UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

TESE
apresentada à UTFPR para
obtenção do título de

DOUTOR EM CIÊNCIAS

por

ANGELA OLANDOSKI BARBOZA

SIMULAÇÃO E TÉCNICAS DA COMPUTAÇÃO EVOLUCIONÁRIA APLICADAS A PROBLEMAS DE PROGRAMAÇÃO LINEAR INTEIRA MISTA

Banca Examinadora:

Presidente e Orientador:

Profº Dr. Flavio Neves Junior	UTFPR

Examinadores:

Profª Dra. Elizabeth Ferreira Gouvêa Goldbarg	UFRN
Profª Dra. Lúcia Valéria Ramos de Arruda	UTFPR
Profº Dr. Marco César Goldbarg	UFRN
Profª Dra. Maria Teresinha Arns Steiner	UFPR
Profº Dr. Nélio Domingues Pizzolato	PUC-RJ

Curitiba, novembro de 2005

ANGELA OLANDOSKI BARBOZA

SIMULAÇÃO E TÉCNICAS DA COMPUTAÇÃO EVOLUCIONÁRIA APLICADAS A PROBLEMAS DE PROGRAMAÇÃO LINEAR INTEIRA MISTA

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de "Doutor em Ciências" – Informática Industrial.

Orientador: Profº Doutor Flavio Neves Junior

Curitiba
2005

Ficha catalográfica elaborada pela Biblioteca da UTFPR – Campus Curitiba

B239s Barboza, Angela Olandoski
Simulação e técnicas da computação evolucionária aplicadas a problemas de programação linear inteira mista / Angela Olandoski Barboza. – Curitiba : [s.n.], 2005.
xvii, 217 f. : il. ; 30 cm

Orientador : Prof. Dr. Flavio Neves Junior
Tese (Doutorado) – UTFPR. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2005.
Bibliografia : f. 192-203

1. Programação linear. 2. Programação inteira. 3. Computação evolucionária. 4. Otimização matemática. 5. Algoritmos genéticos. 6. Métodos de simulação. 7. Petróleo – Refinarias. 8. Administração da produção – Processamento de dados. I. Neves Junior, Flávio, orient. II. Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD : 665.540285
CDU : 665.6

AGRADECIMENTOS

Agradeço primeiramente aos meus filhos Juliana e Wilson e ao meu marido Wilson, pela paciência, amor, incentivo e por compreenderem a falta de atenção a eles dispensada, principalmente nestes últimos dois anos.

Não tenho como expressar minha gratidão aos meus pais Adelaide e Arnaldo Olandoski, que sempre enfatizaram o valor da educação e estiveram ao meu lado nos momentos difíceis desta caminhada.

Aos meus sogros Ivette e Wilson Barboza (*in memorium*) pelo apoio e interesse constantes.

Agradeço à minha irmã Marcia Olandoski, o essencial apoio e incentivo, e por ter compartilhado comigo as vitórias e percalços na elaboração deste trabalho.

À professora Maria Teresinha Arns Steiner pelo precioso estímulo em momentos de dúvida e desânimo e também, por ter participado da banca de qualificação.

Devo também agradecimentos ao casal Elizabeth e Marco César Goldbarg pelo precioso auxílio que me prestaram, mesmo que distantes.

Aos professores da Banca Examinadora, Profª Dra. Elizabeth Ferreira Gouvêa Goldbarg, Profª Dra. Lúcia Valéria Ramos de Arruda, Profº Dr. Marco César Goldbarg, Profª Dra. Maria Teresinha Arns Steiner e Profº Dr. Nélio Domingues Pizzolato pela colaboração na avaliação deste trabalho.

A todos aqueles que contribuíram, direta ou indiretamente, na conclusão desta jornada. Em especial, agradeço aos meus companheiros de mestrado Maria Eugênia de Carvalho e Silva, Paulo Henrique Siqueira e Elizabeth Cristina Adamowicz pelo incentivo nestes últimos anos.

Ao meu orientador Flavio Neves Junior, tenho a agradecer a orientação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE TERMOS

Branching	-	Ramificação
Branch-and-bound	-	ramificar e delimitar
Branch-And-cut	-	ramificar e cortar
Buffer	-	tempo entre o término de uma atividade e início de outra
Data Mining	-	mineração de dados
Fitness	-	Adequabilidade
Flow Shop	-	processamento de *n* tarefas por *m* máquinas na mesma ordem
hardware	-	parte física do computador
Interface	-	interligação homem-máquina
Job Shop	-	processamento de *n* tarefas por *m* máquinas não necessariamente na mesma ordem
Just-in-time	-	resposta instantânea
Links	-	ligação ou atalho
Makespan	-	tempo total para executar um conjunto de tarefas
marketing	-	Negócios
NP-hard	-	NP- difícil
Planning	-	planejamento de produção
Scheduling	-	programação da produção
Shift	-	Mudança
Simulated Annealing	-	têmpera simulada
Slots	-	intervalos de tempo
Softwares	-	programas e aplicativos desenvolvidos para computadores
Solvers	-	Aplicativos para resolver problemas
Steady state	-	estado estacionário
String	-	termo utilizado como sinônimo de cromossomo
Suply Chain	-	cadeia de suprimento
Swap	-	troca

LISTA DE SIGLAS

AG	-	Algoritmo Genético
AGEEH	-	Algoritmo Genético de Estado Estacionário Híbrido
AND	-	Ácido Desoxirribonucléico
APS		*Advanced Planning and Scheduling*
ATEIs	-	Algoritmos Transgenéticos Extra Intracelulares
BIG	-	Banco de Informações Genéticas
Big-M	-	Método das penalidades ou do M grande
CE	-	Computação Evolucionária
EEs	-	Estratégias Evolucionárias
GDH	-	*Genetic descendent hybrid*
GLP	-	Gás Liquefeito de Petróleo
GRASP	-	*Graphical representation and analysis of structural properties*
ItMax	-	Número máximo de iterações
MPS	-	*Mathematical Programming Standard*
PE	-	Programação Evolucionária
PG	-	Programação Genética
PIM	-	Programação Inteira Mista
PL	-	Programação Linear
PLI	-	Programação Linear Inteira
PLIM	-	Programação Linear Inteira Mista
PNL	-	Programação Não Linear
PNLIM	-	Programação Não Linear Inteira Mista
ProtoG	-	Proto Gene
RET	-	Regras Transgenéticas
Rnd	-	Número aleatório no intervalo [0,1]
TC	-	Transgenética Computacional

RESUMO

As empresas vivem hoje uma realidade de transformações econômicas advindas da globalização. O crescimento do comércio internacional de produtos e serviços, a troca constante de informações e o intercâmbio cultural vêm desafiando os administradores a definir novos rumos para suas empresas. Esta dinâmica e a crescente competitividade exigem novos conhecimentos e habilidades dos profissionais. Desta forma, buscam-se novas tecnologias para conseguir-se a melhoria da eficiência operacional. Em especial, a indústria petrolífera brasileira tem investido na pesquisa aplicada, no desenvolvimento e na capacitação tecnológica para manter-se competitiva no mercado internacional. Muitos são os problemas que ainda devem ser estudados neste setor produtivo. Dentre estes, pode-se destacar os problemas de transferência e estocagem de produtos.

Este trabalho aborda um problema de programação da produção (*scheduling*) envolvendo estocagem e distribuição de óleo diesel em uma refinaria de petróleo. Para solucionar este problema foram utilizados, a princípio, modelos de Programação Linear Inteira Mista (PLIM) com abordagens para a representação nos tempos discreto e contínuo. Os modelos desenvolvidos foram resolvidos com o uso do aplicativo computacional LINGO 8.0, por meio do algoritmo *branch and bound*. Devido à natureza combinatorial destes, o tempo computacional despendido na resolução mostrou-se excessivo. Desta forma, foram desenvolvidas quatro novas metodologias para amenizar este problema: Algoritmo Genético de Estado Estacionário Híbrido (AGEEH) e Algoritmo Transgenético ProtoG integrados à Programação Linear (PL) para a representação de tempo discreto; simulação com otimização através de Algoritmo Genético (AG) e simulação com otimização usando Algoritmo Transgenético ProtoG, na representação de tempo contínuo.

Os resultados obtidos através de vários testes com as novas metodologias mostraram que estas podem encontrar bons resultados em um tempo computacional aceitável. Para a representação de tempo discreto, as duas abordagens obtiveram desempenho satisfatório em termos de qualidade da solução e do tempo computacional. Dentre elas, a metodologia que utilizou o Algoritmo Transgenético ProtoG apresentou os melhores resultados. Ainda, o simulador com otimização usando o AG e aquele que utilizou o Algoritmo Transgenético ProtoG na representação de tempo contínuo mostraram-se adequados para substituir a resolução por meio de PLIM, pois encontram soluções em um tempo computacional muito aquém do tempo despendido na resolução com o *branch and bound*.

ABSTRACT

Presently, companies live a reality of rapid economic transformations generated by globalization. The growth of the products and services international trade, the constant exchange of information and the cultural interchange challenge administrators to define new paths for their companies. This dynamics and the increasing competitiveness demand new knowledge and abilities from professionals. In this way, new technologies are researched in order to improve operational efficiency. The Brazilian oil industry in particular has invested in applied research, as well as on development and technological qualification to keep its competitiveness in the international market. Many are the problems that must still be studied in this production sector. Among these, and due their importance, the problems of products storage and transference can be pointed out.

This work approaches a scheduling problem that involves diesel oil storage and distribution in an oil refinery. The Mixed Integer Linear Programming (MILP) techniques with representation in the discrete and continuous time were used. The models that were developed were solved by the LINGO 8.0 software, using the branch and bound algorithm. However, due to their combinatorial nature, the expended computational time used for the solution was excessive. Thus, four new methodologies were developed: Hybrid Steady State Genetic Algorithm (HSSGA) and Transgenetic ProtoG Algorithm, both integrated to Linear Programming (LP), for the representation of discrete time; simulation with optimization using the Genetic Algorithm (GA) and simulation with optimization using the Transgenetic ProtoG Algorithm, for the representation of continuous time.

The results obtained through several tests with these new methodologies have shown that they can reach good results in an acceptable computational time. The two techniques for the representation of discrete time have shown satisfactory performance in terms of quality of solution and computational time. Among these, the methodology that uses the Transgenetic ProtoG Algorithm showed the best results. Also, the simulator with optimization using GA and the one that used the Transgenetic ProtoG Algorithm for the representation of continuous time were adequate to substitute the resolution through PLIM, because they reach solutions with a reduced computational time when compared with the time used for the solution with branch and bound.

CAPÍTULO 1

INTRODUÇÃO

1.1 JUSTIFICATIVA

O mercado mundial vem se modificando em decorrência dos avanços tecnológicos, da globalização, das grandes fusões de empresas e da maior conscientização ecológica. As transformações econômicas, a dinâmica dos mercados e a crescente competitividade fazem parte da globalização mundial, a qual intensifica o comércio internacional de produtos e serviços, promove intercâmbio cultural acentuado e a constante troca de informações. Tais mudanças implicam em um aumento da competitividade, obrigando as organizações a criarem soluções inovadoras para se manterem no mercado. Diante deste perfil, as empresas buscam um novo modelo de gestão baseado, principalmente, na redução dos custos dos produtos e das margens de lucratividade. Almejam, ainda, uma melhoria substancial do nível de serviços relacionados à distribuição para poderem competir com outras empresas. De maneira geral, os custos produtivos e a qualidade dos produtos tendem a se equiparar, independentemente da empresa que os produz e, por isso, o grande diferencial está na otimização das operações, ou seja, na capacidade dos produtos chegarem ao cliente final na quantidade certa, no tempo esperado e a um preço justo.

No contexto das indústrias, um dos problemas é o de otimização do sistema de produção no qual, o planejamento e a programação de produção são de fundamental importância. Esta otimização é hoje essencial para a competitividade das indústrias e inclui a minimização de custos de operação, que envolve, a determinação da taxa de produção, a política de manutenção de equipamentos, as estratégias de regulagem, a alocação e designação de produtos às máquinas, a alocação de produtos finais e, finalmente, a política de entrega a clientes.

Hoje, as empresas brasileiras vivem uma realidade em que ganhos de produtividade significam sobrevivência. Os sistemas de produção *just in time* (enxuta), isto é, sistemas preparados para atender a demanda por produtos a qualquer momento e com estoques reduzidos, apresentam vantagens com relação à produtividade, eficiência e qualidade. Desta forma, torna-se útil a criação de técnicas que proporcionem aos empresários e gerentes a certeza de que estão utilizando um sistema de produção que, se bem administrado, pode gerar

vantagens competitivas. Algumas destas ferramentas quantitativas ao processo de tomada de decisões são fornecidas pela Pesquisa Operacional (PO) que engloba as áreas de Programação Linear, Computação Evolucionária (CE), Teoria das Filas, Simulação, Programação Dinâmica, Teoria dos Jogos e outras.

Este trabalho aborda um problema de programação da produção (*scheduling*) envolvendo estocagem e distribuição de produtos em refinaria de petróleo, cuja solução ótima é difícil de ser encontrada devido à sua característica combinatorial. Em geral, este é modelado como problema de Programação Linear Inteira Mista (PLIM) e resolvido com auxílio de pacotes computacionais comerciais (MORO, 2000). Contudo, pela sua natureza, o aumento do número de variáveis inteiras torna impraticável o uso destes aplicativos, por exigirem um tempo computacional excessivo. Desta forma, não são aplicáveis em sistemas *just in time*, pois a tomada de decisão nestes casos deve ser imediata.

Diante deste cenário, justifica-se a procura por outras abordagens para a resolução deste problema. Em substituição às abordagens tradicionais para a resolução do problema de Programação Linear Inteira Mista, metodologias aplicando técnicas da Computação Evolucionária (CE), integradas com Programação Linear (PL) e otimização baseada em simulação com a utilização de técnicas da CE, foram desenvolvidas.

Duas técnicas da CE, consideradas como meta-heurísticas, foram utilizadas neste trabalho. A primeira delas, o AG, teve sua fundamentação na Teoria da Evolução Biológica dos seres vivos. É amplamente utilizada na resolução de problemas computacionais que freqüentemente requerem uma busca através de um amplo espaço de soluções candidatas. A partir da década de 80, o AG recebeu um grande impulso em diversas áreas de aplicação científica devido, principalmente, à sua versatilidade e excelentes resultados apresentados. A popularização dos computadores e o aparecimento de sistemas cada vez mais rápidos e potentes também impulsionaram muito o seu desenvolvimento.

A segunda técnica, a Transgenética Computacional, é uma nova abordagem da CE. Esta técnica utiliza a inserção planejada de informações no aprimoramento de soluções. Por utilizar conhecimento prévio sobre o problema, consegue-se convergir para uma boa solução mais rapidamente, o que ocasiona uma diminuição no tempo computacional de processamento.

Por outro lado, a simulação é uma técnica de modelagem e análise amplamente utilizada para avaliar e aprimorar os sistemas dinâmicos de todos os tipos. Permite a experimentação computacional de um modelo de um sistema num curto espaço de tempo,

propiciando a capacidade de tomada de decisões que não seria possível com a utilização de outras metodologias.

Mais recentemente vêm sendo utilizados os modelos que aplicam conceitos de simulação e otimização simultaneamente. Consegue-se, desta forma, integrar os objetivos da otimização às vantagens da simulação. Dentro das empresas, as decisões envolvendo produção, compra, políticas de estocagem e reposição, movimentação de materiais e distribuição física devem ser tomadas de forma sistêmica e integrada. Por isso, a modelagem e a simulação são as ferramentas apropriadas para avaliar os ganhos de cada alternativa e os efeitos dessas relações, uma vez que, antes de se implementar uma nova operação ou processo, precisa-se ter uma idéia prévia dos possíveis resultados e conseqüências a fim de identificar pontos de melhoria visando otimizá-los. Outra vantagem que pode ser apontada com relação ao uso de simulação é que esta técnica permite que várias proposições com combinações e diferentes quantidades de recursos sejam realizadas computacionalmente, permitindo a escolha da melhor alternativa em termos de investimento, estratégia e produtividade.

1.2 OBJETIVOS

O objetivo geral deste trabalho é encontrar metodologias mais eficientes em termos computacionais para resolução de problemas modelados em Programação Linear Inteira Mista (PLIM) do que os métodos clássicos, entre estes, o método *branch and bound*.

Os objetivos específicos são:

- Desenvolver um modelo em PLIM para um problema de transferência e estocagem de diesel em uma refinaria de petróleo com representação de tempo discreto e resolver aplicando o método *branch and bound*;
- Desenvolver um segundo modelo em PLIM para um problema de transferência e estocagem de diesel em uma refinaria de petróleo com representação de tempo contínuo e resolver aplicando o método *branch and bound*;
- Resolver o problema da primeira modelagem aplicando as técnicas de Algoritmo Genético (AG) e Transgenética Computacional integradas à Programação Linear (PL);
- Resolver o problema da segunda modelagem aplicando a técnica de simulação com otimização através de AG e Transgenética Computacional;

1.3 ESTRUTURA DO TRABALHO

Neste primeiro capítulo são destacados os problemas que motivaram o desenvolvimento deste trabalho, mostrando a necessidade em buscar novas abordagens, utilizando técnicas da Pesquisa Operacional (PO) para a otimização de processos industriais.

O Capítulo 2 apresenta uma revisão bibliográfica dos seguintes assuntos: Programação Matemática destacando-se modelagem, problemas de busca e otimização, PL, Programação Inteira (PI) e PLIM, métodos e complexidades da PI; cadeia de suprimentos, planejamento (*planning*) e programação da produção (*scheduling*), com enfoque em refinarias de petróleo; CE com detalhamento de AG e Transgenética Computacional e por último o tópico que trata de Simulação de Sistemas.

O Capítulo 3 descreve a modelagem dos modelos em PLIM com representação de tempo discreto e contínuo.

No Capítulo 4 é feita a descrição das metodologias que integram AG e Transgenética Computacional com o modelo em PL para resolver o problema de PLIM com representação de tempo discreto.

O Capítulo 5 detalha as metodologias que utilizam AG e Transgenética Computacional com Simulação para resolução do modelo PLIM com representação contínua do tempo.

A implementação dos programas computacionais e os resultados obtidos são mostrados no Capítulo 6.

Finalmente, o Capítulo 7 apresenta a análise dos resultados, as principais conclusões obtidas, as contribuições deste trabalho e sugestões para trabalhos futuros.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

O mundo globalizado faz com que as empresas procurem auxílio das tecnologias existentes para estarem aptas a competir no mercado. Atualmente, a tecnologia da comunicação permite uma rápida propagação de informações entre pesquisadores e especialistas, fazendo com que estes se atualizem e incorporem novos modelos em seus planos de gestão e negócios, com o objetivo de tornar empreendimentos cada vez mais lucrativos.

Hoje, a competitividade e a sobrevivência de uma companhia dependem, principalmente, da habilidade em controlar os desafios na administração do tempo e dos custos, aumentando níveis de qualidade de serviços e produtos.

O uso de técnicas de Pesquisa Operacional na modelagem de processos das empresas tem-se mostrado um fator decisivo para o desenvolvimento de políticas otimizadas de operação industrial. O desenvolvimento de modelos, em especial os que empregam técnicas de otimização, tem possibilitado que procedimentos operacionais complexos sejam avaliados de forma criteriosa, fazendo com que recursos críticos possam ser utilizados da melhor maneira possível.

Por um lado, no contexto da gestão de processos, estão os problemas relacionados às cadeias de suprimento, seus recursos e tarefas, e à necessidade de se estabelecer adequadamente o planejamento e a programação da produção nas empresas. Por outro lado, estão as ferramentas matemáticas de otimização, oferecendo soluções a estes problemas. Embora seja amplo o espectro de métodos e técnicas matemáticas hoje disponíveis para aplicação em problemas dessa natureza, são descritas neste capítulo aquelas que foram particularmente importantes na abordagem deste estudo.

Inicialmente, são apresentados alguns mecanismos clássicos de Pesquisa Operacional ligados à Programação Matemática e, logo após, são destacados aspectos relacionados ao planejamento e programação de produção, com ênfase especial ao setor petrolífero. Em seguida, é feita uma descrição detalhada de Computação Evolucionária (CE), em especial Algoritmo Genético (AG) e Transgenético. Por último, são apresentados conceitos, modelagem e métodos de simulação.

2.1 PROGRAMAÇÃO MATEMÁTICA

2.1.1 Introdução

A Programação Matemática é o ramo da Pesquisa Operacional que trata de métodos de otimização (minimização ou maximização) de uma função objetivo com um número finito de variáveis de decisão sujeita a certas restrições. Estas restrições podem ser de origem financeira, tecnológica, *marketing*, organizacional ou outras.

De um modo geral, Programação Matemática pode ser definida como uma representação matemática dedicada à programação ou planejamento da melhor possibilidade de alocação de recursos escassos (BRADLEY, HAX e MAGNANTI, 1977). A Programação Matemática utiliza técnicas e algoritmos para solucionar problemas modelados matematicamente.

Segundo Williams (1999), há três motivos principais para a elaboração de modelos em Programação Matemática.

(i) O procedimento de construção de um modelo revela relacionamentos que, em geral, não são evidentes, propiciando um melhor entendimento do objeto que está sendo modelado;
(ii) Em geral é possível analisar matematicamente um modelo, sugerindo novas tendências e procedimentos que, de outra forma, não seriam percebidos;
(iii) Experiências que não são possíveis ou desejáveis na realidade, podem ser efetuadas a partir do modelo formulado.

2.1.2 Modelos de programação

Segundo Goldbarg e Luna (2000), um modelo é um veículo para uma visão bem estruturada da realidade. Um modelo também pode ser visto, com os devidos cuidados, como uma representação substitutiva da realidade. Modelos são construídos para diversos tipos de problemas. Segundo Magatão (2001), o papel dos modelos é configurar uma importante ferramenta de auxílio ao processo de tomada de decisões, ampliando a capacidade de percepção dos especialistas envolvidos, com o melhor aproveitamento possível dos componentes do processo industrial.

Goldbarg e Luna (2000) classificam os modelos com relação à sua natureza. A Figura 1 ilustra esta classificação.

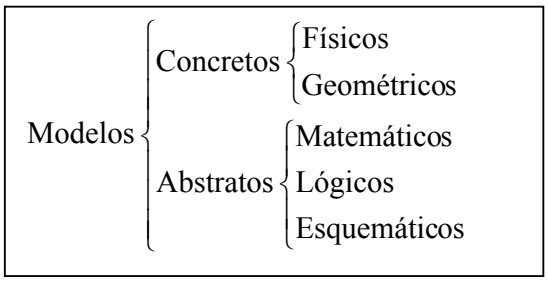

Figura 1: Classificação Geral dos Modelos

Várias abordagens e arquiteturas são sugeridas na literatura para a elaboração de um modelo. Goldbarg e Luna (2000) sugerem os passos mostrados na Figura 2.

Na primeira fase do processo de modelagem é feita a definição do problema e a caracterização dos dados iniciais. Em seguida, está a fase de formulação e construção do modelo que envolve a definição dos tipos de variáveis a serem utilizadas na representação, bem como o nível apropriado de agregação destas. Nesta fase também são representadas as restrições do modelo e é definida uma função de desempenho denominada de função objetivo. A construção de modelos determina a inclusão de parâmetros e constantes que serão responsáveis pela definição e dimensionamento das relações entre as variáveis a serem incluídas. Na validação do modelo é verificado se este está retratando o problema real. Nesta fase são efetuados testes com o modelo. Se este não estiver condizente com a realidade, as distorções são corrigidas na fase de reformulação e volta-se para as fases de testes e validação. Quando o modelo se mostra satisfatório, então se passa à última fase que é a de aplicação do modelo.

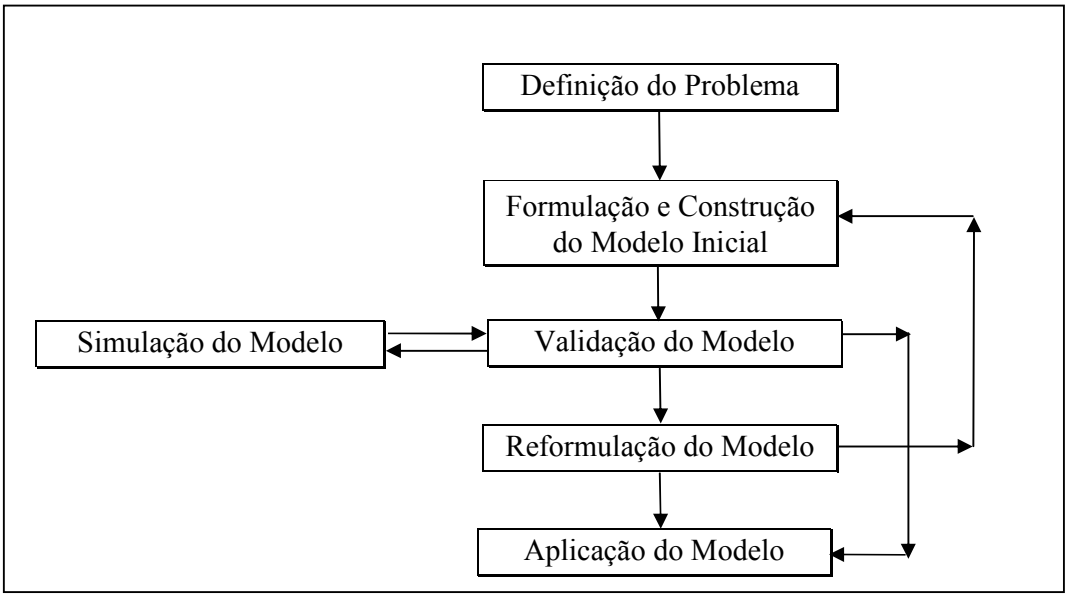

Figura 2: Processo de Construção de Modelos (GOLDBARG e LUNA, 2000)

2.1.3 Problemas de busca e otimização

Grande parte dos problemas científicos pode ser modelada como problemas de busca e otimização: basicamente, existe uma série de fatores influenciando o desempenho de um dado sistema e estes podem assumir um número limitado ou ilimitado de valores e podem estar sujeitos a certas restrições. O objetivo é encontrar a combinação de fatores que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o chamado espaço de busca. Não é difícil perceber que existe uma dualidade entre os conceitos de busca e otimização, de tal modo que todo problema de busca pode ser considerado um problema de otimização e vice-versa. (TANOMARU, 1995).

Dada uma função $f : \Re^n \rightarrow \Re$ e um espaço de busca $S \subseteq \Re^n$, o problema de otimização pode ser formulado como mostra a equação (1):

$$\text{Maximizar } f(x) \mid x \in S \tag{1}$$

Também pode ser escrito da forma apresentada na equação (2):

$$\text{encontrar } x^* \mid f(x^*) \geq f(x), \forall x \in S \tag{2}$$

A função $f(\cdot)$ pode ser derivável ou não, uni ou multidimensional e o espaço de busca S pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo. O problema é chamado unimodal quando há somente um ponto máximo x^* no espaço de busca ou multimodal, caso contrário. Além disso, a função $f(\cdot)$ pode ser estacionária ou não estacionária, isto é, variante com o tempo.

Existem diversos métodos de otimização. Tanomaru (1995) classifica-os em métodos probabilísticos, numéricos e enumerativos. Há ainda um grande número de métodos híbridos. Os métodos probabilísticos empregam a idéia de busca probabilística. Os métodos numéricos podem ser divididos em analíticos ou baseados em cálculo numérico. Os analíticos de otimização são aplicáveis quando a função $f(\cdot)$ é explicitamente conhecida e derivável ou pode ser aproximada por alguma função derivável até o grau desejado de precisão. Neste caso, basta resolver as equações que resultam da igualdade das derivadas da função à zero dentro do espaço de busca. Os métodos baseados em cálculo numérico podem utilizar técnicas de Pesquisa Operacional como o método simplex quando o espaço de busca é linear e técnicas de gradiente ou de estatística quando o problema é não linear. Finalmente, os métodos enumerativos de otimização examinam cada ponto do espaço de busca, um por um, em busca dos pontos ótimos. A idéia pode ser intuitiva, mas torna-se inviável quando existe um número infinito ou muito grande de pontos a examinar.

Em otimização, uma boa capacidade de exploração do espaço de busca é requerida se o que se deseja é encontrar um extremo (máximo ou mínimo) global, ou, pelo menos, uma boa solução. Existem problemas aparentemente simples que exigem um grande esforço de cálculo para sua resolução, devido à complexidade de seu algoritmo de busca, à estrutura e tipo de dados utilizados, aos recursos computacionais disponíveis e, principalmente, ao tamanho (ou dimensão) de sua entrada. Sabe-se que, para uma determinada classe de problemas, existem diversos métodos para resolução. Pode-se dizer que o método mais eficiente é aquele que apresenta o melhor resultado, usando o mínimo de recursos, no menor tempo possível. Nem sempre é possível certificar que o resultado obtido satisfaz a condição de melhor, porém, se dois os mais métodos apresentam o mesmo resultado usando os recursos existentes, pode-se considerar o mais eficiente àquele que encontrou a solução de forma mais rápida.

Segundo Goldbarg e Luna (2000), o campo da Programação Matemática é amplo e suas técnicas consagraram-se em face à sua grande utilidade na solução de problemas de otimização. Em virtude das várias peculiaridades inerentes aos diversos contextos de

programação (planejamento), os métodos de solução sofreram especializações e particularizações. O processo de modelagem matemática, em si, pouco varia, contudo as técnicas de solução acabaram agrupadas em várias subáreas como: PL, Programação não Linear (PNL) e PI.

2.1.4 Programação Linear (PL)

Programação Linear (PL) é uma das mais importantes e mais utilizadas técnicas de Pesquisa Operacional. A simplicidade do modelo envolvido e a disponibilidade de uma técnica de solução programável em computador como o método Simplex descrito por Dantzig (1963) facilitam sua aplicação. Esta técnica é amplamente utilizada, pois possui habilidade para modelar importantes e complexos problemas de decisão e o método Simplex pela capacidade de produzir soluções rapidamente. A descrição do método Simplex pode ser encontrada em Zionts (1974).

Um problema de PL é composto por:

1) Uma função linear formada com as variáveis de decisão chamada de Função Objetivo, cujo valor deve ser otimizado;
2) Relações de interdependência entre as variáveis de decisão que se expressam por um conjunto de equações ou inequações lineares, chamadas de restrições do modelo;
3) Variáveis de decisão que devem ser positivas ou nulas.

Em (3) tem-se a formulação para um problema de PL:

$$
\begin{aligned}
&\text{maximize (ou minimize)} \\
&z = \sum_{j \in N} c_j x_j,\ N = \{1,...,n\} \\
&\text{sujeito a} \\
&\sum_{j \in N} a_{ij} x_j \ (\leq, = ou \geq)\, b_i,\ i \in M = \{1,2,...,m\} \\
&x_j \geq 0,\ j \in N
\end{aligned}
\tag{3}
$$

no qual c_j, a_{ij} e b_i são constantes conhecidas para todo i e j, e x_j são variáveis não negativas.

As restrições do problema podem ser transformadas em equações adicionando-se uma variável de folga (não negativa) x_{n+i}, se a i-ésima desigualdade é do tipo $\leq$ e subtraindo uma variável de folga (não negativa), x_{n+k} se a k-ésima desigualdade é do tipo $\geq$. Considerando

que ao serem acrescentadas as variáveis de folga, obtém-se um total de $m + n$ variáveis, pode-se escrever o problema na forma matricial apresentada em (4):

$$\begin{aligned} &\text{maximize (ou minimize)} \\ &\quad z = cx \\ &\text{sujeito a} \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{4}$$

no qual, c é um vetor linha de ordem $(m+n)$, A é uma matriz $m \times (m + n)$, x é um vetor coluna de ordem $(m + n)$ e b é um vetor coluna de ordem m.

2.1.5 Programação Inteira (PI)

Alguns problemas reais requerem o uso de variáveis que assumem somente valores inteiros. Quando isto acontece tem-se um problema de Programação Inteira (PI). Este problema está definido na formulação (5) a seguir:

$$\begin{aligned} &\text{maximize (ou minimize)} \\ &\quad z = g_0(x_1, x_2, ..., x_n) \\ &\text{sujeito a} \\ &\quad g_i(x_1, x_2, ..., x_n)(\leq ou = ou \geq) b_i,\ i \in M = \{1,2,...,m\} \\ &\quad x_j \geq 0,\ j \in N = \{1,2,...,n\} \\ &\quad x_j \text{ inteira},\ j \in I \subseteq N \end{aligned} \tag{5}$$

no qual, $x_j,\ j \in N$ são as variáveis, $g_i,\ i \in M \cup \{0\}$ são funções das variáveis $x_1, x_2,...,x_n$, e $b_i,\ i \in M$ são constantes conhecidas. Se $I = N$, isto é, todas as variáveis são inteiras, então o problema é dito de PI. Caso contrário, se $I \subset N$, então chama-se de problema de Programação Inteira Mista (PIM).

Em muitos dos problemas abordados em PI, as funções $g_i,\ i \in M \cup \{0\}$ são lineares e o modelo pode então ser descrito como mostra a formulação (6):

$$
\begin{aligned}
&\text{maximize (ou minimize)} \\
&\quad z = \sum_{j \in N} c_j x_j \,,\; N = \{1,\ldots,n\} \\
&\text{sujeito a} \\
&\quad \sum_{j \in N} a_{ij} x_j \,(\leq, \geq ou =) b_i \,,\; i \in M = \{1,\ldots,m\} \\
&\quad x_j \geq 0,\; j \in N \\
&\quad x_j \text{ inteira},\; j \in I \subseteq N
\end{aligned}
\tag{6}
$$

onde x_j são variáveis não negativas e c_j, a_{ij} e b_i são constantes conhecidas, para todo i e j. Se $I = N$, isto é, todas as variáveis são inteiras, então temos um problema Programação Linear Inteira (PLI). Se $I \subset N$, então o problema é de Programação Linear Inteira Mista (PLIM).

Muitos modelos práticos de PLI restringem algumas das variáveis inteiras para valores "0" ou "1" e, neste caso, tem-se um problema de Programação Linear Inteira Binária (PLIB). Estas variáveis são usadas para decisão: sim ("1") e não ("0").

2.1.5.1 Programação Linear Inteira Mista (PLIM)

Existem muitos problemas de programação de produção (*scheduling*) que podem ser colocados como problemas de Programação Linear Inteira Mista, pois os modelos matemáticos de otimização correspondentes envolvem variáveis contínuas e discretas que devem satisfazer um conjunto de restrições lineares de igualdade e desigualdade (MORO, 2000).

A resolução para problemas de otimização linear inteira mista, entendida como a obtenção de uma solução ótima, pode ser difícil, pela sua natureza combinatorial. Num primeiro contato com este tipo de problema, a abordagem seria a de resolver o problema para todas as combinações de variáveis inteiras utilizando a PL e extrair a solução como o menor valor da função objetivo (problemas de minimização). Porém, o número de combinações cresce exponencialmente com o número de variáveis binárias. Logo, para problemas práticos onde se tem um grande número de variáveis inteiras, esta abordagem é inviável. Uma outra alternativa seria a de relaxar as restrições de integralidade e tratar as variáveis inteiras como contínuas, mas não se tem a garantia de que, resolvendo o problema com esta relaxação, encontre-se uma solução com valores inteiros para as variáveis discretas. O arredondamento para valores mais próximos também não leva, em geral, ao resultado correto. Taha (1975) mostra, através de um exemplo, que esta abordagem não é conveniente.

2.1.5.2 Métodos e complexidade de PI

Segundo Taha (1975) as técnicas para a resolução de problemas de PLI são geralmente classificadas em dois tipos: Métodos de Enumeração e Métodos de Corte.

O primeiro tipo baseia-se no fato de que o espaço de soluções inteiras pode ser considerado como constituído de um número finito de pontos. Mesmo no caso misto, o espaço de busca é primeiramente controlado pelas variáveis inteiras. Na forma mais simples, os métodos de enumeração analisam todos os pontos. É o que se chama de busca exaustiva. Um simples método de busca exaustiva pode se tornar mais eficiente se enumerar apenas uma parte das soluções candidatas enquanto descarta pontos que não são promissores. A eficiência de um algoritmo de busca depende de sua capacidade em descartar pontos de solução não promissores. Dentre estas técnicas podem ser citadas: separação e avaliação progressiva ou *branch and bound* (ZIONTS, 1974) e enumeração implícita (TAHA, 1975). Estas técnicas podem utilizar uma relaxação do problema para obter em tempo razoável uma estimativa para o valor da melhor solução que pode ser encontrada em cada ramo da enumeração. Esta estimativa pode ser obtida a partir da relaxação linear ou de uma relaxação Lagrangeana. Pode também ser utilizada a relaxação *surrogate* que consiste em reduzir algumas restrições do problema original a uma só restrição, denominada de restrição *surrogate*; sua função é substituir essas restrições (ESPEJO e GALVÃO, 2002).

Os métodos de corte foram a princípio desenvolvidos para problemas lineares inteiros mistos ou puros. A motivação destes métodos está no fato de que a solução para problemas de PL ocorre em um ponto extremo. A idéia é acrescentar restrições especialmente desenvolvidas que infringem a solução atual não inteira, mas não qualquer das soluções inteiras factíveis. A aplicação sucessiva deste procedimento pode eventualmente gerar um novo espaço convexo, com seu ponto extremo ótimo satisfazendo apropriadamente a condição de integralidade. A denominação método de corte se fundamenta no fato de que as restrições adicionadas "cortam" partes infactíveis do espaço contínuo de soluções. Algumas das técnicas de corte são: cortes inteiros (primais e duais), cortes combinatórios, cortes de intersecção e método de decomposição de Benders.

Goldbarg e Luna (2000) acrescentam ainda as técnicas híbridas tais como: *Branch-And-Cut* e Teoria de Grupos.

As diversas técnicas existentes foram especializadas para problemas de PI, tendo sido desenvolvidas abordagens e algoritmos específicos para cada situação analisada. A descrição

das técnicas mais utilizadas pode ser encontrada em Pinto (2000), Taha (1975), Zionts (1974), Nemhauser e Wolsey (1988) e Bradley, Hax e Magnanti (1977).

As dificuldades que surgem em problemas inteiros têm sido amplamente estudadas por pesquisadores. A principal dificuldade na resolução destes problemas é que não são conhecidas condições de otimalidade para verificar se uma solução encontrada é o ponto ótimo ou não. Para garantir a otimalidade de uma solução é feita a comparação desta com soluções já encontradas. Desta forma, são enumeradas todas as alternativas possíveis, o que gera um custo computacional proibitivo para alguns problemas.

Garey e Johnson (1979) observaram que uma modelagem que utiliza variáveis inteiras possui uma natureza combinatória de difícil resolução. Segundo Taha (1975), a complexidade computacional de um modelo inteiro é afetada primeiramente pelo aumento de variáveis inteiras. Desta forma, deve-se reduzir ao mínimo o número destas variáveis na formulação do modelo. Por outro lado, Shah, Pantelides e Sargent (1993) e Williams (1999) demonstram que, ao serem utilizadas variáveis de decisão, o número destas não é um bom parâmetro para mensurar o tempo de resolução de problemas de PI ou PIM. Se um problema possui n variáveis binárias então sua árvore de busca terá $\left(2^{n+1}-1\right)$ soluções para serem analisadas. Dependendo do algoritmo utilizado, somente uma pequena parcela destes nós é realmente analisada, o que confirma que o número de variáveis binárias não pode ser considerado como um indicador do tempo de resolução computacional.

Por outro lado, contrapondo-se aos problemas de PL, os problemas de PIM podem ser resolvidos em tempos computacionais menores se o número de restrições for aumentado (Magatão, 2001). Ainda, Nemhauser e Wolsey (1988) afirmam que formulações com um número pequeno de restrições constituem uma estratégia pobre para os problemas de PIM. Sherali e Driscoll (2000) confirmam que a adição de restrições para problemas de PI, mesmo com o risco de aumentar a dimensão do problema, é uma prática que contribui para uma melhora no desempenho na obtenção de respostas computacionais.

2.1.6 Ferramentas computacionais para problemas de otimização

Existem vários aplicativos computacionais desenvolvidos para a resolução de problemas de busca e otimização, incorporando diversas técnicas. O progresso na solução destes tipos de problemas tem sido impulsionado pelos avanços obtidos na PL.

Os aplicativos que resolvem problemas de PL utilizam o método Simplex e/ou o método de busca por ponto interior. Já para problemas de PLI e PLIM a maior parte dos aplicativos utiliza o método *branch and bound*. Segundo Pinto (2000), a limitação para a maioria dos aplicativos comerciais existentes é com relação à memória da máquina e o elevado tempo computacional. Várias alternativas vêm sendo testadas para contornar este problema tais como o uso de máquinas mais avançadas e o processamento paralelo em que vários computadores trabalham simultaneamente e alimentam um computador nomeado como principal, cuja tarefa é compilar todas as informações.

