diante de diferentes situações. Para fins de comparação, todas as instâncias foram codificadas na linguagem AMPL e executadas utilizando um método exato disponível no pacote de *solvers* CPLEX durante uma hora.

5.2 Avaliação das Estruturas Populacionais

Para a avaliação das estruturas populacionais, o AG foi executado dez vezes para cada uma das instâncias, com um tempo limite de 30 minutos em cada execução. Estas execuções foram realizadas para três estruturas populacionais diferentes: ternária, binária e não estruturada. A partir das dez execuções realizadas para cada instância, foi calculada a média dos custos das soluções encontradas e esta foi utilizada para a comparação entre as estruturas. Informações sobre o pacote CPLEX podem ser encontradas em AMPL (2009).

A população utilizada em todos os testes é composta por 40 indivíduos, independente da estrutura populacional utilizada. As taxas dos operadores genéticos utilizadas são as mesmas em todas as execuções. A taxa de mutação utilizada é 0,7 e a taxa de *crossover* é 2,0. O tipo de *crossover* utilizado em todos os testes de estruturas populacionais é o *Crossover* de um ponto. Testes com relação a operadores de recombinação são descritos na seção seguinte.

Os resultados para instâncias com apenas uma máquina são apresentados nas tabelas 5.3 e 5.4. Cada célula da tabela apresenta a média das soluções encontradas para a instância. Caso este valor não seja igual ao resultado obtido pelo método exato (AMPL) na mesma instância, um segundo valor é apresentado na célula. Este segundo valor representa o desvio desta solução em relação à solução obtida pelo método exato, sendo calculado segundo a equação (5.1).

$$Desvio = \frac{100\,(soluçãoAnalisada - soluçãoMetodoExato)}{soluçãoMetodoExato} \qquad (5.1)$$

Assim, quanto menor o desvio, melhor o desempenho do algoritmo analisado em relação ao método exato. Um desvio negativo significa que o método analisado atingiu soluções melhores que o método exato.

Na tabela 5.3, o AG com população binária ou ternária obteve resultados na maioria das instâncias com menos de 1% de desvio em relação ao AMPL. Apenas a estrutura binária apresentou desvio acima de 1% em S1-4 e S2-0. O AG com população não estruturada apresentou desvios acima de 1% em 4 das 20 instâncias avaliadas. Na tabela 5.4, o AG com população estruturada ainda apresentou desvios abaixo de 1% na maioria das instâncias, com soluções em média melhores que o AMPL em S4-7. Por outro lado, o AG com população não estruturada continuou apresentando desvios acima de 1% em 9 das 20 instancias avaliadas.

S1	AMPL	Ternária	Binária	Não Est.	S2	AMPL	Ternária	Binária	Não Est.
S1-0	1658,00	1658,00	1658,00	1702,80 2,70	**S2-0**	1729,00	1746,00 0,98	1747,90 1,09	1778,00 2,83
S1-1	1421,00	1421,00	1421,00	1421,00	**S2-1**	1758,00	1758,00	1758,00	1767,70 0,55
S1-2	1700,00	1700,00	1700,00	1700,00	**S2-2**	2162,00	2162,00	2162,00	2190,50 1,32
S1-3	1459,00	1459,00	1459,00	1459,00	**S2-3**	1747,00	1748,20 0,07	1748,20 0,07	1762,30 0,88
S1-4	1543,00	1551,40 0,54	1565,40 1,45	1548,60 0,36	**S2-4**	1638,00	1638,00	1638,00	1638,00
S1-5	1402,00	1402,00	1402,00	1402,00	**S2-5**	1534,00	1534,00	1534,00	1534,00
S1-6	1477,00	1486,50 0,64	1486,50 0,64	1482,60 0,38	**S2-6**	1533,00	1533,00	1533,00	1533,00
S1-7	1463,00	1463,00	1463,00	1463,00	**S2-7**	1822,00	1833,80 0,65	1837,20 0,83	1855,40 1,83
S1-8	1480,00	1487,00 0,47	1487,00 0,47	1494,30 0,97	**S2-8**	1863,00	1864,70 0,09	1869,80 0,37	1865,80 0,15
S1-9	1626,00	1633,40 0,46	1635,20 0,57	1631,50 0,34	**S2-9**	1751,00	1751,00	1751,00	1763,10 0,69
Desvios Médios S1		**0,21**	**0,31**	**0,47**	**Desvios Médios S2**		**0,18**	**0,24**	**0,83**

Tabela 5.3 – Resultados obtidos para o PGDLPP para cada estrutura populacional em S1 e S2.

S3	AMPL	Ternária	Binária	Não Est.	S4	AMPL	Ternária	Binária	Não Est.
S3-0	1269,00	1.269,00	1.269,00	1.278,90 0,78	**S4-0**	1833,00	1.861,50 1,55	1.858,00 1,36	1.854,70 1,18
S3-1	1491,00	1.491,00	1.491,00	1.492,30 0,09	**S4-1**	1986,00	1.989,60 0,18	1.995,80 0,49	2.014,60 1,44
S3-2	1338,00	1.338,00	1.338,00	1.368,50 2,28	**S4-2**	1825,00	1.827,20 0,12	1.828,70 0,20	1.843,10 0,99
S3-3	1530,00	1.530,00	1.530,00	1.530,00	**S4-3**	1674,00	1.681,80 0,47	1.674,00	1.681,80 0,47
S3-4	1663,00	1.667,80 0,29	1.667,10 0,25	1.666,60 0,22	**S4-4**	1974,00	1.974,00	1.979,90 0,30	1.999,60 1,30
S3-5	1542,00	1.542,00	1.547,70 0,37	1.544,00 0,13	**S4-5**	1994,00	2.011,30 0,87	2.012,20 0,91	2.015,60 1,08
S3-6	1553,00	1.553,00	1.553,00	1.553,00	**S4-6**	1790,00	1.793,30 0,18	1.798,20 0,46	1.843,30 2,98
S3-7	1518,00	1.532,10 0,93	1.547,50 1,94	1.538,00 1,32	**S4-7**	1786,00	1.784,80 -0,07	1.784,80 -0,07	1.805,80 1,11
S3-8	1526,00	1.526,00	1.526,00	1.529,40 0,22	**S4-8**	2150,00	2.195,40 2,11	2.193,60 2,03	2.259,40 5,09
S3-9	1441,00	1.441,00	1.441,00	1.447,20 0,43	**S4-9**	1782,00	1.788,70 0,38	1.788,70 0,38	1.788,50 0,36
Desvios Médios S3		**0,12**	**0,26**	**0,55**	**Desvios Médios S4**		**0,58**	**0,61**	**1,60**

Tabela 5.4 – Resultados obtidos para o PGDLPP para cada estrutura populacional em S3 e S4.

A partir destes resultados foi estabelecido o gráfico da figura 5.1. Este gráfico ilustra os desvios médios dos resultados de cada estrutura populacional em relação a cada grupo de instâncias. Os desvios médios observados revelam uma superioridade de desempenho da estrutura populacional ternária em relação às demais, visto que ela possui desvios mais baixos em todos os grupos de instâncias.