Os componentes indispensáveis de um aplicativo para resolver problemas de PLIM incluem (PINTO, 2000):

- Rotina do método Simplex: resolve a modelagem de PL e gera limites inferiores;
- Pré-processamento: o modelo é analisado de forma a fixar ou eliminar variáveis desnecessárias, gerar limites, reformular restrições, eliminar restrições redundantes. Esta etapa muitas vezes pode reduzir a dimensão do problema;
- Planos de corte: restrições são adicionadas ao modelo para eliminar soluções dos modelos de PL, mas não do problema inteiro, com o objetivo de reduzir o espaço de busca.
- Heurísticas: são utilizadas para gerar soluções viáveis satisfatórias. Heurísticas primais geram soluções inteiras, reduzindo o limite superior. Heurísticas duais podem gerar soluções viáveis do dual do problema linear relaxado (limites inferiores) mais rapidamente que o Simplex.
- Separação: a enumeração sistemática de soluções é obtida fixando-se as variáveis ou adicionando restrições de modo que a região viável seja sub-dividida em regiões disjuntas;
- *Branching*: determina qual região deverá ser escolhida para a busca.
- *Interface* com o usuário: cada vez mais as empresas fornecedoras de aplicativos nesta área buscam oferecer ao usuário fácil entendimento com relação ao uso de seus produtos. O padrão de códigos de otimização é o sistema de programação matemática (*MPS- Mathematical Programming System*) que representa o modelo na forma matricial. A maioria dos aplicativos possui ao menos uma linguagem algébrica de entrada do modelo e um gerador de matrizes.

Pinto (2000) listou em tabelas vários aplicativos com especificações tais como: dados da empresa que desenvolveu o produto, plataformas, formatos de entrada, tipos de problemas que resolve e outros.

Dentre estes aplicativos, vale destacar:

- LINGO 8.0 (LINDO *Systems Inc*., 2002): é uma ferramenta de simples utilização e serve para resolver e analisar soluções para problemas de grande porte em Programação Linear e não Linear. Possui linguagem de modelagem própria e trabalha com 4 *solvers*: direto, linear, não linear e um gerenciador de *branch and bound*. O *solver* linear utiliza o método Simplex enquanto que o não linear emprega os algoritmos de programação linear sucessiva e gradiente reduzido generalizado. Os modelos inteiros são resolvidos através do método *branch and bound*.
- ILOG CPLEX 8.0 (CPLEX *Optimization Inc*., 2002): foi desenvolvido para revolver problemas de PL. Resolve também problemas de Fluxos em Rede, Programação Quadrática e PIM. Possui três variações, dependendo das necessidades do usuário, a saber: o CPLEX *Interactive Optimizer*, um programa executável que pode carregar o problema de forma interativa de um arquivo, resolvê-lo e então enviar o resultado a um arquivo de texto; o *Concert Technology* que é composto por um conjunto de bibliotecas na linguagens C++ e Java e possibilitam ao usuário a inserção do otimizador do CPLEX na sua própria programação; finalmente, o CPLEX *Callable Library* que é uma biblioteca na linguagem C que permite ao usuário inserir o otimizador do CPLEX em aplicativos programados em C, Visual Basic, FORTRAN ou outros compiladores que tenham conexão com a linguagem C.

2.2 CADEIAS DE SUPRIMENTO, PLANEJAMENTO E PROGRAMAÇÃO DA PRODUÇÃO

O termo Cadeia de Suprimento ou *Suply Chain* pode ser utilizado para definir uma série de etapas e operações que transformam a matéria-prima em produtos finais para o consumidor (CHWIF, BARRETTO e SALIBY, 2002).

Segundo Joines, Gupta, Gokce et al. (2002), uma cadeia de suprimento, analisada sob a perspectiva operacional, possui quatro componentes: o controle de obtenção de matéria-prima, a manufatura, a distribuição e as decisões no que se refere aos estoques. O principal objetivo em uma cadeia de suprimento é produzir e entregar produtos para o consumidor final com o mínimo custo e o máximo de eficiência.

Dentro das cadeias de suprimento estão os problemas de Planejamento (*Planning*) e Programação da Produção (*Scheduling*) que coordenam as operações em todos os estágios da cadeia (KREIPL e PINEDO, 2004).

A partir de 1980, as indústrias, percebendo que o ganho de qualidade, o corte de custos e o melhor atendimento aos clientes eram essenciais para a competitividade dentro do mercado globalizado, começaram a dar maior importância às atividades relacionadas ao planejamento, programação da produção e distribuição.

Nas indústrias que trabalham com processos químicos, tanto o planejamento como a programação da produção são considerados procedimentos de alocação de recursos e de equipamentos para executar o processamento de tarefas de natureza química ou física necessárias para a produção dos produtos químicos, num determinado período de tempo (PEKNY E ZENTNER, 1994).

As diferenças entre planejamento e programação da produção são abordadas por vários autores. Segundo Kreipl e Pinedo (2004), modelos de planejamento diferem de modelos de programação da produção de várias formas:

1) Modelos de planejamento envolvem, freqüentemente, múltiplos estágios e otimizam num horizonte de tempo médio, enquanto que modelos de programação da produção são geralmente desenvolvidos para um só estágio e otimizam num horizonte de tempo pequeno;
2) Modelos de planejamento usam muitas informações agregadas, enquanto que modelos de programação da produção utilizam informações detalhadas;
3) O objetivo no modelo de planejamento está centrado tipicamente na minimização do custo total, enquanto que o objetivo da programação da produção se preocupa em minimizar a função do tempo para a conclusão de tarefas.

Joly (1999) expõe que o termo Planejamento é utilizado na definição de objetivos e de decisões agregadas de alocação, relativos a períodos de tempo mais longos, como por exemplo, meses ou anos e é geralmente elaborado com base na política da empresa. Por outro lado, a Programação de Produção é uma função de refinamento e faz referência a um período de tempo menor para o qual decisões detalhadas de seqüenciamento de tarefas e alocação de equipamentos são definidas para serem executadas.

Ainda segundo Birewar (1989), planejamento é um problema de nível macroscópico que tem por objetivo, metas de produção relativas a períodos longos de tempo (meses ou anos), considerando, para isto, previsões do mercado (preço dos produtos e suas demandas), bem como disponibilidade de recursos (equipamentos, mão-de-obra e insumos) e estoques. Já a programação da produção é definida pelo autor como um problema de nível microscópico que está imerso no problema de planejamento e que considera períodos de tempo menores (dias ou semanas). Desta forma, o principal objetivo da programação da produção é a geração

de informações detalhadas sobre decisões de seqüenciamento de tarefas e sua alocação de acordo com os equipamentos disponíveis, visando atender as metas definidas pelo planejamento.

2.2.1 Problemas e técnicas de programação da produção

Magalhães, Moro, Smania et al. (1998) observam que, em geral, a programação da produção é alvo de estudo de profissionais diretamente ligados à etapa produtiva e está estritamente relacionada à simulação/otimização computacional.

Atualmente, o estudo de problemas de programação da produção vem se especializando no desenvolvimento, aplicação e resolução de procedimentos combinatórios, técnicas de simulação, métodos em redes e soluções heurísticas. A seleção da técnica apropriada depende da complexidade do problema, da natureza do modelo e da escolha dos critérios e outros fatores (BAKER, 1974).

Nas indústrias, a programação da produção é necessária toda vez em que há competição entre atividades, dada a limitação de recursos disponíveis num período finito de tempo. Isto envolve três elementos principais: alocação de recursos, seqüenciamento de atividades e determinação do tempo de utilização dos recursos pelas atividades (REKLAITIS, 1992). A alocação envolve a seleção de um grupo apropriado de recursos para uma dada atividade. O seqüenciamento se refere ao ordenamento da execução das atividades alocadas aos recursos, enquanto que a temporização envolve a determinação do início e final específico para cada uma das atividades programadas.

Os problemas de programação de produção podem ser considerados como de otimização de recursos e tarefas, sujeitos a restrições. Em geral, os recursos são caracterizados em termos de capacidades quantitativas e qualitativas, e o modelo estipula o tipo e a quantidade de cada recurso. Já as tarefas podem ser interpretadas como restrições tecnológicas existentes entre os elementos.

Segundo Baker (1974), para classificar a maioria dos modelos de programação da produção é necessário caracterizar a configuração dos recursos e o comportamento das tarefas. O modelo é dito de estágio único se possui um único tipo de recurso. Quando possui vários tipos de recursos é chamado de múltiplo-recurso. Com relação ao conjunto de tarefas, o sistema é dito estático se este conjunto não se altera no decorrer do tempo e é chamado de dinâmico se novas tarefas surgem no decorrer do tempo.

Já para Pinto e Grossmann (1998), as técnicas de programação da produção podem ser classificadas quanto ao padrão de demanda em programação da produção cíclica e de curto prazo. Segundo Shah, Pantelides e Sargent (1993), a programação da produção cíclica é recomendada para um mercado consumidor estável, onde as demandas pelos produtos não apresentam grandes variações, podendo assim ser aplicada para períodos de aproximadamente um mês. A aplicação da programação da produção de curto prazo resume-se a períodos diários ou a horizontes de no máximo um mês. Tanto na programação da produção a curto prazo quanto cíclico é importante definir a política de inventários, o que por sua vez requer a alocação da produção para atendê-la.

Ainda, segundo Moro (2000), um aspecto fundamental em modelos de programação de produção diz respeito à representação do domínio do tempo. Existe a representação de tempo discreta e contínua. Na abordagem de tempo discreto, é comum o uso de intervalos de tempo (*slots*) com duração fixa e igual. É definido *a priori* um certo número de intervalos que cubram todo o horizonte de tempo operacional. O número e a duração dos intervalos de tempo devem ser determinados de modo que as decisões relevantes ao problema ocorram na fronteira entre dois intervalos de tempo. O inconveniente encontrado na aplicação de tempo discreto é a geração de um número muito grande de intervalos, que acaba ocasionando uma demora na obtenção da solução e, em alguns casos, provocando a impossibilidade da obtenção desta. Já na representação de tempo contínuo, os intervalos de tempo (*slots*) possuem duração variável. Segundo Pinto e Grossmann (1998), os eventos (ou *slots*) podem ser associados a equipamentos, tarefas ou podem até serem definidos globalmente. Para Más (2001), a formulação em eventos possibilita a discretização das operações de um sistema, ou seja, cada componente do processo é composto por eventos independentes dentro do horizonte de tempo pré-determinado, os quais são ativados quando estes componentes tomam parte de uma operação. Para a representação contínua no tempo, deve-se utilizar um número suficiente de eventos (*slots*) para preencher o horizonte de tempo operacional.

Baker (1974), fazendo uma abordagem usando máquinas como recursos, agrupou os problemas de programação da produção da seguinte forma:

- Seqüenciamento de uma máquina com tarefas independentes: é um problema de seqüenciamento puro no qual uma ordenação completa das tarefas determina a programação da produção. Existe, para este tipo de problema, uma correspondência um a um entre a seqüência das n tarefas e uma permutação da numeração das tarefas 1,2,..., n. O número de soluções é $n!$, que é o total de permutações das n tarefas;

- Modelos de máquinas paralelas: é um problema de seqüenciamento de estágio simples com várias máquinas. Neste tipo de modelo, *n* tarefas estão disponíveis para serem executadas no início da programação. Existem *m* máquinas e cada tarefa deve ser processada por qualquer destas máquinas.
- Modelos de *Flow Shop*: consiste de *m* máquinas e *n* tarefas independentes que devem ser processadas na mesma ordem. Em outras palavras, a *j*-ésima operação de todas as tarefas deve ser sempre executada na máquina *j*. Existem $(n!)^m$ diferentes alternativas para ordenar *n* tarefas em *m* máquinas.
- Modelos de *Job Shop*: numa abordagem clássica, uma tarefa *j* possui m_j operações para serem executadas em seqüência; cada operação é designada a uma máquina, que pode efetuar aquela operação; uma tarefa não pode utilizar a mesma máquina mais de uma vez. O número de soluções distintas para este tipo de problema é de aproximadamente $\left[\left(\frac{nq}{m}\right)!\right]^m$, sendo *m* o número de máquinas, *n* o número de tarefas e *q* o número médio de operações por tarefa.

Morton e Pentico (1993) abordam diversas técnicas que podem ser utilizadas para a resolução de problemas de programação da produção. Dentre estas, são citadas: Abordagem Manual, Simulação Computacional, Programação Inteira, Programação Dinâmica, Busca em Vizinhança, Busca Tabu, Têmpera Simulada (*Simulated Annealing*), Algoritmo Genético (AG), Enumeração Parcial, Relaxação Lagrangiana, Método *Bottleneck*, Sistemas Especialistas e Redes Neurais.

Especificamente sobre as técnicas de Programação Inteira e Dinâmica, sabe-se que estas resolvem problemas pequenos de forma ótima, mas para problemas maiores torna-se intratável com relação ao tempo computacional demandado para obtenção da solução ótima. Para contornar esta dificuldade, novas técnicas estão sendo criadas para a solução destes problemas, tanto na modelagem clássica de programação da produção quanto na solução de problemas especializados.

2.2.2 Aplicações em cadeias de suprimento, planejamento e programação de produção

Armentano e Ronconi (2000) aplicaram a meta-heurística Busca Tabu para minimizar o tempo total de atraso no problema de *flow shop* com *buffer* zero, em processo de manufatura

em série em empresas que não possuem espaço para armazenamento de intermediários. Buzzo e Moccellin (2000) desenvolveram um método heurístico híbrido combinando Algoritmo Genético e Têmpera Simulada para resolver um problema de programação de tarefas *flow shop* permutacional com o objetivo de minimizar o tempo total despendido na execução de todas as tarefas (*makespan*) e compararam a eficiência do mesmo com a aplicação dos métodos de AG e Têmpera Simulada puros.

Jeng e Lin (2005) estudaram um problema de programação da produção com minimização do tempo total para conclusão das tarefas usando uma única máquina onde o tempo de processamento de cada tarefa é do tipo função escada. Esta função depende do momento de início da tarefa. A data de entrega é a mesma para todas as tarefas. Para encontrar soluções ótimas de forma prática os autores desenvolveram um limite inferior e duas regras de eliminação para aplicar ao algoritmo *branch and bound*. A técnica se mostrou efetiva em eliminar a exploração desnecessária durante o processo de busca de soluções conseguindo, desta forma, que problemas com mais de 100 tarefas sejam resolvidos em poucos segundos.

Gupta, Koulama, Kyparisis et al. (2004) aplicaram dois algoritmos para uma variação do problema clássico de alocação de tarefas em duas máquinas com o objetivo de minimizar o tempo total de execução destas.

Aloulou, Kovalyouv e Portmann (2004) estudaram problemas de programação da produção com n tarefas a serem executadas em uma única máquina em que relações de precedência são conhecidas para o conjunto de tarefas e os momentos de início de cada uma destas são diferentes. O objetivo é encontrar uma programação semi-ativa que maximize algumas funções objetivo determinadas.

Babu, Peridy e Pinson (2004) consideraram um problema de minimização do retardo ponderado total no processamento de um conjunto de tarefas em um processador único. Apresentaram um limite mínimo baseado em decomposição Lagrangeana e uma estratégia *branch and bound*.

Chen (2004) estudou um modelo de programação da produção com máquinas paralelas e tempo de processamento das tarefas contínuo e discreto, controlados pela alocação dos recursos. O objetivo foi minimizar o custo total, incluindo o custo associado com a programação das tarefas e o custo de alocação dos recursos. Para resolver o problema, o autor desenvolveu um método *branch and bound* baseado em geração de colunas.

Blazewicz, Machowiak, Weglarz et al. (2004), com o objetivo de minimizar o *makespan,* estudaram o problema de programação ótima de n tarefas em um sistema de

processadores paralelos. As *n* tarefas podem ser executadas por vários processadores simultaneamente e a velocidade de processamento de uma tarefa é uma função linear do número de processadores alocados por esta. Um algoritmo é apresentado para resolver este problema quando todas as funções de velocidade de processamento são convexas. Com o mesmo objetivo, Vakhania (2004), estudou o problema de programação de tarefas com tempo de início e término determinados para uma única máquina. Estes problemas são do tipo *NP-hard*, entretanto, apresentam solução em tempo polinomial se todas as tarefas têm o mesmo tempo de processamento. Mokotoff e Jimeno (2002) desenvolveram um algoritmo heurístico baseado em enumeração parcial para resolver o problema de programação da produção determinístico clássico de minimizar o *makespan* de processadores paralelos desconexos. Este algoritmo consiste, basicamente, na construção de sub-problemas inteiros mistos, considerando a integralidade de alguns subconjuntos de variáveis, formulados com base em informações obtidas da solução do problema com relaxação linear.

2.2.3 Planejamento e programação de produção em refinarias de petróleo

Desde os primórdios da Pesquisa Operacional, muitos trabalhos aplicados ao Planejamento e Programação de Produção em refinarias de petróleo vêm sendo desenvolvidos na área de produção de derivados de petróleo, no gerenciamento de inventário de refinarias, na otimização do uso dos parques de tancagem e também no processo de distribuição de produtos finais aos clientes.

Diversas abordagens da área de Pesquisa Operacional vêm sendo utilizadas para resolver estes problemas de forma a conseguir resultados que gerem economia às indústrias petrolíferas. Dentre estas abordagens podem ser citadas: Programação Linear (PL), Programação Linear Inteira (PLI), Programação Linear Inteira Mista (PLIM), Programação Não Linear Inteira Mista (PNLIM), Sistemas Especialistas, meta-heurísticas, Computação Evolucionária (CE), Simulação e outras.

A utilização de técnicas de otimização para a programação da produção em refinarias é a abordagem que vem atraindo atenção crescente das companhias petrolíferas, apesar de a maioria destas ainda basearem a programação da produção de suas refinarias na utilização de planilhas de produção (BONNELLE e FELDMAN, 1999).

Conforme Magalhães, Moro, Smania et al. (1998), existem muitas razões de ordem técnica e financeira que justificam o desenvolvimento e a implantação de ferramentas computacionais direcionadas à programação da produção de refinarias de petróleo:

- a seleção de misturas ótimas de óleo cru e produtos intermediários auxilia na obtenção de produtos finais, com parâmetros dentro de uma faixa estreita e estável de especificação;
- a utilização otimizada do parque de armazenagem reduz as perdas em tancagem. Estas perdas podem ocorrer em função do tempo que o produto fica depositado no tanque, contribuindo para a formação de emulsões (dispersão de gotículas de um líquido em outro imiscível);
- expansão da capacidade de monitoramento/tomada de decisão no processo como um todo, bem como em situações de instabilidade. Como exemplo desta situação pode-se citar uma oscilação no processo de refino, tirando de especificação alguns produtos finais ou intermediários;
- possibilidade de rápida adaptação ao fornecimento de produtos requeridos em situações especiais;
- possibilidade teórica de obtenção de qualidade assegurada ao produto que sai do processo, o que pode conduzir ao fornecimento direto às distribuidoras;
- possibilidade de adoção de uma política *just-in-time* indo ao encontro das necessidades das distribuidoras;
- redução da necessidade de inventários aliada à melhoria na comunicação e resposta entre as unidades operacionais da planta.

Nos últimos 20 anos, a implantação de sistemas de controle avançado em refinarias proporcionou incrementos de produção significativos nas unidades de processo. O aumento de produtividade resultante gerou interesse por sistemas capazes de levar em conta objetivos de produção mais complexos (MORO, 2000).

2.2.4 Aplicações em planejamento e programação de produção em refinarias de petróleo

Moro (2000) desenvolveu em seu trabalho uma formulação constituída de uma estrutura genérica para modelagem das unidades de processo que compõem uma refinaria e para o relacionamento entre estas unidades. Tal formulação foi aplicada a dois problemas distintos de planejamento de refinarias de petróleo: programação das operações na área de óleo cru e na de gás liquefeito de petróleo (GLP). Estes problemas foram resolvidos por Programação Não Linear (PNL).

Faro (2001) elaborou modelos matemáticos de programação da produção para plantas químicas em batelada que incorporam modelos de desempenho, visando otimizar a sua operação por meio de técnicas de Programação Inteira Mista (PIM).

Pinto (2000) desenvolveu um modelo de planejamento não linear para a produção em refinarias com o objetivo de definir novos pontos de operação, aumentando assim a produção de derivados de maior valor agregado e, ao mesmo tempo, satisfazendo as restrições de especificações. Também aborda problemas de programação de produção em refinarias de petróleo que são formulados como modelos de otimização mista inteira e se baseiam em representações contínuas e discretas do tempo.

Giglio (2001) apresentou um modelo para maximização da margem de contribuição de uma planta petroquímica aplicando técnicas de PLIM no qual foram incluídas todas as principais alternativas e restrições operacionais. Os resultados apresentados, bem como uma análise de sensibilidade do modelo mostraram ser factível a aplicação do modelo proposto.

No final da cadeia de suprimentos em indústrias petrolíferas tem-se o setor de distribuição de produtos acabados a clientes. Neste setor, os problemas de gerenciamento e programação da produção da estocagem e distribuição de derivados são de fundamental importância, pois os produtos demandados devem ser entregues na quantidade e data estipuladas pelos seus clientes para abastecer o mercado. Stebel (2001) elaborou dois modelos para as operações de transferência e estocagem de GLP com o intuito de melhorar a capacidade de tomada de decisão do operador. Estes modelos incluem o processo de recebimento de produtos em esferas de armazenamento e posterior distribuição para cumprimento da demanda.

O setor petrolífero utiliza diversos meios para transportar a matéria-prima e seus produtos finais. Alguns destes meios são: transporte rodoviário, ferroviário, naval e dutoviário. O transporte de produtos através de dutos é eficiente, de baixo custo e evita danos ao meio ambiente. Como vários tipos de produtos podem ser transportados por dutos, a sua utilização deve ser otimizada. Magatão (2001) desenvolveu em seu trabalho modelos de auxílio ao gerenciamento das operações de um poliduto. Utilizou a metodologia baseada em PLIM com discretização uniforme do tempo. A solução do modelo proporcionou ganhos operacionais significativos se comparados à operação baseada na experiência dos operadores. Assim como Magatão (2001), Rejowski Junior (2001) também tratou do problema de seqüenciamento de produtos em dutos. Construiu seus modelos usando PLIM para o sistema dutoviário respeitando as condições de balanço de massa, de distribuição e de demanda pelos produtos a serem transportados e armazenados. Utilizou na modelagem a representação

uniforme de intervalos de tempo e, para a obtenção da solução, aplicou metodologias *Big-M* e uma alternativa baseada em relações lógicas.

Com a finalidade de desenvolver modelos matemáticos que reflitam o processo produtivo, Souzani (1999) estudou o caso real de uma planta multi-produto que produz resinas e solventes e opera em modo batelada, possibilitando gerar uma política de produção de curto prazo que atenda às demandas do mercado em um dado horizonte de tempo. Aplicou para a modelagem a representação da planta em Rede Estado-Tarefa. Modelos de PIM foram gerados e resolvidos com o método *branch and bound.*

Más (2001) desenvolveu uma estratégia de decomposição a partir de modelos de PLIM de tempo contínuo baseada em eventos, para resolver problemas de programação de suprimento de petróleo em complexo petrolífero contendo portos, refinaria e uma infra-estrutura de oleodutos. Utilizou o método *branch and bound* baseado na relaxação por PL.

Shah (1996) também tratou do problema de programação da produção para suprimento de óleo cru em uma refinaria usando técnicas de programação matemática. As decisões relacionadas a este problema incluem a alocação de óleo cru para a refinaria e tanques no porto, a conexão dos tanques da refinaria com as unidades de destilação de crus, o seqüenciamento e quantidade de crus bombeados do porto para a refinaria e os detalhes relacionados com descarga dos tanques no porto. Mostrou como as técnicas de programação matemática podem ser aplicadas neste tipo de problema e as vantagens na sua utilização.

Castro, Barbosa-Póvoa e Matos (2002) estudaram a otimização de um problema de programação da produção de curto prazo em uma planta com três linhas paralelas de produção de polímeros. Os equipamentos desta planta requerem limpeza entre as trocas de produto. Uma ferramenta computacional de uso fácil foi desenvolvida e consiste de um modelo geral de programação da produção aliado às capacidades do pacote computacional *Microsoft Excel*. O modelo de programação da produção baseia-se em Rede Estado-Tarefa com discretização do tempo que conduz a problemas de PLIM.

Lima, Barbosa e Beal (2003) concluíram que a aplicação de técnicas de simulação a problemas de transferência e estocagem em indústrias petrolíferas são apropriadas, pois através da utilização de modelos probabilísticos que representam as trocas (mercado, suprimento, produção, laboratório, etc), é possível simular os custos logísticos e o nível de serviço em diferentes cenários, subsidiando decisões técnicas e gerenciais que levam à otimização dos estoques.

Finalmente, Giannelos e Georgiadis (2002) propuseram uma nova formulação matemática para a programação da produção de processos contínuos com múltiplos propósitos usando uma representação de tempo por eventos, gerando um modelo PLIM.

2.3 COMPUTAÇÃO EVOLUCIONÁRIA

Nas últimas três décadas, pesquisadores vêm estudando com mais afinco características e princípios da natureza para criar modelos computacionais inteligentes. Físicos, biólogos e outros cientistas tentam desvendar os princípios que regem os fenômenos da natureza, enquanto que os matemáticos, cientistas da computação e engenheiros buscam idéias que possam ser copiadas ou pelo menos imitadas, e então aplicadas para solucionar problemas que a ciência atual ainda não consegue resolver satisfatoriamente (TANOMARU, 1995). Destes estudos surgiu a CE que é considerada um ramo da ciência da computação que se fundamenta em um novo paradigma para a resolução de problemas, não exigindo o conhecimento de uma sistemática prévia de resolução e que se baseia nos mecanismos encontrados na natureza, à luz da teoria da evolução natural de Darwin.

Segundo Coelho e Coelho (1999), o paradigma computacional da CE imita um modelo rudimentar e simplificado da natureza como um processo adaptativo de busca e otimização que possibilite implementações computacionais. Esses modelos produzem sistemas baseados no princípio da evolução e hereditariedade, e possuem uma população de soluções potenciais, trabalham com alguns processos de seleção baseados na aptidão dos indivíduos e alguns operadores genéticos. Os algoritmos evolutivos se baseiam num processo de aprendizagem coletivo de uma população de indivíduos, cada um destes representando um ponto do espaço de busca de soluções potenciais de um determinado problema (BÄCK e SCHWEFEL, 1993). Esses indivíduos visam melhorar a sua adequação em relação ao meio ambiente, ou seja, o seu desempenho geral com respeito a um dado problema (GOLDBERG, 1989).

Tanomaru (1995) observa que a CE engloba um número crescente de paradigmas e métodos, dos quais os mais importante são o AG, Programação Evolucionária (PE), Estratégias Evolucionárias (EEs), Programação Genética (PG) e Sistemas Classificadores, entre outros. Destes, o mais difundido e pesquisado é o AG, pela sua flexibilidade, relativa simplicidade de implementação e eficácia em realizar busca global em ambientes diversos. Recentemente, o escopo da CE foi ampliado pela inclusão de novas abordagens, entre elas a Transgenética Computacional (GOLDBARG e GOLDBARG, 2002).

2.4 ALGORITMO GENÉTICO (AG)

2.4.1 Introdução

O AG foi inicialmente proposto por Holland (1975) que se inspirou no mecanismo de evolução das espécies, tendo como base os trabalhos de Darwin sobre a origem das espécies e a genética natural devida principalmente a Mendel. De acordo com a teoria Darwiniana de evolução das espécies, uma população sujeita a um ambiente qualquer, sofrerá influências deste, de tal forma que os mais aptos terão maior probabilidade de sobreviver a tal ambiente. Já os trabalhos de Mendel mostram como o material genético dos pais podem ser passados para os descendentes. Desta forma, a cada geração haverá uma população mais adaptada ao ambiente em questão. Em contraste com estratégias evolutivas e programação evolutiva, o objetivo original de Holland (1975) não foi criar algoritmos para resolver problemas específicos, mas sim formalizar o estudo do fenômeno de adaptação como ocorre na natureza e desenvolver caminhos nos quais os mecanismos da adaptação natural sejam inseridos em sistemas computacionais. No entanto, ao representar o processo evolutivo partindo do modelo de cromossomos, Holland foi capaz de encontrar um caminho de grande e imediata aplicação prática na determinação de máximos e mínimos de funções matemáticas, facilitando a aceitação do AG no meio acadêmico (GOLDBARG E LUNA, 2000).

Dentre as definições de AG encontradas na literatura tem-se a de Tanomaru (1995) que o define como métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em AG, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.

O AG é considerado muito eficiente para busca de soluções ótimas, ou quase ótimas em uma grande variedade de problemas, pois não impõe muitas das limitações encontradas nos métodos de busca tradicionais. Miranda (2005) destaca uma das vantagens do AG como sendo a simplificação que estes permitem na formulação e solução de problemas de otimização, sendo desta forma indicado para a solução de problemas de otimização NP-Completos envolvendo um grande número de variáveis e, conseqüentemente, espaços de soluções de dimensões elevadas. Além disso, em muitos casos em que outras estratégias de otimização falham na busca de uma solução, o AG converge.

Tanomaru (1975) classificou o AG como método probabilístico de busca e otimização, embora sejam muito diferentes dos algoritmos aleatórios, pois combinam elementos de buscas diretas e estocásticas. Mesmo parecendo simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca mais poderosos e robustos do que os métodos de busca direta existentes.

Segundo Goldberg (1989), o AG difere dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

- opera numa população de pontos e não a partir de um ponto isolado;
- opera num espaço de soluções codificadas e não no espaço de busca diretamente;
- necessita somente de informação sobre o valor de uma função objetivo para cada membro da população e não requerem derivadas ou qualquer outro tipo de conhecimento;
- usa transições probabilísticas e não regras determinísticas.

2.4.2 Componentes básicos e estrutura do Algoritmo Genético

O AG emprega uma terminologia originada da teoria da evolução natural e da genética. Um indivíduo da população pode ser formado por um ou mais cromossomos. Quando este indivíduo é representado por um único cromossomo, pode-se utilizar o termo indivíduo ou cromossomo indistintamente. Os cromossomos, também denominados *string*, em geral, são implementados na forma de vetores, onde cada componente do vetor é conhecido como gene (Figura 3). Os possíveis valores que um determinado gene pode assumir são denominados alelos. Cada gene possui um local fixo no cromossomo, denominado de lócus. O conjunto composto por cromossomo, genes e alelos denomina-se genótipo e as características conferidas por este formam o fenótipo.

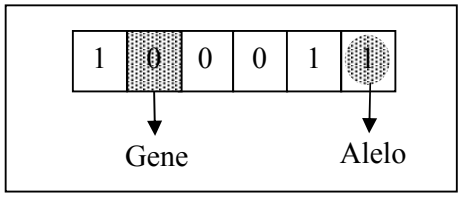

Figura 3: Exemplo de um Cromossomo

A terminologia comum ao AG e à biologia aparece resumida na Tabela 1.

Tabela 1: Relação da Terminologia do AG com a Biologia

Biologia	**Algoritmo Genético (AG)**
Cromossomo	Indivíduo (*string*)
Gene	Bit
Alelo	Valor do bit
Lócus	Posição de um bit específico no indivíduo ou *string*
Genótipo	Indivíduo candidato à solução – x
Fenótipo	Valor da função para um dado indivíduo – $f(x)$

Para aplicação de AG a um problema qualquer se deve num primeiro passo representar cada possível solução $\boldsymbol{x}$ no espaço de busca como uma seqüência de símbolos $\boldsymbol{s}$ (cromossomo), gerados a partir de um dado alfabeto finito A. No caso mais simples, usa-se o alfabeto binário $A = \{0,1\}$ e, no caso geral, tanto o método de representação quanto o alfabeto genético dependem de cada problema. Considera-se o genótipo como sendo a variável independente x e o fenótipo como a variável dependente ou função $f(x)$.

No AG trabalha-se com um conjunto de indivíduos (população) no qual cada elemento é candidato a ser a solução desejada. A função a ser otimizada representa o ambiente no qual a população inicial se insere. Espera-se então que, através dos mecanismos de evolução das espécies e da genética natural, os indivíduos mais aptos tenham maior probabilidade de se reproduzirem e que a cada nova geração estejam mais aptos ao ambiente.

A próxima geração será uma evolução da anterior e para que isso ocorra, os mais aptos deverão possuir maior probabilidade de serem selecionados para dar origem à nova geração. Contudo, alguns poucos não muito aptos também poderão ser selecionados. O mecanismo responsável que faz a escolha seletiva de indivíduos denomina-se seleção. Após a seleção, o próximo passo é a aplicação dos operadores genéticos que atuam sobre os genótipos produzindo novos indivíduos, também denominados de mecanismos de busca (HOLLAND, 1975). Dentre estes mecanismos, os mais comumente empregados são o cruzamento ou recombinação e a

mutação. O esquema de seleção e os mecanismos de recombinação e mutação são chamados de operadores do AG. Se as operações de seleção, recombinação e mutação forem bem conduzidas, espera-se que a nova geração seja, em média melhor do que a que lhe deu origem.

Um algoritmo genérico capaz de englobar a maioria dos Algoritmos Genéticos existentes seria (BARBOSA, 1997):

Algoritmo AG Genérico
Inicialize a população
Avalie indivíduos na população
Repita
Selecione indivíduos para reprodução
Aplique operadores de recombinação e mutação
Avalie indivíduos na população
Selecione indivíduos para sobreviver
Até critério de parada satisfeito
Fim

Existem inúmeras estruturas de AG. O que distingue uma estrutura da outra são as variações que ocorrem com relação aos procedimentos e operadores empregados (CASTRO, 2001).

Dois tipos distintos de AG podem ser citados: geracional e de estado estacionário (*steady state*), os quais diferem em relação à maneira como os indivíduos criados são inseridos na população.

2.4.3 AG geracional

Neste algoritmo, toda a população é substituída por novos indivíduos gerados pelo processo de seleção e aplicação dos operadores genéticos. O algoritmo para esta estrutura é da seguinte forma (CASTRO, 2001).

Algoritmo AG Geracional

Inicialize a população P de cromossomos

Avalie indivíduos na população P

Repita

Selecione dois indivíduos em P para reprodução

Aplique o operador de recombinação com probabilidade p_c

Aplique o operador de mutação com probabilidade p_m

Insira novo indivíduo em uma nova população P'

Até que *P'* esteja completa

Avalie indivíduos na população P'

$P \leftarrow P'$

Até objetivo final ou máximo de gerações

Fim

Nesta estrutura, pode-se observar que, como toda a geração anterior (de pais) é inteiramente substituída pela outra (de filhos), não existe convivência, ocorrendo desta forma uma perda de bons indivíduos no processo. Por este motivo, especialmente em problemas de otimização, um procedimento freqüentemente empregado é o elitismo, ou seja, o(s) melhor(es) indivíduo(s) de uma geração é(são) preservado(s): uma cópia é passada diretamente para a geração seguinte.

2.4.4 AG de estado estacionário

Nesta estrutura apenas um indivíduo é criado de cada vez. Depois de sua avaliação, este será inserido na população em substituição a algum outro elemento como, por exemplo, aquele que apresenta a pior aptidão. Caso ele seja pior que todos os já existentes, então nada é alterado e procede-se uma nova criação de indivíduo. Em geral, neste tipo de estrutura costuma-se ordenar a população por valor de aptidão, para facilitar a substituição do pior indivíduo (CASTRO, 2001). O algoritmo para este tipo de AG é:

Algoritmo AG de estado estacionário

Inicialize a população *P* de cromossomos

Avalie indivíduos na população *P*

Repita

Selecione operador genético

Selecione indivíduo(s) para reprodução

Aplique o operador genético selecionado

Avalie o indivíduo(s) gerado(s)

Selecione indivíduo para sobreviver

Se este indivíduo é melhor que o pior elemento de *P* **então**

Insira este indivíduo em *P* de acordo com sua aptidão

Até objetivo final ou máximo de gerações

Fim

2.4.5 Geração da população Inicial

Tendo em vista que o AG trabalha com manipulação de caracteres de determinados alfabetos, deve-se especificar a codificação do cromossomo (indivíduo) para que este represente de forma conveniente pontos do espaço de busca do problema. Na maior parte das aplicações, a população inicial é gerada aleatoriamente ou através de algum processo heurístico que evite indivíduos infactíveis. É importante que a população inicial cubra a maior parte possível do espaço de busca.

Reeves (1995) relata que direcionar a construção da população com auxílio de alguma heurística, de modo a obter indivíduos com boa qualidade, pode auxiliar o AG a encontrar melhores soluções mais rapidamente, se comparada com a geração aleatória dos cromossomos. Entretanto, pode ocorrer um aumento nas chances de uma convergência prematura, o que não é conveniente para a resolução do problema.

2.4.6 Função aptidão e avaliação da população

O grau de adaptação de cada indivíduo é chamado de aptidão ou *fitness*. É obtido pela avaliação do cromossomo através da função a ser otimizada. Pode-se definir a aptidão como

sendo a quantificação da adaptação do indivíduo, ou seja, é o valor obtido com a aplicação deste à função custo. Se o objetivo for maximizar, a aptidão é diretamente proporcional ao valor da função. Caso o objetivo seja a minimização, a aptidão será inversamente proporcional ao valor da função. Contudo, deve-se observar que o termo minimização não é bem aceito por alguns pesquisadores por não ter inspiração biológica, pois os indivíduos mais aptos é que devem ter maiores chances de sobreviver.

Quanto maior for o valor de aptidão, maiores as chances de o indivíduo sobreviver no ambiente e reproduzir-se, passando parte de seu material genético a gerações posteriores (TANOMARU, 1995).

Segundo Falcão e Borges (2001), nos problemas de otimização sem restrições, o valor de aptidão de um indivíduo pode corresponder ao valor da função objetivo. Nos problemas de otimização com restrições, a abordagem mais comum é a utilização da função de aptidão associada a uma função de penalidade.

A avaliação da população é efetuada após cada aplicação dos operadores seleção, recombinação e mutação. Consiste na obtenção do grau de adaptação da população atual para análise da convergência e/ou continuidade do processo. Quando o grau de adaptação for aceitável, o melhor indivíduo desta geração provavelmente será a solução desejada. Este grau pode ser determinado pela diferença entre a aptidão do primeiro e a do último elemento da lista de cromossomos. A definição dessa diferença aceitável está intimamente ligada ao conhecimento do problema.

2.4.7 Operadores de seleção

O operador denominado de seleção é considerado um operador Darwiniano, pois tem inspiração na teoria da evolução das espécies de Darwin (1981). Segundo Bäck, Fogel e Michalewicz (2000) este operador é usado para direcionar o processo para as melhores regiões do espaço de busca. Sua função é selecionar indivíduos da população para a reprodução, dando preferência aos indivíduos mais adaptados ao ambiente (MITCHELL, 1996).

A seguir estão descritos os principais operadores de seleção.

2.4.7.1 Roleta simples ou seleção proporcional

Este método foi proposto por Holland (MITCHELL,1996) e ainda hoje é uma dos mais empregados em AG. Esta técnica estabelece que a probabilidade p_i do *i*-ésimo indivíduo da população vir a ser selecionado para reprodução é proporcional à sua aptidão relativa. Uma possível implementação corresponde a tomar:

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j} \tag{7}$$

onde $f_i = f(x_i)$ é assumida positiva e *m* é o número de indivíduos da população. Uma vez definida a forma de quantificação da probabilidade de sobrevivência de cada indivíduo da população, emprega-se o "método da roleta". Neste método, cada indivíduo da população é representado proporcionalmente ao seu índice de aptidão. Assim, os indivíduos com alta aptidão recebem uma porção maior da roleta, enquanto que os de baixa aptidão ocuparão uma porção relativamente menor. O sorteio da roleta é efetuado *m* vezes para escolher os indivíduos que farão parte da população na próxima geração.

O método da roleta tem a desvantagem de possuir uma alta variância, podendo levar a um grande número de cópias de um cromossomo com alto valor de aptidão, o que faz diminuir a diversidade da população. Esta falha pode ocasionar uma convergência prematura do algoritmo para uma solução local. Por outro lado, quando a evolução está avançada, observa-se uma estagnação do algoritmo, isto é, uma baixa pressão de seleção entre os indivíduos.

2.4.7.2 Roleta ponderada ou seleção por ordenação

Esta técnica apresenta os indivíduos ordenados conforme desempenho. Contudo, a distância entre os indivíduos próximos é reduzida, ou seja, a pressão seletiva é atenuada. Cada indivíduo recebe uma nota, sendo que o pior recebe nota "1" e o melhor fica com a nota igual ao tamanho da população (distância de "1" entre cada indivíduo vizinho). O primeiro em aptidão tem a maior probabilidade de seleção. Nesta técnica a pressão seletiva é atenuada.

2.4.7.3 Seleção por torneio

Nesta forma de seleção, a competição não ocorre entre todos os indivíduos da população, mas em um subconjunto desta. A idéia consiste em escolher-se *k* indivíduos da população, extrair o melhor indivíduo deste grupo para uma população intermediária. Este procedimento é repetido até que a população intermediária tenha o mesmo número de indivíduos da população da geração anterior.

2.4.7.4 Seleção por truncamento

Nesta técnica, uma população de pais com *n* indivíduos gera $m > n$ filhos, dos quais os *n* melhores são selecionados como pais para a próxima geração (BÄCK, FOGEL e MICHALEWICZ, 2000). Esta seleção não é dependente do valor de aptidão dos indivíduos da população, ou seja, os *n* melhores indivíduos sempre serão escolhidos.