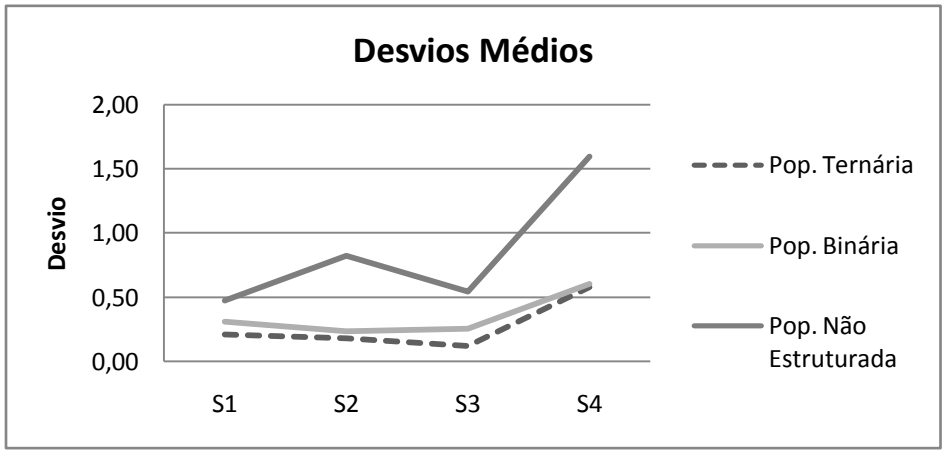

Figura 5.1 – Gráfico de desvios médios das estruturas populacionais para instâncias de máquina simples.

Os resultados obtidos nas execuções em instâncias com mais de uma máquina são apresentados nas tabelas 5.5, 5.6 e 5.7.

P1	AMPL	Ternária	Binária	Não Est.	P2	AMPL	Ternária	Binária	Não Est.
P1-0	1765,55	2218,00 25,63	2281,20 29,21	2340,40 32,56	**P2-0**	1002,84	1713,30 70,85	1698,20 69,34	1766,20 76,12
P1-1	1752,46	2175,30 24,13	2056,10 17,33	2204,40 25,79	**P2-1**	1161,31	1819,70 56,69	1909,90 64,46	2170,00 86,86
P1-2	1953,25	1919,20 -1,74	1885,20 -3,48	2008,30 2,82	**P2-2**	1625,11	2702,40 66,29	2569,60 58,12	2704,50 66,42
P1-3	1598,30	1850,80 15,80	1837,30 14,95	1967,20 23,08	**P2-3**	1544,75	2337,70 51,33	2715,50 75,79	2770,60 79,36
P1-4	1498,54	1903,20 27,00	1861,10 24,19	2048,40 36,69	**P2-4**	1163,31	2022,60 73,87	1974,80 69,76	2113,80 81,71
Desvios Médios P1		**18,16**	**16,44**	**24,19**	**Desvios Médios P2**		**63,81**	**67,49**	**78,09**

Tabela 5.5 – Resultados obtidos para o PGDLPP para cada estrutura populacional em P1 e P2.

P3	AMPL	Ternária	Binária	Não Est.	P4	AMPL	Ternária	Binária	Não Est.
P3-0	5819,75	5.020,50 -13,73	4.857,80 -16,53	5.229,60 -10,14	**P4-0**	5730,19	4.690,80 -18,14	4.668,00 -18,54	4.891,10 -14,64
P3-1	5561,00	4.682,60 -15,80	4.842,90 -12,91	4.965,60 -10,71	**P4-1**	5364,30	4.427,90 -17,46	4.354,40 -18,83	4.789,80 -10,71
P3-2	4371,46	4.165,10 -4,72	4.206,10 -3,78	4.365,40 -0,14	**P4-2**	6375,31	5.449,20 -14,53	5.236,80 -17,86	5.449,80 -14,52
P3-3	5043,78	4.406,50 -12,63	4.772,00 -5,39	4.751,60 -5,79	**P4-3**	7045,37	4.933,50 -29,98	4.852,50 -31,12	5.122,20 -27,30
P3-4	4536,87	4.477,40 -1,31	4.514,80 -0,49	4.718,30 4,00	**P4-4**	6666,00	5.289,60 -20,65	5.235,10 -21,47	5.598,30 -16,02
Desvios Médios P1		**-9,64**	**-7,82**	**-4,56**	**Desvios Médios P4**		**-20,15**	**-21,56**	**-16,64**

Tabela 5.6 – Resultados obtidos para o PGDLPP para cada estrutura populacional em P3 e P4.

P5	AMPL	Ternária	Binária	Não Est.
P5-0	5679,64	4.857,60 -14,47	5.359,50 -5,64	5.570,00 -1,93
P5-1	7713,08	5.252,20 -31,91	5.507,90 -28,59	5.758,00 -25,35
P5-2	7806,11	5.445,90 -30,24	5.434,90 -30,38	5.669,20 -27,37
P5-3	7836,67	5.828,00 -25,63	5.679,90 -27,52	5.944,10 -24,15
P5-4	7130,80	5.554,80 -22,10	5.521,40 -22,57	5.606,10 -21,38
Desvios Médios P5		**-24,87**	**-22,94**	**-20,04**

Tabela 5.7 – Resultados obtidos para o PGDLPP para cada estrutura populacional em P5.

A maioria dos desvios está acima de 10% na tabela 5.5. O AG com população não estruturada apresenta desvios acima de 20% e o AG com população estruturada apresenta desvios abaixo de 20 % na maior parte dos resultados. Por outro lado, o método exato encontra mais dificuldade em determinar boas soluções nas instâncias mais complexas apresentadas nas tabelas 5.6 e 5.7. Assim, o AG consegue melhores resultados que o AMPL, onde o AG com população estruturada continua obtendo melhores soluções que o AG não estruturado.

O gráfico de desvios médios para estruturas populacionais e instâncias com mais de uma máquina é apresentado na figura 5.2.

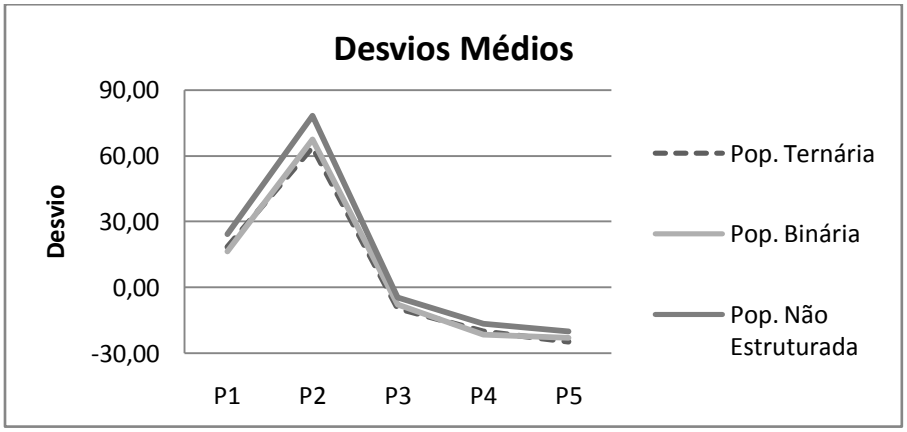

Figura 5.2 – Gráfico de desvios médios das estruturas populacionais para instâncias de máquinas paralelas.