2.4.7.5 Seleção proposta por Mayerle (1994)

Muitas vezes costuma-se empregar uma função de seleção, na qual os indivíduos com melhor aptidão têm maior probabilidade de serem escolhidos para operações genéticas. Considerando uma população com seus indivíduos dispostos em ordem crescente (se o problema for de minimização) ou decrescente (se o problema for de maximização) com relação aos valores de aptidão, a seleção de um indivíduo é feita de forma a privilegiar indivíduos com melhor valor de aptidão, considerando uma distribuição de probabilidade inversamente proporcional ao índice dos indivíduos na população. Assim, quanto menor o índice, maior é a sua probabilidade de escolha. Correa, Steiner, Freitas et al. (2004) aplicaram em seu trabalho a seleção proposta por Mayerle (1994) em um AG do tipo estado estacionário. A fórmula para este operador de seleção é:

$$Select(R) = \left\{ r_j \in R \,/\, j = m + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.rnd.\left(m^2 + m\right)}}{2} \right\rceil \right\} \tag{8}$$

onde:

- R é o conjunto dos m cromossomos;
- r_j é o j-ésimo cromossomo;
- $rnd \in [0,1)$ é um número aleatório uniformemente distribuído;
- $\lceil x \rceil$ é a função que retorna o menor inteiro maior que x.

2.4.7.6 Seleção elitista

Este tipo de seleção é normalmente acoplado a outros métodos de seleção. Consiste em copiar ou reproduzir os melhores indivíduos da população atual para a próxima geração, garantindo que estes cromossomos não sejam destruídos nas etapas de recombinação e mutação.

A principal vantagem é o fato de se garantir a convergência, ou seja, caso o ótimo global seja descoberto durante o processo de busca, o AG deve convergir para tal solução. Sua desvantagem é a possibilidade de forçar a busca, pela presença de mais de uma cópia do melhor indivíduo em direção de algum ponto ótimo local que tenha sido descoberto antes do global, embora normalmente o AG consiga "escapar de tais armadilhas". O que pode ser feito é guardar separadamente a melhor solução encontrada durante a evolução, para no final da execução designá-la como o indivíduo ótimo encontrado, mesmo que ele não esteja presente na última geração do processo.

2.4.8 Operadores genéticos

Os operadores genéticos são mecanismos de busca que se destinam à manipulação dos indivíduos selecionados de uma geração anterior, visando à obtenção de indivíduos mais aptos. Tais operadores são basicamente de dois tipos: recombinação ou cruzamento e mutação.

2.4.8.1 Recombinação ou cruzamento

A recombinação é o operador responsável pela propagação das características dos indivíduos mais aptos da população (pais) por meio de troca de segmentos de informações entre estes, o que dará origem a novos indivíduos. Diferentes mecanismos de recombinação

são utilizados. Existem vários tipos de cruzamento e estes diferem entre si pela escolha do lócus do cromossomo a ser trocado entre os cromossomos pais e pela maneira como esta troca é efetuada. Os tipos de recombinação ou cruzamento descritos na literatura variam de acordo com a codificação do cromossomo. Alguns destes cruzamentos, separados por tipo de codificação são:

Codificação Binária:

- Cruzamento Uniforme: consiste no emparelhamento dos dois cromossomos pais e cada lócus do cromossomo têm 50% de chance de ser trocado. A Figura 4 mostra um exemplo onde supõe-se que um determinado cromossomo que possui 8 lócus, sofreu cruzamento uniforme em 3 deles: o primeiro, o quarto e o quinto lócus.

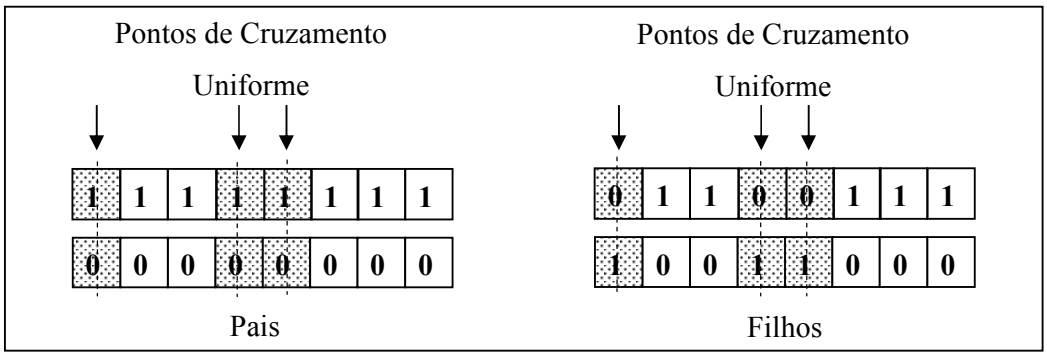

Figura 4: Exemplo 1 de Cruzamento Uniforme

- Cruzamento de um Ponto: Um ponto de corte é escolhido aleatoriamente e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores a este ponto no outro pai (Figura 5).

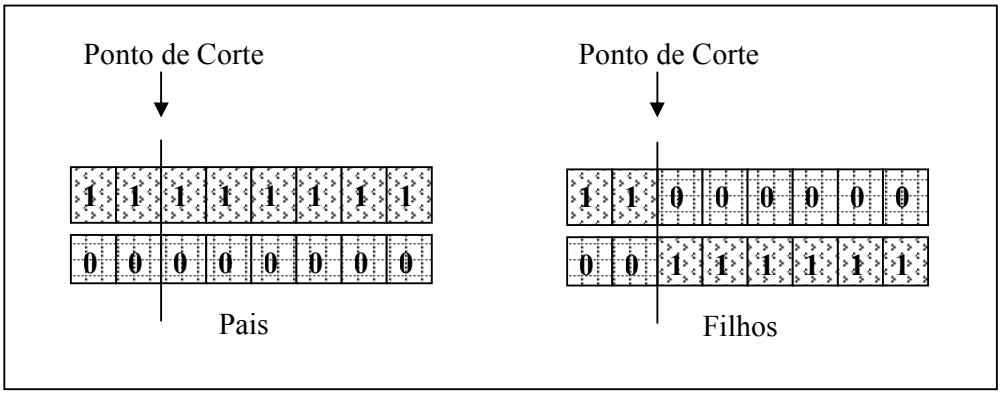

Figura 5: Exemplo de Cruzamento de um Ponto

- Cruzamento de dois Pontos: Dois pontos de corte são escolhidos aleatoriamente e a partir destes pontos as informações genéticas dos pais serão trocadas. Todo o material genético dos pais existente entre os dois pontos de corte são trocados e o restante mantém-se inalterado (Figura 6).

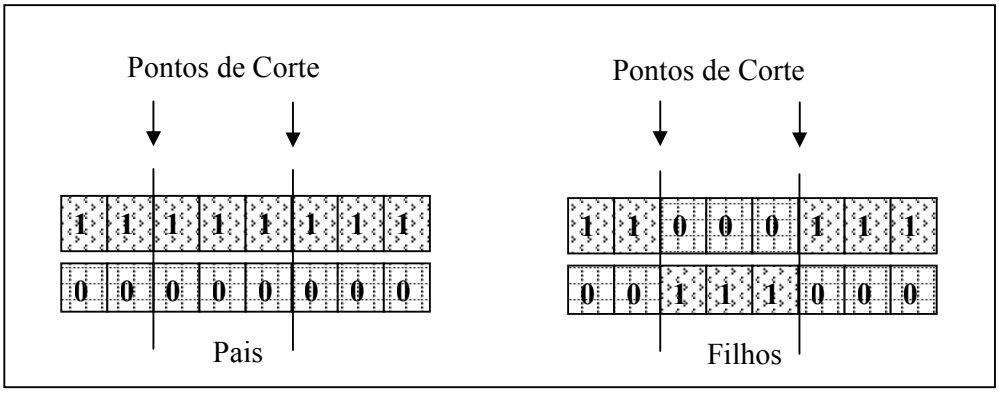

Figura 6: Exemplo de Cruzamento de dois Pontos

Codificação Real :

O AG pode ser construído utilizando-se uma codificação real para as variáveis, o que é mais natural quando estas já são, por definição, contínuas. Assim, o cromossomo passa a ser um vetor em $\Re^n$ que coleciona as n variáveis reais que caracterizam a solução do problema. Uma vantagem da codificação real é o fato de, em geral, ser mais intuitivo conceber operadores de recombinação para um dado problema lançando mão de conhecimento já previamente adquirido no domínio da aplicação. Nos operadores mostrados a seguir utiliza-se a seguinte notação para os cromossomos (BARBOSA, 1997):

$$\text{Cromossomo 1: } C_1 = \left(c_1^1, c_2^1, ...c_n^1\right) \tag{9}$$

$$\text{Cromossomo 2: } C_1 = \left(c_1^2, c_2^2, ...c_n^2\right) \tag{10}$$

- Cruzamento Discreto: Este operador de recombinação simplesmente escolhe para cada variável um dos valores assumidos por esta variável nos cromossomos "paternos". Assim é gerado o cromossomo "filho" $F = \left(f_1, f_2, ..., f_n\right)$ onde cada f_i é igual a c_i^1 ou c_i^2 com igual probabilidade.
- Cruzamento Uniforme: Neste caso o valor de cada componente f_i é escolhido de maneira aleatória (com probabilidade uniforme) entre os limites definidos pelos valores encontrados nos cromossomos "paternos", isto é, $c_i^1 \le f_i \le c_i^2$.
- Cruzamento de um Ponto: Em analogia ao cruzamento de um ponto para a codificação binária pode-se definir um operador tal que produza os "filhos":

$$F_1 = \left(c_1^1, c_2^1, ...c_i^1, c_{i+1}^2, ..., c_n^2\right) \text{ e } F_1 = \left(c_1^2, c_2^2, ..., c_i^2, c_{i+1}^1, ..., c_n^1\right) \tag{11}$$

  quando é sorteada a i-ésima posição para o corte.
- Cruzamento Aritmético: Neste caso os "filhos" $F_i = \left(f_1^i, f_2^i, ...f_n^i\right)$ são definidos por:

$$f_i^1 = \lambda c_i^1 + (1-\lambda)c_i^2 \text{ e } f_i^2 = \lambda c_i^2 + (1-\lambda)c_i^1 \tag{12}$$

  onde $\lambda \in [0,1]$.
- Cruzamento BLX$_\alpha$: Gera o "filho" $F = \left(f_1, f_2, ..., f_n\right)$ com f_i escolhido, com probabilidade uniforme, no intervalo $\left[c_{min} - I.\alpha, c_{max} + I.\alpha\right]$ onde $c_{min} = min\left(c_i^1, c_i^2\right)$, $c_{max} = max\left(c_i^1, c_i^2\right)$ e $I = c_{max} - c_{min}$. Pode-se observar que para $\alpha = 0$ tem-se o cruzamento uniforme.
- Cruzamento Heurístico de Wright: Para este operador sorteia-se um número real $r \in [0,1]$ com probabilidade uniforme e, denotando por C_1 o indivíduo com maior aptidão entre os cromossomos selecionados como "pais", define-se o cromossomo "filho" com componentes:

$$f_i = r.\left(c_i^1 - c_i^2\right) + c_i^1 \tag{13}$$

Problemas de Permutação:

Uma classe de problemas que apresenta características especiais quando resolvidos pelo AG é aquela em que um alfabeto de cardinalidade $n > 2$ é utilizado para resolver problemas em que se busca uma permutação ótima de um certo conjunto de cardinalidade também n.

- Cruzamento PMX^2 (cruzamento com mapeamento parcial): Dois pontos de corte são escolhidos aleatoriamente. A Figura 7 ilustra este tipo de cruzamento para um cromossomo com $n = 8$.

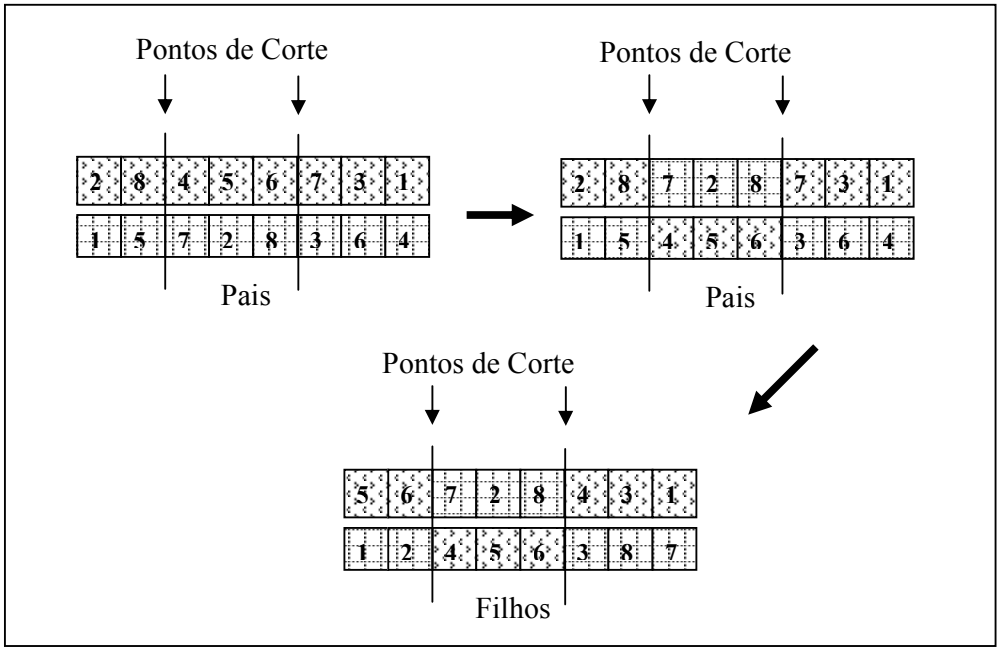

Figura 7: Exemplo de cruzamento PMX^2

- Cruzamento C1: Um ponto de corte é escolhido aleatoriamente. Para formar o primeiro "filho" toma-se as variáveis antes do corte do primeiro "pai" e completa-se o cromossomo com as variáveis do segundo "pai" na ordem deste cromossomo, excetuando-se os que vieram do primeiro "pai". A Figura 8 mostra um exemplo de geração de "filhos" através deste operador (REEVES, 1995).

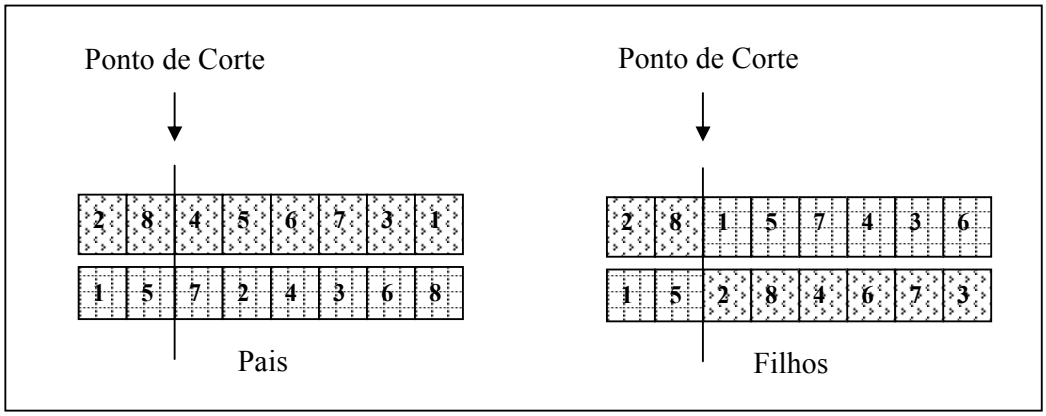

Figura 8: Exemplo de Cruzamento C1

- Cruzamento Uniforme: é gerada uma máscara formada com os dígitos "0" e "1" com o mesmo número de elementos do cromossomo. O elemento igual a "1" indica que o cromossomo do primeiro "filho" herdará o valor da variável encontrada naquele lócus do primeiro "pai". Os outros lócus deste "filho", onde a máscara possui valor "0", serão preenchidos com os valores encontrados no segundo "pai" na mesma ordem, da esquerda para a direita, não se repetindo as variáveis já preenchidas. O segundo "filho" é gerado de maneira semelhante (REEVES, 1995), (Figura 9).

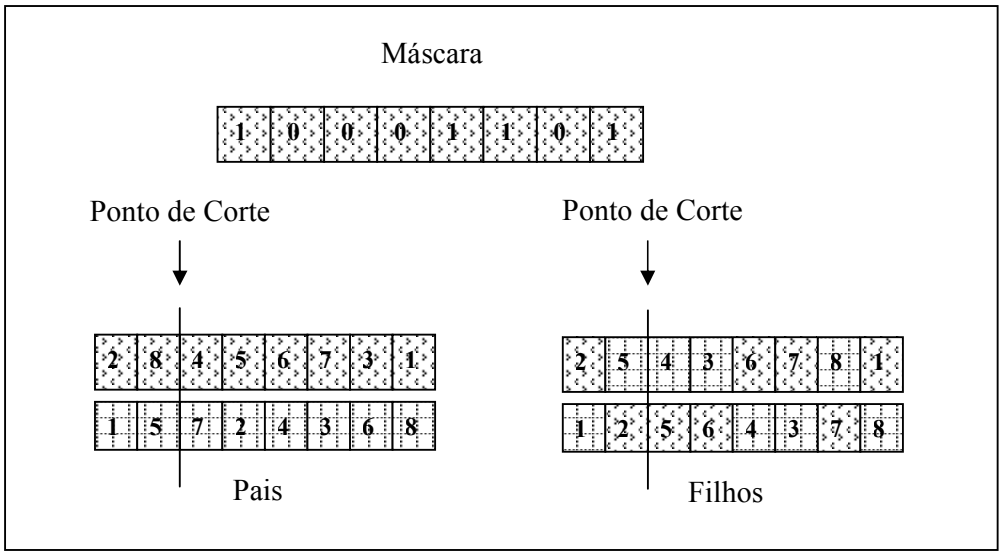

Figura 9: Exemplo 2 de Cruzamento Uniforme

Outros operadores de recombinação podem ser encontrados em Souza (2004), Mitchell (1996), Hartmann (2001), Bernal-Haro, Azzaro-Pantel, Domenech et al. (1998), Deb e Pal (2004), Ilkyeong e Jieom (2000), Correa, Steiner, Freitas et al. (2004) e Velloso, Mendonça, Pacheco et al. (1995). Muitos operadores de recombinação que não se adaptam aos operadores convencionais são construídos por especialistas para serem aplicados a problemas específicos.

2.4.8.2 Mutação

Segundo Barbosa (1997), a idéia que gerou o operador de mutação também é oriunda da Genética e visa preservar alguma diversidade genética na população que vai se perdendo com a evolução e os indivíduos vão se tornando mais e mais semelhantes, tendendo a uma solução que, nos casos de sucesso do AG, é uma boa solução do problema. Goldberg (1989) confirma que, no AG, o operador de mutação executa um papel secundário, porém necessário, pois possibilita restaurar a diversidade genética eventualmente perdida durante o processo evolutivo.

O operador de mutação é equivalente à busca aleatória. Basicamente, seleciona-se uma posição de um cromossomo e muda-se aleatoriamente o valor do gene correspondente para um outro alelo possível. Desta maneira, consegue-se inserir novos elementos na população. É possivelmente o operador genético mais simples de ser implementado.

Esse operador é importante para que ocorra a introdução e manutenção da diversidade genética na população, possibilitando inclusive, recuperar algum bom material genético que possa ter sido perdido após sucessivas recombinações. Ainda, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca seja zero e serve para evitar a convergência prematura do AG para soluções sub-ótimas.

Dentre os principais mecanismos de alteração genética que recebem a denominação global de mutação podem ser citados:

Codificação binária:

- Mutação Simples: Uma posição do cromossomo é sorteada e o gene correspondente é invertido, isto é, se for “1” ele passa a ser “0” e vice-versa (Figura 10).

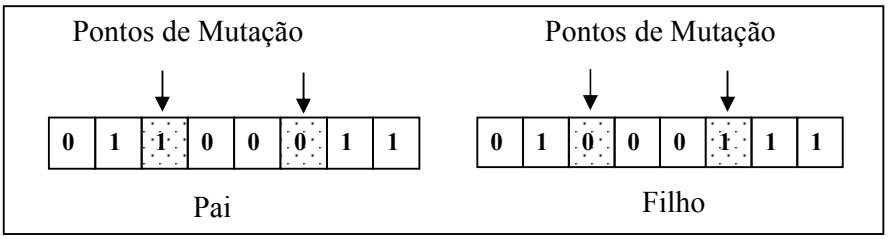

Figura 10: Exemplo de Mutação Binária Simples

Codificação Real (BARBOSA, 1997):

- Mutação Aleatória: Escolhe-se aleatoriamente um elemento c_i do cromossomo e altera-se este valor para algum $c_i^0 \in [a_i, b_i]$ sorteado com probabilidade uniforme.

Problemas de Permutação:

- Mutação por Troca (*Swap)*: Consiste na troca aleatória de posições entre dois genes (Figura 11).

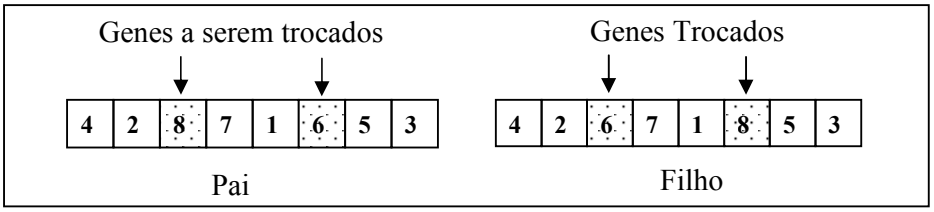

Figura 11: Exemplo de Mutação por Troca (*Swap*)

- Mutação por Mudança (*Shift*): Consiste em escolher dois genes aleatoriamente, retirar o primeiro gene escolhido, deslocar todos os genes entre os dois genes escolhidos em direção à posição do gene retirado. Em seguida, inserir o segundo gene escolhido na posição que ficou vazia e finalmente trocar o segundo gene pelo primeiro (SOUZA, 2004), (Figura 12).

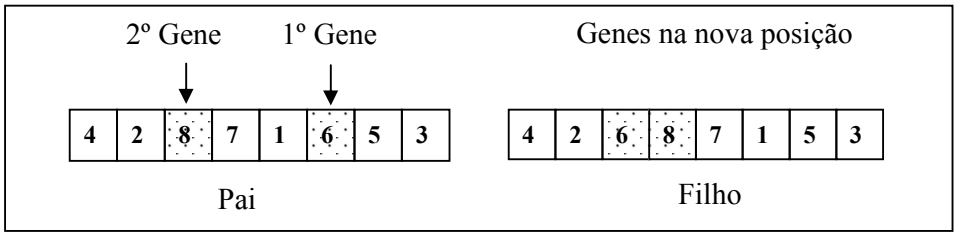

Figura 12: Exemplo de Mutação por Mudança (*Shift*)

- Mutação por Inversão: Uma cadeia de genes é retirada do cromossomo, invertida e recolocada na mesma posição (Figura 13).

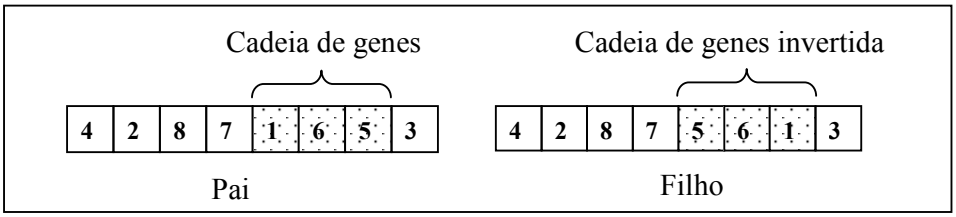

Figura 13: Exemplo de Mutação por Inversão

- Mutação por Translocação: Nesta operação, uma cadeia de genes é retirada do cromossomo e colocada em outra posição, na mesma ordem em que foi retirada (Figura 14).

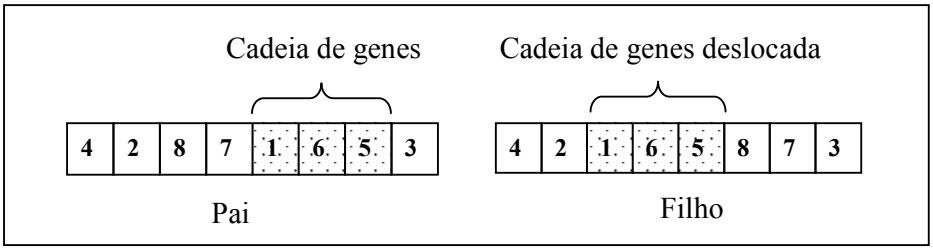

Figura 14: Exemplo de Mutação por Translocação

- Hipermutação Heurística: este operador utiliza conhecimento do problema que está sendo resolvido. É aplicado logo após a geração da população e nas iterações do AG, com uma probabilidade determinada. Primeiro, uma porcentagem de indivíduos da população é selecionada. Então, para cada um destes indivíduos, faz-se a troca de cada gene por valores que não estejam no cromossomo e avalia-se. Os melhores resultados obtidos para todos os indivíduos são inseridos na população. Correa, Steiner e Freitas et al. (2004) utilizaram este operador em um problema de localização de facilidades.

2.4.9 Parâmetros do AG

Além da escolha da codificação do cromossomo, da função de aptidão, do método de geração da população inicial e dos operadores de seleção, cruzamento e mutação, existem

vários parâmetros do AG que podem ser escolhidos para melhorar o seu desempenho. A influência de cada parâmetro no desempenho do AG depende da classe de problemas que se está tratando. Assim, a determinação de um conjunto de valores otimizado para estes parâmetros dependerá da realização de um grande número de experimentos e testes. Existem vários estudos sobre a escolha dos parâmetros na literatura. Infelizmente, não há regras claras para a escolha desses parâmetros, e experimentos com bons resultados são tomados como base. Entre os parâmetros que serão aqui analisados, são considerados os mais importantes: o tamanho da população, o número de gerações, a probabilidade de cruzamento e a probabilidade de mutação.

2.4.9.1 Tamanho da população

O número de indivíduos da população afeta o desempenho global e a eficiência do AG. Segundo Reeves (1995), pequenas populações podem provocar um sério risco de não obter cobertura do espaço de busca, enquanto que grandes populações podem exigir um esforço computacional excessivo para a resolução do problema. Goldberg (1985), concluiu que para cromossomos binários de comprimento l, o tamanho ótimo deve ser uma função exponencial de l. Ainda, Tanomaru (1995) afirma que, na prática, valores da ordem de 50 a 200 cromossomos resolvem a maior parte dos problemas, mas populações maiores podem ser necessárias para problemas mais complexos. Em casos em que a avaliação de um cromossomo é excessivamente lenta é mais conveniente o uso de pequenas populações. Os experimentos de De Jong, apud Mitchell (1996), indicam que o melhor tamanho para a população é entre 50 e 100 indivíduos. Esta faixa de valores é amplamente utilizada em aplicações com AG.

2.4.9.2 Número de gerações

O número de gerações depende da complexidade do problema de otimização e deve ser determinado experimentalmente. Como o AG resolve problemas de otimização, o ideal seria que o algoritmo terminasse assim que o ponto ótimo fosse encontrado. No entanto, na prática, o que ocorre na maioria das aplicações é que não se pode afirmar com certeza se um dado ponto ótimo corresponde a um ótimo global. Desta forma, algum critério deve ser

especificado para o término do processamento do algoritmo. Normalmente, usa-se o critério do número máximo de gerações ou um tempo limite de processamento para encerrar o processo. Outro critério utiliza a idéia de estagnação, ou seja, o processo é encerrado quando não se observa melhoria da população depois de várias gerações consecutivas. Neste caso, a análise da convergência pode ser feita através de vários fatores, tais como: valores máximo, mínimo, médio, desvio padrão da população e outros. Goldberg (1989) considera o desvio padrão dos valores de aptidão dos indivíduos da população como um dos parâmetros mais utilizados. Assim, tem-se uma comparação do desempenho da geração atual com a anterior e se o desvio padrão for igual ou menor ao estabelecido como aproximação aceitável, o processo de busca é finalizado. Nesses casos, o número de gerações não é fornecido no início do processo e é conhecido somente quando o critério de parada for satisfeito.

2.4.9.3 Taxa de cruzamento

É a taxa que determina se será feito o cruzamento entre dois cromossomos. Para tanto, gera-se um número aleatório no intervalo $[0,1]$e compara-se à taxa. Se o número for menor que a taxa, o cruzamento é efetuado. Na literatura, geralmente encontram-se valores entre 0,6 e 0,65 para taxa de cruzamento ou recombinação (p_c). Tanomaru (1995) relata que, em relação à probabilidade de cruzamento, estudos empíricos têm mostrado que bons resultados geralmente são obtidos com alto valor da taxa de recombinação, isto é, $p_c \geq 0{,}7$. Já De Jong, apud Mitchell (1996), sugere uma taxa p_c = 0,6. Pode-se observar experimentalmente que quanto maior esta taxa, mais rapidamente novos indivíduos serão introduzidos na população. Mas se esta for muito alta, a maior parte da população será substituída e pode ocorrer perda de estruturas de boa aptidão. Com um valor muito baixo, o algoritmo pode tornar-se muito lento.

2.4.9.4 Taxa de mutação

Esta taxa determina se os genes dos cromossomos da população selecionada sofrerão mutação ou não. Para a tomada de decisão, gera-se um número aleatório no intervalo $[0,1]$ para cada um dos genes de todos os cromossomos e compara-se com a taxa de mutação. Se este número for menor que a taxa, o gene será modificado de acordo com o operador escolhido. Com relação aos valores da taxa de mutação, pode-se observar que uma baixa taxa

de mutação previne que uma dada posição fique estagnada em um valor, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Com uma taxa muito alta, a busca torna-se essencialmente aleatória. Os valores encontrados na literatura para a taxa de mutação (p_m) geralmente encontram-se entre 0,001 e 0,05. Tanomaru (1995) sugere um baixo valor para a taxa de mutação, geralmente $p_m \leq 0{,}01$. Já De Jong, apud Mitchell (1996) sugere $p_m = 0{,}001$.

2.4.9.5 Taxas adaptativas para cruzamento e mutação

Existem duas características que podem ser destacadas no AG. A primeira é a capacidade de convergir para um ótimo (local ou global), após a localização da região contendo este ótimo. A segunda é a capacidade de explorar novas regiões do espaço de solução em busca do ótimo global. O equilíbrio entre estas características é dado pelos valores das taxas de mutação e cruzamento e também pelo tipo de mutação e cruzamento utilizados.

Srinivas e Patnaik (1994), com o objetivo de prevenir uma convergência prematura do AG para um ótimo local, utilizaram em seu trabalho as fórmulas (14), (15), (16) e (17). Estas fórmulas calculam, a cada iteração, a taxa de mutação e cruzamento.

Considerando um problema de maximização, seja f' o maior valor de aptidão entre os indivíduos escolhidos para pais, $\overline{f}$ a média dos valores de aptidão de todos os cromossomos da população e f_{max} o valor de aptidão do melhor indivíduo, tem-se:

$$p_c = k_1.(f_{max} - f')/(f_{max} - \overline{f}) \; se \; f' \geq \overline{f} \tag{14}$$

$$p_c = k_3, \; se \; f' < \overline{f} \tag{15}$$

$$p_m = k_2.(f_{max} - f')/(f_{max} - \overline{f}) \; se \; f' \geq \overline{f} \tag{16}$$

$$p_m = k_4, \; se \; f' < \overline{f} \tag{17}$$

Os parâmetros $k_1, k_2, k_3 \; e \; k_4$ devem estar no intervalo $[0,1]$ e são determinados através de testes com o algoritmo.

O desempenho do AG proposto por Srinivas e Patnaik (1994) foi comparado com o AG padrão e os resultados mostraram-se significativamente melhores.

2.4.10 Hibridização

O AG possui um bom desempenho em buscas globais, mas são lentos com relação à convergência para um ótimo local. Por outro lado, os métodos de busca local podem encontrar o ótimo local em uma pequena região do espaço de busca, mas não possuem bom desempenho em buscas globais (TORABI, FATEMI-GHOMI e KARIMI, 2005). Desta forma, um melhor desempenho do AG pode ser obtido através de hibridização, ou, como é denominada por alguns autores, Algoritmo Memético. Esta estratégia visa acoplar algoritmos distintos com a finalidade de tirar o melhor proveito de cada um destes. Como Castro (2001) observa em seu trabalho, com a hibridização é possível acoplar-se ao AG qualquer outro método matemático para efetuar uma busca local mais agressiva. Ainda, esta alternativa híbrida resultante confere capacidade evolucionária de uma grande exploração global da região viável, aliada a uma boa eficiência nas buscas locais.

Segundo Barbosa (1997), pode-se pensar na hibridização como um operador de mutação generalizado que efetua uma busca local, ou seja, tomando um indivíduo como ponto de partida, utiliza um algoritmo específico para o problema em questão e efetua uma busca local adequada ao problema. O autor acrescenta ainda que um algoritmo de busca local para o caso de codificação binária corresponde a mudar todos os bits do genótipo, um por vez, decodificá-lo (se for o caso), avaliar a aptidão do vizinho assim gerado e substituir o genótipo por aquele que forneceu a melhor aptidão. Outra possibilidade é sortear um bit, mudar seu valor, decodificar o genótipo (se for o caso), avaliar a aptidão do vizinho assim gerado e continuar o processo até encontrar o primeiro vizinho que apresente uma aptidão superior à inicial. Estes dois procedimentos são gerais no sentido de não dependerem do problema e efetuam modificações apenas sobre o genótipo, sem necessidade de informações adicionais. O algoritmo resultante da hibridização mantém a capacidade de exploração global do espaço de soluções, característica inicial do AG, e tem aumentada sua capacidade de busca local.

Com relação aos métodos que podem ser acoplados para formar AG Híbrido (AGH) podem ser citados: Busca Tabu, *Simulated Annealing* (SA), *Hill-Climbing*, busca na vizinhança e outros.

2.4.11 Aplicações do AG

Desde os conceitos básicos definidos por Holland (1975), inúmeras aplicações têm sido desenvolvidas em várias áreas de pesquisa e em situações do mundo real com resultados satisfatórios.

Em geral, o AG têm sido utilizado na solução de funções de otimização, principalmente pela sua versatilidade na obtenção de soluções ótimas globais. Podem-se citar como exemplos de aplicações do AG:

- Problemas de otimização complexos: problemas com muitas variáveis e espaços de soluções de dimensões elevadas;
- Otimização evolutiva multi-critério: otimização de funções com múltiplos objetivos conflitantes entre si;
- Mineração de dados (*Data Mining*);
- Robótica: robôs utilizam AG para tomar decisões;
- Redes Neurais: AG é combinado com Redes Neurais para treiná-las ou para encontrar sua topologia;
- Ciências Biológicas: modela processos biológicos para o entendimento do comportamento de estruturas genéticas;
- Engenharia de Construções: possui aplicações na otimização discreta de estruturas;
- Síntese de Circuitos Analógicos: o AG é utilizado para gerar a topologia, o tipo e o valor dos componentes do circuito;
- Síntese de Protocolos: determina as funções de protocolo de *hardware* e *software* para que um certo desempenho seja alcançado;
- Gerenciamento de Rede: O AG faz a supervisão do tráfego nos *links* e das filas nos *buffers* de roteadores para descobrir rotas ótimas e para reconfigurar as rotas existentes no caso de falha de algum *link*;
- Outras áreas: evolução interativa de imagens, composição musical e outras.

A maior parte dos problemas de programação de produção (*scheduling*) envolve a determinação de uma seqüência de produção que gera um problema de otimização combinatória onde o número de soluções possíveis cresce exponencialmente à medida que aumenta o tamanho do problema. Dentre estes, podem ser citados os de *flowshop* e *jobshop*, considerados como sendo problemas NP-completos. Para resolver estes problemas de forma ótima, pode-se fazer a formulação como problemas de PLIM. Entretanto, para um grande

número de variáveis, a busca da solução usando o algoritmo *branch and bound* torna-se inviável computacionalmente. Por este motivo, diversas técnicas heurísticas têm sido desenvolvidas para resolver estes problemas, e mesmo que não se consiga a solução ótima, espera-se que uma solução satisfatória seja obtida em tempo computacional aceitável.

Vários trabalhos podem ser encontrados na literatura, relatando aplicações de AG para este tipo de problemas. Baudet, Azzaro-Pantel, Domenech et al (1995) aplicaram uma metodologia envolvendo dois estágios: primeiro, uma simulação por eventos discretos para avaliar dinamicamente o sistema de produção e, no segundo, a aplicação do AG para resolver o problema do processo de programação da produção em batelada. Os resultados computacionais obtidos mostraram que o uso desta metodologia pode gerar uma melhoria na eficiência no sistema de produção da indústria química envolvida no estudo. Wang, Zhang e Zheng (2003) propuseram em seu trabalho uma classe de AG modificado para resolver problemas de programação de produção do tipo *flow shop*. Conseguiram mostrar que, através da metodologia por eles desenvolvida, pode-se obter uma solução satisfatória e com alto grau de confiança, em tempo computacional reduzido. Pasupathy, Rajendran e Suresh (2005) consideraram o problema de programação de produção *flow shop* permutacional com o objetivo de minimizar o tempo de execução de todas as tarefas e o tempo total de fluxo de cada tarefa. Propuseram um AG, usando uma classificação de Pareto baseada em AG com múltiplos objetivos e um arquivo de soluções não dominantes, sujeitas a uma heurística de busca local. Ilkyeong e Jieom (2000) propuseram várias aplicações de AG para resolver problemas de programação de produção do tipo *job shop* com rotinas alternativas. Compararam os resultados obtidos com os encontrados através da resolução do problema de PLIM e consideraram melhores os resultados obtidos por meio das novas metodologias. Souza (2004) desenvolveu um modelo de AG para permitir aos tomadores de decisões em planejamento de colheita, determinar o período de intervenção das equipes de corte nos pontos de produção. O objetivo de seu trabalho foi minimizar os custos com as atividades relacionadas à colheita e ao transporte principal de madeira. Hartmann (2001) apresentou um novo AG para resolver um problema de programação de produção de projetos com restrições de recursos envolvendo modos de execução múltiplos para cada atividade e função objetivo, buscando minimizar o tempo total de execução de todos os projetos.

Ainda, podem ser encontradas diversas aplicações envolvendo AG híbrido. Buzzo e Moccellin (2000) aplicaram um método heurístico híbrido usando AG e SA para minimizar a duração total da programação de tarefas *flow shop* permutacional. Fizeram a comparação deste método híbrido com a aplicação dos métodos AG e SA puros e através dos resultados

concluíram que um método híbrido pode ser desenvolvido de maneira a agregar as características vantajosas dos métodos meta-heurísticos puros. Em seu artigo, Torabi, Fatemi-Ghomi e Karimi (2005) investigaram o problema de programação de lotes e entregas em uma cadeia de suprimento onde um fornecedor único produz vários componentes em uma linha de fluxo flexível e os entrega diretamente a uma linha de montagem. Para resolver este problema criaram um Algoritmo Genético Híbrido (AGH) incorporando um método eficiente de busca na vizinhança junto às operações de seleção e recombinação. Resultados computacionais indicaram que o desempenho do AGH foi muito promissor e mostrou-se melhor que o método de enumeração no que se refere ao tempo computacional e qualidade de solução na maior parte dos testes. Cheung e Zhou (2001) desenvolveram um algoritmo híbrido baseado na integração de AG e regras heurísticas. O principal objetivo desta metodologia foi conseguir uma exploração mais efetiva e eficiente na solução de problemas de programação de produção do tipo *job shop* com seqüenciamento dependente do tempo de ajuste. A hibridização proposta mostrou um direcionamento eficiente na busca de soluções, conseguindo com isto uma maior rapidez do algoritmo. Outros trabalhos aplicando estratégias de hibridização para melhorar o desempenho do AG podem ser encontrados na literatura (CHENG e GEN, 1997; HO e JI, 2003; LIAW, 2000; WANG e WU, 2003; WANG e WU, 2004; WEI e ZHAO, 2005).

Numerosos trabalhos relacionados a outras áreas de estudo relatam aplicações de AG. Brown e Sumichrast (2005) avaliaram AG Agrupados, com relação ao desempenho em problemas de otimização com restrições; Ribeiro Filho e Lorena (2000) fizeram um uso combinado de AG e geração de colunas para resolver de forma aproximada problemas de coloração de grafos; Salhi e Gamal (2003) propuseram uma nova implementação de AG, incluindo estratégias para seleção e remoção de pais e esquemas para união e reprodução, de forma a manter a diversidade dentro da população de indivíduos. Foi utilizada como plataforma de testes um problema de locação-alocação contínuo; Deb e Pal (2004) desenvolveram um AG generalizado com quatro variações, usando o conceito de múltiplos pais, para resolver problemas de PLI em grande escala; Castro (2001) utilizou para a otimização de estruturas com multi-objetivos, o AG proposto por Pareto; Velloso, Mendonça, Pacheco et al. (1995) aplicaram AG em um problema de otimização de alocação do espaço físico de uma universidade para as atividades acadêmicas nos seus respectivos horários.

2.5 TRANSGENÉTICA COMPUTACIONAL

Transgenética Computacional (TC) é uma nova técnica da Computação Evolucionária (CE). É considerada uma abordagem evolucionária, pois emprega populações de indivíduos, as populações trocam informações dinamicamente e somente os mais aptos sobrevivem em cada ciclo de troca de informações.

O paradigma da TC está contextualizado na evolução dos micro-organismos e células e fundamentam-se no processo da simbiogênese (GOLDBARG, GOLDBARG e MEDEIROS NETO, 2005).

A simbiogênese é uma teoria evolucionária onde indivíduos de naturezas distintas se unem para formar um novo indivíduo (MOROWITZ, 1992). Esta teoria destaca mais os efeitos positivos resultantes das inter-relações entre indivíduos do que a sobrevivência e reprodução do mais apto.

Margulis (1998) apud Goldbarg, Goldbarg. e Medeiros Neto (2005) enunciou a teoria da simbiogênese da seguinte forma: “A evolução do genoma no longo prazo é muito mais influenciada por interações extra-intracelulares do que pela mutação ou seleção natural”. Ainda, afirma que a evolução permite modificações no genoma devido a “fusões” de duas ou mais criaturas para formar uma outra, mais bem adaptada ao ambiente.

As alterações do genoma nos micro-organismos são efetuadas por mecanismos e vetores. Uma simplificação da classificação de vetores seria:

- bacteriófagos: vírus que possuem bactérias como hospedeiros;
- plasmídios: pequenos segmentos de ADN (Ácido Desoxirribonucléico) que podem ser transferidos de uma bactéria a outra, permitindo a transmissão de genes;
- transponsons: seqüências de ADN que fazem parte de outros elementos genéticos – cromossomo ou plasmídio;

Estes três vetores são considerados partículas genéticas móveis.

Uma classificação simplista dos vetores sob a ótica dos processos seria:

- mutações: modificações no material genético;
- plasmídios: moléculas de ADN existente em bactérias ou fungos, independentes do ADN do cromossomo;
- transformação: “pedaços” de DNA estranhos, existentes no meio que entram na bactéria e a modificam;
- transdução: os genes são transportados por um vírus;

- conjugação: a transferência de genes de uma bactéria para outra ocorre através de um canal de comunicação que se forma entre elas;
- recombinação: os genes de duas ou mais bactérias são misturados e incorporados em uma terceira;
- plasmídios/recombinação: partículas genéticas móveis que realizam recombinações do DNA disponível no fluxo intracelular.

Os plasmídios, a transformação e a transdução são chamados de processos de transferência horizontal. Este tipo de transferência é uma via de alteração do genoma das bactérias e permite o efeito de simbiose.

2.5.1 Algoritmos Transgenéticos

Dentro do contexto da simbiogênese, a aplicação da metáfora aos algoritmos transgenéticos é feita de forma que:

1) o processo evolucionário dos algoritmos transgenéticos esteja baseado em interações simbióticas desenvolvidas entre uma população de cromossomos e uma população de vetores do fluxo intracelular;
2) o resultado da simbiose seja avaliado pela adequação alcançada pelos cromossomos;
3) os mecanismos evolucionários empregados nos algoritmos transgenéticos mimetizem os processos de conjugação e de recombinação (processos de transferência horizontal);
4) os vetores empregados mimetizem os plamídios, vírus e transponsons;
5) o processo evolucionário seja gerenciado pelos três tipos de regras que definem:
    - como construir os vetores do fluxo intra-extracelular;
    - de que forma as populações em evolução vão interagir;
    - como será obtida a informação que vai circular no processo simbiótico;
6) o paradigma da transgenética possua os seguintes princípios gerais:
    - a evolução pode ocorrer sem reprodução sexual ou mutações;
    - os vetores do fluxo intra-extracelular cooperam com a evolução dos cromossomos;
    - o fluxo de informações do processo evolucionário percorre vários níveis de decisão;

A metáfora transgenética possui, basicamente, três níveis onde as informações são armazenadas:

- Primeiro nível: é composto pela população de cromossomos e representa a memória atual do processo de busca;

- Segundo nível: é composto por uma população de indivíduos de natureza distinta da dos cromossomos. Estes indivíduos são chamados de vetores transgenéticos e operam na diversificação e intensificação da busca. Os vetores transgenéticos são detalhados a seguir, na Seção 2.5.1.2.
- Terceiro Nível: é formado por regras que direcionam a interação entre as duas populações (cromossomos e vetores transgenéticos). Implementam o formato dos vetores transgenéticos e coordenam o relacionamento entre as populações distintas. Este nível de informação pode ser pensado como uma modelagem do processo co-evolucionário. Pode-se entender por processo co-evolucionário a idéia de alguma troca evolucionária recíproca integrando espécies.

2.5.1.1 Cromossomos

Um cromossomo é composto por uma cadeia de informação e tem associados um ou mais valores de adequação.

2.5.1.2 Vetores transgenéticos

Com o objetivo de explorar o espaço de busca, os algoritmos transgenéticos obtêm informação e a inserem nos cromossomos através dos vetores transgenéticos.

Um vetor transgenético λ é uma dupla $\lambda = (I, \Phi)$, onde I é uma cadeia de informação e Φ é um método de manipulação.

A cadeia de informação I é composta por um ou mais fragmentos de ADN. Esta informação pode ser *a priori* ou *a posteriori*. A informação *a priori* pode vir de diversas fontes, tais como conhecimento teórico e heurístico. A informação *a posteriori* é obtida através do próprio processo de busca. De forma geral, estas informações podem ser obtidas de:

- células doadoras (cromossomos);
- recombinações;
- banco de DNA.

Estas cadeias podem ser reunidas em um banco de informações genéticas. Este banco de informações visa acelerar o processo de evolução.

O método de manipulação $\Phi = (p_j)$ com $j = 1,...,s$ é composto pelos procedimentos p_j que definem a atuação do vetor (GOLDBARG, GOLDBARG e MEDEIROS NETO, 2003). A Tabela 2 resume os procedimentos que compõem o método de manipulação dos vetores transgenéticos.

Tabela 2: Procedimentos Gerais de um Vetor Transgenético

Procedimento	Denominação	Descrição
Procedimento 1 (p_1)	Ataque (A)	Define o critério de avaliação que estabelece quando um cromossomo é suscetível à manipulação do vetor. $A : S_i \rightarrow \{\text{falso, verdadeiro}\}$, $i = 1,..., N$.
Procedimento 2 (p_2)	Operador de Transcrição (Γ)	Se $A(S_i)$ = "verdadeiro", o procedimento define como a informação I, transportada pelo vetor, será transferida para o cromossomo.
Procedimento 3 (p_3)	Bloqueio da informação transcrita (Ψ)	Torna o resultado da manipulação inviolável por um certo período de tempo – número de iterações, gerações de cromossomos, etc.
Procedimento 4 (p_4)	Desbloqueio da informação transcrita ($\overline{\Psi}$)	Torna o resultado da manipulação sem restrições.
Procedimento 5 (p_5)	Operador de identificação (Λ)	Identificam uma cadeia ou um trecho da cadeia.
Procedimento 6 (p_6)	Operador de recombinação (Π)	Recombinam os genes da cadeia.

Em geral, quando um vetor transgenético manipula um cromossomo, modifica sua estrutura e desta forma um novo ponto do espaço de busca é investigado (GOUVÊA e GOLDBARG, 2001). Esta manipulação ocorre de acordo com o procedimento A (ataque), definido em p_1, que pode ser determinístico ou probabilístico. Um vetor λ transcreve sua informação I usando o operador de transcrição definido em p_2 somente se $A(S_i) = "verdadeiro"$. Caso contrário, o cromossomo não é manipulado por λ e diz-se que S_i resistiu ao ataque de λ.

Os plasmídios, considerados como partículas genéticas móveis, somente consolidam sua transcrição se a referida operação resulte na melhoria da adequação do cromossomo. Já os vírus são vetores de manipulação que podem realizar transcrição de sua informação, ainda que resulte em redução da adequação do cromossomo.

Existem casos nos quais a manipulação se torna inviolável por um certo período de tempo. Este período, determinado por p_3, pode ser medido em número de iterações, número de gerações de cromossomos ou outros. Depois de transcorrido o tempo determinado por p_3, o procedimento p_4 torna o resultado da manipulação sem restrições.

Os vetores $\lambda = (I, \Phi)$ com $\Phi = (p_1, p_2, ...p_s)$ podem possuir um dos seguintes processos:

- Vírus (V): $\Phi = (p_1, p_2, p_3, p_4)$;
- Plasmídio (PI): $\Phi = (p_1, p_2)$;
- Transponson (T): $\Phi = (p_1, p_2, p_5, p_6)$
- Plasmídio Recombinado (Pr): $\Phi = (p_1, p_2, p_6)$

2.5.1.3 Regras de administração (regras transgenéticas)

Estas regras gerenciam o processo de infiltração intra-extracelular. Existem três tipos de regras que administram o relacionamento entre cromossomos e vetores transgenéticos (GOLDBARG, GOLDBARG e MEDEIROS NETO, 2004).

- Regra Tipo 1: Direciona a constituição da cadeia de informação *I* que irá ser carregada no vetor transgenético. Pode usar fontes teóricas ou heurísticas para obter o conhecimento para formação do vetor. Podem existir várias regras deste tipo;
- Regra Tipo 2: Define como a cadeia de informação *I* será transcrita em um cromossomo ou como o conteúdo do cromossomo será rearranjado. Formaliza os operadores de manipulação de um vetor λ. Podem ser regras heurísticas. O algoritmo pode possuir mais de uma regra deste tipo;
- Regra Tipo 3: Determina como utilizar as regras do Tipo1 e Tipo2 e gerenciar a simbiose estabelecendo:
  1) como escolher um número *q* de cromossomos para manipulação;
  2) como escolher um número *r* de vetores de manipulação;

3) quais as regras Tipo1 e Tipo2 serão utilizadas em cada manipulação;
4) como organizar e obter informações genéticas;
5) como realimentar o processo evolucionário;
6) quando parar a evolução.

Um algoritmo transgenético na sua forma geral pode ser descrito da seguinte forma:

Algoritmo Transgenético

Gerar uma população inicial;
Carregar as Regras Transgenéticas (RET);
Carregar Banco de Informações Genéticas (BIG);
Repita
Gerar vetores de manipulação
Selecionar cromossomos para manipulação;
Manipular os cromossomos conforme RET
Atualizar RET e BIG
Até critério de parada ser satisfeito

2.5.2 Algoritmo Proto-Gene (ProtoG)

O algoritmo ProtoG é baseado em população na qual os cromossomos não trocam informações de forma direta. Nesse algoritmo, os cromossomos são expostos a um ataque direto pelos vetores do fluxo intracelular para concretizarem a transcrição de seus códigos.

Como em qualquer algoritmo evolucionário, gera-se a população inicial aleatoriamente. O pseudo-código de um algoritmo ProtoG é o seguinte:

Algoritmo ProtoG

Gerar população $(S_1, S_2, ..., S_N)$;
Carregue_Regras_Transgenéticas (Tipo1, Tipo2, Tipo3, q, r);
Repita
Escolha_λ = vetor_transgenético (r);
Escolha_S = seleção_população $(S_1, S_2, ..., S_N, q)$;
Para i = 1 até q, **faça**
Para k = 1 até r **faça**
Se ataque(S_i, λ_k) é verdadeiro, **então**
S_i = manipular (S_i, λ_k);
Se critério_realimentação (S_i), **então**
Incluir_fonte_informação (S_i);
fim_Se;
fim_Se
fim_Para_*k*
fim_Para_*i*
até critério_parada ser satisfeito

Os procedimentos do algoritmo e sua atuação são:

- Carregue_Regras_Transgenéticas ( ): Determina como os vetores transgenéticos são formados, informando o conjunto de regras de administração que serão utilizadas. Define também, o número de vetores (r) a serem formados em cada iteração e o tamanho da sub-população (q) selecionada para ser atacada por estes r vetores;
- vetor_transgenético ( ): gera r vetores transgenéticos de acordo com as regras determinadas previamente. Cada um dos r vetores carrega uma informação obtida de alguma regra do Tipo 1 e possui um operador definido por uma regra do Tipo 2;
- seleção_população ( ): gera um conjunto com q cromossomos para serem atacados por todos os r vetores transgenéticos ;
- ataque ( ): implementa o procedimento p_1. Caso o procedimento retorne valor "verdadeiro" em alguma iteração, λ_k completa a manipulação de S_i, o i-ésimo cromossomo do conjunto definido pelo procedimento seleção_população.
- manipular ( ): faz a manipulação através do vetor transgenético escolhido;
- critério_realimentação: se o cromossomo S_i é considerado como um bom exemplo de evolução de acordo com algum critério definido no algoritmo, então este procedimento retorna o valor "verdadeiro";

- Incluir_fonte_informação: se o valor gerado pelo critério_realimentação for “verdadeiro”, então este procedimento implementa a inclusão da solução representada em S_i como uma fonte de informação não genética para ser considerada pelas regras de administração na formação de futuras gerações de vetores transgenéticos;
- critério_parada: pode ser estabelecido como sendo um número máximo de iterações sem melhoria da melhor solução atual, um tempo máximo de processamento, um número máximo de iterações, etc.

Os procedimentos p_2, p_3 e p_4 já estão definidos no algoritmo, porque são características inerentes ao vetor.

Vários critérios podem ser utilizados para selecionar um cromossomo para ser inserido como fonte de informação. Um dos mais utilizados é considerar cromossomos que apresentem melhor valor de aptidão que qualquer outro encontrado até então.

Uma variação para o algoritmo ProtoG foi proposta por Costa, Goldbarg e Goldbarg (2003) onde considerou-se a estimulação do cromossomo como a estratégia prioritária. Os vetores transgenéticos, ao invés de atacarem os cromossomos da população selecionada, são direcionados a um único cromossomo. Este cromossomo é estimulado sucessivamente por cada um dos vetores criados na iteração, buscando-se uma maior intensificação do processo de busca.

2.5.3 Aplicações de algoritmos transgenéticos

Os Algoritmos Transgenéticos vêm sendo aplicados em diversas áreas. Têm mostrado bons resultados, principalmente em problemas de otimização combinatória.

Goldbarg e Goldbarg (2002) aplicaram quatro tipos de algoritmos transgenéticos para resolver problemas quadráticos de alocação. Os resultados obtidos foram comparados com as metodologias: Colônia de Formigas, Aprendizado por reforço (*Reinforcement Learning*), *GRASP* (*Graphical representation and analysis of structural properties*) e GDH (método híbrido genético descendente). Os autores concluíram que, na maior parte dos testes, os algoritmos transgenéticos apresentaram desempenho superior aos outros citados.

Costa, Goldbarg e Goldbarg (2003) relatam em seu artigo a aplicação de duas versões do algoritmo ProtoG para resolver o problema do *flow-shop* de permutação. Compararam os resultados obtidos com os limites superiores das instâncias da literatura para o problema até o presente momento e concluíram que a qualidade de soluções é comparável com a meta-heurística busca tabu. A variante do algoritmo ProtoG apresentou os melhores resultados entre os algoritmos comparados.

Ramos, Goldbarg, Goldbarg et al. (2003b) aplicaram o algoritmo ProtoG ao Problema do Caixeiro Viajante e compararam os resultados obtidos com os resultados de dois algoritmos baseados na técnica de *Simulated Annealing*. Um processo de ajuste do algoritmo ProtoG foi desenvolvido para mostrar que este algoritmo é sensível com relação à variação dos parâmetros população e tamanho da cadeia do agente transgenético. Para o conjunto de instâncias testado, o experimento mostra que o tamanho máximo do agente não deve ultrapassar cerca de 10% do cromossomo. O algoritmo ProtoG mostrou um desempenho computacional superior ao algoritmo *Simulated Annealing*, tanto em tempo computacional como em qualidade de solução obtida.

Ramos, Goldbarg, Goldbarg et al. (2003a) fizeram um experimento computacional com o algoritmo transgenético ProtoG em 25 instâncias encontradas na literatura, para o Problema do Caixeiro Viajante. Este algoritmo foi comparado com os seguintes algoritmos: *simulated annealing* híbrido, busca local e algoritmo memético. Os resultados mostraram que o algoritmo transgenético encontrou melhores soluções e processou em menor tempo computacional quando comparado aos outros citados. O algoritmo ProtoG aplicado utilizou uma geração de árvore mínima e uma análise de componentes para gerar cadeias de informação que foram utilizadas para direcionar o processo evolucionário.

Goldbarg, Goldbarg e Medeiros Neto (2005) apresentaram um caso especial do problema geral de determinação de uma configuração para um sistema de co-geração de energia que utiliza o gás natural como fonte energética. O problema tem como objetivo a determinação de uma configuração que gere um custo mínimo para o sistema. Foram aplicados para a resolução deste problema, Algoritmos Meméticos, AG e a Transgenética Computacional. Após a aplicação a um conjunto de 35 instâncias, os resultados indicaram uma dominância de melhores resultados para o algoritmo transgenético.

Goldbarg e Gouvêa (2001) aplicaram o Algoritmo Transgenético ATEIs ao Problema de Coloração de Grafos. Através dos experimentos, puderam mostrar que a manipulação de cromossomos por agentes transgenéticos é uma ferramenta potente para direcionar a busca dentro do espaço de soluções.

Goldbarg, Castro e Goldbarg (2004) aplicaram o Algoritmo Transgenético ProtoG para um problema de redes de dutos. A seleção de tipos de tubos é considerada como um problema de otimização com restrições e normalmente é abordado como um problema de otimização de custo mínimo, tendo como variáveis de decisão o diâmetro dos tubos. Os experimentos computacionais foram efetuados para o algoritmo proposto e para AG. Os resultados mostraram uma vantagem na obtenção de resultados do ProtoG em relação ao AG.

2.6 SIMULAÇÃO

Simulação, considerada por Saliby (1989) como uma técnica da área de Pesquisa Operacional, é uma das melhores ferramentas de análise existentes para auxiliar no desenvolvimento e construção de processos ou sistemas. Esta técnica tem sido cada vez mais aceita e utilizada por administradores, engenheiros e outros especialistas, como ferramenta para verificar ou encaminhar soluções de problemas. Muitas vezes, estes profissionais utilizam simulação para procurar formas de analisar o impacto de mudanças potencialmente positivas em sistemas complexos.

De acordo com Banks, Carson, Nelson et al. (2000), simulação é a imitação das operações de um sistema ou processo real no decorrer do tempo. Feita manualmente ou computacionalmente, a simulação envolve a geração de uma história artificial de um sistema e a observação desta história, para extrair dados sobre as características operacionais do sistema real.

Uma das vantagens da técnica de simulação é a versatilidade de aplicação. Segundo Taha (1997), esta técnica tem sido utilizada em todos os ramos da ciência e tecnologia, tais como:

1) Ciência básica:
    a) Cálculo da área abaixo da curva ou mais genericamente, cálculo de integrais múltiplas;
    b) Cálculo do número π;
    c) Inversão de matrizes;
    d) Estudo de difusão de partículas.
2) Situações práticas:
    a) Problemas em indústrias, incluindo projetos de sistemas de filas, redes de comunicação, controle de estoque e processos químicos;
    b) Problemas econômicos e de negócios, incluindo comportamento do consumidor, determinação de preços e previsão econômica;
    c) Comportamento e problemas sociais, incluindo população dinâmica, estudos epidemiológicos e comportamento de grupos;
    d) Sistemas biomédicos, incluindo equilíbrio de fluídos, distribuição eletrolítica no corpo humano, proliferação de células sanguíneas e atividades cerebrais;
    e) Táticas e estratégias de guerra.

2.6.1 Sistemas de Simulação

Na definição de simulação foi utilizado o termo sistema. Dentro do contexto de simulação, define-se sistema como um conjunto de objetos, como pessoas ou máquinas, por exemplo, que atuam e interagem com a intenção de alcançar um objetivo ou um propósito lógico (FREITAS FILHO, 2001). O interesse num sistema específico pode ser em uma ou mais dentre as atividades: análise, projeto, controle ou melhoria da compreensão ou desempenho.

Na indústria podem ser citados os seguintes sistemas:

- Área de produção: sistemas de manufatura e montagem, movimentação de peças e matéria-prima, alocação de mão-de-obra, áreas de armazenagem e outros;
- Transporte e estocagem: sistemas de redes de distribuição, armazéns e entrepostos, frotas e outros.

Segundo Banks, Carson, Nelson et al. (2000), são cinco os componentes em um sistema: entidade, atributo, atividade, evento e estado do sistema. Uma entidade é um objeto de interesse do sistema e pode ser uma pessoa ou um objeto (real ou imaginário) que se movimenta, causando mudanças no estado do sistema. Um atributo é uma propriedade da entidade. Uma atividade representa um período de tempo de duração específica e é ativada por uma entidade. Um evento é definido como uma ocorrência instantânea que pode mudar o estado do sistema. O estado do sistema é definido como o conjunto de variáveis necessárias para descrever o sistema em qualquer instante e é relativo aos objetivos do estudo.

Por exemplo, na área de produção de uma indústria tem-se:

- Máquinas como as entidades;
- Velocidade, capacidade e probabilidade de ocorrência de defeitos de uma máquina como os atributos;
- Pintura e estampagem como as atividades;
- Defeito em uma máquina como um evento;
- Estado das máquinas (ocupada, desocupada ou parada) como as variáveis de estado.

2.6.2 Modelos de um sistema de simulação

Muitas vezes, estuda-se um sistema para entender o relacionamento entre seus componentes ou para prever como o sistema se comportará com uma nova configuração.

Quando não é possível fazer as modificações no próprio sistema, utiliza-se um modelo. Banks, Carson, Nelson et al. (2000) definem um modelo como uma representação de um sistema com o propósito de estudá-lo. A modelagem pressupõe um processo de criação e descrição envolvendo um determinado grau de abstração que, na maioria das vezes, acarreta numa série de simplificações sobre a organização e o funcionamento do sistema real. Deve ser detalhado o suficiente para permitir que sejam extraídas conclusões válidas a respeito do sistema real.

Ainda, segundo Banks, Carson, Nelson et al. (2000), os modelos podem ser classificados em matemáticos ou físicos. Modelos físicos são construídos com componentes concretos (tangíveis). São representações de sistemas físicos como sistemas elétricos, térmicos, hidráulicos, etc. São descritos por variáveis como voltagens, correntes, potência, temperaturas, comprimentos, pressão, fluxo, força, velocidade etc. Modelos matemáticos usam notação simbólica e equações matemáticas para representar o sistema. Um modelo de simulação é um tipo particular de modelo matemático de um sistema.

Modelos também podem ser classificados em estáticos ou dinâmicos, determinísticos ou estocásticos e discretos ou contínuos. Um modelo de simulação estático, muitas vezes chamado de simulação de Monte Carlo, representa um sistema em um ponto particular no tempo. Modelos de simulação dinâmica representam sistemas no decorrer do tempo.

Modelos de simulação que não contêm variáveis aleatórias são classificados como determinísticos. São modelos que possuem um conjunto conhecido de entradas as quais irão resultar em um único conjunto de saídas. Um modelo estocástico possui uma ou mais variáveis aleatórias de entrada que geram saídas também aleatórias.

Um modelo de simulação discreto é aquele em que mudanças no estado do sistema ocorrem somente em pontos isolados de tempo. Neste caso, supõe-se que o estado do sistema não se altera ao longo do intervalo compreendido entre dois eventos consecutivos. Numa simulação contínua, o modelo trata mudanças de comportamento continuamente no tempo. A dinâmica da população mundial é um exemplo típico de modelo contínuo. Modelos de simulação contínua normalmente são representados em termos de equações diferenciais que descrevem a interação entre os diferentes elementos do sistema. Ainda, se parte do sistema é representado como contínuo e outra parte como discreto, então o sistema é chamado de combinado ou híbrido (PEGDEN, SHANNON e SADOWSKI, 1995).

2.6.3 Etapas em um estudo envolvendo modelagem e simulação

Freitas Filho (2001) após um estudo detalhado de várias referências bibliográficas resumiu os passos necessários para modelagem e simulação, separando-os em etapas:

1) Etapa de Planejamento:
   a) Formulação e Análise do Problema: todo estudo de simulação inicia com a formulação do problema, a definição dos propósitos e dos objetivos do estudo. Devem ser respondidas questões do tipo:
      - Por que o problema está sendo estudado?
      - Quais serão as respostas que o estudo espera alcançar?
      - Quais são os critérios para avaliação do desempenho do sistema?
      - Quais restrições e limites são esperados das soluções obtidas?
   b) Planejamento do Projeto: com o planejamento do projeto pretende-se ter a certeza de que existem recursos suficientes no que diz respeito a pessoal, suporte, gerência, *hardware* e *software* para a realização do trabalho proposto. Além disso, o planejamento deve incluir uma descrição dos vários cenários que serão investigados e um cronograma temporal das atividades que serão desenvolvidas, indicando os custos e necessidades relativas aos recursos anteriormente citados.
   c) Formulação do Modelo Conceitual: traçar um esboço do sistema, de forma gráfica (fluxograma, por exemplo) ou algorítmica (pseudo-código), definindo componentes, descrevendo as variáveis e interações lógicas que constituem o sistema. É recomendado que o modelo inicie de forma simplificada e vá crescendo até alcançar algo mais complexo, contemplando todas as suas peculiaridades e características. O usuário deve participar intensamente desta etapa. Algumas das questões que devem ser respondidas:
      - Qual a estratégia de modelagem? Discreta? Contínua? Híbrida?
      - Que quantidade de detalhes deve ser incorporada ao modelo?
      - Como o modelo reportará os resultados? Relatórios pós-simulação? Animações durante a execução?
      - Que nível de personalização de cenários e ícones de entidades e recursos deve ser implementado?
      - Que nível de agregação dos processos (ou de alguns) deve ser implementado?
      - Como os dados serão colocados no modelo? Manualmente? Leitura de arquivos?

d) Coleta de Macro-Informações e Dados: macro-informações são fatos, informações e estatísticas fundamentais, derivados de observações, experiências pessoais ou de arquivos históricos. Em geral, as macro-informações servem para conduzir os futuros esforços de coleta de dados voltados à alimentação de parâmetros do sistema modelado. Algumas questões que se apresentam são:
   - Quais são as relações e regras que conduzem a dinâmica do sistema? O uso de diagramas de fluxos é comum para facilitar a compreensão destas inter-relações.
   - Quais são as fontes dos dados necessários à alimentação do modelo?
   - Os dados já se encontram na forma desejada? O mais comum é encontrar-se os dados disponíveis de maneira agregada (na forma de médias, por exemplo), o que não é interessante para a simulação;
   - E quanto aos dados relativos a custos e finanças? Incorporar elementos de custos em um projeto torna sua utilização muito mais efetiva.

2) Etapa de Modelagem
   a) Coleta de Dados: a busca por dados específicos do sistema deve ser feita desde o início do projeto, pois utiliza uma grande porção do tempo disponível para o desenvolvimento do modelo de simulação.
   b) Tradução do Modelo: codificar o modelo numa linguagem de simulação apropriada. Embora hoje os esforços de condução desta etapa tenham sido minimizados em função dos avanços em *hardware* e, principalmente nos *softwares* de simulação, algumas questões básicas devem ser propriamente formuladas e respondidas.
      - Quem fará a tradução do modelo conceitual para a linguagem de simulação? É fundamental a participação do usuário se este não for o responsável direto pelo código;
      - Como será realizada a comunicação entre os responsáveis pela programação e a gerência do projeto?
      - E a documentação? Os nomes de variáveis e atributos estão claramente documentados? Outros, que não o programador responsável, podem entender o programa?
   c) Verificação e Validação: confirmar que o modelo opera de acordo com a intenção do analista (sem erros de sintaxe e lógica) e que os resultados por ele fornecidos possuam crédito e sejam representativos dos resultados do modelo real. Nesta etapa, as principais questões são:
      - O modelo gera informações que satisfazem aos objetivos do estudo?

- As informações geradas são confiáveis?
- A aplicação de testes de consistência e outros confirma que o modelo está isento de erros de programação?

3) Etapa de Experimentação
   a) Projeto Experimental Final: projetar um conjunto de experimentos que produza a informação desejada, determinando como cada um dos testes deve ser realizado. O principal objetivo é obter mais informações com menos experimentações. As principais questões são:
      - Quais os principais fatores associados aos experimentos?
      - Em que níveis devem estar os fatores variados, de forma que se possa melhor avaliar os critérios de desempenho?
      - Qual o projeto experimental mais adequado ao quadro de respostas desejadas?
   b) Experimentação: executar as simulações para a geração dos dados desejados e para a realização das análises de sensibilidade.
   c) Interpretação e Análise Estatística dos Resultados: traçar inferências sobre os resultados alcançados pela simulação. Estimativas para as medidas de desempenho nos cenários planejados são efetuadas. As análises poderão resultar na necessidade de um maior número de execuções (replicações) do modelo para que se possa alcançar a precisão estatística sobre os resultados desejados. Também pode ser necessária a inclusão de um período de aquecimento que consiste num período de tempo que o modelo deve processar para que vestígios decorrentes da inicialização sejam removidos antes que a coleta de dados estatísticos seja iniciada. Algumas questões que devem ser apropriadamente respondidas:
      - O sistema modelado é do tipo terminal ou não terminal?
      - Quantas replicações são necessárias?
      - Qual deve ser o período simulado para que se possa alcançar o estado de regime?
      - E o período de aquecimento?
4) Etapa de Tomada de Decisão e Conclusão do Projeto
   a) Comparação de Sistemas e Identificação das melhores soluções: muitas vezes, o emprego da técnica de simulação visa à identificação de diferenças existentes entre diversas alternativas de sistemas. Em algumas situações, o objetivo é comparar um sistema existente ou considerado como padrão, com propostas alternativas. Em outras,

a idéia é a comparação de todas as propostas entre si com o propósito de identificar a melhor ou mais adequada delas. As questões próprias deste tipo de problema são:

- Como realizar este tipo de análise?
- Como proceder para comparar alternativas com um padrão?
- Como proceder para comparar todas as alternativas entre si?
- Como identificar a melhor alternativa de um conjunto?
- Como garantir estatisticamente os resultados?

b) Documentação: a documentação do modelo é sempre necessária. Primeiro para servir como um guia para que alguém, familiarizado ou não com o modelo e os experimentos realizados, possa fazer uso do mesmo e dos resultados já produzidos. Segundo, porque se forem necessárias futuras modificações no modelo, toda a documentação existente vem a facilitar, e muito, os novos trabalhos. A implementação bem sucedida de um modelo depende, fundamentalmente, que o analista, com a maior participação possível do usuário, tenha seguido os passos descritos. Os resultados das análises devem ser reportados de forma clara e consistente, também como parte integrante da documentação do sistema. Como linhas gerais, pode-se dizer que os seguintes elementos devem constar em uma documentação final de um projeto de simulação.

   - Descrição dos objetivos e hipóteses levantadas;
   - Conjunto de parâmetros de entrada utilizados (incluindo a descrição das técnicas adotadas para adequação de curvas de variáveis aleatórias);
   - Descrição das técnicas e métodos empregados na verificação e na validação do modelo;
   - Descrição do projeto de experimentos e do modelo fatorial de experimentação adotado;
   - Resultados obtidos e descrição dos métodos de análise adotados;
   - Conclusões e recomendações. Nesta última etapa, é fundamental tentar descrever os ganhos obtidos na forma monetária.

c) Apresentação dos Resultados e Implementação. A apresentação dos resultados do estudo de simulação deve ser realizada por toda a equipe participante. Os resultados do projeto devem refletir os esforços coletivos e individuais realizados, considerando os seus diversos aspectos, isto é, levantamento do problema, coleta de dados, construção do modelo, etc. Durante todo o desenvolvimento e implementação do projeto, o processo de comunicação entre a equipe e os usuários finais, deve ser total.

Os itens abaixo devem estar presentes como forma de encaminhamento das questões técnicas, operacionais e financeiras no que diz respeito aos objetivos da organização:

- Restabelecimento e confirmação dos objetivos do projeto;
- Identificação de problemas que foram resolvidos;
- Rápida revisão da metodologia;
- Benefícios alcançados com a(s) solução(ões) proposta(s);
- Considerações sobre o alcance e precisão dos resultados;
- Alternativas rejeitadas e seus motivos;
- Animações das alternativas propostas quando cabíveis;
- Estabelecimento de conexões entre o processo e os resultados alcançados com o modelo simulado e outros processos de reengenharia ou de reformulação existentes no negócio;
- Assegurar que os responsáveis pelo estabelecimento de mudanças organizacionais ou processuais tenham compreendido a abordagem utilizada e seus benefícios;
- Tentar demonstrar que a simulação é uma espécie de ponte entre a idéia e sua implementação.

2.6.4 Simulação de eventos discretos

Em simulação de eventos discretos, um sistema é modelado em termos de seu estado em cada ponto no tempo. Os componentes estruturais de uma simulação de eventos discretos incluem: entidades, atributos, atividades e eventos, recursos, variáveis globais, um gerador de números aleatórios, um calendário, um relógio da simulação, variáveis de estado do sistema e acumulador estatístico.

- Entidades: é o termo utilizado para pessoas ou objetos, reais ou imaginários, que movimentam-se através do sistema provocando mudanças no estado deste. Sem entidades, nada acontece em uma simulação. Um exemplo de entidade são as partes a serem processadas em uma indústria. Estas são criadas quando chegam, movem-se através da fila (se necessário), são processadas por uma máquina e por fim saem do sistema.
- Atributos: um atributo é uma característica das entidades. Entidades semelhantes podem possuir os mesmos atributos. Os valores dos atributos é que as diferenciam umas das outras (KELTON, SADOWSKI E SADOWSKI, 1998). O uso de atributos permite não

apenas caracterizar e individualizar entidades, como também possibilita a obtenção de estatísticas importantes sobre o comportamento dos sistemas sob investigação.

- Atividades e Eventos: segundo Freitas Filho (2001), uma atividade corresponde a um período de tempo pré-determinado. Logo, uma vez iniciada, seu final pode ser programado. A duração de uma atividade, no entanto, não é, necessariamente, uma constante. Poderá ser o resultado de uma expressão matemática ou um valor aleatório com base em uma distribuição de probabilidade. Eventos são condições que ocorrem em um ponto do tempo, o qual causa uma mudança no estado do sistema. Entidades interagindo com atividades criam eventos (INGALLS, 2002). Exemplos de eventos podem ser: a chegada de peças ou clientes, o início do processamento de uma máquina, a saída de peças ou clientes do sistema e outros.
- Recursos: em simulação, recursos representam algum tipo de serviço que possui uma capacidade limitada. São exemplos de recursos: funcionários, equipamentos, espaço numa área de estocagem com tamanho limitado e outros.
- Variáveis Globais: Kelton, Sadowski e Sadowski (1998) definem variável global como sendo uma parte de informação que reflete alguma característica do sistema. Uma variável global pode capturar algo que seja de interesse do sistema como um todo. Estas variáveis são utilizadas para diversos propósitos. Como exemplo, pode-se citar a variável que acumula o número de partes em um sistema. Se uma nova parte chega ao sistema adiciona-se uma unidade à variável. Se uma parte sai do sistema, subtrai-se uma unidade.
- Gerador de números aleatórios: um gerador de números aleatórios é uma rotina que gera números aleatórios num intervalo $[a,b]$, com $a, b \in \Re$. Todos os métodos de geração de variáveis aleatórias baseiam-se na prévia geração de um número aleatório, uniformemente distribuído no intervalo $(0,1)$. Banks, Carson, Nelson et al. (2000) descrevem cinco métodos básicos voltados para a geração de amostras de variáveis aleatórias: transformação inversa, transformação direta, convolução, aceitação/rejeição e propriedades especiais. O método a ser empregado na geração das variáveis aleatórias depende do tipo de distribuição desejada e da eficiência que se está buscando no processo. Programas computacionais que geram números aleatórios utilizam um algoritmo. Desta forma, se este algoritmo é conhecido pode-se prever qual é o próximo número aleatório. Por este motivo, esse números são chamados de pseudo-aleatórios, pois uma mesma seqüência de números aleatórios pode ser sempre replicada. Para finalidade estatística, um

bom gerador de números pseudo-aleatórios é satisfatório. Os simuladores comerciais existentes utilizam números pseudo-aleatórios para atribuir às variáveis.

- Calendário: é determinado por uma lista de eventos ordenados por data e hora, agendados para ocorrerem no futuro. Em qualquer simulação existe somente um calendário, e qualquer evento a ocorrer deve estar neste calendário.
- Relógio da Simulação: o valor atual do tempo em uma simulação é atribuído a uma variável chamada de relógio do simulador. Esta variável é alterada quando ocorre uma mudança em algum evento. Como nenhuma mudança ocorre entre eventos, somente são registrados os tempos de mudança dos eventos.
- Variáveis de estado do sistema: constituem o conjunto de informações necessárias à compreensão do que está ocorrendo no sistema num determinado instante no tempo, com relação aos objetos de estudo. A determinação destas variáveis é função do propósito do estudo. Variáveis de estado definidas numa determinada investigação de um sistema podem ser completamente diferentes daquelas definidas em outro estudo sobre o mesmo sistema (FREITAS FILHO, 2001). Como exemplo podem ser citadas as variáveis de estado que controlam o número de peças esperando para serem processadas na máquina ou o estado da máquina, isto é, ocupada ou livre, número de clientes esperando na fila do caixa e outras.
- Acumulador Estatístico: é parte do simulador e serve para efetuar estatísticas de certos estados tais como, o estado dos recursos ou o valor das variáveis globais ou ainda, para avaliar o desempenho estatístico dos atributos das entidades.

Kheir (1996) aponta duas abordagens básicas para simulação de sistemas discretos: agendamento de eventos e interação de processos. O agendamento de eventos concentra-se nos eventos e seu efeito sobre o estado do sistema. As mudanças provocadas pelos eventos devem ser gravadas para posterior estudo estatístico. Na segunda abordagem, o processo é composto por uma coleção ordenada no tempo de eventos, atividades e durações de tempo relativos a uma entidade. Em simulações orientadas a processo, muitos processos ocorrem simultaneamente e envolvem interações extremamente complexas entre estes.

2.6.5 Otimização baseada em simulação

2.6.5.1 Introdução

Alguns sistemas em áreas como manufatura, gerenciamento de cadeias de suprimento e gerenciamento financeiro, são muito complexos para serem modelados analiticamente e resolvidos usando Programação Matemática. Para analisar estes sistemas, os especialistas têm utilizado a simulação. Entretanto, quando são aplicadas as técnicas de simulação para melhorar o desempenho destes sistemas, surgem várias limitações com relação à incapacidade de se avaliar mais do que uma fração do espaço de soluções viáveis. Como uma simples análise do desempenho pode ser insuficiente, recentemente, novas metodologias integrando otimização e simulação vêm sendo desenvolvidas, gerando diversos métodos híbridos para resolver estes problemas. Law e McComas (2002) apresentam uma introdução à simulação baseada em otimização, assunto que consideram a mais importante nova tecnologia em simulação desenvolvida nos últimos cinco anos.

Alguns dos primeiros trabalhos fazendo a integração de simulação de sistemas e metodologias de otimização foram apresentados por Fu (1994) e Andradóttir (1998) que aplicaram métodos de aproximação estocástica para ajustar parâmetros contínuos. Desde então, muitos pesquisadores têm se dedicado a esta área. Dentre estes, podem ser citados Ólafsson e Kim (2002), que definem otimização baseada em simulação como o processo para encontrar o melhor valor para alguma variável de decisão em um sistema onde o desempenho é avaliado através dos resultados de um modelo de simulação deste sistema.

2.6.5.2 Configuração de problemas de otimização baseada em simulação

Um problema de otimização baseada em simulação, da mesma forma que um problema geral de otimização, é constituído por:

- Variáveis de decisão;
- Função objetivo;
- Restrições.

As variáveis de decisão são denominadas θ. As restrições representadas por estas variáveis formam uma região factível Θ tal que $\theta \in \Theta$. A função objetivo é dada por uma

função real $f : \Theta \rightarrow \Re$. Devido à complexidade e natureza estocástica de certos sistemas, não existe uma forma analítica para *f(.)*. É feita então uma estimativa usando uma função com dados da simulação. Esta função é denominada de $X(\theta)$.

O processo de otimização baseada em simulação envolve um algoritmo que manipula as soluções factíveis envolvendo as variáveis de decisão θ em busca da melhor destas. As soluções geradas pelo processo são avaliadas através do simulador que o realimenta com os resultados obtidos. Este ciclo é repetido até que o critério de parada do algoritmo esteja satisfeito. A Figura 15 ilustra o processo da otimização baseada em simulação.

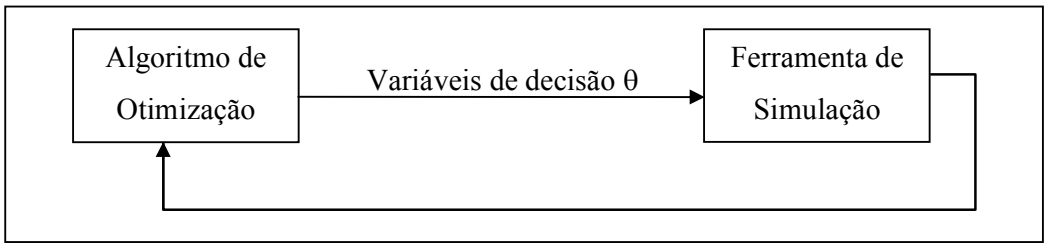

Figura 15: Processo de Otimização baseada em Simulação

2.6.5.3 Técnicas para otimização baseada em simulação

Segundo Andradóttir (FU, ANDRADÓTTIR, CARSON et al. 2000), a classe de métodos de otimização baseada em simulação inclui diferentes técnicas tais como métodos estatísticos, métodos que utilizam a estimação do gradiente para otimização com parâmetros contínuos, métodos de busca aleatória e outros.

Estas técnicas de otimização baseada em simulação podem ser classificadas de acordo com a natureza da região factível. Se esta região é de natureza contínua, isto é $\Theta \subset \Re$, então pode ser utilizado o método gradiente de busca como uma aproximação estocástica. Se a região é finita e razoavelmente pequena com $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ com $m < 30$, então é possível o uso de métodos de classificação e seleção. Se a região é finita e grande é apropriado o uso de meta-heurísticas.