Observa-se que o AG, utilizando populações não estruturadas, novamente apresenta um desempenho inferior. As estruturas binária e ternária obtiveram desvios bastante próximos, visto que suas linhas no gráfico quase se sobrepõem. Para uma análise mais profunda e uma melhor comparação entre estas estruturas é apresentado o gráfico de desvio médio total (figura 5.3). Este

gráfico apresenta a média de todos os desvios médios para cada estrutura populacional. A partir dele é possível notar que a estrutura ternária obteve uma leve superioridade de desempenho em relação à estrutura binária.

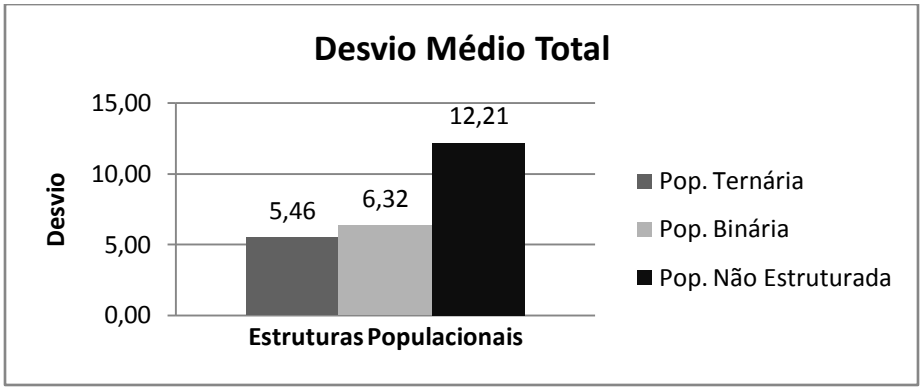

Figura 5.3 – Gráfico de desvio médio total das estruturas populacionais para instâncias de máquinas paralelas.

Os resultados nos permitem concluir que há uma superioridade de desempenho do AG quando utilizada população hierarquicamente estruturada. Tanto a estrutura binária quanto a ternária obtiveram resultados consideravelmente melhores do que aqueles alcançados com população não estruturada. Esta superioridade pode ser justificada pelo fato de que, nas estruturas populacionais hierárquicas, os operadores de recombinação atuam em *clusters* selecionados aleatoriamente, e nem sempre envolvem o melhor indivíduo da população nesta recombinação. Isto permite que a evolução da população seja mais abrangente com relação ao espaço de busca, evitando que a convergência do método seja prejudicada por ótimos locais. De acordo com os resultados, dentre as estruturas populacionais hierárquicas utilizadas, a estrutura ternária é a melhor configuração para esta abordagem do AG quando aplicado ao conjunto de instâncias proposto.

5.3 Avaliação dos Operadores de Recombinação

Na avaliação dos operadores de recombinação (ou *crossover*), o AG foi executado utilizando a estrutura populacional que obteve melhores resultados: estrutura ternária. Foram realizadas dez execuções para cada uma das instâncias, com um tempo limite de 30 minutos em cada execução. Estas execuções foram realizadas para dois operadores: *crossover* de um ponto e *crossover* uniforme. As taxas de incidência dos operadores genéticos e o tamanho da população são os mesmos utilizados nos testes para estruturas populacionais. As tabelas com resultados nesta seção possuem estrutura similar àquelas apresentadas na seção anterior. Os valores obtidos com a utilização de cada operador e seus respectivos desvios com relação aos resultados alcançados pelo método exato são reportados.

Os resultados obtidos nos testes com instâncias de uma única máquina são apresentados nas tabelas 5.8 e 5.9.

S1	AMPL	Um Ponto	Uniforme	S2	AMPL	Um Ponto	Uniforme
S1-0	1658,00	1658,00	1658,00	**S2-0**	1729,00	1746,00 0,98	1737,00 0,46
S1-1	1421,00	1421,00	1421,00	**S2-1**	1758,00	1758,00	1758,00
S1-2	1700,00	1700,00	1700,00	**S2-2**	2162,00	2162,00	2162,00
S1-3	1459,00	1459,00	1459,00	**S2-3**	1747,00	1748,20 0,07	1747,00
S1-4	1543,00	1551,40 0,54	1543,00	**S2-4**	1638,00	1638,00	1638,00
S1-5	1402,00	1402,00	1402,00	**S2-5**	1534,00	1534,00	1534,00
S1-6	1477,00	1486,50 0,64	1480,00 0,20	**S2-6**	1533,00	1533,00	1533,00
S1-7	1463,00	1463,00	1463,00	**S2-7**	1822,00	1833,80 0,65	1822,00
S1-8	1480,00	1487,00 0,47	1487,00 0,47	**S2-8**	1863,00	1864,70 0,09	1863,00
S1-9	1626,00	1633,40 0,46	1628,90 0,18	**S2-9**	1751,00	1751,00	1751,00
Desvios Médios S1		**0,21**	**0,09**	**Desvios Médios S2**		**0,18**	**0,05**

Tabela 5.8 – Resultados obtidos para o PGDLPP para cada operador de recombinação em S1 e S2.

S3	AMPL	Um Ponto	Uniforme	S4	AMPL	Um Ponto	Uniforme
S3-0	1269,00	1.269,00	1.269,00	**S4-0**	1833,00	1.861,50 1,55	1.851,00 0,98
S3-1	1491,00	1.491,00	1.491,00	**S4-1**	1986,00	1.989,60 0,18	1.988,40 0,12
S3-2	1338,00	1.338,00	1.338,00	**S4-2**	1825,00	1.827,20 0,12	1.824,70 -0,02
S3-3	1530,00	1.530,00	1.530,00	**S4-3**	1674,00	1.681,80 0,47	1.674,00
S3-4	1663,00	1.667,80 0,29	1.663,00	**S4-4**	1974,00	1.974,00	1.974,00
S3-5	1542,00	1.542,00	1.542,00	**S4-5**	1994,00	2.011,30 0,87	2.000,40 0,32
S3-6	1553,00	1.553,00	1.553,00	**S4-6**	1790,00	1.793,30 0,18	1.790,00
S3-7	1518,00	1.532,10 0,93	1.522,00 0,26	**S4-7**	1786,00	1.784,80 -0,07	1.777,00 -0,50
S3-8	1526,00	1.526,00	1.526,00	**S4-8**	2150,00	2.195,40 2,11	2.182,20 1,50
S3-9	1441,00	1.441,00	1.441,00	**S4-9**	1782,00	1.788,70 0,38	1.788,00 0,34
Desvios Médios S3		**0,12**	**0,03**	**Desvios Médios S4**		**0,58**	**0,27**

Tabela 5.9 – Resultados obtidos para o PGDLPP para cada operador de recombinação em S3 e S4.

O uso do AG com *crossover* uniforme ou *crossover* de um ponto apresentou pouca diferença. As duas abordagens continuam obtendo desvios abaixo de 1% na maioria das instancias. A figura 5.4 apresenta um gráfico com os desvios médios para estes grupos de instâncias.