Quando a região factível é discreta deve-se analisar previamente se o problema em questão possui uma região que permite uma enumeração completa de todas as possíveis soluções. Se a busca exaustiva não é viável, uma maneira de simplificar esta busca, considerando todas as possíveis combinações entre fatores e níveis, é anexar, ao simulador, ferramentas que permitam analisar os resultados das simulações e apontar caminhos que

assegurem resultados ainda melhores, descartando aqueles que apenas consumirão tempo e recursos do analista. Muitas ferramentas de otimização existentes hoje podem ser utilizadas. Dentre estas, pode-se citar: algoritmos evolucionários, meta-heurísticas, inteligência artificial, método gradiente, algoritmos de busca aleatória, métodos de programação matemática e técnicas de busca estatísticas (FU, ANDRADÓTTIR, CARSON et al. 2000). Das meta-heurísticas conhecidas, o AG, o SA e a busca tabu são as mais utilizadas e entre técnicas de inteligência computacional encontram-se algumas aplicações utilizando redes neurais. Estes métodos, que são geralmente aplicados para resolver problemas de otimização combinatória em contextos determinísticos, têm sido aplicados com sucesso na otimização baseada em simulação.

2.6.6 Aplicativos computacionais para simulação

Os primeiros trabalhos de simulação, desenvolvidos entre 1955 e 1960, utilizaram as linguagens de programação de propósito geral. Embora estas linguagens permitissem a modelagem em várias aplicações diferentes, o tempo e as habilidades de programação requeridos desencorajavam muitos modelistas em potencial. Segundo Bateman, Bowden, Gogg et al. (1999), estas linguagens de programação ainda são usadas em algumas aplicações, mas a tendência é o uso de linguagens de última geração.

As linguagens de simulação foram introduzidas a partir de 1960 e oferecem sentenças de programação especificamente projetadas para gerenciar a lógica das filas e outros fenômenos comuns aos sistemas computacionais.

Com o crescente interesse por esses métodos, começaram a ser desenvolvidos pacotes de simulação ou simuladores projetados para facilitar a modelagem rápida em um ambiente específico. Um desenvolvimento subseqüente em *software* de simulação foi o de integrar em um pacote único, a flexibilidade das linguagens com a facilidade de uso dos simuladores. Telas de entrada de dados orientadas por menu e construções direcionadas a aplicações específicas proporcionaram a modelagem rápida. As ferramentas de simulação atuais têm reduzido significativamente o esforço necessário ao processo de construção de um modelo. Habilidades em programação de computadores, embora benéficas, não são mais imprescindíveis. Outra vantagem na utilização destas ferramentas é que possuem facilidades decorrentes da animação gráfica e a saída de informações estatísticas que facilitam a crítica de modelos.

Banks, Carson, Nelson et al. (2000) dividem os aplicativos que podem ser utilizados para simulação em três categorias:

1) Linguagens de programação de propósito geral como Fortran, C, C++, *Visual Basic* e outras;
2) Linguagens de programação para simulação tais como GPSS, GPSS/H™, SIMSCRIPT 11.5, MODSIM II, SIMAN V® e SLAMSYSTEM;
3) Pacotes de simulação. Esta categoria inclui muitos produtos que se diferenciam de acordo com suas aplicações, mas que possuem características comuns.

Existem atualmente diversos pacotes de simulação também conhecidos apenas por simuladores. Podem ser citados: SIMFACTORY II.5 (GLOBE, 1991), ProModel (PRICE E HARRELL, 1999), AutoMod (ROHRER, 1999), FACTOR/AIM (LILEGDON, 1993), WITNESS (MEHTA e RAWLES, 1999), ARENA (SADOWSKI e BAPAT, 1999), Deneb/QUEST (DONALD, 1998), Extend (KRAHL, 1999), Micro Saint (BLOECHLE e LAUGHERY, 1999) e Taylor ED (HULLINGER, 1999).

Recentemente, novos aplicativos têm sido desenvolvidos para a otimização de sistema simulados, utilizando novas técnicas de busca direta associadas a algoritmos evolutivos. Embora sejam relativamente fáceis de se utilizar, tais ferramentas de otimização podem ser aplicadas de maneira mais eficiente quando utilizadas com um entendimento básico de como estas buscam uma solução ótima para um problema (BATEMAN, BOWDEN, GOGG et al. 1999). Entre os aplicativos para otimização da simulação estão: OptQuest, AutoStat, Extend Optimizer, WITNESS Optimizer e SimRunner2 (LAW e KELTON, 2000). Law e McComas (2002) enumeram os aplicativos computacionais para otimização baseada em simulação, as empresas que os desenvolveram, os simuladores nos quais são acoplados e os procedimentos heurísticos utilizados (Tabela 3).

Tabela 3: Aplicativos de Otimização baseada em Simulação

Aplicativo de otimização	Empresas	Simuladores acoplados	Procedimentos Heurísticos Utilizados
AutoStat	Brooks-PRI Automation	AutoMod, AutoSched	Estratégias Evolutivas
Extend Optimizer	Imagine That	Extend	Estratégias Evolutivas
OptQuest	Optimization Technologies	ARENA, Flexsim ED, Micro Saint, ProModel, QUEST, SIMUL8	Busca Dispersa, Busca Tabu, Redes Neurais
WITNESS	Lanner Group	WITNESS	*Simulated Annealing*, Busca Tabu

Vários estudos abordam as aplicações, vantagens e desvantagens destes pacotes computacionais. Podem ser citados Concannon, Hunter e Tremble (2003) que estudaram a aplicação do SIMUL8 em planejamento e programação de produção; Harrell e Price (2003) analisaram os aplicativos de modelagem em simulação da ProModel e relataram que estes são potentes e de fácil utilização para modelagem de todo tipo de sistemas e processos; Bloechle e Schunk (2003) relataram que o simulador Micro Saint Sharp tem sido utilizado para auxiliar as forças armadas e também companhias comerciais. Este aplicativo auxilia em questões de como melhorar o desempenho e utilização de processos; Krahl (2003) analisa o aplicativo Extend e relata que este fornece ferramentas para todos os níveis de modeladores para eficientemente criar modelos precisos, com credibilidade e aplicáveis a diversos sistemas; Nordgren (2003) classifica o simulador FlexSim como um aplicativo orientado a objeto que pode desenvolver, modelar, simular, visualizar e monitorar atividades de sistemas com processos de fluxo dinâmico. Descreve também as aplicações e benefícios deste aplicativo; Williams e Gunal (2003) apresentam uma revisão e um manual resumido do aplicativo FlexSim com enfoque em análise e simulação de cadeias de suprimento; Rohrer (2003) enumera como o AutoMod pode ser utilizado para aumentar o retorno de investimentos através de simulação; Bapat e Sturrock (2003) introduzem o pacote de produtos ARENA para modelar, simular e otimizar, destacando os pontos marcantes da arquitetura e tecnologia deste aplicativo.

A análise para escolha de um simulador por parte das empresas não tem sido uma tarefa fácil, pois existe uma grande variedade de aplicativos e cada um com suas características. Tewoldeberhan, Verbraeck, Valentin et al (2002) propuseram uma metodologia para auxiliar na seleção de um novo simulador por eventos discretos para companhias internacionais que possuem sua própria equipe de simulação. Observaram que preferências e áreas de aplicação diferem entre países e os simuladores já em uso influenciam a decisão de um processo de seleção. A metodologia por eles proposta é composta de duas fases: a primeira reduz rapidamente a extensa lista de opções para uma pequena lista de aplicativos. A segunda compara as necessidades da companhia com as características de cada simulador. Após terem testado a metodologia, concluíram que a mesma mostrou-se efetiva em termos de qualidade e eficiente com relação ao tempo. Pode ser facilmente aplicada para grandes organizações com uma equipe de especialistas em simulação.

Além das opções oferecidas pelos próprios simuladores existentes para a otimização, pode-se optar por utilizar outros aplicativos computacionais existentes, juntamente com o simulador, se a junção destes se mostrar conveniente para o problema em questão. Vamanan,

Wang, Batta et al (2004) descrevem em seu artigo a integração dos aplicativos CPLEX e ARENA. O simulador ARENA foi utilizado para simular a logística de estoque de centros de distribuição e os parâmetros encontrados são incorporados a um problema inteiro misto que é então resolvido pelo CPLEX. A dificuldade encontrada pelos autores foi o elevado tempo computacional quando o número de centros de distribuição foi aumentado. Uma alternativa por eles sugerida foi a de se acatar a solução com um nível de otimalidade aceitável.

2.6.7 Aplicações de simulação

Existem diversos artigos em revistas científicas e anais de congressos, encontros e conferências referentes à simulação. Dentre estes, alguns foram selecionados e são aqui citados, por estarem de alguma forma relacionados a este trabalho.

Paolucci, Sacile e Boccalatte (2002) desenvolveram seu trabalho focados no problema de alocação do óleo cru descarregado por navios em tanques do porto e da refinaria. Dois aspectos de alocação discretos influem neste processo: a chegada dos navios e a seqüência de lotes de óleo cru processados na refinaria. Foi proposta pelos autores uma abordagem de simulação que pode ser aplicada como um simulador do fluxo de óleo cru no sistema da refinaria, como suporte de aprendizado para novos operadores e também como sistema de suporte à decisão. Aplicaram a metodologia desenvolvida em uma refinaria de pequeno a médio porte. Os resultados mostraram que a técnica abordada é um ponto de partida para o estudo de estratégias mais complexas que possam fornecer aos tomadores de decisão, soluções que sejam factíveis e aceitáveis simultaneamente.

Stebel (2001) desenvolveu um modelo em redes de Petri para modelar as operações de transferência e estocagem de gás liquefeito de petróleo em uma indústria petrolífera com o objetivo de melhorar a capacidade de tomada de decisão do operador. Este modelo de simulação pode ser utilizado para experimentar planos de envio e recebimento, apesar da complexidade das operações a serem realizadas, e permite o diagnóstico de pontos críticos do problema e a experimentação de diferentes abordagens para sua solução.

Analisando a crescente demanda por produtos individualizados, Graupner, Richter e Sihn (2002) concluíram que a configuração de sistemas está se tornando cada vez mais importante. Alguns sistemas de configuração que podem ser acessados pela *Internet* são utilizados por vendedores e clientes independente da localização e do horário de acesso. Os autores propõem em seu trabalho um novo serviço eletrônico para configuração, simulação e

animação de um sistema de manufatura. Este serviço permite apresentar, testar e otimizar o sistema de manufatura via *Internet*. O configurador testado mostrou uma redução de tempo no contato inicial do cliente na elaboração do pedido. Os autores concluem que este sistema de configuração provavelmente será um sucesso de mercado e em breve irá tornar-se um serviço essencial em vendas e distribuição.

Chwif, Barretto e Saliby (2002) apresentaram duas técnicas de avaliação para a análise de cadeias de suprimento: análise de planilha e simulação por eventos discretos. A primeira é muito simples e clara para implementação, mas não considera as características dinâmicas da cadeia e não leva em consideração a variabilidade dos parâmetros. Por outro lado, a segunda considera estes elementos. Comparando as duas abordagens em um estudo de caso, os autores chegaram à conclusão de que o efeito provocado pela variação de alguns parâmetros, como tempo de transporte, pode não interferir significativamente no resultado da cadeia, mas a variação da demanda mostrou-se um fator chave no desempenho da cadeia. Neste ponto, a simulação mostrou vantagens sobre a técnica através da planilha na análise de cadeias de suprimento.

Higuchi e Troutt (2004) estudaram a dinâmica de uma cadeia de suprimento para o caso de um produto com ciclo de vendas curto. Desenvolveram um modelo de simulação que incorpora um processo de retro-alimentação e demonstraram o impacto da dinâmica de uma cadeia de suprimento. Observam que sem modelos de simulação é muito difícil compreender esta dinâmica. Concluíram que o modelo por eles desenvolvido pode contribuir para tomadas de decisão, tais como: níveis de capacidade de manufatura e publicidade e também a decisão sobre o momento apropriado para oferecer o produto ao mercado estrangeiro.

Musselman, O'Reilly e Duket (2002) observam em seu artigo que a tarefa de planejar e programar a produção em manufatura tem evoluído de um simples sistema de planejamento de pedidos de material para sofisticados sistemas avançados de planejamento e programação de produção (*Advanced Planning and Scheduling – APS*). A combinação da tecnologia de planejamento *APS* com processo de gerenciamento de empreendimento através de simulação permite: aprimorar o serviço ao cliente; cumprir datas de entrega de pedidos; reduzir o trabalho, as horas extras e estoque; incrementar qualidade; aumentar lucros. A função de programação da produção por simulação em sistemas deste tipo é apropriada em grande parte por sua habilidade em imitar o mundo real.

Fábrica virtual é uma ferramenta para programação de produção de problemas do tipo *job shop* e foi desenvolvida por Thoney, Joines, Manninagarajan et al. (2002) no estado da Carolina do Norte nos Estados Unidos da América. Esta ferramenta serve para gerar soluções

próximas do ótimo para problemas industriais em segundos, através da comparação com um limite inferior computado. É um procedimento iterativo baseado em simulação cujo objetivo é minimizar o máximo de atraso. Foi testada para problemas em indústrias com vários pedidos diferentes; novos pedidos chegando enquanto os antigos estão sendo atendidos. Mostrou um bom desempenho para estes casos com uma variedade de condições diferentes.

A otimização baseada em simulação vêm sendo empregada para auxílio no gerenciamento de empresas em vários ramos. Algumas destas aplicações são relatadas a seguir.

Piera, Narciso, Guasch et al. (2004) desenvolveram um simulador em Redes de Petri para gerar uma política de agendamento com o objetivo de satisfazer certos objetivos em um sistema complexo logístico. O grande número de variáveis de decisão deste problema leva a uma árvore de busca com um número elevado de ramificações, o que torna a resolução deste, praticamente impossível em termos computacionais. Desta forma, o simulador foi desenvolvido, tal que diferentes algoritmos heurísticos podem ser implementados para orientar o simulador em direção à melhor política de agendamento.

Allaoui e Artiba (2004) estudaram o problema de alocação de *flow shop* híbrido com o objetivo de minimizar o tempo de fluxo e o atraso nas datas de entrega, enquanto satisfaz as restrições. Tempos de preparação, limpeza e transporte foram também considerados. Três objetivos foram desenvolvidos: o primeiro foi mostrar como podem ser integradas a simulação e a meta-heurística SA para resolver o problema; o segundo objetivo foi ilustrar através de um estudo experimental que o desempenho da heurística aplicada ao problema pode ser afetada por certas porcentagens de tempo de preparação, limpeza e transporte; finalmente, mostrou-se que a abordagem utilizada gera melhores resultados que a heurística NEH (NAWAZ, ENSCORE e HAM, 1983), sob determinadas condições.

Cheng, Feng e Chen (2005) propuseram um mecanismo de otimização híbrido que integra algoritmos heurísticos, AG e simulação, para encontrar a melhor combinação de recursos que produza o desempenho ótimo do sistema. Um estudo de caso mostrou que este novo mecanismo híbrido, juntamente com a implementação computacional pode, de forma eficiente e precisa, gerar a solução ótima, auxiliando desta forma engenheiros a agilizar o processo de planejamento das operações da construção.

Em seu trabalho, Cave, Nahavandi e Kouzani (2002) apresentaram uma otimização da simulação para um problema real de *scheduling* em uma indústria. Utilizaram nesta aplicação a meta-heurística SA. Os autores mostraram que a utilização deste método gerou alocações de alta qualidade em tempo considerado razoável.

Aoki, Nakayama, Yamamoto et al. (1991) desenvolveram um novo método de alocação para processo industrial baseado em um algoritmo de otimização heurística e simulação de eventos discretos. O algoritmo de otimização determina a seqüência em que os produtos devem ser enviados para a produção e o simulador especifica o fluxo para cada produto. A função do simulador é fazer apenas uma otimização local. Os autores concluíram, através de uma aplicação real, que a arquitetura "seqüenciamento de produtos" e "simulação" foi satisfatória.

No artigo de Baesler, Moraga e Ramis (2002), é apresentado o resultado obtido através da aplicação de uma metodologia de otimização baseada em simulação para a linha de produção de uma indústria de processamento de troncos de madeira. O modelo de simulação construído no aplicativo ARENA foi integrado à técnica de AG. Os resultados obtidos mostraram que, utilizando uma configuração diferente de recursos da planta, é possível reduzir a média de tempo total do ciclo em 18%. A configuração de recursos do resultado foi obtida avaliando-se apenas 1,6% do número total de possíveis combinações.

Finke e Traband (2002) relataram os primeiros resultados do desenvolvimento de um procedimento de *scheduling* para uma indústria de chapas de aço. A abordagem utiliza a meta-heurística Busca Tabu, combinada com simulação, para alocar os produtos a um conjunto de máquinas. O desempenho deste procedimento foi avaliado em comparação com a solução ótima para problemas de pequena instância e comparados com outras heurísticas para problemas maiores. Os resultados mostraram que a busca tabu funcionou bem para este problema. Esta técnica aplicada com simulação permitiu a incorporação de restrições mais realistas da operação do sistema.

O artigo de Law e McComas (2002) abordando uma introdução à otimização baseada em simulação, mostra um exemplo de um sistema de manufatura. Com o objetivo de comparar aplicativos, foi feita a otimização de alguns dos parâmetros da simulação utilizando o OptQuest com o ARENA e o WITNESS Optimizer com o WITNESS. Os autores concluíram que o desempenho foi satisfatório para o problema abordado e para o conjunto de parâmetros utilizados.

Pichitlamken e Nelson (2002) propuseram em seu trabalho um algoritmo de otimização através de simulação, para ser utilizado quando a medida de desempenho é estimada por uma simulação de eventos discretos estocástica e as variáveis de decisão podem estar sujeitas a restrições inteiras lineares e determinísticas. A abordagem consiste em um sistema de direcionamento global, um procedimento de seleção do melhor resultado e um aperfeiçoamento local. Este algoritmo mostrou-se promissor em testes numéricos aplicados.

Aomar (2002) desenvolveu uma metodologia para resolver problemas de planejamento paramétrico em sistemas de produção e comércio com uma complexidade considerável. A solução é focada na determinação do conjunto ótimo de parâmetros críticos do sistema tal que cada resposta do sistema esteja no seu nível de desempenho ótimo com a menor variabilidade possível. Os autores propuseram uma metodologia de solução que integra quatro módulos com métodos comprovados: modelagem por simulação, AG, método de entropia e módulo robusto de Taguchi.

Um sistema automatizado de apoio à decisão foi desenvolvido por Bruzzone, Orsoni, Mosca et al. (2002) para otimizar a logística do transporte marítimo de uma grande companhia de produtos químicos. O artigo relata o projeto e implementação de um módulo de otimização baseado em AG para complementar um sistema de suporte à decisão, incluindo banco de dados dinâmico, heurística de decisão e um processo dinâmico de simulação. O objetivo deste módulo é gerar sistematicamente uma configuração da frota que seja capaz de atender à demanda de recursos da indústria.

Joines, Gupta, Gokce et al. (2002) consideram crítica a decisão sobre a aquisição de matéria-prima nas indústrias. Para resolver este problema acoplaram a um simulador de cadeia de suprimento já existente, uma metodologia que utiliza AG para otimizar os parâmetros do sistema. Vetores são construídos com as variáveis de decisão, formando os cromossomos, que são então utilizados para a aplicação da técnica de AG. A função do simulador é a de encontrar o valor de desempenho dos cromossomos. A eficiência da técnica de otimização da simulação foi testada com um conjunto de dados reais que são normalmente analisados pela empresa. Segundo os autores, pelo fato da rotina de otimização ter funcionado bem, o código da simulação está sendo expandido para englobar níveis de serviços diferentes em função do volume de demanda.

CAPÍTULO 3

PLIM: MODELOS DE TEMPO DISCRETO E CONTÍNUO

3.1 MODELAGEM MATEMÁTICA DO PROBLEMA

Neste trabalho foram desenvolvidas algumas técnicas para solução de problemas de *scheduling* de curto termo em um sub-sistema de uma refinaria de petróleo. A busca por técnicas mais eficientes se justifica, principalmente, pela complexidade das operações de programação de produção que geram problemas combinatórios de grande porte e também pelas limitações computacionais encontradas no processamento. Em geral, problemas de *scheduling* em refinarias de petróleo são formulados como modelos de otimização inteira mista e resolvidos utilizando-se técnicas exatas, mas o tempo despendido para chegar à solução é inviável para a implementação dos mesmos.

O sub-sistema estudado neste trabalho envolve as operações de transferência e estocagem de diesel. São considerados o desenvolvimento e a solução de modelos de otimização para o *scheduling* de um conjunto de operações que incluem: o recebimento do diesel nos tanques, armazenamento e envio deste produto a clientes. A Figura 16 ilustra este sub-sistema.

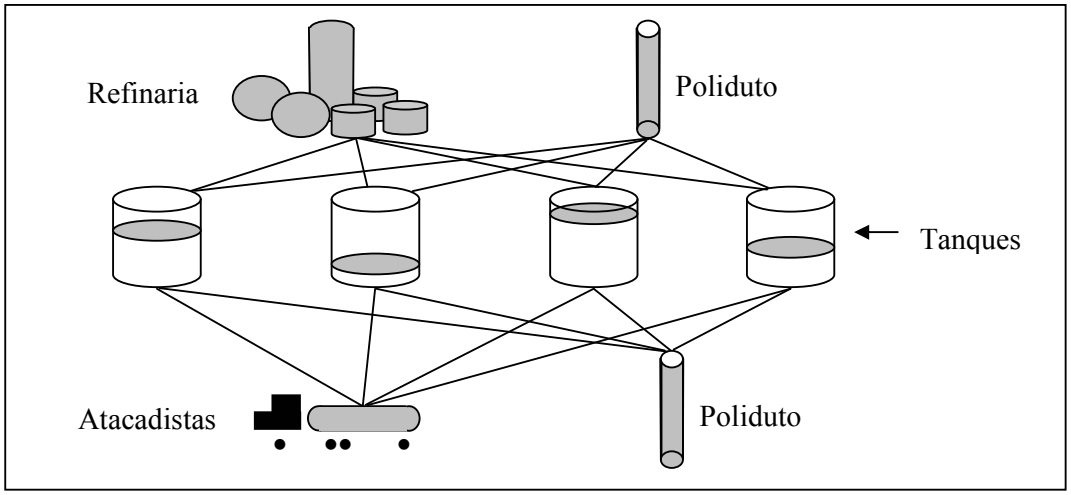

Figura 16: Representação esquemática do sub-sistema de transferência e estocagem de diesel

Dois modelos foram desenvolvidos para o problema de *scheduling* deste sub-sistema:

- O primeiro foi modelado usando discretização uniforme do tempo;
- O segundo teve sua modelagem baseada em tempo contínuo;

O objetivo geral para a primeira modelagem é a obtenção da seqüência para o fluxo e armazenamento do diesel de forma a atender às restrições de operações e de demandas a um custo mínimo. A outra modelagem minimiza o tempo total para a realização de todos os envios de diesel aos clientes.

Os parâmetros envolvidos na modelagem deste problema incluem a configuração do parque de tancagem, as restrições de operação e a demanda de diesel como dados de entrada e o gerenciamento de atividades como dados de saída. A Figura 17 ilustra o relacionamento destes parâmetros com o modelo matemático.

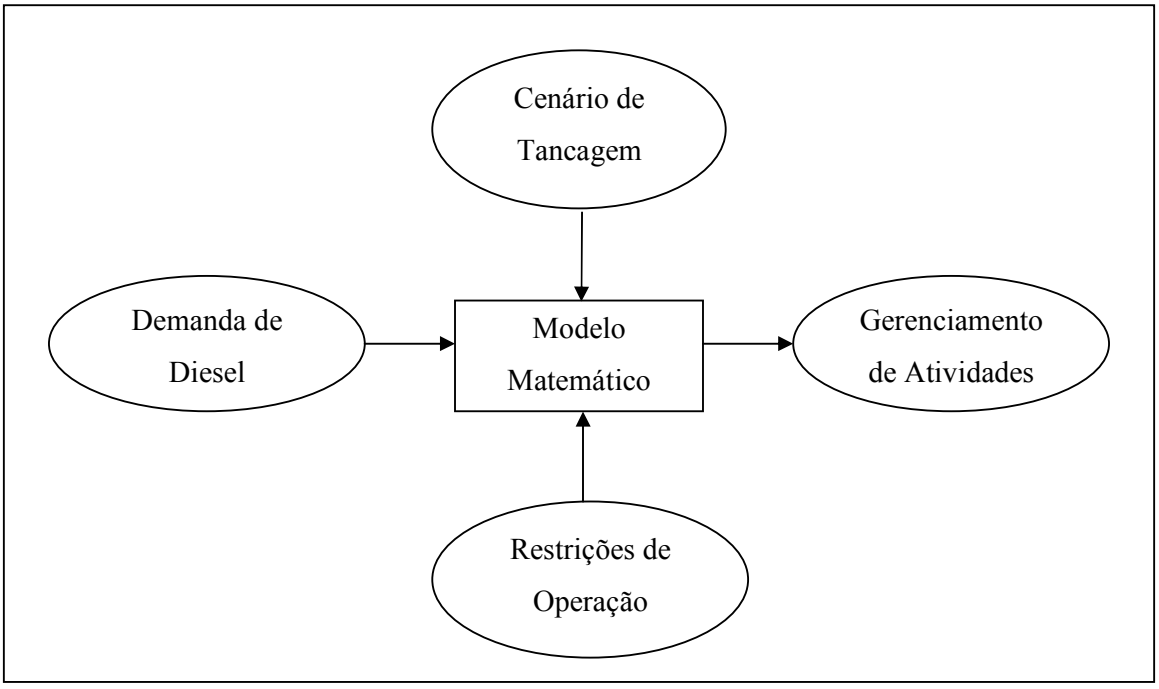

Figura 17: Modelo Matemático – Entradas e Saída

Para a resolução dos modelos foram utilizadas as seguintes técnicas: *branch and bound*, AG, Algoritmo Transgenético ProtoG e Simulação.

Os dois modelos desenvolvidos consideraram a demanda requerida de diesel por parte de clientes finais, pré-definida pelo *planning* da refinaria, o cenário do parque de tancagem da refinaria e as fontes de recebimento de diesel nos tanques.

As seguintes premissas e restrições operacionais foram consideradas nas modelagens do problema:

- Os volumes de diesel estocados nos tanques são conhecidos;
- Os tanques são dedicados, ou seja, armazenam um único tipo de diesel;
- Todos os tanques devem estar disponíveis durante o horizonte de planejamento, ou seja, não devem sofrer manutenção ou qualquer operação que impossibilite seu uso;
- Os tempos de transição entre tarefas não são considerados, por serem desprezíveis em relação às demais operações;
- Cada um dos tanques não pode efetuar operações de carga e descarga simultaneamente;
- Cada tanque pode receber de uma única origem num determinado intervalo de tempo;
- Cada tanque pode enviar para um único destino num determinado intervalo de tempo;
- As restrições operacionais de vazão mínima/máxima de fluxo de recebimento e envio de produtos devem ser respeitadas;
- Os tanques não poderão estar com volume abaixo do mínimo e nem acima do máximo estipulados;
- Quando um tanque iniciar o recebimento de carga, deverá fazê-lo até o enchimento total do mesmo;
- Após o enchimento, o tanque deverá ficar parado por um período mínimo estipulado para repouso e análise do produto estocado. Após este período o tanque estará disponível para envio;
- Todo o lote requerido pelo cliente deve ser bombeado de forma ininterrupta;
- As demandas de recebimento e envio podem ser satisfeitas em qualquer instante do horizonte de planejamento.

3.2 MODELO COM DISCRETIZAÇÃO UNIFORME DE TEMPO

O primeiro modelo matemático para o problema de transferência e estocagem utiliza a modelagem PLIM com discretização uniforme no tempo. São utilizadas variáveis contínuas, inteiras e binárias. As variáveis binárias representam as decisões a serem tomadas. Eventos de qualquer tipo devem começar e terminar no início de cada intervalo de tempo. A principal vantagem desta formulação é a facilidade na modelagem que fornece o posicionamento de todas as operações que competem por recursos compartilhados. A principal desvantagem é com relação ao tamanho do intervalo de tempo. Se o intervalo for muito grande poderão

ocorrer perdas devido à diferença entre o tempo real e o tempo discreto utilizado. Por outro lado, se o intervalo for pequeno, isto gerará uma grande quantidade de variáveis, aumentando consideravelmente o tempo de processamento.

Como já citado, o objetivo deste modelo é o de minimizar custos operacionais. As restrições foram construídas respeitando-se os procedimentos operacionais, as restrições físicas do processo e a demanda. A restrição que trata da necessidade de enchimento total do tanque quando recebe produto e a restrição para o tempo de certificação não foram considerados neste modelo.

3.2.1 Nomenclatura utilizada no Modelo

Índices:

- tq - representa o número do tanque: $tq = 1,2,...,TQ$
- c - representa o número do cliente: $c = 1,2,...,C$
- t - representa o número do intervalo de tempo: $t = 1,2,...,T$

Conjuntos:

- CTQ - conjunto formado pelos tanques tq
- CC - conjunto formado pelos clientes c
- CT - conjunto formado pelos intervalos de tempo t

Vale observar o uso de índices subscritos, para representar os tanques, os clientes ou os intervalos de tempo envolvidos nos parâmetros e variáveis.

Parâmetros:

- CB_c - custo de bombeio para o cliente c (unidades monetárias/mil m^3);
- CA_{tq} - custo de armazenamento no tanque tq (unidades monetárias/mil m^3);
- CTR_{tq} - custo de troca do tanque tq que recebe da produção (unidades monetárias);
- $QRmin$ - vazão mínima de recebimento pelos tanques tq (mil m^3/h);
- $QRmax$ - vazão máxima de recebimento pelos tanques tq (mil m^3/h);
- $QEmin_c$ - vazão mínima de envio ao cliente c (mil m^3/h);
- $QEmax_c$ - vazão máxima de envio ao cliente c (mil m^3/h);
- $Volmin_{tq}$ - volume mínimo permitido no tanque tq (mil m^3);
- $Volmax_{tq}$ - volume máximo permitido no tanque tq (mil m^3);
- $Volini_{tq}$ - volume no tanque tq no início do processo (mil m^3);
- DEM_c - demanda de diesel do cliente c (mil m^3).

Variáveis Binárias:

$$Rec_{tq,t} = \begin{cases} 1, \text{se o tanque } tq \text{ recebe diesel da produção no intervalo de tempo } t \\ 0, \text{caso contrário} \end{cases}$$

$$Env_{tq,c,t} = \begin{cases} 1, \text{se o tanque } tq \text{ envia diesel para o cliente } c \text{ no intervalo de tempo } t \\ 0, \text{caso contrário} \end{cases}$$

$$XI_{c,t} = \begin{cases} 1, \text{se o cliente } c \text{ começa a receber diesel no intervalo de tempo } t \\ 0, \text{caso contrário} \end{cases}$$

$$XF_{c,t} = \begin{cases} 1, \text{se o cliente } c \text{ termina de receber diesel no intervalo de tempo } t \\ 0, \text{caso contrário} \end{cases}$$

$$TR_{tq,tq',t} = \begin{cases} 1, \text{se o tanque } tq \text{ termina de receber e o tanque } tq' \text{ começa a receber no intervalo de tempo } t \\ 0, \text{caso contrário} \end{cases}$$

Variáveis Contínuas:

$QR_{tq,t}$ - volume de diesel recebido pelo tanque *tq* no intervalo de tempo *t*;

$QE_{tq,c,t}$ - volume de diesel enviado pelo tanque *tq* ao cliente *c* no intervalo de tempo *t*;

$Vol_{tq,t}$ - volume de diesel estocado no tanque *tq* no tempo *t*;

TI_c - marca o momento de início do recebimento do cliente *c*;

TF_c - marca o momento de término do recebimento do cliente *c*;

3.2.2 Função Objetivo

Para este modelo assume-se que a operação ótima do problema de transferência e estocagem é aquela que minimiza os custos operacionais do processo, definidos pelos custos de bombeamento de envio adicionado ao custo de armazenamento de produto nos tanques e, finalmente, adicionado ao custo de transição por troca de tanques na operação de recebimento do processo.

$$Custos = \sum_{tq \in CTQ} \sum_{c \in CC} \sum_{t \in CT} CB_c . QE_{tq.c.t} + \sum_{tq \in CTQ} \sum_{t \in CT} CA_{tq} . Vol_{tq,t} + \sum_{tq \in CTQ} \sum_{tq' \neq tq \in CTQ} \sum_{t \in CT} CTR_{tq} . TR_{tq,tq',t} \quad (18)$$

3.2.3 Restrições

As seguintes restrições são utilizadas no modelo:

1) Restrições de operação:

Um tanque não pode receber e enviar diesel ao mesmo tempo. Portanto, são três os estados possíveis para o tanque: recebendo, enviando ou ocioso.

$$Rec_{tq,t} + \sum_{c \in CC} Env_{tq,c,t} \leq 1 \qquad \forall\, tq = 1,...,TQ;\ t = 1,...,T \tag{19}$$

Como o fluxo de produto do processo é contínuo, haverá sempre um tanque recebendo diesel em cada intervalo de tempo.

$$\sum_{tq \in CTQ} Rec_{tq,t} = 1 \qquad \forall\, t = 1,...,T \tag{20}$$

Somente um tanque poderá enviar diesel para um determinado cliente em cada intervalo de tempo.

$$\sum_{tq \in CTQ} Env_{tq,c,t} \leq 1 \qquad \forall\, c = 1,...,C;\ t = 1,...,T \tag{21}$$

Quando um cliente iniciar seu recebimento de diesel, deverá fazê-lo sem interrupção. Portanto, haverá apenas uma variável de início e de término igual a 1.

$$\sum_{t \in CT} XI_{c,t} \leq 1 \qquad \forall\, c = 1,...,C \tag{22}$$

$$\sum_{t \in CT} \left(XI_{c,t} - XF_{c,t} \right) = 0 \qquad \forall\, c = 1,...,C \tag{23}$$

O momento de início e término de bombeio são armazenados nas variáveis auxiliares TI_c e TF_c.

$$\sum_{t \in CT} \left(t.XI_{c,t} \right) = TI_c \qquad \forall\, c = 1,...,C \tag{24}$$

$$\sum_{t \in CT} \left(t.XF_{c,t} \right) = TF_c \qquad \forall\, c = 1,...,C \tag{25}$$

2) Restrições de Fluxo:

O fluxo de recebimento deve estar entre o fluxo mínimo e máximo estipulados se a variável $Rec_{tq,t}$ for igual a "1". Se o valor da variável $Rec_{tq,t}$ for "0", o fluxo $QR_{tq,t}$ também será igual a "0".

$$QR\,min.Rec_{tq,t} \le QR_{tq,t} \le QR\,max.Rec_{tq,t} \qquad \forall\ tq = 1,...,TQ;\ t = 1,...,T \tag{26}$$

O fluxo de envio deve estar entre o fluxo mínimo e máximo estipulados se a variável $Env_{tq,c,t}$ for igual a "1". Se o valor da variável $Env_{tq,c,t}$ for "0", o fluxo $QE_{tq,c,t}$ também será igual a "0".

$$QE\,min_c.Env_{tq,c,t} \le QE_{tq,c,t} \le QE\,max_c.Env_{tq,c,t} \quad \forall\ tq = 1,...,TQ;\ c = 1,...,C;\ t = 1,...,T \tag{27}$$

3) Balanço de Massa:

O volume de um tanque num certo intervalo de tempo deve ser igual ao volume inicial, mais o volume recebido do processo menos o volume enviado a clientes até este intervalo de tempo.

$$Vol_{tq,t} = Volini_{tq} + \sum_{\substack{t' \in CT \\ t' \le t}} \left(QR_{tq,t'} - \sum_{\substack{c \in CC \\ t' \le t}} QE_{tq,c,t'} \right) \qquad \forall\ tq = 1,...,TQ;\ t = 1,...,T \tag{28}$$

O volume dos tanques deve estar sempre entre o volume mínimo e o máximo determinados.

$$Vol\,min_{tq} \le Vol_{tq,t} \le Vol\,max_{tq} \qquad \forall\ tq = 1,...,TQ;\ t = 1,...,T \tag{29}$$

4) Restrição de Demanda:

A demanda dos clientes deverá ser cumprida em sua totalidade.

$$\sum_{tq \in CTQ} \sum_{t \in CT} QE_{tq,c,t} = DEM_c \qquad \forall\ c = 1,...,C \tag{30}$$

5) Restrições de Transição:

A transição ocorre quando num intervalo de tempo um certo tanque está recebendo do processo e no intervalo seguinte um outro tanque é que recebe.

$$TR_{tq,tq',t} \leq Rec_{tq,t-1} \qquad \forall\, t = 2,...,T;\ tq,tq' = 1,...,TQ,\ tq \neq tq' \tag{31}$$

$$TR_{tq,tq',t} \leq Rec_{tq',t} \qquad \forall\, t = 2,...,T;\ tq,tq' = 1,...,TQ,\ tq \neq tq' \tag{32}$$

$$TR_{tq,tq',t} \geq Rec_{tq,t-1} + Rec_{tq',t} - 1 \qquad \forall\, t = 2,...,T;\ tq,tq' = 1,...,TQ,\ tq \neq tq' \tag{33}$$

As restrições a seguir fazem com que a variável $XI_{c,t}$ assuma valor "1" se no intervalo (t-1) o cliente não está recebendo e passa a receber no intervalo t e assume valor "0" para qualquer outra situação.

$$XI_{c,t} \leq \sum_{tq \in CTQ} Env_{tq,c,t} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{34}$$

$$XI_{c,t} \leq 1 - \sum_{tq \in CTQ} Env_{tq,c,t-1} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{35}$$

$$XI_{c,t} \geq \sum_{tq \in CTQ} Env_{tq,c,t} - \sum_{tq \in CTQ} Env_{tq,c,t-1} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{36}$$

As restrições a seguir fazem com que a variável $XF_{c,t}$ assuma valor "1" se no intervalo (t - 1) o cliente está recebendo e passa a não receber no intervalo t e assume valor "0" para qualquer outra situação.

$$XF_{c,t} \leq \sum_{tq \in CTQ} Env_{tq,c,t-1} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{37}$$

$$XF_{c,t} \leq 1 - \sum_{tq \in CTQ} Env_{tq,c,t} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{38}$$

$$XF_{c,t} \geq \sum_{tq \in CTQ} Env_{tq,c,t-1} - \sum_{tq \in CTQ} Env_{tq,c,t} \qquad \forall\, t = 2,...,T;\ c = 1,...,C \tag{39}$$

As restrições (34) a (39) não contemplam o primeiro intervalo de tempo. Se no início do período algum tanque estiver bombeando, este será considerado o início do bombeio. Nenhum bombeio poderá encerrar no primeiro intervalo.

$$XI_{c,1} = \sum_{tq \in CTQ} Env_{tq,c,1} \qquad \forall\, c = 1,...,C \tag{40}$$

$$XF_{c,1} = 0 \qquad \forall\, c = 1,...,C \tag{41}$$

3.3 MODELO COM REPRESENTAÇÃO DE TEMPO CONTÍNUO

Neste modelo, as variáveis contínuas que determinam os instantes de tempo são seqüenciadas em eventos que representam as operações de recebimento e envio de produto pelos tanques. O controle da temporização do processo é dependente das decisões representadas pelas variáveis binárias do modelo. Cada elemento do sistema possui sua própria temporização em eventos, representadas por variáveis contínuas que marcam o tempo de início e término da operação. Estas variáveis estão ligadas às decisões do modelo, obedecendo à seguinte regra: para um dado evento, se houver decisões envolvendo dois dos elementos do sistema, as temporizações desses devem estar vinculadas e devem obedecer às temporizações determinadas em eventos precedentes. Se comparado com o primeiro modelo em PLIM, este modelo apresenta vantagens, pois retrata melhor a realidade do processo descrito e aceita mais de um tipo de processo (produção ou poliduto) e de produto.

O objetivo deste modelo é o de minimizar o tempo total despendido na entrega de produtos aos clientes. As restrições operacionais a serem cumpridas são as descritas na Seção 3.1.

3.3.1 Nomenclatura utilizada no Modelo

Este modelo está organizado através dos seguintes índices, conjuntos, parâmetros e variáveis:

Índices:

- tq - representa o número do tanque: $tq = 1,2,...,TQ$
- c - representa o número do cliente: $c = 1,2,...,C$
- p - representa o tipo de processo (refinaria ou poliduto): $p = 1,2,...,P$
- d - representa o tipo de diesel: $d = 1,2,...,D$
- e - representa o evento de tempo: $e = 1,2,...,E$

Vale observar o uso de índices subscritos para representar os tanques, os clientes, os processos, os tipos de produto ou os eventos de tempo envolvidos nos conjuntos, parâmetros e variáveis.

Conjuntos:

- CTQ - conjunto dos tanques *tq*;
- CC - conjunto dos clientes *c*;
- CP - conjunto dos processos *p*;
- CD - conjunto dos tipos de diesel *d*;
- CE - conjunto dos eventos de tempo *e*;
- $CTQD_{tq}$ - conjunto de tipos de diesel que podem ser armazenados no tanque *tq*;
- CCD_c - conjunto de tipos de diesel com demanda pelo cliente *c* maior que zero;
- CPD_p - conjunto de tipos de diesel com lote de recebimento do processo *p* maior que zero;
- $CDTQ_d$ - conjunto de tanques *tq* que recebem diesel do tipo *d*.

Parâmetros:

- QR_p - vazão de recebimento pelos tanques do processo *p* (mil m^3/h);
- QE_c - vazão de envio ao cliente *c* (mil m^3/h);
- $Volmin_{tq}$ - volume mínimo permitido no tanque *tq* (mil m^3);
- $Volmax_{tq}$ - volume máximo permitido no tanque *tq* (mil m^3);
- $Volini_{tq}$ - volume no tanque *tq* no início do processo (mil m^3);
- $Dem_{c,d}$ - demanda do cliente *c* de diesel do tipo *d* (mil m^3);
- $LoteProc_{p,d}$ - lote de diesel do tipo *d* a ser recebido do processo *p* no horizonte de planejamento (mil m^3/h);
- TC - tempo para certificação do diesel após o enchimento do tanque (h);
- $TqProd_{tq,d}$ - assume valor "1" se o tanque *tq* é dedicado ao recebimento do diesel do tipo *d*. Caso contrário, seu valor será "0".
- EPS - número positivo próximo de zero.

Variáveis Binárias:

$$Rec_{tq,p,d,e} = \begin{cases} 1, \text{se o tanque } tq \text{ recebe do processo } p \text{ o diesel } d \text{ no evento } e \\ 0, \text{caso contrário} \end{cases}$$

$$Env_{tq,c,d,e} = \begin{cases} 1, \text{se o tanque } tq \text{ envia para o cliente } c \text{ o diesel } d \text{ no evento } e \\ 0, \text{caso contrário} \end{cases}$$

$$DemVol_{tq,c,d,e} = \begin{cases} 1, \text{se a demanda do cliente } c \text{ pelo diesel } d \text{ no evento } e \text{ é maior} \\ \quad \text{ou igual ao volume no tanque } tq \text{ no evento } e \\ 0, \text{caso contrário} \end{cases}$$

Variáveis Contínuas não Negativas:

H	-	limitante superior para tempo;
$VolAtual_{tq,e}$	-	volume no tanque *tq* no início do evento *e*;
$VolFinal_{tq}$	-	volume no tanque *tq* no final do evento *E* (último evento);
$DemAtual_{c,d,e}$	-	saldo de demanda do cliente *c* pelo produto *d* no evento *e*;
$DemFinal_{c,d}$	-	saldo de demanda do cliente *c* pelo produto *d* no final do evento *E*;
$XITanque_{tq,e}$	-	marca o momento de início de operação do tanque *tq* no evento *e*;
$XFTanque_{tq,e}$	-	marca o momento de término de operação do tanque *tq* no evento *e*;
$XICliente_{c,e}$	-	marca o momento de início de recebimento do cliente *c* no evento *e*;
$XFCliente_{c,e}$	-	marca o momento de término de recebimento do cliente *c* no evento *e*;
$XIProcesso_{p,e}$	-	marca o momento de início do envio do processo *p* no evento *e*;
$XFProcesso_{p,e}$	-	marca o momento de término de envio do processo *p* no evento *e*;
$VolEnv_{tq,c,d,e}$	-	volume de diesel do tipo *d* enviado do tanque *tq* ao cliente *c* no evento *e*;
$VolRec_{tq,p,d,e}$	-	volume de diesel do tipo *d* recebido no tanque *tq* do processo *p* no evento *e*;
$MaxXFCliente$	-	maior valor entre todas as variáveis $XFCliente_{c,e}$ no último evento

3.3.2 Função Objetivo

O objetivo para este modelo é enviar toda a quantidade demandada pelos clientes, no menor tempo possível. Pode-se observar que a restrição (80) faz com que a variável *MaxXFCliente* assuma o maior valor encontrado entre as variáveis $XFCliente_{c,E}$, ao final do último evento. Esta é a variável a ser minimizada.

$$min \;\; TempoMáximo = MaxXFCliente \tag{42}$$

3.3.3 Restrições

1) Balanço de Massa:

O volume no início do primeiro evento é igual ao volume existente nos tanques no início do processo.

$$VolAtual_{tq,1} = Volini_{tq} \qquad \forall\, tq = 1,...,TQ \tag{43}$$

O volume no início de um evento (e + 1) é calculado da seguinte forma: volume no início do evento anterior e, mais o volume recebido no evento e, menos o volume enviado no evento e.

$$VolAtual_{tq,e+1} = VolAtual_{tq,e} + \sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Vol\,Rec_{tq,p,d,e} - \sum_{c \in CC} \sum_{d \in (CTQD_{tq} \cap CCD_c)} VolEnv_{tq,c,d,e} \quad \forall\, tq = 1,...,TQ;\ e = 1,...,E-1 \tag{44}$$

Uma restrição adicional é inserida para o cálculo do volume final.

$$VolFinal_{tq} = VolAtual_{tq,E} + \sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Vol\,Rec_{tq,p,d,E} - \sum_{c \in CC} \sum_{d \in (CTQD_{tq} \cap CCD_c)} VolEnv_{tq,c,d,E} \quad \forall\, tq = 1,...,TQ \tag{45}$$

O volume dos tanques deve estar sempre entre os volumes mínimo e máximo permitidos.

$$Vol\,min_{tq} \le VolAtual_{tq,e} \le Vol\,max_{tq} \qquad \forall\, tq = 1,...,TQ;\ e = 1,...,E \tag{46}$$

$$Vol\,min_{tq} \le VolFinal_{tq} \le Vol\,max_{tq} \qquad \forall\, tq = 1,...,TQ \tag{47}$$

2) Restrições de demanda e lotes de envio:

Cada cliente deve receber toda a demanda de produto requerida.

$$\sum_{tq \in CDTQ_d} \sum_{e \in CE} VolEnv_{tq,c,d,e} = Dem_{c,d} \qquad \forall\, c = 1,...,C;\ d \in CCD_c \tag{48}$$

A demanda de cada cliente pelos tipos de diesel no primeiro evento é igual à demanda inicial.

$$DemAtual_{c,d,e} = Dem_{c,d} \qquad \forall c = 1,...,C;\ d \in 1,...,D;\ e = 1 \tag{49}$$

A demanda do cliente c pelo produto d no evento (e + 1) é dada pela demanda no evento e subtraída da quantidade de diesel enviada no evento e.

$$DemAtual_{c,d,e+1} = DemAtual_{c,d,e} - \sum_{tq \in CDTQ_d} VolEnv_{tq,c,d,e} \qquad \forall c = 1,...,C;\ d = 1,...,D;\ e = 1,...,E-1 \tag{50}$$

A demanda do cliente *c* pelo produto *d* no evento *E* é dada pela demanda no evento (*E* - 1) subtraída da quantidade de diesel enviada no evento *E*.

$$DemFinal_{c,d} = DemAtual_{c,d,e} - \sum_{tq \in CDTQ_d} VolEnv_{tq,c,d,e} \qquad \forall c = 1,...,C; d = 1,...,D; e = E \tag{51}$$

Os lotes dos processos devem ser totalmente enviados aos tanques.

$$\sum_{tq \in CDTQ_d} \sum_{e \in CE} Vol\,Rec_{tq,p,d,e} = Lote\,Proc_{p,d} \qquad \forall\ p = 1,...,P; d \in CPD_p \tag{52}$$

3) Restrição de ativação da variável $DemVol_{tq,c,d,e}$.

$$DemAtual_{c,d,e} >= VolAtual_{tq,e} - Vol\,min_{tq} - \left(\sum_{c \in CC} \sum_{d \in CD} Dem_{c,d} \right).\left(1 - DemVol_{tq,c,d,e}\right) \tag{53}$$
$$\forall\ tq \in CDTQ_d; c = 1,...,C; d = 1,...,D; e = 1,...,E$$

$$DemAtual_{c,d,e} <= VolAtual_{tq,e} - Vol\,min_{tq} + \left(\left(\sum_{c \in CC} \sum_{d \in CD} Dem_{c,d} \right) + EPS \right).\left(DemVol_{tq,c,d,e}\right) - EPS \tag{54}$$
$$\forall\ tq \in CDTQ_d; c = 1,...,C; d = 1,...,D; e = 1,...,E$$

4) Restrições de Operação:

Cada processo pode enviar produto para somente um tanque em cada evento.

$$\sum_{tq \in CTQ} \sum_{d \in \left(CTQD_{tq} \cap CPD_p\right)} Rec_{tq,p,d,e} \le 1 \qquad \forall\ p = 1,...,P; e = 1,...E \tag{55}$$

Em cada evento *e* somente um tanque *tq* poderá enviar diesel para um único cliente *c*.

$$\sum_{tq \in CTQ} \sum_{c \in CC} \sum_{d \in \left(CTQD_{tq} \cap CCD_c\right)} Env_{tq,c,d,e} \le 1 \qquad \forall\ e = 1,...,E \tag{56}$$

Cada cliente *c* pode receber um determinado tipo de diesel *d* de um único tanque no evento *e*.

$$\sum_{tq \in CTQ} \sum_{d \in \left(CTQD_{tq} \cap CCD_c\right)} Env_{tq,c,d,e} \le 1 \qquad \forall\ c = 1,...,C; e = 1,...,E \tag{57}$$

Cada tanque pode estar ocioso, descarregando ou carregando diesel, ou seja, as operações de carregamento e de descarregamento não podem ser efetuadas simultaneamente.

$$\sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Rec_{tq,p,d,e} + \sum_{c \in CC} \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e} \leq 1 \qquad \forall\, tq = 1,...,TQ;\ e = 1,...,E \tag{58}$$

Quando um tanque recebe produto de um processo, deverá fazê-lo até o volume máximo permitido.

$$VolAtual_{tq,e+1} \geq \sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} \left(Rec_{tq,p,d,e}.VolMax_{tq} \right) \qquad \forall\, tq = 1,...,TQ;\ e = 1,...,E-1 \tag{59}$$

Se $DemVol_{tq,c,d,e} = 1$ e $Env_{tq,c,d,e} = 1$ então deve-se garantir o esvaziamento do tanque até o seu volume mínimo. Se uma ou ambas destas variáveis assumir valor "0", então a restrição é relaxada.

$$\begin{gathered} VolAtual_{tq,e+1} \leq Vol\,min_{tq} + (Vol\,max_{tq}).(2 - Env_{tq,c,d,e} - DemVol_{tq,c,d,e}) \\ \forall\, tq \in CDTQ_d;\ c = 1,...,C;\ d = 1,...,D;\ e = 1,...,E-1 \end{gathered} \tag{60}$$

5) Restrições de ativação das variáveis $Rec_{tq,p,d,e}$ e $Env_{tq,c,d,e}$.

A variável $Rec_{tq,p,d,e}$ é ativada se houver volume recebido em determinado evento. Da mesma forma a variável $Env_{tq,c,d,e}$ é ativada se houver envio a cliente. Quando qualquer destas variáveis é ativada, seu valor será igual a "1". Caso contrário o valor será "0".

$$\begin{gathered} Vol\,Rec_{tq,p,d,e} \leq (Vol\,max_{tq} - Vol\,min_{tq}).Rec_{tq,p,d,e} \\ \forall\, tq = 1,...,TQ;\ p = 1,...,P;\ d \in (CTQD_{tq} \cap CPD_p);\ e = 1,...,E \end{gathered} \tag{61}$$

$$\begin{gathered} Vol\,Rec_{tq,p,d,e} \geq (-Vol\,max_{tq} - EPS).(1 - Rec_{tq,p,d,e}) + EPS \\ \forall\, tq = 1,...,TQ;\ p = 1,...,P;\ d \in (CTQD_{tq} \cap CPD_p);\ e = 1,...,E \end{gathered} \tag{62}$$

$$\begin{gathered} VolEnv_{tq,c,d,e} \leq (Vol\,max_{tq} - Vol\,min_{tq}).Env_{tq,c,d,e} \\ \forall\, tq = 1,...,TQ;\ c = 1,...,C;\ d \in (CTQD_{tq} \cap CCD_c);\ e = 1,...,E \end{gathered} \tag{63}$$

$$\begin{gathered} VolEnv_{tq,c,d,e} \geq (-Vol\,max_{tq} - EPS).(1 - Env_{tq,c,d,e}) + EPS \\ \forall\, tq = 1,...,TQ;\ c = 1,...,C;\ d \in (CTQD_{tq} \cap CCD_c);\ e = 1,...,E \end{gathered} \tag{64}$$

6) Restrições envolvendo variáveis de temporização:

- Variáveis de temporização dos processos:

Como os processos são contínuos, isto é, uma vez iniciado o envio do lote, este deverá ser enviado do começo ao fim, sem interrupções, a seguinte restrição deve ser adicionada.

$$XI\,Processo_{p,e+1} = XF\,Processo_{p,e} \qquad \forall\ p = 1,...,P;\ e = 1,...,E-1 \tag{65}$$

Para o cálculo da variável $XFProcesso_{p,e}$ se faz a adição da variável $XIProcesso_{p,e}$ ao quociente do $VolRec_{tq,p,d,e}$ pela vazão QR_p. Este quociente calcula o tempo despendido no recebimento.

$$XF\,Processo_{p,e} = XI\,Processo_{p,e} + \sum_{tq \in CTQ} \sum_{d \in (CTQD_{tq} \cap CPD_p)} \left(Vol\,Rec_{tq,p,d,e}\right) / QR_p \ \forall\ p = 1,..,P;\ e = 1,...,E \tag{66}$$

- Variáveis de temporização dos clientes:

Cada cliente deve receber todo o lote de seu pedido, sem interrupções. Portanto, o início do envio a cada cliente no evento (e + 1) deve ser igual ao término do envio no evento e.

$$XICliente_{c,e+1} = XFCliente_{c,e} \qquad \forall\ c = 1,...,C;\ e = 1,...,E-1 \tag{67}$$

O cálculo da variável $XFCliente_{c,e}$ é feito adicionando-se à variável $XICliente_{c,e}$, o quociente do $VolEnv_{,tq,c,d,e}$ pela vazão QE_c. Este quociente calcula o tempo despendido para o envio.

$$XFCliente_{c,e} = XICliente_{c,e} + \sum_{tq \in CTQ} \sum_{d \in (CTQD_{tq} \cap CCD_c)} \left(VolEnv_{tq,c,d,e}\right) / QE_c \quad \forall\ c = 1,..,C;\ e = 1,...,E \tag{68}$$

- Variáveis de temporização dos tanques:

Os tanques podem ter intervalos de tempo em que estão ociosos. Portanto, a restrição a seguir, aplicada às variáveis $XITanque_{tq,e}$ e $XFtanque_{tq,e}$, servem para ordenar os eventos no tempo. Quando a operação no evento e é de recebimento, deverá ser respeitado o intervalo de certificação do diesel (TC).

$$XITanque_{tq,e+1} \geq XFTanque_{tq,e} + TC. \sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Rec_{tq,p,d,e} \qquad \forall\ tq = 1,...,TQ;\ e = 1,...,E-1 \tag{69}$$

As duas restrições a seguir são acrescentadas para que, quando não houver operação em um evento e com um determinado tanque tq, as variáveis $XItanque_{tq,e}$ e $XFtanque_{tq,e}$

assumam o mesmo valor. Ainda, estas restrições garantem que, se no evento *e* ocorre uma operação, as restrições são relaxadas. O objetivo da inserção destas duas restrições ao modelo foi o de facilitar a interpretação do resultado.

$$XITanque_{tq,e} \leq XFTanque_{tq,e} + H.\left(\sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Rec_{tq,p,d,e} + \sum_{c \in CC} \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e} \right) \quad (70)$$
$$\forall\ tq = 1,...,TQ;\ e = 1,...,E-1$$

$$XITanque_{tq,e} \geq XFTanque_{tq,e} - H.\left(\sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CPD_p)} Rec_{tq,p,d,e} + \sum_{p \in CP} \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e} \right) \quad (71)$$
$$\forall\ tq = 1,...,TQ;\ e = 1,...,E-1$$

- Vinculação das variáveis $XITanque_{tq,e}$ com $XIProcesso_{p,e}$ e $XFTanque_{tq,e}$ com $XFProcesso_{p,e}$

As restrições (72) e (73) vinculam as variáveis $XITanque_{tq,e}$ com $XIProcesso_{p,e}$ e as restrições (74) e (75) vinculam $XFTanque_{tq,e}$ com $XFProcesso_{p,e}$. Estas, garantem que, se houver operação de recebimento pelo tanque *tq* do processo *p*, as variáveis para o início e término de operação para o tanque e o processo envolvidos ficam atreladas ao evento *e* considerado. Se não houver vínculo entre estas variáveis, as restrições são relaxadas.

$$XITanque_{tq,e} \leq XI\,Processo_{p,e} + H.\left(1 - \sum_{d \in (CTQD_{tq} \cap COD_p)} Rec_{tq,p,d,e} \right) \quad (72)$$
$$\forall\ tq = 1,...TQ;\ p = 1,...,P;\ e = 1,...,E$$

$$XITanque_{tq,e} \geq XI\,Processo_{p,e} - H.\left(1 - \sum_{d \in (CTQD_{tq} \cap COD_p)} Rec_{tq,p,d,e} \right) \quad (73)$$
$$\forall\ tq = 1,...TQ;\ p = 1,...,P;\ e = 1,...,E$$

$$XFTanque_{tq,e} \leq XF\,Processo_{p,e} + H.\left(1 - \sum_{d \in (CTQD_{tq} \cap COD_p)} Rec_{tq,p,d,e} \right) \quad (74)$$
$$\forall\ tq = 1,...TQ;\ p = 1,...,P;\ e = 1,...,E$$

$$XFTanque_{tq,e} \geq XF\,Processo_{p,e} - H.\left(1 - \sum_{d \in (CTQD_{tq} \cap COD_p)} Rec_{tq,p,d,e} \right) \quad (75)$$
$$\forall\ tq = 1,...TQ;\ p = 1,...,P;\ e = 1,...,E$$

- Vinculação das variáveis $XITanque_{tq,e}$ com $XICliente_{c,e}$ e $XFTanque_{tq,e}$ com $XFCliente_{c,e}$

As restrições (76) e (77) vinculam as variáveis $XITanque_{tq,e}$ com $XICliente_{c,e}$ e as restrições (78) e (79) vinculam $XFTanque_{tq,e}$ com $XFCliente_{c,e}$. Estas, garantem que se houver operação de envio do tanque *tq* ao cliente *c*, as variáveis para o início e término de operação para o tanque e o cliente envolvidos ficam atreladas ao evento *e* considerado. Se não houver vínculo entre estas variáveis, as restrições são relaxadas.

$$XITanque_{tq,e} \leq XICliente_{c,e} + H.\left(1 - \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e}\right) \quad \forall\, tq = 1,...TQ;\ c = 1,...,C;\ e = 1,...,E \quad (76)$$

$$XITanque_{tq,e} \geq XICliente_{c,e} - H.\left(1 - \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e}\right) \quad \forall\, tq = 1,...TQ;\ c = 1,...,C;\ e = 1,...,E \quad (77)$$

$$XFTanque_{tq,e} \leq XFCliente_{c,e} + H.\left(1 - \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e}\right) \forall\, tq = 1,...TQ;\ c = 1,...,C;\ e = 1,...,E \quad (78)$$

$$XFTanque_{tq,e} \geq XFCliente_{c,e} - H.\left(1 - \sum_{d \in (CTQD_{tq} \cap CCD_c)} Env_{tq,c,d,e}\right) \forall\, tq = 1,...TQ;\ c = 1,...,C;\ e = 1,...,E \quad (79)$$

7) Restrições adicionais

Para poder encontrar o maior valor para o tempo final de envio entre todos os clientes acrescenta-se a seguinte restrição.

$$MaxXFCliente \geq XFCliente_{c,E} \qquad \forall\, c = 1,...,C \quad (80)$$

A variável *H* é utilizada como limitante superior para o tempo.

$$H = \sum_{p \in CP}\left(\sum_{d \in CD} Lote\,Proc_{p,d} / QR_p\right) + \sum_{c \in CC}\left(\sum_{d \in CD} Dem_{c,d} / QE_c\right) \quad (81)$$

CAPÍTULO 4

COMPUTAÇÃO EVOLUCIONÁRIA – PLIM: MODELAGEM E METODOLOGIA

4.1 INTRODUÇÃO

Os aplicativos computacionais comerciais voltados à solução de problemas de PL, PLI e PLIM, em geral, resolvem os problemas de PL em tempo computacional aceitável. Mas, tanto para problemas de PLI como de PLIM, o aumento do número de variáveis inteiras, geralmente resulta em um aumento no tempo computacional para a resolução destes problemas, pois estes aplicativos utilizam a técnica de *branch and bound*. Muitas vezes este tempo computacional torna inviável o uso do aplicativo, pois na maior parte das aplicações práticas em gestão de processos, o resultado do problema reflete em decisões que devem ser tomadas de forma imediata.

Por estas razões, foram desenvolvidas neste trabalho metodologias que aplicam técnicas de Computação Evolucionária ao problema de PLIM. O modelo utilizado para o problema é gerado a partir do modelo de PLIM com discretização uniforme do tempo (Seção 3.2). Para a resolução, este modelo em PLIM é modificado para ser resolvido por PL. Esta modificação inclui a retirada de algumas restrições e variáveis binárias, e relaxação linear das variáveis binárias e inteiras restantes. Algumas das variáveis retiradas do modelo são tratadas por técnicas de Computação Evolucionária e outras através da aplicação de um procedimento. Os valores encontrados são inseridos como dados no modelo em PL, que é então resolvido pelo aplicativo LINGO 8.0. O valor da função objetivo resultante é utilizado como valor da função de aptidão para o algoritmo da CE. Depois de terminado o processo da técnica de CE, o resultado encontrado é inserido como dados no modelo em PLIM. Este modelo é então resolvido pelo LINGO 8.0 usando a técnica de *branch and bound*. A Figura 18 mostra um fluxograma simplificado desta modelagem.

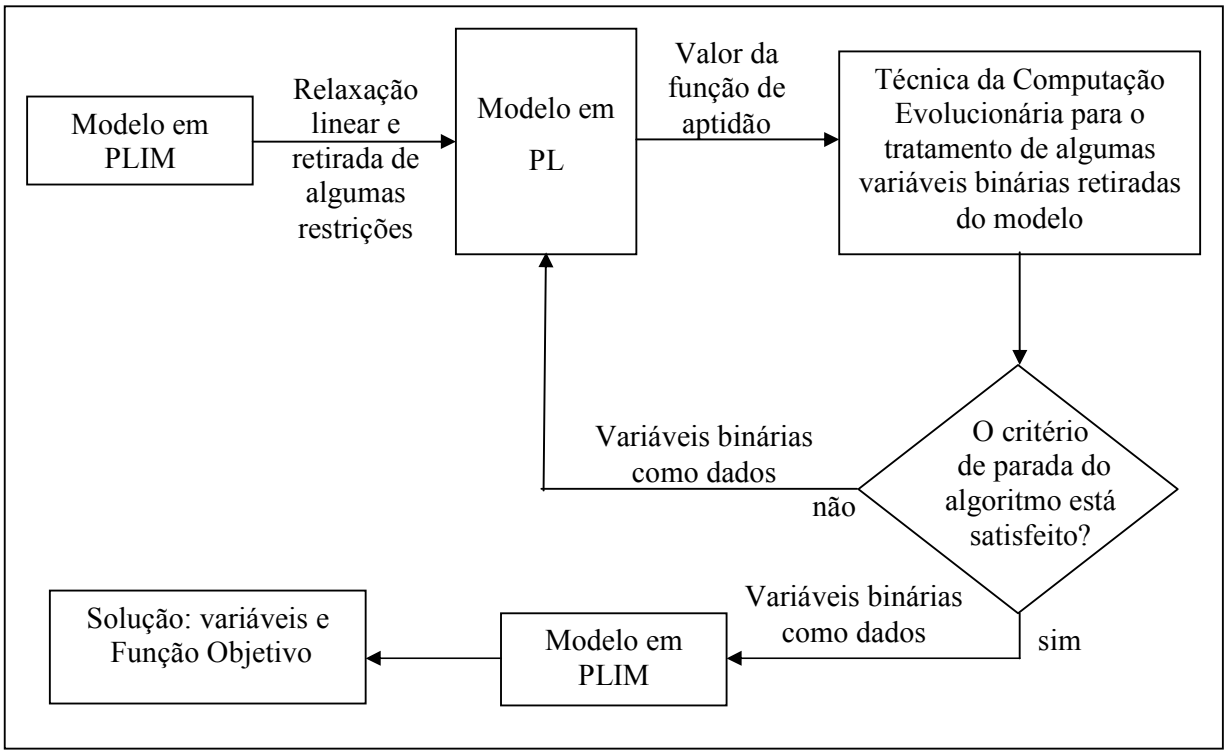

Figura 18: Fluxograma de PLIM – Computação Evolucionária

As técnicas de Computação Evolucionária utilizadas neste trabalho são: AG de Estado Estacionário Híbrido (AGEEH) e Algoritmo Transgenético Proto Gene (ProtoG). A seguir será feito o detalhamento de como estas técnicas foram utilizadas para o problema descrito na Seção 3.1.

4.2 AG DE ESTADO ESTACIONÁRIO HÍBRIDO (AGEEH) – PLIM

A modelagem utilizando AGEEH e o modelo PLIM com discretização uniforme de tempo para a resolução do problema descrito na Seção 3.1 foi feita com base no fluxograma da Figura 18. Para melhor compreensão, as etapas para a resolução desta modelagem estão separadas em oito módulos descritos a seguir.

4.2.1 Módulo 1: Construtor de Indivíduos

Este módulo gera aleatoriamente vetores compostos pelos dígitos binários "0" e "1" para formar os indivíduos da população. Estes dígitos representam os valores das variáveis $Rec_{tq,t}$, $tq = 1,...,TQ$, $t = 1,...,T$. Um procedimento foi desenvolvido para gerar estes cromossomos de forma a satisfazer a restrição (20) que impõe que haverá sempre um, e somente um tanque recebendo da produção em cada intervalo de tempo. Logo, para a construção de um indivíduo deve-se ter em cada intervalo de tempo $t \in \{1,...,T\}$, um e somente um dos tanques $tq \in \{1,...,TQ\}$ com valor da variável binária igual a "1". Os passos a seguir foram utilizados na programação deste módulo.

Passo1:

Fazer $t = 1$

Passo 2:

Gerar um número aleatório $tq \in \{1,...,TQ\}$ usando a fórmula (82) para determinar o número do tanque que estará recebendo no intervalo t.

$$tq = Int(TQ.Rnd) + 1 \tag{82}$$

onde *Rnd* é um número aleatório no intervalo [0, 1] e *Int(.)* é a função que retorna o valor inteiro de um número.

Passo 3:

Construir a cadeia constituída por TQ variáveis binárias na qual o tanque de número tq terá valor "1" enquanto que todos os outros terão valor "0";

Passo 4:

Acrescentar a nova cadeia ao cromossomo;

Passo 5:

Fazer $t = t + 1$. Se $t \leq T$, voltar ao Passo 2. Caso contrário, o processo é encerrado e o cromossomo está completo.

A Figura 19 ilustra o exemplo de um cromossomo gerado pelo Módulo 1 com $T = 5$ e $TQ = 4$.

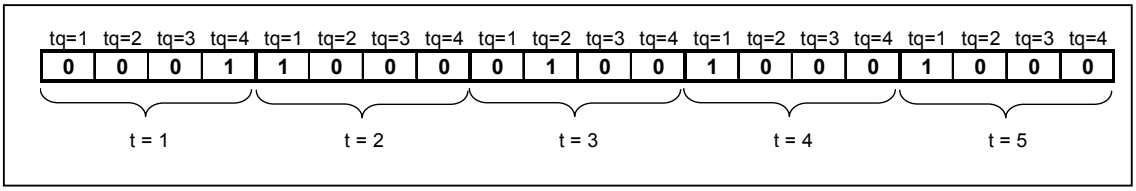

Figura 19: Exemplo de cromossomo para o AGEEH

4.2.2 Módulo 2: Procedimento para encontrar as variáveis binárias $TR_{tq,tq',t}$

Como estas variáveis dependem exclusivamente das variáveis $Rec_{tq,t}$, um procedimento foi implementado para encontrá-las, satisfazendo as restrições (31), (32) e (33).

O procedimento verifica, para todas as combinações duas a duas dos tanques, se as variáveis $Rec_{tq,t}$ no intervalo de tempo t e $(t - 1)$ são iguais a "1". Em caso afirmativo, a variável $TR_{tq,tq',t}$ assumirá valor "1" no intervalo t. Para qualquer outro caso, a variável assume valor "0". Esta verificação é feita para os intervalos de tempo $t \in \{2,...,T\}$.

4.2.3 Módulo 3: Avaliador dos indivíduos

Este módulo é composto pelas etapas: extração dos dados do cromossomo (variáveis $Rec_{tq,t}$, $tq = 1,..., TQ$, $t = 1,...,T$), aplicação do Módulo 2 para gerar as variáveis $TR_{tq,tq',t}$, inserção dos dados obtidos no modelo do LINGO 8.0 em PL, resolução deste modelo e obtenção do valor da função objetivo para ser utilizado como valor da função aptidão. O fluxograma da Figura 20 ilustra os passos deste módulo.

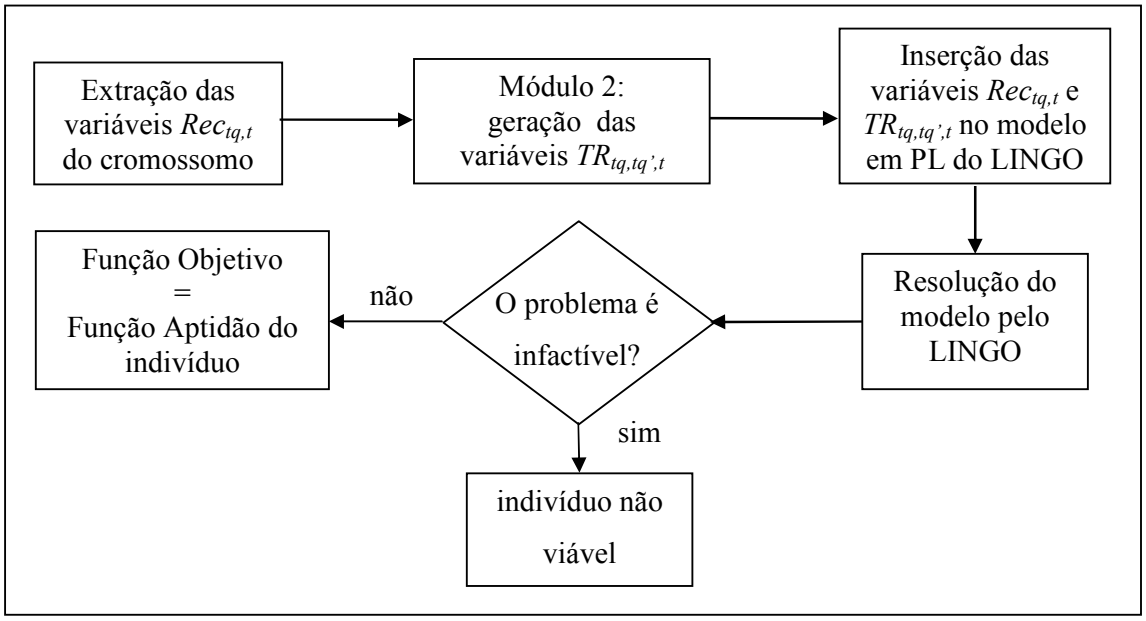

Figura 20: Fluxograma do Módulo 3: AGEEH - PLIM

O modelo em PL utilizado neste módulo é obtido a partir do modelo descrito na Seção 3.2. retirando-se as restrições (20), (31), (32) e (33). São retiradas também, da modelagem do LINGO 8.0 as restrições para variáveis binárias. Para resolver este modelo, as variáveis $Rec_{tq,t}$ e $TR_{tq,tq',t}$ são inseridas neste como dados do problema e o aplicativo LINGO 8.0 é processado usando apenas PL (método Simplex). Se o problema for factível, o resultado da função objetivo será utilizado para valor da função de aptidão do AGEEH.

4.2.4 Módulo 4: Ordenador da população

Ordena os indivíduos, no caso de minimização, em ordem crescente de acordo com o valor da função de aptidão. Os passos descritos a seguir são utilizados tanto na construção da população como após as operações de cruzamento, mutação e busca local quando novos indivíduos são encontrados.

Seja N o número de indivíduos na população. Esta variável terá valor igual ao tamanho da população após a construção desta. Então:

Passo 1:

Se $N > 1$, executar os passos 2 e 3.

Passo 2:

O novo indivíduo é inserido na *N*-ésima posição no lugar do indivíduo com pior valor de aptidão.

Passo 3:

Se este indivíduo possui função de aptidão pior que o indivíduo de posição N - 1, então ficará na *N*-ésima posição e está encerrado o processo de ordenação. Caso contrário, é feita a troca de posição entre eles e o indivíduo em questão passará a ocupar a posição de ordem N - 1. A seguir é feita a comparação com o indivíduo de ordem $N - 2$. Se o indivíduo inserido é pior que este indivíduo, fica na posição que está e o processo está encerrado. Caso contrário, é feita a troca com o indivíduo que está na posição $N - 2$. Procede-se desta forma até que o valor de aptidão do indivíduo na posição imediatamente acima deste seja menor que seu valor de aptidão. Se o valor de aptidão deste indivíduo é menor que o valor de todos os outros indivíduos da população, este ocupará ao final do processo, a primeira posição.

4.2.5 Módulo 5: Construtor da população inicial com *N* indivíduos

Para formar este módulo, foram utilizados os Módulos 1, 3 e 4. O fluxograma da Figura 21 ilustra o funcionamento deste módulo.

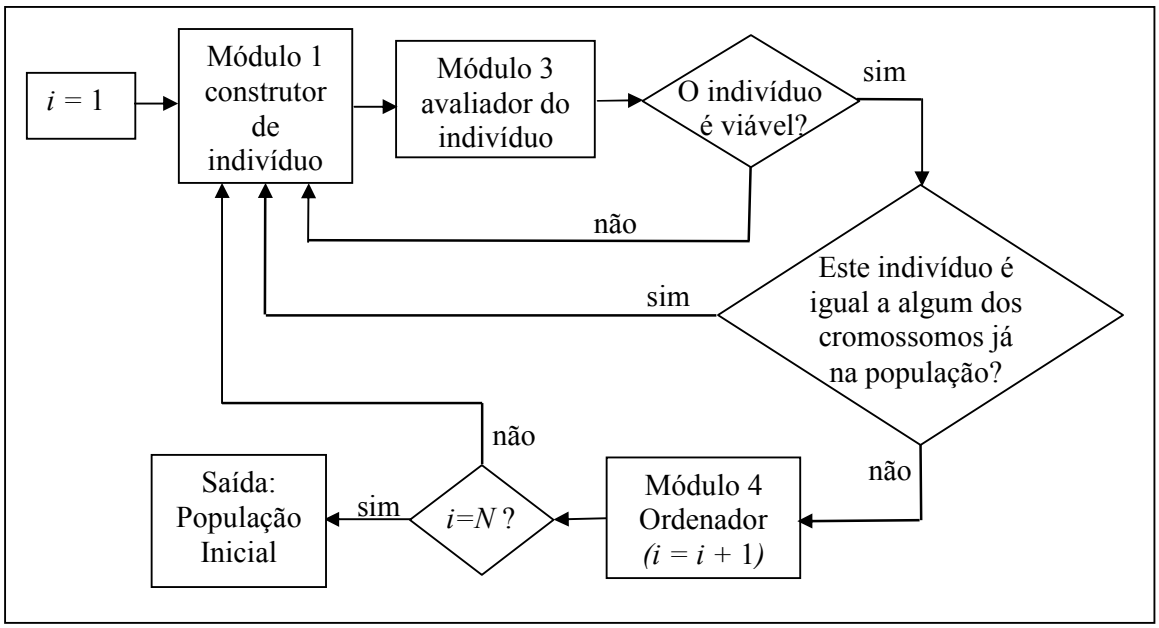

Figura 21: Fluxograma do Módulo 5: AGEEH – PLIM

Deve-se observar que, em alguns casos, o cromossomo composto com as variáveis $Rec_{tq,t}$ pode gerar um problema infactível, mesmo obedecendo à restrição (20). Neste caso, este cromossomo é descartado e outro deve ser gerado. Como o Módulo 1 gera cromossomos aleatoriamente, é possível um mesmo indivíduo ser gerado mais de uma vez. Para que a população não tenha indivíduos iguais, cada novo cromossomo gerado é comparado a todos os já inseridos na população.

4.2.6 Módulo 6: Busca local

O AG implementado mostrou a necessidade da inclusão de uma busca local mais agressiva, após um certo número de iterações, com o objetivo de sair de um mínimo local. O número de iterações *ItHibr* para iniciar a busca local foi considerado como um parâmetro variável.

O processo da busca local nesta metodologia é feito através dos seguintes passos:

Passo 1:

Escolha um indivíduo da população usando a fórmula (8) proposta por Mayerle.

Passo 2:

Escolha aleatoriamente um intervalo de tempo $t \in \{1,...,T\}$ para marcar o ponto de início da busca, usando a fórmula (83).

$$t = Int(T.Rnd) + 1 \tag{83}$$

onde *Rnd* é um número aleatório no intervalo [0, 1] e *Int(.)* é a função que retorna o valor inteiro de um número. Iniciam-se as variáveis *i* com valor inicial "0", *j* com valor "1", *k* com valor "0" e *T'* com valor inicial *T*.

Passo 3:

Para o intervalo de tempo ($t + i$), coloca-se o valor "1" para o tanque *j* e "0" para todos os outros;

Passo 4:

Avalia-se este indivíduo.

Passo 5:

Se for melhor que o anterior, este substitui o indivíduo a ser manipulado;

Passo 6:

Faça $j = j + 1$. Se $j \leq TQ$ então volte ao Passo 3;

Passo 6:

Faça $i = i + 1$ e $j = 1$. Se $t + i \leq T'$ então volte ao Passo 3;

Passo 7:

Se $k = 0$, faça $i = 0$, $j = 1$, $T' = t - 1$, $t = 1$, $k = 1$ e volte ao Passo 3.

Passo 8:

O indivíduo encontrado e seu valor de aptidão são o resultado da busca local.

4.2.7 Módulo 7: AGEEH

Este módulo é constituído pelo AG do tipo estado estacionário com hibridização propriamente dito. Utiliza o resultado obtido pelo Módulo 5 para dados de entrada e os Módulos 3, 4 e 6 para a execução do algoritmo. O parâmetro *ItMax* representa o número máximo de iterações do algoritmo. Este módulo está ilustrado no fluxograma da Figura 22.

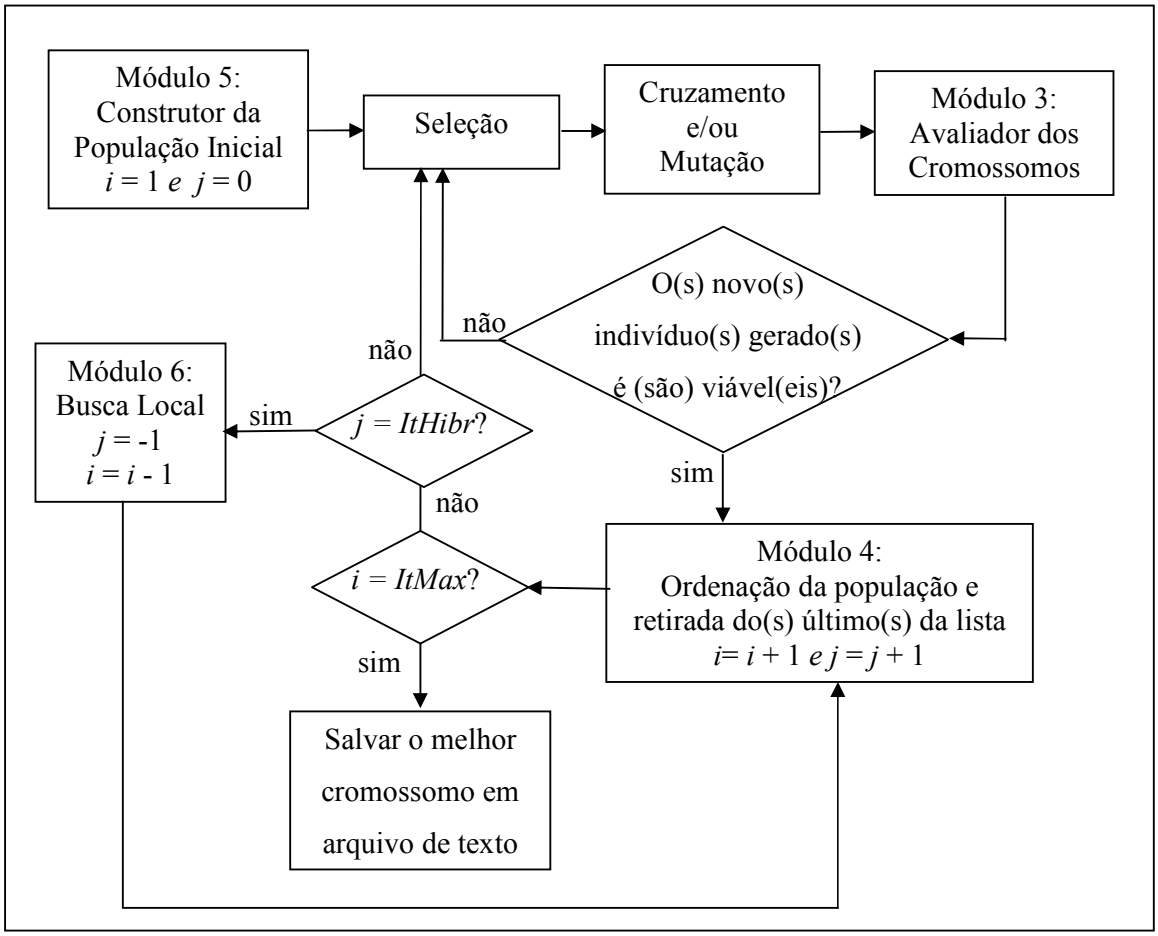

Figura 22: Fluxograma do Módulo 7: AGEEH – PLIM

Os operadores de seleção, cruzamento e mutação utilizados no AGEEH são:

- O operador de seleção utilizado foi proposto por Mayerle (Seção 2.4.7.5) através da fórmula (8).
- O cruzamento de dois pontos descrito na Seção 2.4.8.1 foi o utilizado neste modelo de AG. Este cruzamento foi especializado para a configuração dos cromossomos em questão, de forma a gerar novos indivíduos viáveis face à restrição (20). A Figura 23 exemplifica este operador genético para $T = 5$ *e* $TQ = 4$.