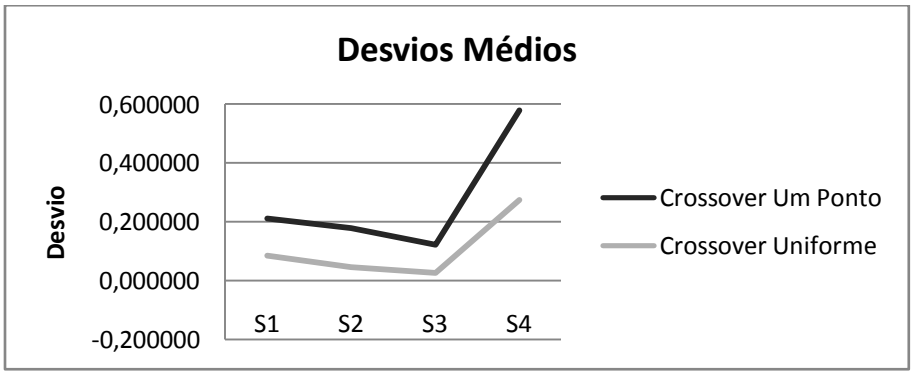

Figura 5.4 – Gráfico de desvios médios dos operadores de recombinação para instâncias de uma única máquina.

A partir destes resultados é possível constatar um melhor desempenho do AG utilizando *crossover* uniforme para instâncias com uma única máquina, onde o *crossover* de um ponto não superou o uniforme em nenhum dos grupos de instancias.

Os resultados para instâncias com máquinas paralelas são apresentadas nas tabelas 5.10, 5.11 e 5.12.

P1	AMPL	Um Ponto	Uniforme	P2	AMPL	Um ponto	Uniforme
P1-0	1765,55	2218,00 25,63	2047,60 15,98	**P2-0**	1002,84	1713,30 70,85	1447,30 44,32
P1-1	1752,46	2175,30 24,13	2022,80 15,43	**P2-1**	1161,31	1819,70 56,69	1718,80 48,00
P1-2	1953,25	1919,20 -1,74	1884,90 -3,50	**P2-2**	1625,11	2702,40 66,29	2401,70 47,79
P1-3	1598,30	1850,80 15,80	1840,20 15,13	**P2-3**	1544,75	2337,70 51,33	2170,40 40,50
P1-4	1498,54	1903,20 27,00	1837,60 22,63	**P2-4**	1163,31	2022,60 73,87	1839,80 58,15
Desvios Médios P1		**18,16**	**13,13**	**Desvios Médios P2**		**63,81**	**47,75**

Tabela 5.10 – Resultados obtidos para o PGDLPP para cada operador de recombinação em P1 e P2.

P3	AMPL	Um Ponto	Uniforme	P4	AMPL	Um Ponto	Uniforme
P3-0	5819,75	5.020,50 -13,73	5.021,20 -13,72	**P4-0**	5730,19	4.690,80 -18,14	4.875,30 -14,92
P3-1	5561,00	4.682,60 -15,80	4.768,40 -14,25	**P4-1**	5364,30	4.427,90 -17,46	4.736,40 -11,71
P3-2	4371,46	4.165,10 -4,72	4.165,10 -4,72	**P4-2**	6375,31	5.449,20 -14,53	5.411,30 -15,12
P3-3	5043,78	4.406,50 -12,63	4.406,50 -12,63	**P4-3**	7045,37	4.933,50 -29,98	5.020,10 -28,75
P3-4	4536,87	4.477,40 -1,31	4.636,50 2,20	**P4-4**	6666,00	5.289,60 -20,65	5.397,50 -19,03
Desvios Médios P1		**-9,64**	**-8,63**	**Desvios Médios P4**		**-20,15**	**-17,90**

Tabela 5.11 – Resultados obtidos para o PGDLPP para cada operador de recombinação em P3 e P4.

P5	AMPL	Um Ponto	Uniforme
P5-0	5679,64	4.857,60 -14,47	6.196,30 9,10
P5-1	7713,08	5.252,20 -31,91	6.035,30 -21,75
P5-2	7806,11	5.445,90 -30,24	6.124,30 -21,54
P5-3	7836,67	5.828,00 -25,63	5.995,10 -23,50
P5-4	7130,80	5.554,80 -22,10	5.964,75 -16,35
Desvios Médios P5		**-24,87**	**-14,81**

Tabela 5.12 – Resultados obtidos para o PGDLPP para cada operador de recombinação em P5.

Na tabela 5.10, o uso do *crossover* uniforme apresenta resultados melhores que o *crossover* de um ponto. Nas tabelas 5.11 e 5.12, esse comportamento se repete no conjunto P4, mas o *crossover* de um ponto é melhor em média no conjunto P3 e P5.

O gráfico representando os desvios médios dos operadores de recombinação para as instâncias com mais de uma máquina é apresentado na figura 5.5.

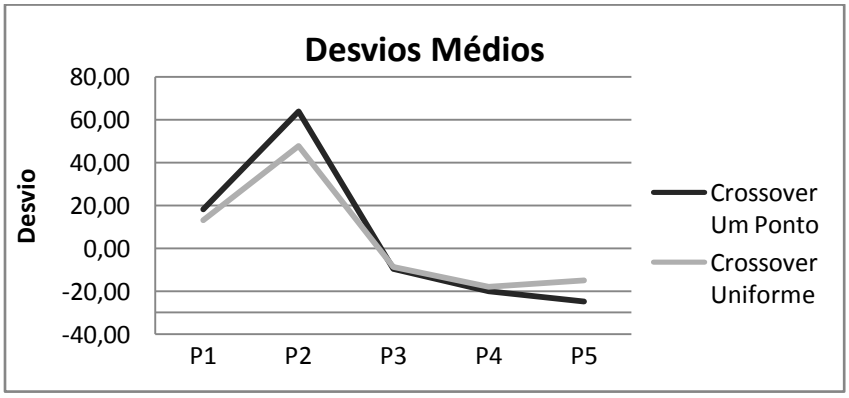

Figura 5.5 – Gráfico de desvios médios dos operadores de recombinação para instâncias com máquinas paralelas.

Apesar do melhor desempenho do *crossover* uniforme nas instâncias de uma única máquina, esta superioridade não é tão evidente para as instâncias com mais de uma máquina. O *crossover* uniforme é superior nos primeiros grupos de instâncias (P1 e P2). Porém, nos grupos que contém instâncias mais complexas (P4 e P5) o *crossover* de um ponto possui melhor desempenho. Analisando estes resultados, juntamente com as curvas apresentadas no gráfico da figura 5.5, é possível constatar que o *crossover* de um ponto tende a alcançar melhores resultados em instâncias mais complexas. Para uma melhor análise de qual operador de recombinação possui um melhor resultado geral, é apresentado o gráfico de desvio médio total, contendo a média de todos os desvios médios para cada operador (figura 5.6). Este gráfico mostra que, na média, o *crossover* uniforme possui melhor desempenho para instâncias com máquinas paralelas.