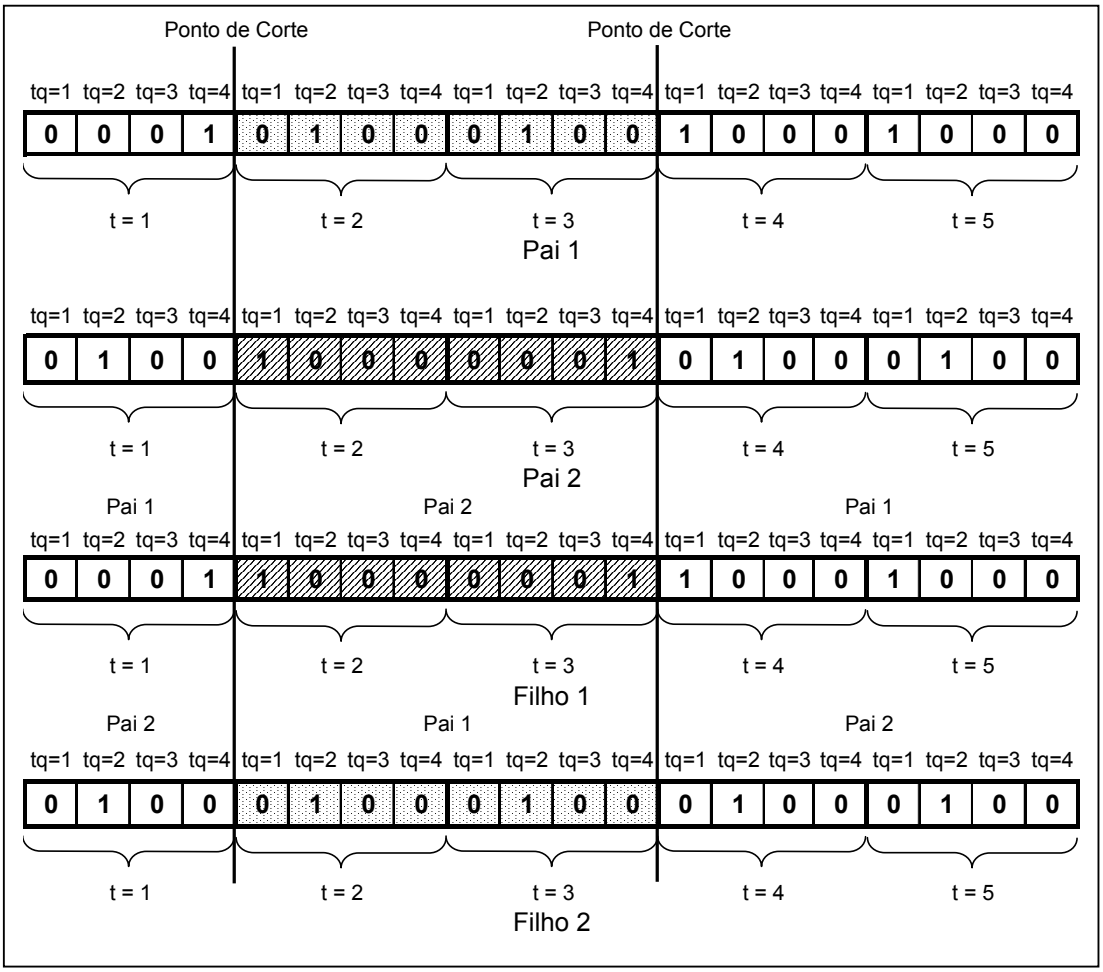

Figura 23: Exemplo de Cruzamento para a metodologia AGEEH – PLIM

- Um operador de mutação especializado foi gerado a partir da mutação simples, da seguinte forma: um intervalo de tempo é escolhido aleatoriamente para que sua cadeia binária seja modificada. Sorteia-se então um tanque, com a restrição de que o seu valor deve ser "0". Troca-se este valor para "1" e o tanque que estava com valor "1" passa a ser "0". A mutação aplicada desta forma garante a validade da restrição (20). A Figura 24 exemplifica este operador com $T = 5$ *e* $TQ = 4$.

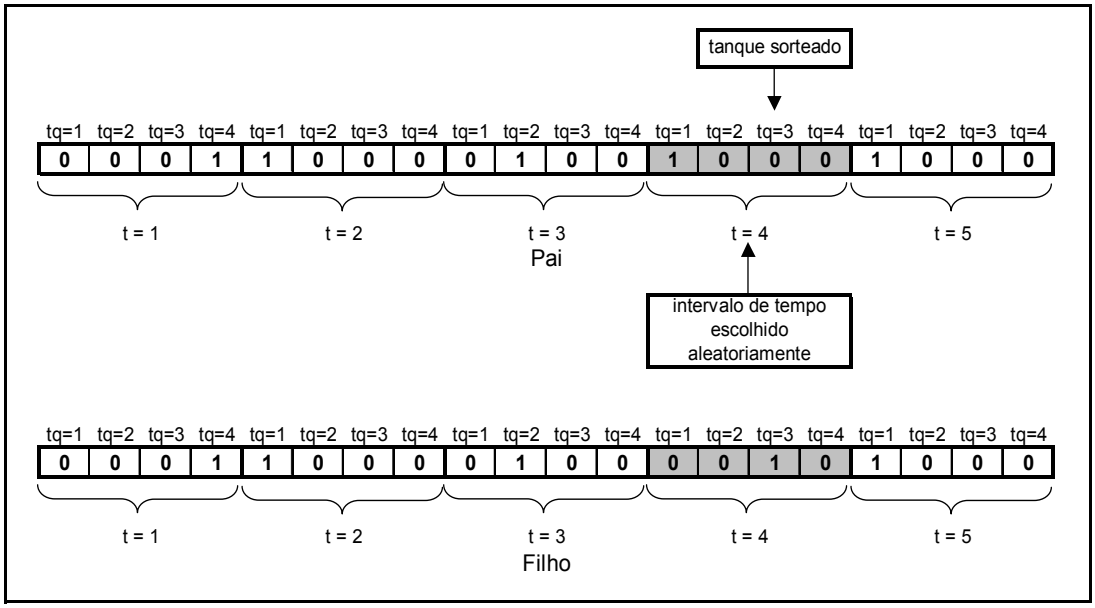

Figura 24: Exemplo de Mutação para metodologia AGEEH – PLIM

O algoritmo para o AGEEH aplicado neste modelo baseou-se no algoritmo AG de Estado Estacionário. Algumas modificações foram implementadas para inserir a Busca Local.

Foram aplicadas taxas adaptativas (Seção 2.4.9.5) para a probabilidade de recombinação (cruzamento) p_c e a probabilidade de mutação p_m, com o objetivo de se evitar uma convergência prematura do AG para um ótimo local. As fórmulas (84) e (85) calculam em cada iteração as taxas de recombinação e mutação. Estas fórmulas calculam as probabilidades de recombinação e mutação de forma que, indivíduos com valor de aptidão próximo do melhor resultado terão taxas menores do que os que estão afastados. Desta forma, indivíduos com piores valores de aptidão são induzidos a fazerem operações de cruzamento e mutação.

Sendo de minimização o problema abordado, seja *f'* o menor valor de aptidão entre os indivíduos escolhidos para pais, f_{min}, o menor valor de aptidão e f_{max} o maior valor de aptidão, respectivamente, entre todos os indivíduos da população. As fórmulas (84) e (85) calculam as probabilidades de cruzamento e mutação, respectivamente.

$$p_c = \left[(k_2 - k_1).f' + f_{max}.k_1 - f_{min}.k_2\right] / \left(f_{max} - f_{min}\right) \tag{84}$$

$$p_m = \left[(k_4 - k_3).f' + f_{max}.k_3 - f_{min}.k_4\right] / \left(f_{max} - f_{min}\right) \tag{85}$$

Os parâmetros k_1, k_2, k_3 e k_4 são números reais no intervalo $[0,1]$ com $k_1 < k_2$ e $k_3 < k_4$.

Algoritmo AGEEH - PLIM

Inicialize a população *P* com *N* cromossomos

Avalie indivíduos na população *P*

Ordene a população *P* em ordem crescente pelo valor de aptidão

Repita

Se a aplicação do operador de recombinação com probabilidade p_c é verdadeira **então**

Selecione dois indivíduos em *P*

Aplique o operador de recombinação

Avalie os indivíduos gerados

Insira estes indivíduos em *P* de acordo com sua aptidão

Se a aplicação do operador de mutação com probabilidade p_m é verdadeira **então**

Selecione um indivíduo em *P*

Aplique o operador de mutação

Avalie o indivíduo gerado

Insira este indivíduo em *P* de acordo com sua aptidão

Se o número de iterações para a hibridização foi alcançado **então**

Faça busca local

Até máximo de gerações

Fim

4.2.8 Módulo 8: Obtenção do resultado final

O melhor indivíduo obtido ao final da execução do Módulo 7 é utilizado para gerar as variáveis $Rec_{tq,t}$. Através do Módulo 2, obtém-se as variáveis $TR_{tq,tq',t}$. Estas variáveis são inseridas no modelo PLIM sem as restrições (20), (31), (32) e (33) e o problema é resolvido pelo LINGO 8.0. A Figura 25 ilustra o fluxograma deste módulo.

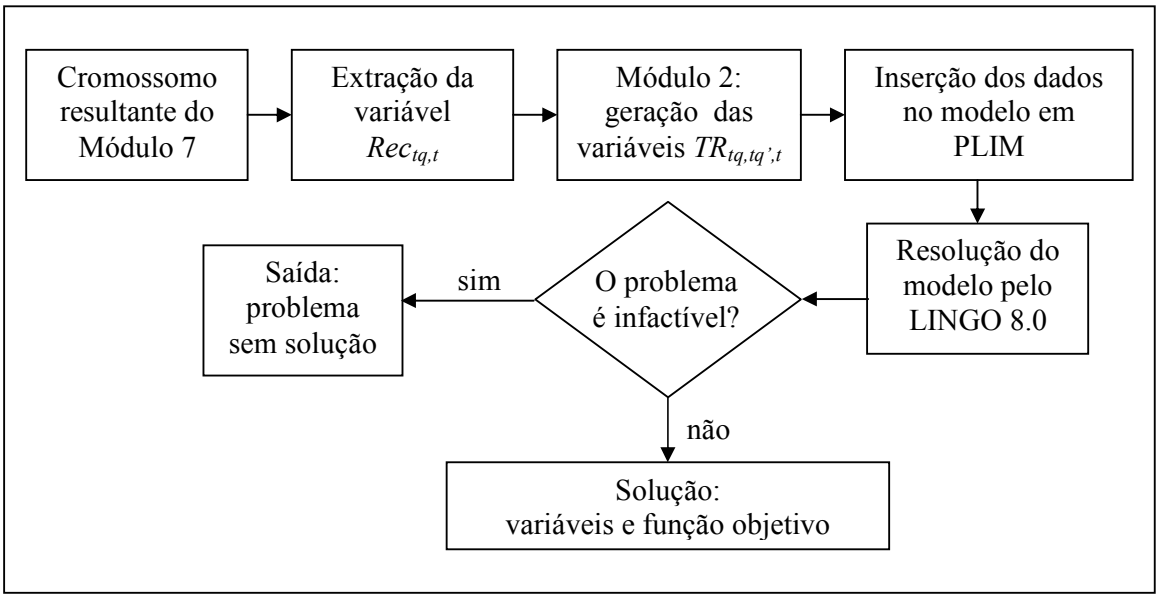

Figura 25: Fluxograma do Módulo 8: AGEEH – PLIM

4.3 ALGORITMO ProtoG – PLIM

A modelagem para o problema descrito na Seção 3.1, utilizando o algoritmo transgenético ProtoG e PLIM com discretização uniforme do tempo, é semelhante à descrita para AGEEH-PLIM na Seção 4.2. Os Módulos 1, 2, 3, 4, 5 e 8 são os mesmos. Serão descritos a seguir os Módulos 6 e 7 para esta aplicação.

4.3.1 Módulo 6: Construtor das partículas genéticas móveis

Para formar as partículas genéticas móveis $\lambda = (I, \Phi)$, deve-se definir a cadeia de informação I e o método de manipulação Φ, com $\Phi = \{p_1, p_2\}$ (Seção 2.5.1.2). O procedimento p_1 estabelece quando um cromossomo é suscetível à manipulação ou ataque pelas partículas genéticas móveis. Na aplicação deste trabalho, um cromossomo é suscetível à manipulação se a operação promover melhoria no valor de aptidão. Já o procedimento p_2, em caso de ataque, define como a informação I, que é transportada pela partícula genética móvel, é transferida para o cromossomo.

Como citado na Regra Tipo 1 (Seção 2.5.1.3), a constituição da cadeia de informação *I* que será carregada na partícula genética móvel pode fazer uso de conhecimento sobre o problema. Neste caso, sabe-se que o número de trocas entre tanques é reduzido se cada tanque que recebe da produção o faz por vários intervalos de tempo consecutivos. Portanto, a cadeia *I* para este problema será formada por um certo número de intervalos consecutivos com o mesmo tanque recebendo da produção. O número de intervalos consecutivos é considerado como parâmetro variável do modelo. A Figura 26 ilustra uma cadeia *I* com número de intervalos $T = 3$ e $TQ = 4$.

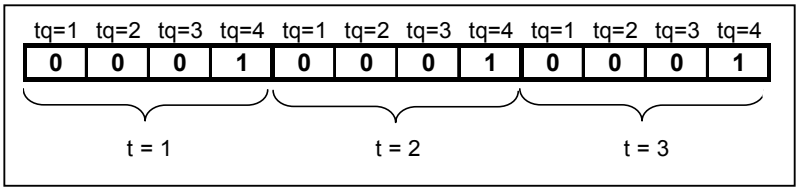

Figura 26: Exemplo de cadeia *I* para o algoritmo ProtoG

A Regra Tipo 2 (Seção 2.5.1.3) define como a cadeia de informação é transcrita em um cromossomo. Para esta aplicação, o operador de transcrição age de acordo com os passos descritos a seguir:

Passo 1:

Seja P o número de intervalos da cadeia. Gera-se aleatoriamente um número $t \in \{1,...,T-P\}$ aplicando-se a fórmula (86) para determinar o ponto de corte.

$$t = Int\left[(T-P).Rnd\right]+1 \tag{86}$$

onde *Rnd* é um número aleatório no intervalo $[0,1]$ e *Int(.)* é a função que retorna a parte inteira do número.

Passo 2:

Retira-se do cromossomo, a partir do corte, uma cadeia de mesmo comprimento da cadeia *I*.

Passo 3:

Insere-se a cadeia *I* na mesma posição da cadeia retirada.

A Figura 27 ilustra um exemplo desta operação sobre um cromossomo com 6 intervalos de tempo, uma cadeia *I* com 3 intervalos de tempo e $TQ = 4$.

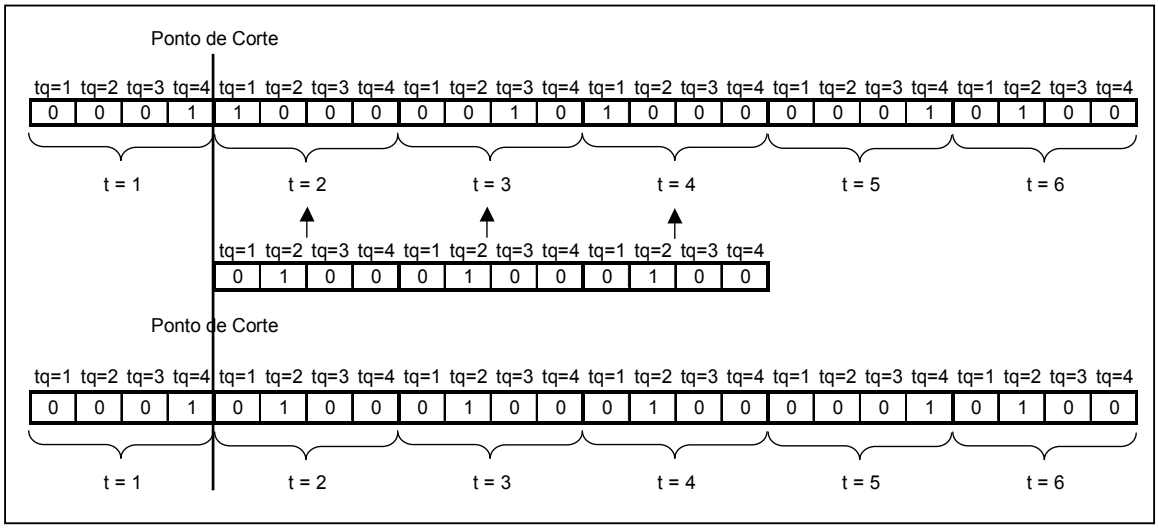

Figura 27: Exemplo de operação de transcrição do algoritmo ProtoG

4.3.2 Módulo 7: Algoritmo ProtoG

Este módulo é constituído pelo Algoritmo Transgenético ProtoG propriamente dito. Utiliza o resultado obtido pelo Módulo 5 para dados de entrada e os Módulos 3, 4 na execução do algoritmo. Fazendo uso da Regra Tipo 3 (Seção 2.5.1.3), define-se o tamanho da sub-população a ser extraída da população. O operador de seleção utilizado para a escolha dos indivíduos da sub-população foi o proposto por Mayerle (Seção 2.4.7.5) através da fórmula (8). A partícula genética móvel $\lambda = (I, \Phi)$ que fará o ataque a um determinado indivíduo da população é escolhida aleatoriamente entre todas as construídas pelo Módulo 6. O parâmetro *ItMax* representa o número máximo de iterações do algoritmo. O fluxograma da Figura 28 ilustra os passos deste módulo.

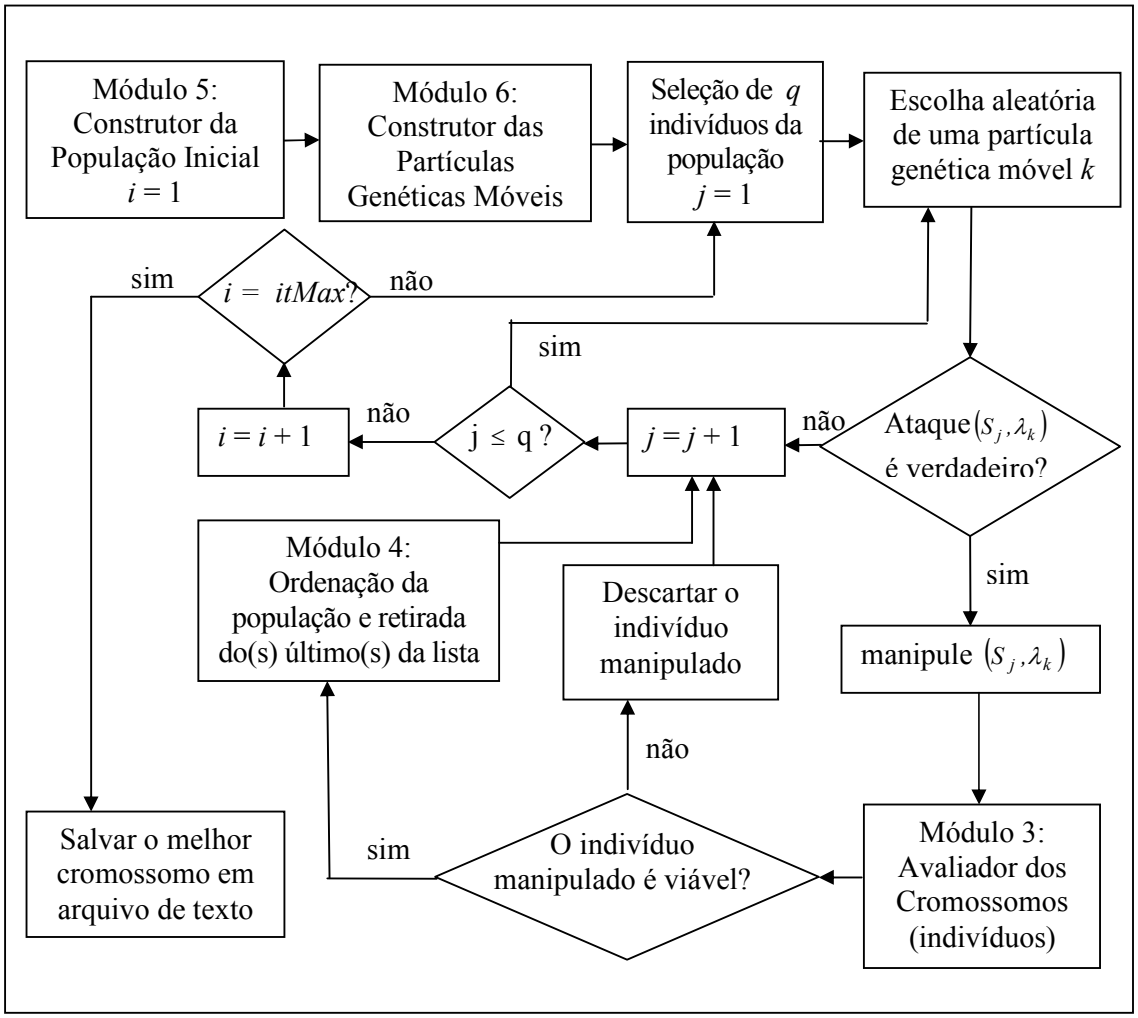

Figura 28: Fluxograma do Módulo 7: ProtoG – PLIM

O algoritmo transgenético ProtoG utilizado neste módulo é o descrito a seguir:

Algoritmo ProtoG

Gerar população $(S_1, S_2, ..., S_N)$

Carregue_Regras_Transgenéticas (*Tipo1*, *Tipo2*, *Tipo3*, *q*, *r*)

Construir_Partículas_Genéticas_Móveis (*r*)

Repita

Escolha_*S* = seleção_população $(S_1, S_2, ..., S_N, q)$

Para *j* = *1* até *q*, **faça**

Escolha_λ = Partícula_genética_móvel (*r*);

Se ataque(S_j, λ) é verdadeiro, **então**

S_j = manipular (S_j, λ)

Inserir S_j na população

fim_Se

fim_Para_*j*

até critério_parada ser satisfeito

Fim

Os passos deste algoritmo, utilizados na aplicação ProtoG – PLIM tem as seguintes funções:

- Carregue_Regras_Transgenéticas (*Tipo1, Tipo2, Tipo3, q, r*): Determina como as partículas genéticas móveis serão formadas, informando o conjunto de regras de administração que serão utilizadas. Define também o número de partículas genéticas (*r*) a serem formadas e quantos serão os indivíduos da sub-população (*q*);
- Construir_Partículas_Genéticas_Móveis (*r*): gera *r* partículas genéticas móveis de acordo com as regras definidas no Módulo 6;
- Escolha_λ: escolhe aleatoriamente uma das *r* partículas genéticas móveis;
- Escolha_*S*: escolhe os indivíduos (cromossomos) para formar a sub-população;
- ataque(S_j, λ): implementa o procedimento p_1.
- manipular (S_j, λ): Caso o ataque seja permitido, a partícula λ efetua a manipulação p_2 no cromossomo S_j;
- critério_parada: o número máximo de iterações *itMax* foi o critério de parada utilizado nesta aplicação.

CAPÍTULO 5

COMPUTAÇÃO EVOLUCIONÁRIA – SIMULAÇÃO: MODELAGEM E METODOLOGIA

5.1 INTRODUÇÃO

A simulação vem sendo empregada com vantagens em sistemas produtivos. As características destes sistemas tais como: mudanças constantes no processo visando melhoria, necessidade de analisar o sistema como um todo e o grau de variabilidade dos dados pertinentes, exigem uma ferramenta de tomada de decisão que englobe estas características de uma forma sistêmica (LIMA, BARBOSA e BEAL, 2003). Vários problemas reais de otimização são muito complexos para serem resolvidos utilizando a Programação Matemática. A recente integração de otimização e simulação tem gerado diversos métodos híbridos para tentar resolver estas dificuldades. Segundo Law e MacComas (2002), o avanço da tecnologia de computadores e o desenvolvimento de técnicas heurísticas de busca e otimização tem permitido nos dias de hoje a integração de técnicas de otimização e simuladores.

Neste capítulo foi criado primeiramente um modelo de simulação para o problema descrito na Seção 3.1, respeitando-se as restrições e função objetivo do modelo em PLIM com representação de tempo contínuo (Seção 3.3). Em seguida, metodologias aplicando otimização baseada em simulação foram desenvolvidas usando-se algoritmos da CE e o modelo de simulação.

5.2 MODELO DE SIMULAÇÃO

A maior parte dos elementos definidos nesta modelagem faz uso da mesma notação do modelo descrito na Seção 3.3.

5.2.1 Elementos do Modelo

- Entidades: clientes (*c*) e processos (*p*);
- Atributos: Os atributos dos clientes são as demandas de diesel representadas pelos parâmetros $Dem_{c,d}$ e o fluxo de envio QE_c. Os processos possuem como atributos as quantidades de diesel a serem carregadas nos tanques, representadas pelos parâmetros $LoteProc_{p,d}$ e o fluxo de recebimento $QR_{p;}$
- Atividades: são atividades do modelo de simulação:
  - a duração do recebimento de produto por um tanque, calculada por:

$$duração\ de\ recebimento = (\ quantidade\ recebida\ ) / QR_p\ ,\ p \in CP \tag{87}$$

  - a duração do envio de produto a um cliente, calculada por:

$$duração\ de\ envio = (\ quantidade\ enviada\ ) / QE_c\ , c \in CC \tag{88}$$

  - tempo de certificação dos tanques.
- Eventos: Os momentos de início e término de recebimento e envio de produtos são eventos que modificam o estado do sistema. Também são eventos os momentos de início e término dos intervalos de tempo dedicados à certificação do produto contido nos tanques.
- Recursos: tanques (*tq*);
- Variáveis: as variáveis deste modelo de simulação são:
  - As variáveis $VolIni_{tq}$, $Volmin_{tq}$, $Volmax_{tq}$, $VolAtual_{tq,e}$ e $VolFinal_{tq}$ utilizadas para os volumes de produtos nos tanques. Estas, coincidem com as do Modelo em PLIM com representação do tempo contínuo;
  - Variáveis de início e término de operação para tanques, clientes e processos representadas por $XI_{tq,c,e}$ e $XF_{tq,c,e}$ onde o índice *c* representa os clientes e os processos da seguinte forma: os índices de 1 a *C* são aplicados aos clientes e os índices de $C + 1$ a $C + P$ são utilizados para os processos. Desta forma, conseguiu-se reduzir o número de índices e utilizar apenas duas variáveis, uma para o início do evento e outra para o término do mesmo;
  - Variável *MaxXFCliente*, obtida através do valor da variável $XF_{tq,c,e}$ no último evento envolvendo clientes;

- Variáveis de estado representadas por $EstadoTanque_{tq}$, $EstadoCliente_c$ e $EstadoProcesso_p$. Estas são atualizadas quando ocorre em algum evento uma operação envolvendo um tanque, um cliente ou um processo, respectivamente.

- Gerador de números aleatórios: foram utilizados no simulador números aleatórios distribuídos uniformemente no intervalo $[0,1]$.

Dois modelos usando otimização baseada em simulação foram desenvolvidos: o primeiro acopla um simulador com AG de Estado Estacionário e o segundo utiliza o algoritmo transgenético ProtoG para a otimização.

5.3 AG - SIMULAÇÃO

Nesta metodologia utilizou-se o simulador para calcular o valor de aptidão dos indivíduos da população do AG. A descrição desta técnica foi desenvolvida a partir de 5 módulos.

5.3.1 Módulo 1: Construtor de indivíduos (cromossomos)

Gera aleatoriamente cadeias que determinam a ordem das operações de transferência e estocagem. Estas cadeias formam os indivíduos. Os passos a seguir descrevem o Módulo 1, utilizado para a obtenção de indivíduos (cromossomos).

Passo 1:

Para a obtenção de um número de cliente, gera-se um número aleatório $c \in \{1,...,C\}$ com a aplicação da fórmula (89).

$$c = Int(C.Rnd) + 1 \tag{89}$$

onde *Rnd* é um número aleatório no intervalo [0, 1] e *Int(.)* é uma função que retorna a parte inteira do número.

Esta fórmula é utilizada até conseguir-se uma cadeia composta por uma seqüência aleatória com cada cliente representado uma única vez. Esta cadeia indica a ordem em que os clientes serão atendidos. Uma cadeia envolvendo 4 clientes é exemplificada a seguir.

c2c4c1c3

Faça $i = 1$.

Passo2:

Extrai-se da cadeia formada pelos clientes, o cliente de ordem *i*.

Passo3:

A partir da demanda do cliente, são formados os eventos, da seguinte forma. Um tipo de produto $d \in \{1,...,D\}$ é gerado aleatoriamente usando a fórmula (90):

$$d = Int(D.Rnd) + 1 \tag{90}$$

onde *Rnd* é um número aleatório no intervalo [0, 1] e *Int(.)* é uma função que retorna a parte inteira do número.

Se a demanda deste cliente por este tipo de diesel é zero então outro número aleatório é gerado até que se encontre um tipo para o qual exista demanda. A seguir, escolhe-se aleatoriamente um tanque $tq \in \{1,...,TQ\}$ usando a fórmula (82).

Se o tanque selecionado não é dedicado ao produto requerido, gera-se outro número aleatório até que o tanque escolhido sirva para o propósito. A seguir, gera-se aleatoriamente um dos processos $p \in \{1,...,P\}$ para enviar ao tanque, usando a fórmula (91).

$$p = Int(P.Rnd) + 1 \tag{91}$$

onde *Rnd* é um número aleatório no intervalo [0, 1] e *Int(.)* é uma função que retorna a parte inteira do número.

Se o processo não possui lote do produto requerido, então outro processo é escolhido aleatoriamente até conseguir-se um que satisfaça a demanda.

Passo 4:

Faz-se então as operações de recebimento e envio do tanque e atualiza-se a demanda do cliente e o lote do processo.

Passo 5:

Se a demanda do cliente ainda não foi toda cumprida, retorna-se ao Passo 3.

Após o primeiro cliente ter sido atendido em toda a sua demanda, obtém-se a primeira parte da cadeia. A seqüência a seguir exemplifica uma cadeia para este cliente.

c2t1p1t3p2

Esta configuração indica que o cliente 2 receberá a primeira parte de sua demanda do tanque 1, carregado pelo processo 1. A segunda parte do seu recebimento virá do tanque 3, carregado pelo processo 2.

Se o cliente de ordem i já foi atendido, faz-se $i = i + 1$ e volta-se ao Passo 2. Este procedimento se repete até que todos os clientes tenham sido atendidos.

A cadeia a seguir mostra um exemplo de indivíduo (cromossomo) gerado neste Módulo, considerando 5 tanques, 4 clientes e 2 processos.

c2t1p1t3p2c4t4p1c1t5p1t2p2t5p1c3t3p1t4p2

5.3.2 Módulo 2: Simulador

Calcula o valor de aptidão do indivíduo (cromossomo) através da simulação das operações de recebimento e envio para tanques, indicadas no cromossomo.

O Módulo 2 (simulador) inicia com a cadeia que forma o indivíduo, gerada pelo Módulo 1. Os passos a seguir descrevem este módulo.

Passo 1:

Extrai-se da cadeia o primeiro cliente com os tanques e processos.

Passo 2:

O simulador verifica, primeiramente, se a quantidade de produto dentro do tanque é suficiente para satisfazer a demanda do cliente por este tipo de produto. Se a quantidade não é suficiente, faz com que o tanque receba o produto até seu enchimento total e então, após o tempo de certificação, envie o volume necessário ao cliente. Este procedimento é aplicado a todas as combinações tanque-processo encontradas na cadeia, deste cliente. As variáveis $XI_{tq,c,e}$, $XF_{tq,c,e}$, $EstadoTanque_{tq}$, $EstadoCliente_{c}$, $EstadoProcesso_{p}$ e $VolAtual_{tq,e}$ são atualizadas após cada operação de recebimento ou envio de produto. Cada operação de recebimento ou envio efetuada pelos tanques é considerada um evento.

Passo 3:

Devido à restrição que impõe que cada cliente deve receber seu pedido de forma ininterrupta, uma rotina foi implementada para fazer a troca de um tanque para o outro de forma imediata, isto é, não deve haver tempo de espera nestas trocas. Muitas vezes, é necessário atrasar o envio de um ou mais tanques para fazer coincidir o término de um envio

com o início do outro. Neste passo, alguns cromossomos se mostram inviáveis e são então rejeitados, muitas vezes, antes mesmo de terminar-se a seqüência imposta pela cadeia.

Passo 4:

Se a atualização imposta pelo Passo 3 foi efetuada com sucesso, verifica-se se existe algum cliente que ainda não foi atendido. Se houver, extraem-se os dados do cromossomo e volta-se ao passo 2.

Passo 5:

Neste passo é feita uma verificação nos lotes de produto enviados pelos processos. Se ainda existe um resíduo dos lotes, é feito o recebimento deste produto pelos tanques. Muitas vezes o cumprimento de uso de todo o lote dos processos não é possível, o que torna o cromossomo inviável e então descartado.

Passo 6:

Este é o último passo do simulador e sua função é calcular o valor da função de aptidão do indivíduo (cromossomo) analisado. A função utilizada para este cálculo é igual à função objetivo do Modelo em PLIM com representação do tempo contínuo, isto é, procura-se entre as variáveis $XF_{tq,c,e}$, $tq = 1,...,TQ;\ c = 1,...,C$ e $e = 1,...,E$, aquela que possui o maior valor. Esse será o valor de aptidão do indivíduo.

5.3.3 Módulo 3: Ordenador de indivíduos

Ordena os indivíduos da população, no caso de minimização, em ordem crescente, de acordo com a função de aptidão. A ordenação da população inicial que se faz à medida que os indivíduos vão sendo gerados e a inserção de indivíduos construídos a partir das operações de cruzamento ou mutação seguem os passos descritos a seguir.

Considerando uma população com *N* indivíduos, e um problema de minimização, no qual a ordenação é feita por ordem crescente de valor de aptidão.

Passo 1:

Se $N > 1$, execute os passos 2 e 3

Passo 2:

Insere-se o indivíduo na *N*-ésima posição no lugar do indivíduo com pior valor de aptidão.

Passo 3:

Se este indivíduo possui função de aptidão pior que o indivíduo de posição N - 1, então ficará na N-ésima posição e está encerrado o processo de ordenação. Caso contrário, é feita a troca de posição entre eles quando então o indivíduo em questão ocupará a posição de ordem N - 1. A seguir, é feita a comparação com o indivíduo de ordem N – 2. Se o indivíduo inserido é pior que este indivíduo, este fica na posição que está e o processo está encerrado. Caso contrário, é feita a troca com o indivíduo que está na posição N – 2. Desta mesma forma procede-se até o primeiro indivíduo da população.

5.3.4 Módulo 4: Construtor da população inicial com *N* indivíduos

Para a formação deste módulo foram utilizados os Módulos 1, 2 e 3 descritos nas Seções 5.3.1, 5.3.2 e 5.3.3. A ordem de aplicação destes módulos é mostrada no Fluxograma da Figura 29.

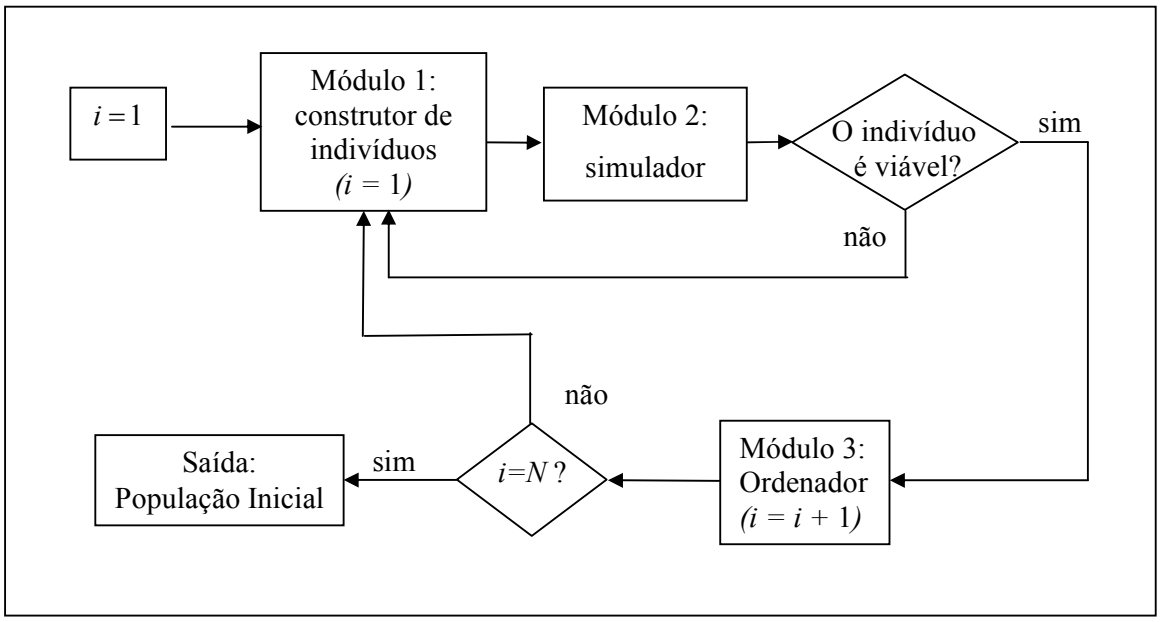

Figura 29: Fluxograma do Módulo 4 – AG - Simulação

Não há necessidade de detalhamento deste módulo, pois este é composto somente pela integração dos Módulos 1, 2 e 3, já descritos.

5.3.5 Módulo 5: Algoritmo Genético (AG)

Este módulo é constituído pelo AG do tipo estado estacionário. Utiliza o resultado obtido pelo Módulo 4 para dados de entrada, o Módulo 2 e o Módulo 3, como mostra a Figura 30. O parâmetro *ItMax* representa o número máximo de iterações do AG.

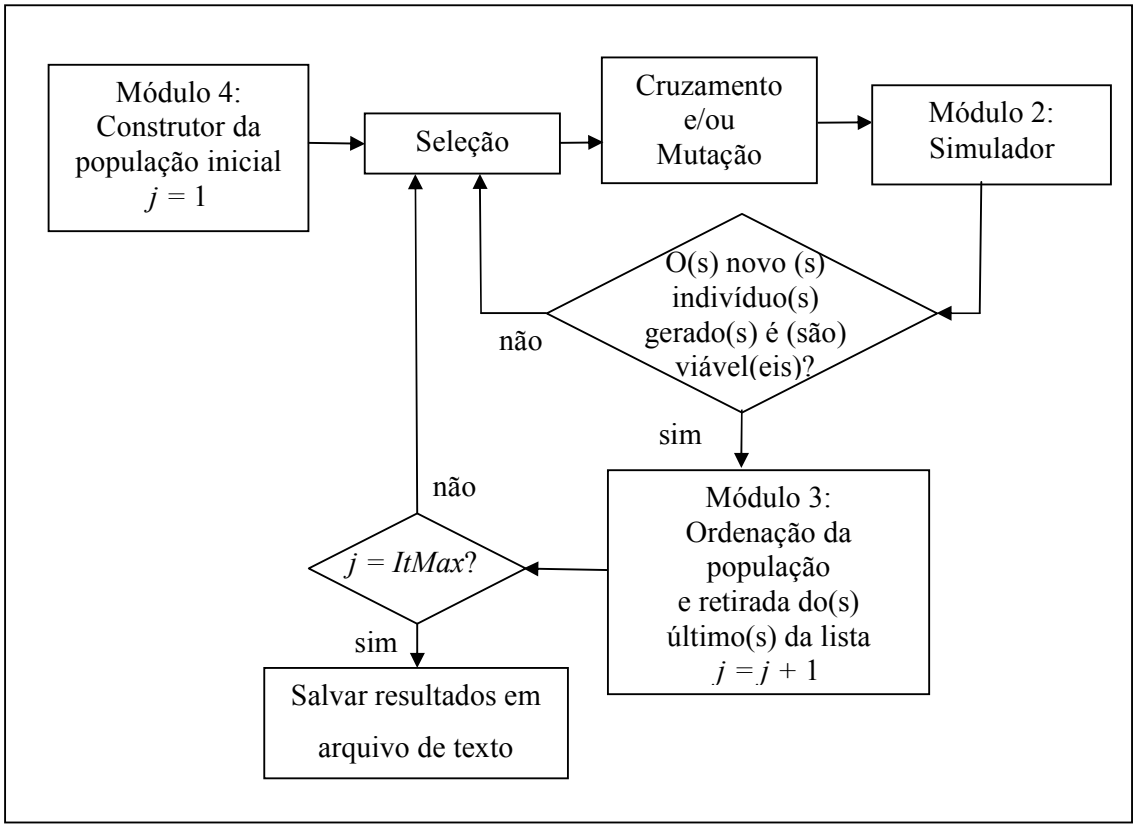

Figura 30: Fluxograma do Módulo 5: AG - Simulação

5.3.5.1 Operadores de seleção do AG

O operador de seleção utilizado foi o proposto por Mayerle (Seção 2.4.7.5) através da fórmula (8).