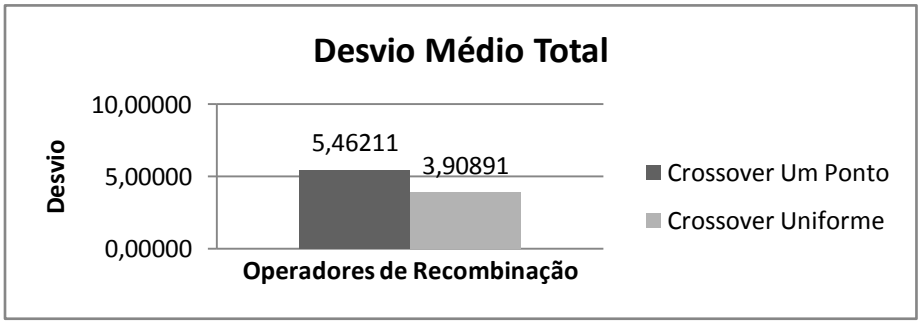

Figura 5.6 – Gráfico de desvio médio total dos operadores de recombinação para instâncias com máquinas paralelas.

Apesar de o crossover uniforme obter melhores resultados em geral, o *crossover* de um ponto também é utilizado nos demais testes. Isto se justifica pelo seu desempenho nas instâncias mais complexas.

5.4 Comparação com Outros Métodos

Além do método exato, nesta seção são apresentados os resultados de teste realizados com outras heurísticas. São elas:

- *Simulated Annealing* (SA), apresentada por Kirkpatrick *et AL* (1983);
- Busca Tabu (BT), apresentada por Glover (1997);
- *Threshold Accepting* (TA), proposta por Fleischmann e Meyr (1997).

O AG utilizado nos testes possui tamanho de população e taxas dos operadores genéticos iguais aos testes apresentados nas seções anteriores. A estrutura populacional utilizada é a estrutura hierárquica ternária. Os dois operadores de *crossover* são utilizados separadamente.

As tabelas 5.13, 5.14, 5.15 e 5.16 apresentam os resultados dos testes realizados para instâncias com uma única máquina. Nestes testes, para cada instância os métodos foram executados dez vezes durante 30 minutos.

S1	AMPL	TA	SA	BT	AG Um ponto	AG Uniforme
S1-0	1658,00	1737,50 4,79	1658,00	1658,00	1658,00	1658,00
S1-1	1421,00	1533,50 7,92	1421,00	1421,00	1421,00	1421,00
S1-2	1700,00	1764,40 3,79	1700,00	1700,00	1700,00	1700,00
S1-3	1459,00	1583,90 8,56	1459,00	1459,00	1459,00	1459,00
S1-4	1543,00	1643,10 6,49	1543,00	1543,00	1551,40 0,54	1543,00
S1-5	1402,00	1527,40 8,94	1402,00	1402,00	1402,00	1402,00
S1-6	1477,00	1567,00 6,09	1480,00 0,20	1480,00 0,20	1486,50 0,64	1480,00 0,20
S1-7	1463,00	1517,80 3,75	1463,00	1463,00	1463,00	1463,00
S1-8	1480,00	1575,60 6,46	1487,00 0,47	1487,00 0,47	1487,00 0,47	1487,00 0,47
S1-9	1626,00	1714,20 5,42	1628,00 0,12	1628,00 0,12	1633,40 0,46	1628,90 0,18
Desvios Médios S1		**6,2215**	**0,0799**	**0,0799**	**0,2116**	**0,0854**

Tabela 5.13 – Resultados obtidos para o PGDLPP com os diversos métodos em S1.

S2	AMPL	TA	SA	BT	AG Um ponto	AG Uniforme
S2-0	1729,00	1948,60 12,70	1737,00 0,46	1737,00 0,46	1746,00 0,98	1737,00 0,46
S2-1	1758,00	1839,30 4,62	1758,00	1758,00	1758,00	1758,00
S2-2	2162,00	2243,20 3,76	2162,00	2162,00	2162,00	2162,00
S2-3	1747,00	1865,00 6,75	1747,00	1747,00	1748,20 0,07	1747,00
S2-4	1638,00	1800,30 9,91	1638,00	1638,00	1638,00	1638,00
S2-5	1534,00	1689,90 10,16	1534,00	1534,00	1534,00	1534,00
S2-6	1533,00	1568,30 2,30	1533,00	1533,00	1533,00	1533,00
S2-7	1822,00	1982,20 8,79	1822,00	1822,00	1833,80 0,65	1822,00
S2-8	1863,00	1910,70 2,56	1863,00	1863,00	1864,70 0,09	1863,00
S2-9	1751,00	1847,60 5,52	1751,00	1751,00	1751,00	1751,00
Desvios Médios S2		**6,7080**	**0,0463**	**0,0463**	**0,1791**	**0,0463**

Tabela 5.14 – Resultados obtidos para o PGDLPP com os diversos métodos em S2.

S3	AMPL	TA	SA	BT	AG Um ponto	AG Uniforme
S3-0	1269,00	1.357,70 6,99	1.269,00	1.269,00	1.269,00	1.269,00
S3-1	1491,00	1.573,00 5,50	1.491,00	1.491,00	1.491,00	1.491,00
S3-2	1338,00	1.528,00 14,20	1.338,00	1.338,00	1.338,00	1.338,00
S3-3	1530,00	1.649,80 7,83	1.530,00	1.530,00	1.530,00	1.530,00
S3-4	1663,00	1.760,20 5,84	1.663,00	1.663,00	1.667,80 0,29	1.663,00
S3-5	1542,00	1.612,40 4,57	1.542,00	1.542,00	1.542,00	1.542,00
S3-6	1553,00	1.616,00 4,06	1.553,00	1.553,00	1.553,00	1.553,00
S3-7	1518,00	1.677,30 10,49	1.522,00 0,26	1.522,00 0,26	1.532,10 0,93	1.522,00 0,26
S3-8	1526,00	1.595,60 4,56	1.526,00	1.526,00	1.526,00	1.526,00
S3-9	1441,00	1.563,50 8,50	1.441,00	1.441,00	1.441,00	1.441,00
Desvios Médios S3		**7,2543**	**0,0264**	**0,0264**	**0,1217**	**0,0264**

Tabela 5.15 – Resultados obtidos para o PGDLPP com os diversos métodos em S3.

S4	AMPL	TA	SA	BT	AG Um ponto	AG Uniforme
S4-0	1833,00	1.936,30 5,64	1.837,00 0,22	1.837,00 0,22	1.861,50 1,55	1.851,00 0,98
S4-1	1986,00	2.164,00 8,96	1.986,00	1.986,00	1.989,60 0,18	1.988,40 0,12
S4-2	1825,00	1.941,30 6,37	1.823,00 -0,11	1.823,00 -0,11	1.827,20 0,12	1.824,70 -0,02
S4-3	1674,00	1.828,30 9,22	1.674,00	1.674,00	1.681,80 0,47	1.674,00
S4-4	1974,00	2.096,20 6,19	1.974,00	1.974,00	1.974,00	1.974,00
S4-5	1994,00	2.118,00 6,22	1.994,00	1.995,50 0,08	2.011,30 0,87	2.000,40 0,32
S4-6	1790,00	2.191,10 22,41	1.790,00	1.790,00	1.793,30 0,18	1.790,00
S4-7	1786,00	1.906,10 6,72	1.777,00 -0,50	1.777,00 -0,50	1.784,80 -0,07	1.777,00 -0,50
S4-8	2150,00	2.433,20 13,17	2.175,00 1,16	2.175,00 1,16	2.195,40 2,11	2.182,20 1,50
S4-9	1782,00	1.920,40 7,77	1.788,00 0,34	1.788,00 0,34	1.788,70 0,38	1.788,00 0,34
Desvios Médios S4		**9,2668**	**0,1104**	**0,1179**	**0,5795**	**0,2738**

Tabela 5.16 – Resultados obtidos para o PGDLPP com os diversos métodos em S4.

Nas tabelas 5.12, 5.13 e 5.14, não há uma diferença de desempenho significativa entre BT, SA e AG uniforme. O AG com *crossover* de um ponto apresenta desvios abaixo de 1%, mas ainda superiores aqueles retornados pela BT, SA e AG uniforme. O método TA apresenta resultados médios piores que todas as demais abordagens. O gráfico representando os desvios médios para os métodos nestes grupos de instâncias é apresentado na figura 5.7. O método *Threshold Accepting* não foi incluído no gráfico já que obteve um desempenho muito abaixo dos demais métodos, com desvios muito maiores.

O AG utilizando *crossover* de um ponto, como observado em testes anteriores, obtém desvios mais altos para estes grupos de instâncias. O AG utilizando crossover uniforme obteve resultados praticamente iguais a Busca Tabu e o *Simulated Annealing* nos grupos S1, S2 e S3. Porém, no grupo S4 estas

heurísticas alcançaram resultados superiores ao AG. Para uma análise mais detalhada, é apresentado na figura 5.8 o gráfico do desvio médio total para cada método. Neste gráfico é possível observar a sutil vantagem do *Simulated Annealing* sobre a Busca Tabu.

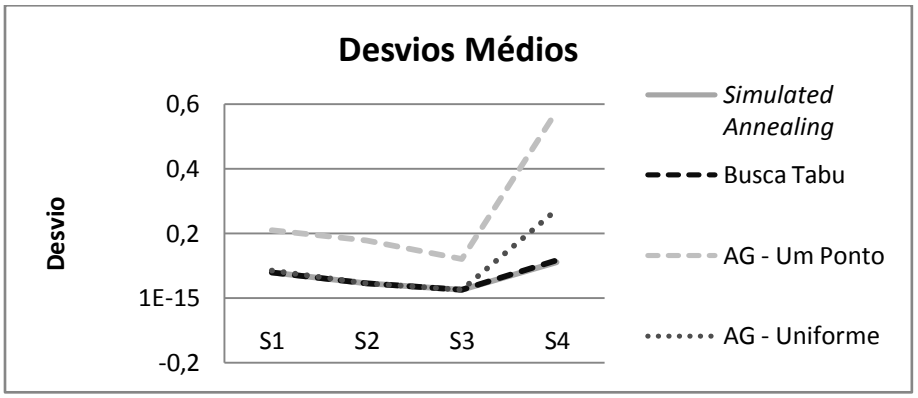

Figura 5.7 – Gráfico de desvios médios dos diversos métodos para instâncias com uma única máquina.

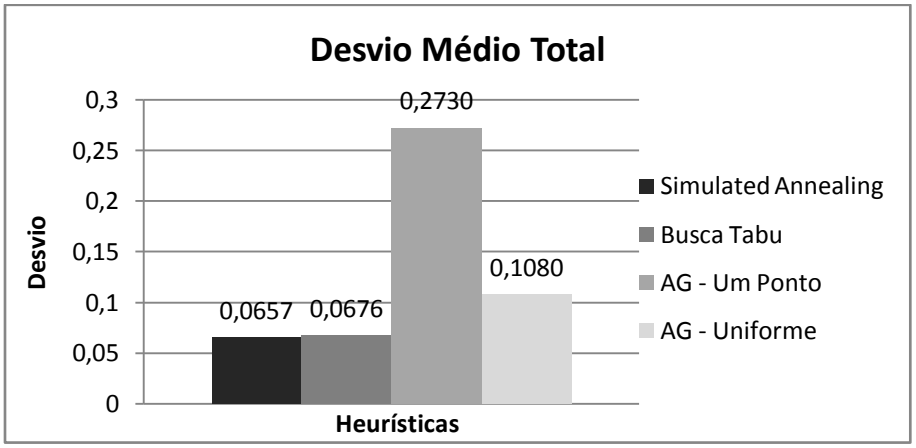

Figura 5.8 – Gráfico de desvio médio total dos métodos para instâncias com uma única máquina.

As tabelas 5.17, 5.18, 5.19, 5.20 e 5.21 apresentam os resultados dos testes realizados para instâncias com máquinas paralelas. Nestes testes, para cada instância os métodos foram executados dez vezes, durante 1 hora em cada execução. Não foram realizados testes com o método *Threshold Accepting* para este grupo de instâncias.

P1	AMPL	BT	SA	AG Um ponto	AG Uniforme
P1-0	1765,547	1.804,60 2,21	2.007,42 13,70	2.166,60 22,72	2.069,10 17,19
P1-1	1752,462	1.805,91 3,05	1.811,44 3,37	2.123,30 21,16	2.034,90 16,12
P1-2	1598,304	1.630,04 1,99	1.746,05 9,24	1.846,70 16,89	1.786,90 11,35
P1-3	1498,543	1.633,95 9,04	1663,396 11,00	1867,4 23,23	1.759,80 19,24
P1-4	1493,318	1.642,65 10,00	1760,098 17,86	1868,3 25,05	1.779,70 17,84
Desvios Médios P1		**5,2567**	**11,0350**	**21,8105**	**16,3493**

Tabela 5.17 – Resultados obtidos para o PGDLPP com os diversos métodos em P1.

P2	AMPL	BT	SA	AG Um ponto	AG Uniforme
P2-0	1002,838	1.315,23 31,15	1.518,71 51,44	1.669,60 66,49	1.408,40 40,44
P2-1	1161,314	1.661,19 43,04	1.982,60 70,72	1.793,40 54,43	1.662,10 43,12
P2-2	1625,106	1.905,95 17,28	2.299,04 41,47	2.601,40 60,08	2.322,90 42,94
P2-3	1544,754	1.867,02 20,86	2056,289 33,11	2276,9 47,40	2.212,30 43,21
P2-4	1163,305	1.601,98 37,71	1938,832 66,67	1932,9 66,16	1.719,50 47,81
Desvios Médios P2		**30,0096**	**52,6824**	**58,9086**	**43,5055**

Tabela 5.18 – Resultados obtidos para o PGDLPP com os diversos métodos em P2.

P3	AMPL	BT	SA	AG Um ponto	AG Uniforme
P3-0	5819,752	4.785,40 -17,77	4.979,56 -14,44	4.869,10 -16,33	4.763,50 -18,15
P3-1	5560,997	4.706,85 -15,36	4.798,83 -13,71	4.653,40 -16,32	4.678,00 -15,88
P3-2	4371,458	4.221,44 -3,43	4.341,34 -0,69	4.052,30 -7,30	4.052,30 -7,30
P3-3	5043,776	4.480,20 -11,17	4753,419 -5,76	4380,5 -13,15	4.380,50 -13,15
P3-4	4536,871	4.636,19 2,19	4945,135 9,00	4411,8 -2,76	4.397,62 -3,07
Desvios Médios P3		**-9,1098**	**-5,1178**	**-11,1728**	**-11,5097**

Tabela 5.19 – Resultados obtidos para o PGDLPP com os diversos métodos em P3.

P4	AMPL	BT	SA	AG Um ponto	AG Uniforme
P4-0	5730,186	5.142,95 -10,25	5.848,29 2,06	4.711,40 -17,78	4.752,60 -17,06
P4-1	5364,301	5.036,63 -6,11	5.888,31 9,77	4.913,70 -8,40	4.678,20 -12,79
P4-2	6375,307	5.593,09 -12,27	5.965,04 -6,44	5.183,60 -18,69	5.169,20 -18,92
P4-3	7045,372	5.326,12 -24,40	6144,673 -12,78	5326,4 -24,40	5.132,60 -27,15
P4-4	6666,003	5.417,91 -18,72	5876,616 -11,84	5552,1 -16,71	5.331,10 -20,03
Desvios Médios P4		**-14,3504**	**-3,8464**	**-17,1961**	**-19,1887**

Tabela 5.20 – Resultados obtidos para o PGDLPP com os diversos métodos em P4.

P5	AMPL	BT	AS	AG Um ponto	AG Uniforme
P5-0	5679,644	6.154,35 8,36	7.691,08 35,41	4.711,40 -17,05	5.856,40 3,11
P5-1	7713,078	6.206,19 -19,54	7.547,76 -2,14	4.913,70 -36,29	5.774,20 -25,14
P5-2	7806,113	6.122,44 -21,57	7.327,65 -6,13	5.183,60 -33,60	5.738,40 -26,49
P5-3	7836,667	6.110,87 -22,02	7262,579 -7,33	5552,1 -29,15	5.744,40 -26,70
P5-4	7130,804	5.846,67 -18,01	7220,466 1,26	5530,833 -22,44	5.608,50 -21,35
Desvios Médios P5		**-14,5556**	**4,2148**	**-27,7054**	**-19,3121**

Tabela 5.21 – Resultados obtidos para o PGDLPP com os diversos métodos em P5.

A busca tabu (BT) destacou-se em relação aos demais métodos nas instâncias dos grupos P1 e P2 (tabelas 5.17 e 5.18). Nas instâncias mais complexas, o AG com *crossover* uniforme e *crossover* de um ponto passam a obter soluções melhores que todas as demais abordagens (tabelas 5.19, 5.20 e 5.11). A figura 5.9 apresenta o gráfico de desvios médios para cada um dos métodos nos grupos de instâncias com máquinas paralelas.

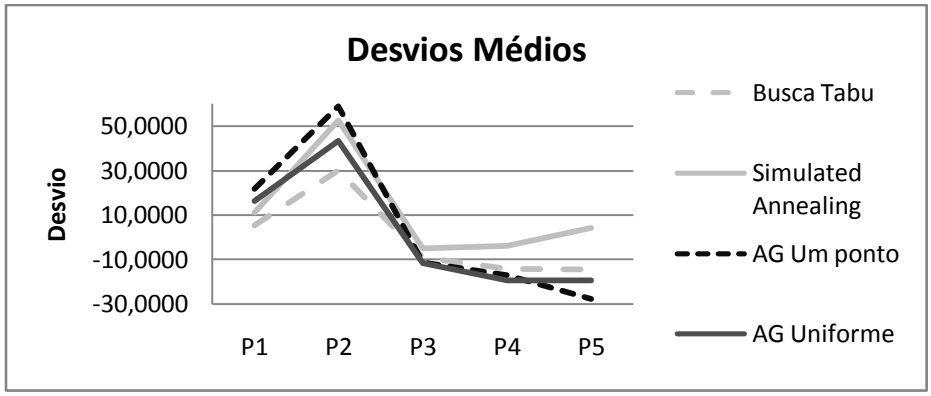

Figura 5.9 – Gráfico de desvios médios dos diversos métodos para instâncias com máquinas paralelas.

Estes resultados mostram que para instâncias mais simples, os métodos *Simulated Annealing* e Busca Tabu apresentam melhor desempenho que o AG. A partir do grupo P3, as duas abordagens de AG passam a alcançar melhores resultados que os outros métodos. Isto mostra que o AG é um método bastante apropriado para instâncias mais complexas. A busca tabu mostra-se um método bastante eficiente para instâncias menos complexas. No geral, o *Simulated Annealing* não apresenta grande eficiência quando comparado a outros métodos. Para uma analise mais geral sobre o comportamento dos métodos é apresentado o gráfico de desvio médio total (figura 5.10).

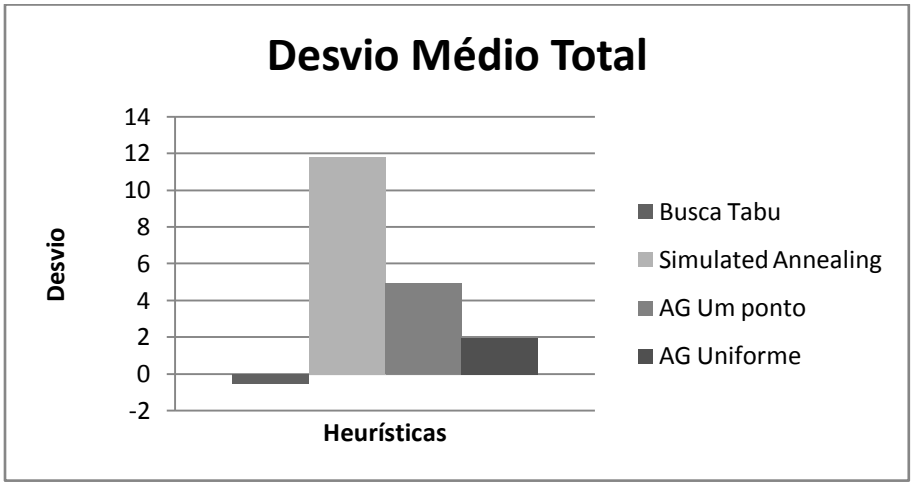

Figura 5.10 – Gráfico de desvio médio total para todos os métodos para instâncias com máquinas paralelas.

Apesar da busca Tabu obter um desvio médio total menor que os dos demais métodos, ela não se apresenta drasticamente melhor que outros métodos. Na verdade, este método foi realmente superior em apenas 2 grupos de instâncias, onde obteve uma diferença considerável de desvio, impactando na média total.

Capítulo 6

Conclusão

O principal objetivo deste trabalho é o estudo e proposta de uma abordagem de Algoritmo Genético capaz de resolver o PGDLPP. Os objetivos propostos envolvem o estudo do problema e a apresentação de um novo modelo matemático para ele, contemplando cenários com várias máquinas trabalhando em paralelo. Também foram geradas instâncias do PGDLPP e, a partir destas, o teste de desempenho do AG foi realizado. Os resultados obtidos pelo AG foram comparados em relação a métodos exatos e heurísticos.

Os modelos matemáticos propostos foram codificados utilizando a linguagem de modelagem matemática AMPL. Isso possibilitou a realização de testes utilizando um método de resolução exato disponível no solver CPLEX. Para a obtenção de instâncias, foi proposto e implementado um gerador de instâncias para o problema. A partir destas instâncias, foram realizados vários testes em busca de uma melhor configuração do AG.

Inicialmente foram avaliadas diferentes estruturas populacionais para o AG: população não-estruturada, estrutura em árvore binária e estrutura em árvore ternária. Nestes testes, o AG mostrou desempenho superior quando utilizou estrutura populacional ternária. Também foram realizados testes para avaliar dois tipos de operadores de recombinação: *crossover* uniforme e *crossover* de um ponto. Esse melhor desempenho pode ser explicado pelos operadores de *crossover* aplicados em clusters selecionados aleatoriamente. Um nó líder com melhor valor de *fitness* é sempre selecionado e recombinado com um dos seus seguidores. Isso não ocorre na população não-estruturada. Nestes testes, o *crossover* uniforme demonstrou melhor desempenho. Porém a

utilização do *crossover* de um ponto não foi descartada já que ele também obteve resultados satisfatórios em instâncias mais complexas. Em seguida foram realizadas comparações entre o desempenho do AG e outros métodos de resolução.

Os resultados apresentados mostram que o AG é um algoritmo bastante eficiente na resolução do PGDLPP, principalmente em instâncias mais complexas. O AG destacou-se em relação aos demais métodos principalmente em instâncias de alta complexidade envolvendo máquinas paralelas. Isto aumenta a viabilidade de sua aplicação em problemas reais, pois estes normalmente apresentam um alto grau de complexidade.

Trabalhos futuros poderão estudar modelos que tempos de troca de configuração da máquina, levando a proposição de um gerador de instancias para esse contexto. Uma nova representação para o indivíduo também poderia ser desenvolvida, considerando seqüência e quantidades de produtos. Por último, outros tipos de operadores de recombinação podem ser avaliados.

Capítulo 7

Referências Bibliográficas

AMPL. 2009. Disponível em www.ampl.com. Acesso em 15 fevereiro 2009.

ANTHONY, R. N. (1965). **Planning and Control Systems: A Framework for Analysis**. Boston: Harvard Business School Press, 1965.

ARENALES, M., ARMENTANO, V., MORABITO, R., YANASSE, H. **Pesquisa Operacional**. Editora Campus, Rio de Janeiro, 2007. 531p.

BÃCK, T.D.B.FOGEL., and MICHALEWICZ, T., editors. **Evolutionary Computation1, Basic Algorithms and Operators.** Institute of Physics Publishing, 2000a.

BERRETTA, Regina Esther **Heurísticas para otimização do planejamento da produção em sistemas MPR.** Tese de Doutorado, Campinas, SP, UNICAMP, 1997.

BITRAN, G. R., YANASSE, H. H. **Computational Complexity of Capacitated Lot Size Problem.** Management Science, n. 28, p. 1178-1186, 1982.

DARWIN , C. **The Origin of Species**, John Murray, 1859 (Penguin Classics, 1985).

DAVIS L, STEENSTRUP M. **Genetic algorithms and simulated annealing: an overview.** In: Davis L, editor. Genetic algorithms and simulated annealing. San Mateo: Morgan Keufman Publishers; 1987. p. 1 – 11.

DAVIS, S.G., **Shedulling economic lost size production runs**. Management Science, 36(8), 1990.

FERREIRA, D., MORABITO, R., RANGEL, S. **Solution approaches for soft drink integrated production lot sizing and scheduling problem**. European Journal of Operational Research. Available on line. doi:10.1016/j.ejor.2008.03.035, 2007.

FLEISCHMANN, B., MEYER, H. **The general lotsizing and scheduling problem.** Department for Production and Logistics, University of Augsburg, Universiffitsstr. 16, Augsburg, Germany, 1997.

FOGEL, D.B. **An Introduction to Simulated Evolutionary Computation**. *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

GLOVER, F., LAGUNA, M., **Tabu Search**, Kluwer, Norwell, MA, 1997.

GOLDBERG, D.E. **Genetic Algorithms in search, optimization, and machine learning.** Addison Wesley, 1989.

HAASE, K. **Capacitated lot-sizing with sequence dependent setup costs**. Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstrasse 40, D-24118 Kiel, Germany, 1995.

HOLLAND, J.H. **Adaptation in natural and artificial systems**, The University of Michigan Press, 1975. 211 p.

JANS, R., DEGRAEVE, Z. **Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches**. European Journal of Operational Research, vol. 177, p. 1855-1875, 2007.

KARIMI, B.; GHOMI, S. M. T. F.; WILSON, J. M., **The capacitated lot sizing problem: a review of models and algorithms**. OMEGA, v. 31, p. 365–378, 2003.

KIRKPATRICK, S., GELATT, C. D., VECCHI, M.P., **Optimization by Simulated Annealing**, Science, 220 (4598), 671-680, 1983.

LINDEN, Ricardo. **Algoritmos genéticos: uma importante ferramenta da inteligência computacional**. Rio de Janeiro: Brasport, 2006. 428 p.

MAYR , E. TOWARD, **A New Philosophy of Biology: Observations of an Evolutionist**, Belknap Press, 1988. 575 p.

MENDES, A.S. **O Framework NP-Opt e suas Aplicações a Problemas de Otimização**. PhD thesis, Universidade Estadual de Campinas, 2003.

MENDES, Jorge José de Magalhães e GONÇALVES, José Fernando. **Um algoritmo genético para o problema do sequenciamento de projectos com recursos limitados**. dez. 2003, vol.23, no.2, p.179-195. ISSN 0874-5161.

MICHALEWICZ, Z. "**Genetic algorithms + Data Structures = Evolution Programs**", 3rd edition, Springer-Verlag, 1996.

NASCIMENTO, M. C. V. **Uma heurística GRASP para o problema de dimensionamento de lotes com múltiplas plantas**. 2007. 66 f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo.

RAMALHO, M. A. P.; SANTOS, J. B.; PINTO, C. A. B. P.. **Genética na agropecuária**. 3. ed. rev. Lavras: UFLA, 2004. 472 p.

TOLEDO, C.F.M. **Problema Conjunto de Dimensionamento de Lotes e Programação da Produção.** Tese de Doutorado, Campinas, SP, UNICAMP, 2005.

TOLEDO, C. F. M., FRANÇA, P. M., ROSA, K. A. **Evaluating Genetic Algorithms with Different Population Structures on a Lot Sizing and Scheduling Problem.** In: 23rd Annual ACM Symposium on Applied Computing, Fortaleza, p. 1777-1781, 2008a.

TOLEDO, C. F. M., FRANÇA, P. M., MORABITO, R., KIMMS, A. **Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem**. International Journal of Production Research, p. 1-23, 2008b.

TOLEDO, C. F. M., FRANÇA, P. M., FERREIRA, J. E. **Meta-Heuristic Approaches for a Soft Drink Industry Problem**. In: 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Alemanha, p. 1384-1391, 2008c.

TOLEDO, Frankilina Maria Bragion de **Dimensionamento de Lotes em Máquinas Paralelas.** Tese de Doutorado, Campinas, SP, UNICAMP 1998.

VON ZUBEN, F.J. **Computação Evolutiva: Uma Abordagem Pragmática**, DCA/FEEC/UNICAMP disponível para acesso em 08/01/2008 endereço em ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tutorial/tutorialEC.pdf.

ZACARELLI, S. B. **Programação e Controle da Produção**. São Paulo: Pioneira, 1979. 292 p.

WARNER, H. M. e T. M. WHITIN, **Dynamic version of the economic lot size model**. Management Science 5, 89-96. 1958.