5.3.5.2 Operador de cruzamento do AG

O cruzamento aplicado foi o C1, descrito na Seção 2.4.8.1. A Figura 31 ilustra como este operador genético foi aplicado com a configuração do cromossomo utilizada no simulador.

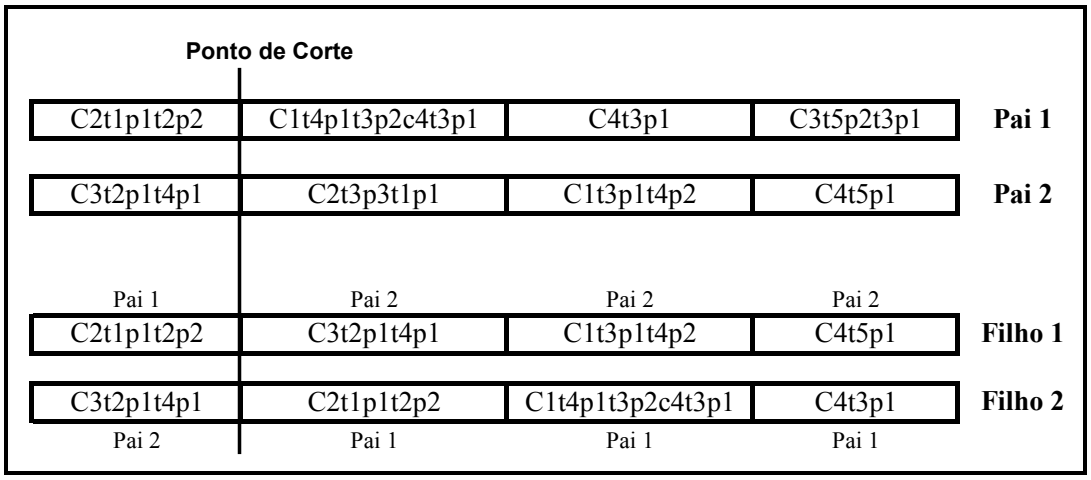

Figura 31: Cruzamento C1 aplicado ao modelo AG – Simulação

5.3.5.3 Operador de Mutação

Para este modelo, foi criado um operador de mutação especializado para ser aplicado aos cromossomos. Um número aleatório *i* entre 1 e *C* é gerado de forma uniforme. A cadeia (gene) que possui a combinação *Ci* é retirada do cromossomo. Uma nova cadeia é gerada usando-se os passos 3, 4 e 5 do Módulo 1, considerando-se apenas um cliente. Esta cadeia é então inserida na mesma posição. A Figura 32 ilustra o procedimento deste operador, com $i = 3$.

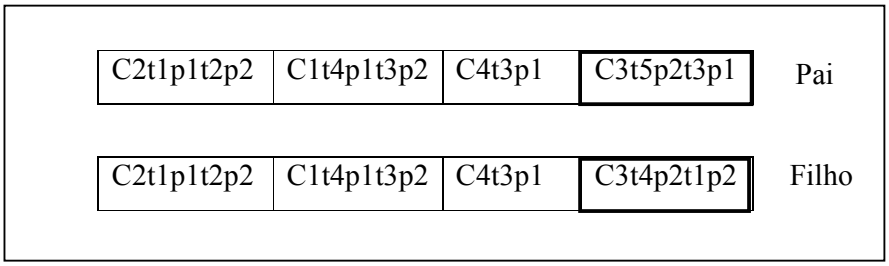

Figura 32: Operador de mutação aplicado ao modelo AG - Simulação

Finalmente, para completar o detalhamento do Módulo 5, será descrito o algoritmo AG de estado estacionário empregado neste modelo.

5.3.5.4 Algoritmo AG de estado estacionário aplicado no modelo de otimização com simulação

Algoritmo AG de estado estacionário

Inicialize a população *P* de cromossomos
Avalie os indivíduos da população *P*
Ordene a população *P* em ordem crescente pelo valor de aptidão
Repita
Selecione operador genético
Se o operador for aplicado de acordo com sua probabilidade **então**
Selecione indivíduo(s) para operação
Aplique o operador genético selecionado
Avalie o(s) indivíduo(s) gerado(s)
Insira este(s) indivíduo(s) em *P* de acordo com sua aptidão
Fim_Se
Até máximo de gerações
Fim

5.4 ALGORITMO ProtoG - SIMULADOR

A metodologia que utiliza o simulador acoplado ao algoritmo ProtoG é semelhante à descrita para o AG baseado em simulação (Seção 5.3). Os módulos 1, 2, 3 e 4 são os mesmos para as duas aplicações. Foram então desenvolvidos, para esta metodologia, os módulos 5, 6 e 7.

Para a construção das partículas genéticas móveis $\lambda = (I, \Phi)$, deve-se definir a cadeia de informação I e o método de manipulação Φ. Nesta metodologia aplicou-se $\Phi = \{p_1, p_2\}$ (Seção 2.5.1.2). Como serão utilizados procedimentos do tipo p_1 e p_2, as partículas genéticas a serem utilizadas são os plasmídios. O procedimento p_1 é o que estabelece quando um cromossomo é suscetível à manipulação ou ataque pelas partículas genéticas móveis. Neste caso, um indivíduo será suscetível ao ataque se houver melhoria no valor de aptidão após a manipulação. Já o procedimento p_2, em caso de ataque, define como a

informação *I*, que é transportada pela partícula genética móvel é transferida para o cromossomo.

O processo de formação das partículas genéticas móveis foi divido nos módulos 5 (construtor das cadeias *I*) e módulo 6 (manipulador de indivíduos Φ).

5.4.1 Módulo 5: Construtor das cadeias *I*

Para definir como serão formadas as cadeias *I* utiliza-se a Regra Tipo 1 (Seção 2.5.1.3). Neste módulo, duas fontes distintas foram utilizadas para estas cadeias: a primeira extrai dos *G* primeiros indivíduos as cadeias para cada um dos clientes, obtendo-se desta forma *C*.*G* partículas genéticas móveis; a segunda faz uso de uma heurística que constrói e avalia um determinado número de cadeias. O módulo 1 (construtor de indivíduos) cria cadeias com apenas um cliente. Essas são avaliadas pelo módulo 2 (simulador). Este processo é repetido para cada cliente e ao final, são extraídas as *C.H* melhores cadeias que farão parte do banco de partículas genéticas móveis. O número total de partículas genéticas móveis será $r = C.(G + H)$. Os passos a seguir descrevem como funciona o Módulo 5.

Passo 1:

Selecionar da população os *G* primeiros indivíduos. Vale observar que os indivíduos estão ordenados em ordem crescente para o valor de aptidão.

Passo 2:

Extrair destes indivíduos, as seqüências de tanques e processos para cada cliente *c*. Estas seqüências serão as *C.G* primeiras cadeias *I*.

Os passos 3 a 6 constroem as *C.H* cadeias restantes. Seja $k = 1,...,H$ e $c = 1,...,C$ (número de clientes) e $it = 1,...,ItM$ (número de vezes que a heurística deverá simular para ao final extrair os melhores resultados).

Passo 3:

Faça $it = 1$.

Passo 4:

Faça $k = 1$.

Passo 5:

Faça $c = 1$.

Passo 6:

A partir da demanda do cliente c, são formados os eventos, da seguinte forma. Um tipo de produto $d \in \{1,...,D\}$ é gerado aleatoriamente usando a fórmula (90). Se a demanda deste cliente por este tipo de diesel é zero, então outro número aleatório é gerado até que se encontre um tipo para o qual exista demanda. A seguir, escolhe-se aleatoriamente um tanque $tq \in \{1,...,TQ\}$ usando a fórmula (82). Se o tanque selecionado não é dedicado ao produto requerido, gera-se outro número aleatório até que o tanque escolhido sirva para o propósito. A seguir, gera-se aleatoriamente um dos processos $p \in \{1,...,P\}$ para enviar ao tanque, usando a fórmula (91). Se o processo não possui lote do produto requerido, então outro processo é escolhido aleatoriamente até conseguir-se um que satisfaça a demanda.

Passo 7:

Faça as operações de recebimento e envio do tanque e atualize a demanda do cliente e o lote do processo. Se a demanda do cliente ainda não foi toda cumprida, retorne ao Passo 6.

Passo 8:

Avalie a cadeia utilizando o módulo 2 (simulador) fixando-se a variável cliente para o valor "1".

Passo 9:

Se esta é a primeira cadeia encontrada, salve-a como cadeia(c, k). Caso contrário, compare seu valor de aptidão com a que está salva. Se for melhor, troque a cadeia(c, k) pela nova cadeia encontrada.

Passo 10: Faça $c = c + 1$.

Se $c \leq C$ retorne ao Passo 6.

Passo 11: Faça $k = k + 1$.

Se $k \leq H$, retorne ao Passo 5.

Passo 12: Faça $it = it + 1$.

Se $it \leq ItM$, retorne ao Passo 4.

Para exemplificar os Passos 1 e 2 considere um problema com parâmetros $C = 2$, $P = 2$, $TQ = 5$ e $H = 2$. Extraem-se os dois primeiros indivíduos da população que estão representados pelas seqüências a seguir:

Indivíduo 1: C1t1p1t5p2t3p1t1p1C2t4p2t2p1

Indivíduo 2: C2t4p2t3p1C1t1p1t5p2t2p1t1p1

As quatro cadeias *I* que são obtidas usando-se os Passos 1 e 2 são:

Cadeia 1: C1t1p1t5p2t3p1t1p1

Cadeia 2: C2t4p2t2p1

Cadeia 3: C2t4p2t3p1

Cadeia 4: C1t1p1t5p2t2p1t1p1

5.4.2 Módulo 6: Manipulador de indivíduos Φ

Este módulo trata das regras de Tipo 2 (Seção 2.5.1.3) que possuem a função de formalizar os operadores de manipulação de um vetor λ. Estas regras definem como as cadeias de informação *I* serão transcritas nos indivíduos (cromossomos) ou como o conteúdo do cromossomo será rearranjado.

Para a metodologia abordada foi necessária a construção de duas regras do Tipo 2. A primeira, para as *G* primeiras cadeias, é composta pelo Passo 1. A segunda, para as *H* cadeias restantes, é formada pelo Passo 2.

Seja $r = C.(G + H)$, j o indivíduo escolhido para manipulação e k o número da partícula genética móvel.

Passo 1:

Se $k > C.G$, então vá ao Passo 3.

Passo 2:

Identificar no indivíduo j a cadeia que possui o mesmo cliente que a partícula k. Retirar esta seqüência do indivíduo e inserir a cadeia k na mesma posição. Vá ao Passo 4.

Passo 3:

Identificar no indivíduo j a cadeia que possui o mesmo cliente que a partícula k. Retirar esta seqüência do indivíduo e inserir a cadeia k no início da cadeia restante do indivíduo.

Passo 4: Fim

5.4.3 Módulo 7: Algoritmo ProtoG

Este módulo é semelhante ao descrito na Seção 4.3.2 para a metodologia Algoritmo ProtoG – PLIM. Utiliza os Módulos 3, 4 da metodologia AG - Simulação e os Módulos 5 e 6 detalhados nas Seções 5.4.1 e 5.4.2. Fazendo uso da Regra Tipo 3 (Seção 2.5.1.3), define-se o tamanho q da sub-população a ser extraída da população e o número r de partículas genéticas móveis a serem geradas. O operador de seleção utilizado para a escolha dos indivíduos da sub-população foi o proposto por Mayerle (Seção 2.4.7.5) através da fórmula (8). Cada indivíduo da sub-população será testado por todas as partículas genéticas móveis $\lambda = (I, \Phi)$. Vale ressaltar que o ataque se consolidará se o valor de aptidão do indivíduo manipulado for melhor que o do indivíduo. O parâmetro *ItMax* representa o número máximo de iterações do algoritmo. O fluxograma da Figura 33 ilustra os passos deste módulo.

O algoritmo transgenético ProtoG utilizado neste módulo é o descrito a seguir:

Algoritmo ProtoG

Gerar população $(S_1, S_2, ..., S_N)$

Carregue_Regras_Transgenéticas(*Tipo1*, *Tipo2*, *Tipo3*, *q*, *r*)

Construir_Partículas_Genéticas_Móveis (*r*)

Repita

Escolha_S = seleção_população $(S_1, S_2, ..., S_N, q)$

Para j = 1 até q, **faça**

Para k = 1 até r, **faça**

Se ataque(S_j, λ_k) é verdadeiro, **então**

S_j = manipular (S_j, λ_k)

Inserir S_j na população de acordo com o valor de aptidão

fim_Se

fim Para k

fim_Para_j

até critério_parada ser satisfeito

Os passos deste algoritmo, utilizados na metodologia ProtoG – Simulação possui as mesmas funções descritas para a abordagem Algoritmo ProtoG – PLIM, excluindo-se apenas a função Escolha_λ.

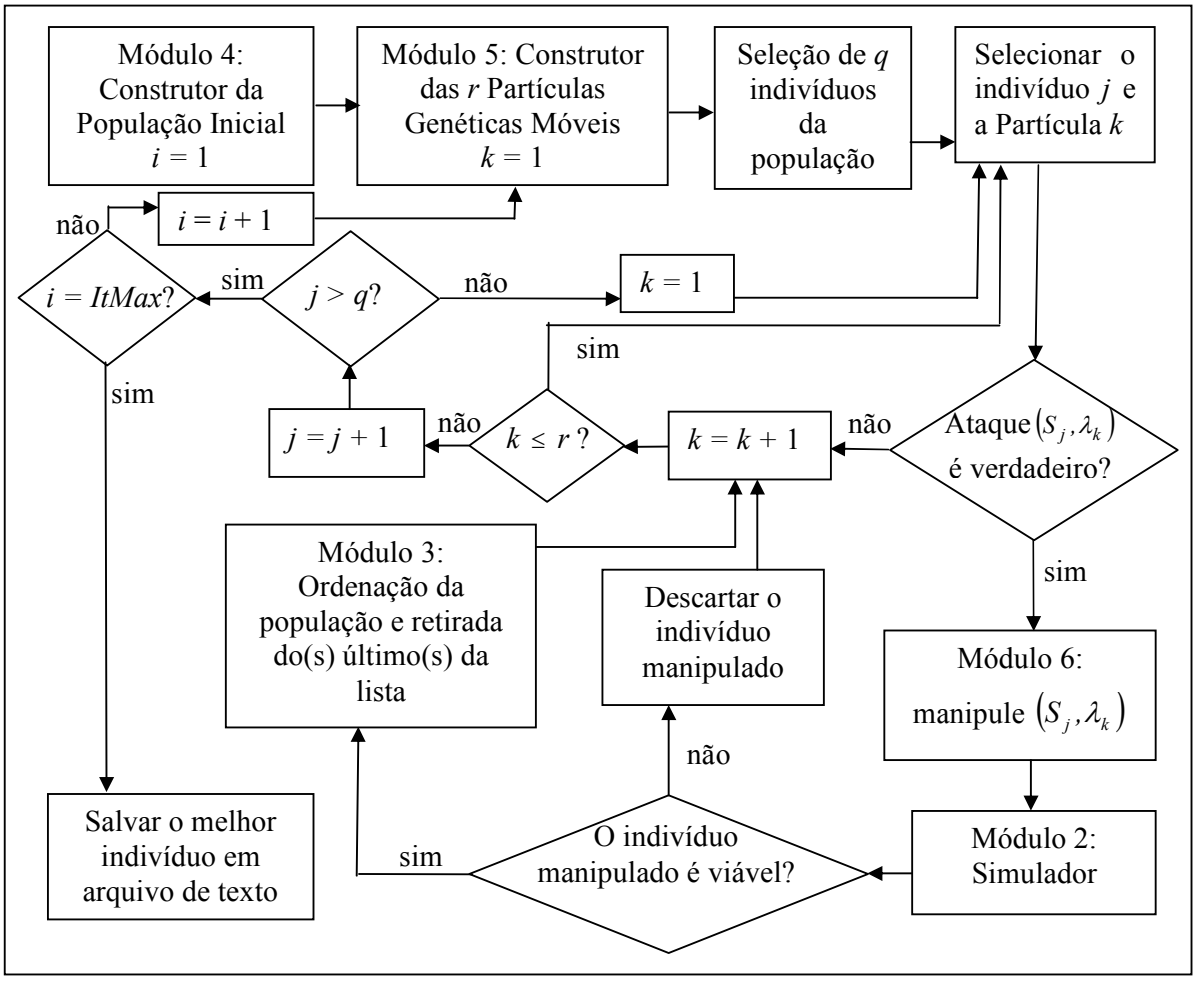

Figura 33: Fluxograma do Módulo 7: Simulador – ProtoG

CAPÍTULO 6

IMPLEMENTAÇÃO E RESULTADOS

6.1 INTRODUÇÃO

Neste capítulo são apresentados os resultados dos testes obtidos através da implementação computacional das metodologias descritas. Estes resultados são apresentados inicialmente para os modelos com representação de tempo discreto com a implementação das metodologias: algoritmo *branch and bound* com o uso do aplicativo LINGO 8.0, AGEEH – PLIM e Algoritmo ProtoG – PLIM. Em seguida, são apresentados os resultados da implementação das metodologias para representação de tempo contínuo: algoritmo *branch and bound* com o uso do aplicativo LINGO 8.0, AG - Simulação e Algoritmo ProtoG - Simulação.

Os programas computacionais implementados foram desenvolvidos com o aplicativo Visual Basic 6.0 (VB 6.0) na versão *Enterprise Edition*. Este aplicativo é uma linguagem de programação integrante do pacote *Microsoft Visual Studio*, de autoria da empresa *Microsoft Corporation* com programação dirigida por eventos (*event driven*) e um ambiente de desenvolvimento integrado (*IDE - integrated development environment*).

Na implementação das abordagens AGEEH – PLIM e Algoritmo ProtoG – PLIM foram utilizados dois módulos em VB, fornecidos pela LINDO *Systems Inc.*, para integração entre VB 6.0 e LINGO 8.0. Já a implementação de AG - Simulação e Algoritmo ProtoG - Simulação foi feita integralmente em VB.

6.2 IMPLEMENTAÇÃO PARA O MODELO COM DISCRETIZAÇÃO UNIFORME DE TEMPO

Para o modelo com discretização uniforme de tempo, descrito na Seção 3.2, foram efetuados três grupos de testes distintos para a obtenção de resultados: resolução do modelo em PLIM utilizando o aplicativo LINGO 8.0, aplicação das metodologias AGEEH – PLIM e Algoritmo ProtoG – PLIM.

Os dados utilizados para estes grupos de testes foram:

- Número de tanques: $TQ = 4$;
- Número de clientes: $C = 2$;
- Número de intervalos de tempo: $T = 24$;
- Custo de bombeio para os clientes: CB_1 = 0,15 unidades monetárias/mil m^3 e CB_2 = 0,2 unidades monetárias /mil m^3 de produto enviado;
- Custo de armazenamento nos tanques: CA_{tq} = 0,01 unidades monetárias/mil m^3 de produto estocado em qualquer tanque;
- Custo de troca: CTR_{tq} = 2,00 unidades monetárias para cada troca efetuada entre tanques na operação de recebimento;
- Vazão mínima de recebimento pelos tanques: $QRmin$ = 0,6 mil m^3/h;
- Vazão máxima de recebimento pelos tanques: $QRmax$ = 0,7 mil m^3/h;
- Vazão mínima de envio ao cliente c: $QEmin_1$ = 0,5 mil m^3/h e $QEmin_2$ = 0,9 mil m^3/h;
- Vazão máxima de envio ao cliente c: $QEmax_1$ = 0,6 mil m^3/h e $QEmax_2$ = 1 mil m^3/h;
- Volume mínimo permitido nos tanques: $Volmin_{tq}$ = 1 mil m^3 para todos os tanques;
- Volume máximo permitido nos tanques: $Volmax_{tq}$ = 16 mil m^3 para todos os tanques;
- Volume no tanque tq no início do processo: $Volini_1$ = 7 mil m^3, $Volini_2$ = 1 mil m^3, $Volini_3$ = 1 mil m^3 e $Volini_4$ = 1 mil m^3;
- Demanda de diesel para cada cliente c: DEM_1 = 5 mil m^3 e DEM_2 = 6 mil m^3;

Para a obtenção dos resultados foi utilizado um computador com processador 950 *MHz* e 128 *MB* de memória *RAM*.

6.2.1 Resultados para a modelagem PLIM com representação de tempo discreto

Os dados acima aplicados ao modelo PLIM, resultaram em 390 variáveis contínuas, 766 inteiras e um total de 2149 restrições. O modelo foi executado 25 vezes pelo LINGO 8.0, até encontrar-se o valor ótimo igual a 6,285 unidades monetárias. A Tabela 4 mostra os resultados obtidos, incluindo, os valores mínimo e máximo, a mediana, a média e o desvio padrão para o número de iterações efetuadas pelo LINGO 8.0 e o tempo despendido no processo.

Tabela 4: Resultados para o modelo PLIM

Nº do Teste	Nº de Iterações	Tempo Computacional (s)
1	506002	963
2	731715	1362
3	630809	1136
4	600538	1194
5	681425	1217
6	566063	1005
7	805159	1460
8	724202	1355
9	1068176	1899
10	859705	1569
11	316400	582
12	1523047	2804
13	1213626	2044
14	350441	622
15	772195	1423
16	515443	925
17	1107641	2044
18	734734	1355
19	697283	1284
20	605848	1166
21	724949	1328
22	822491	1498
23	988042	1706
24	542815	1163
25	489371	881
Mínimo	316400	582
Máximo	1523047	2804
Mediana	724202	1328
Média	743124,8	1359,4
Desvio Padrão	273218,69	480,52

O resultado para a média do tempo computacional expresso em minutos foi de 22 minutos e 39 segundos com desvio padrão de 8 minutos e 52 segundos. Usando as médias de número de iterações e tempo computacional obtém-se um desempenho para o LINGO 8.0 de 547 iterações por segundo.

Para melhor compreensão do resultado encontrado, as operações dos tanques, do processo e dos clientes em cada intervalo de tempo foram ilustradas em uma carta de Gantt, mostrada na Figura 34.

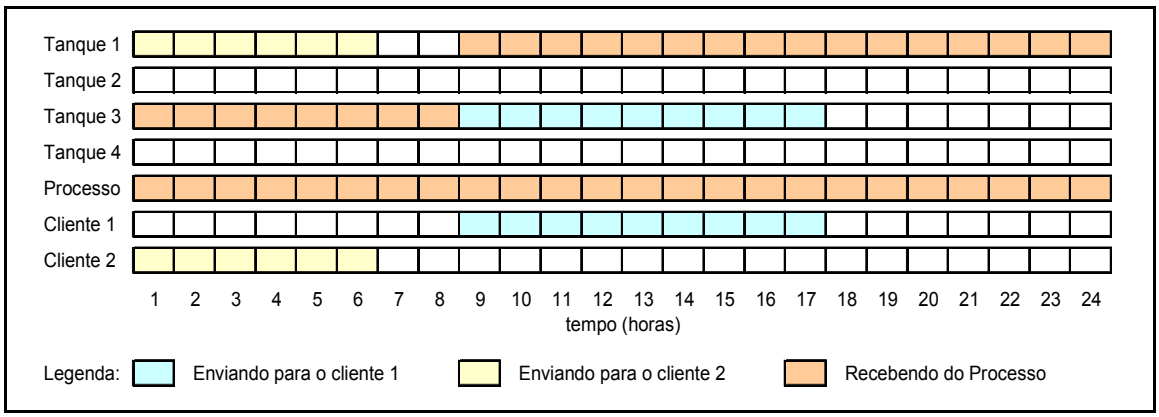

Figura 34: Carta de Gantt – Resultado do Modelo PLIM com representação discreta do tempo

Vale ressaltar que no período de 24 horas, ocorreu apenas uma troca no recebimento pelo processo do tanque 3 para o tanque 1, no início do intervalo 9.

As Figuras 35, 36, 37 e 38 ilustram graficamente a variação do volume destes tanques no período de 24 horas considerado.

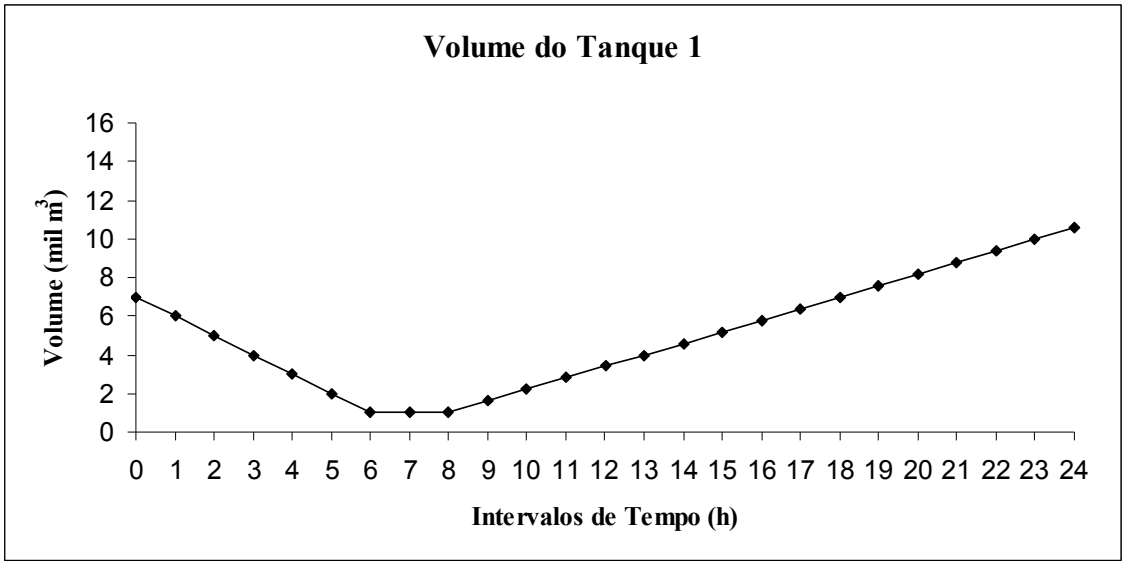

Figura 35: Variação do volume do tanque 1 (PLIM com tempo discreto)

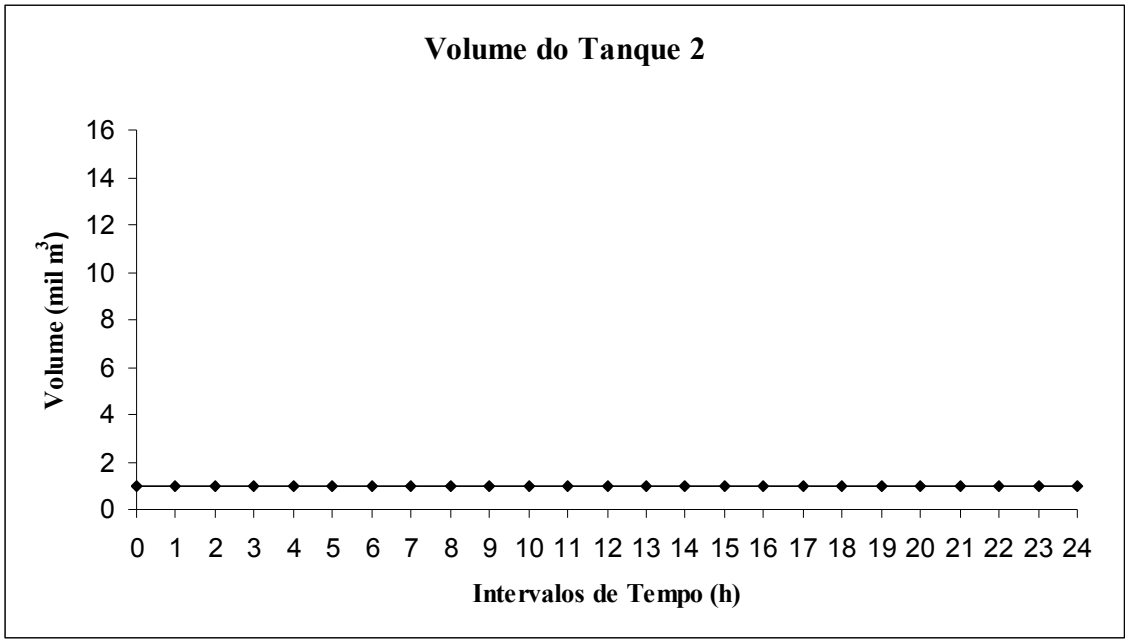

Figura 36: Variação do volume do tanque 2 (PLIM com tempo discreto)

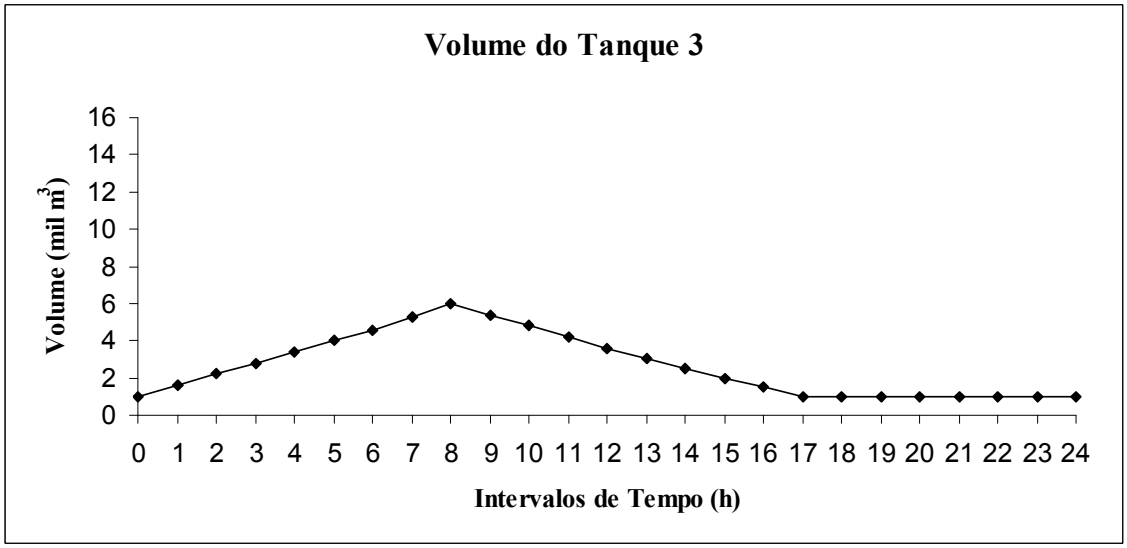

Figura 37: Variação do volume do tanque 3 (PLIM com tempo discreto)

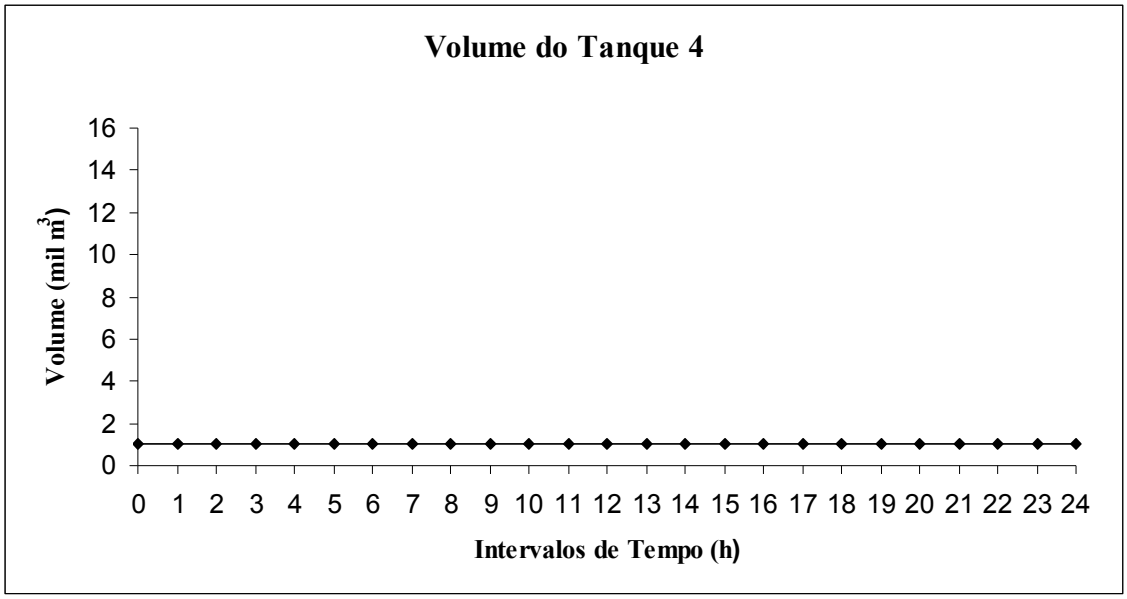

Figura 38: Variação do volume do tanque 4 (PLIM com tempo discreto)

6.2.2 Resultados para a metodologia AGEEH – PLIM

Para a obtenção dos resultados para o modelo AGEEH-PLIM foram efetuados 135 rodadas de testes, variando-se os parâmetros: tamanho da população e o número necessário de iterações do AGEEH para realizar a busca local (hibridização). Os dados referentes ao número de tanques, clientes e intervalos de tempo devem estar em conformidade com o modelo em PL do LINGO 8.0, interligado ao programa. As rodadas de testes foram efetuadas e os resultados coletados pelo programa foram: número de iterações para o AGEEH, número de iterações do LINGO 8.0 na resolução do AGEEH, número de iterações do LINGO 8.0 para a resolução do problema em PLIM, tempos computacionais para o AGEEH e PLIM e, finalmente, valor de aptidão (função objetivo). O tempo computacional para o modelo PLIM é o tempo despendido na resolução do modelo discreto introduzindo-se o resultado obtido pelo AGEEH como dados do problema, conforme descrito na Seção 4.2. Em cada teste foram salvos todos os indivíduos com valor da função de aptidão menor que 6,9, sendo este valor 10% maior que o valor ótimo (resolução em PL) de 6,266523 unidades monetárias. Desta forma, com as 135 rodadas de testes foram obtidos 351 resultados que podem ser visualizados nas Tabelas do Anexo 1.

A Figura 39 ilustra a tela de interface com o usuário do programa computacional desenvolvido em VB para esta abordagem. Deve-se observar que as caixas de texto dedicadas às probabilidades de mutação e cruzamento foram utilizadas para que se pudesse visualizar o valor das probabilidades adaptativas utilizadas em cada iteração.

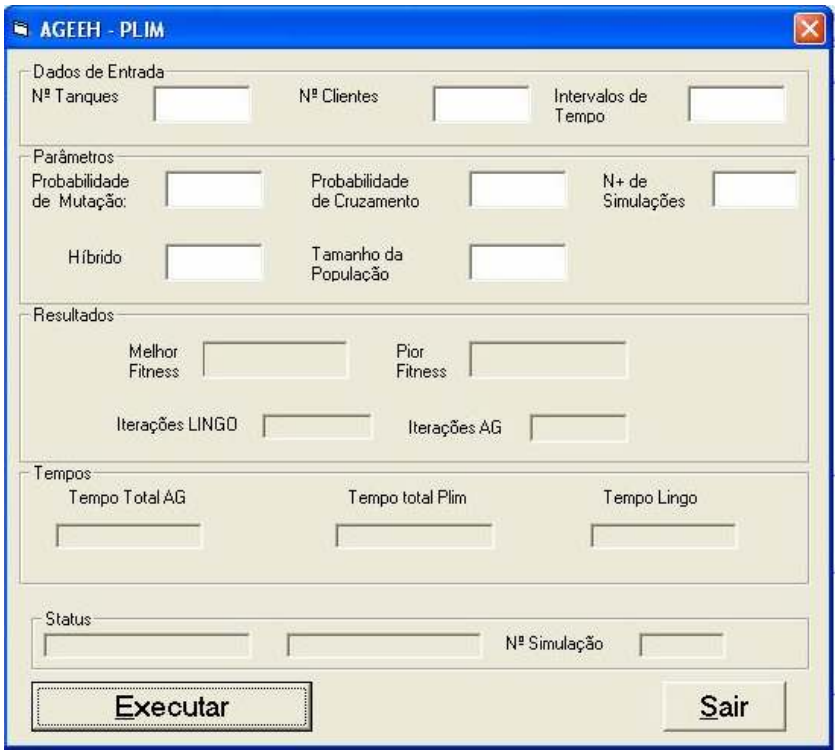

Figura 39: Formulário de interface do programa AGEEH-PLIM

Nas rodadas de testes foram combinados os seguintes valores para os parâmetros:

- Tamanho da População: 40, 45 e 50;
- Número de iterações para a busca local (hibridização): 100, 115 e 125.

Para as probabilidades adaptativas foram utilizados os valores de $k_1 = 0{,}5$, $k_2 = 1$, $k_3 = 0{,}1$ e $k_4 = 0{,}5$.

Na Tabela 5 são apresentadas estatísticas descritivas do número de iterações do AGEEH, do número de iterações do LINGO 8.0, do tempo computacional e da função de aptidão obtidos a partir dos 351 resultados separados por intervalos para o valor da função de aptidão *f*.

Tabela 5: Estatísticas descritivas dos testes da metodologia AGEEH – PLIM

Intervalos da função f	Nº de Iterações do AGEEH	Nº de Iterações do LINGO	Tempo Computacional (s)	Valor da Função de Aptidão
$f = 6{,}285$	908 ± 502	1229718 ± 623219	1219 ± 621	6,285 ± 0
$6{,}285 < f \leq 6{,}47$	679 ± 261	928961 ± 347005	921 ± 352	6,353 ± 0,056
$6{,}47 < f \leq 6{,}66$	634 ± 241	860666 ± 307462	856 ± 313	6,530 ± 0,025
$6{,}66 < f \leq 6{,}9$	601 ± 271	837905 ± 365698	834 ±372	6,790 ± 0,071

Média ± desvio padrão

O resultado para a média geral do tempo computacional expresso em minutos foi de 16 minutos e 59 segundos com desvio padrão de 8 minutos e 19 segundos. Através das médias do número de iterações do LINGO e do tempo computacional do programa desenvolvido calculou-se o desempenho de aproximadamente 1006 iterações do LINGO por segundo.

A Tabela 6 mostra as médias do número de iterações do AG, do número de iterações do LINGO e do tempo computacional para cada combinação dos parâmetros tamanho da população e número de iterações para hibridização, quando o resultado ótimo de 6,285 foi obtido. O melhor desempenho pôde ser observado para tamanho da população igual a 45 e número de iterações para hibridização igual a 125.

Tabela 6: Média para o resultado ótimo da metodologia AG – PLIM

Tamanho População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do LINGO	Tempo Computacional (segundos)
40	100	1211	1626152	1609,0
40	115	971	1306102	1291,7
40	125	940	1219458	1207,2
45	100	786	1100281	1091,5
45	115	845	1152047	1185,3
45	125	732	993706	978,6
50	100	735	1056174	1032,9
50	115	1079	1440459	1428,9
50	125	873	1173083	1149,5
Médias	-	908	1229718	1219,4

A Figura 40 ilustra o comportamento do número de iterações do AGEEH em relação ao valor da função de aptidão.

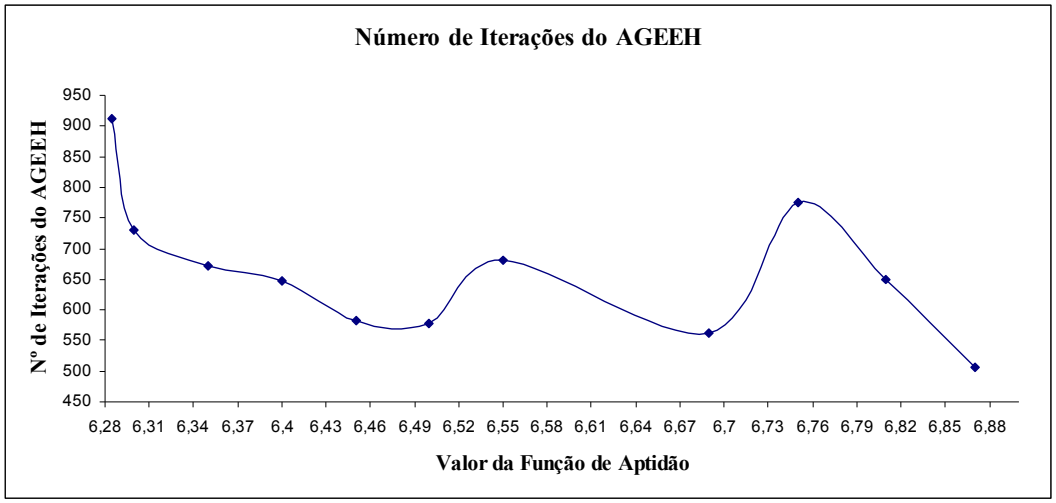

Figura 40: Número de Iterações do AGEEH de acordo com o valor da função de aptidão

Para ilustrar o número de iterações do LINGO 8.0, de acordo com o valor da função de aptidão, construiu-se o gráfico da Figura 41.

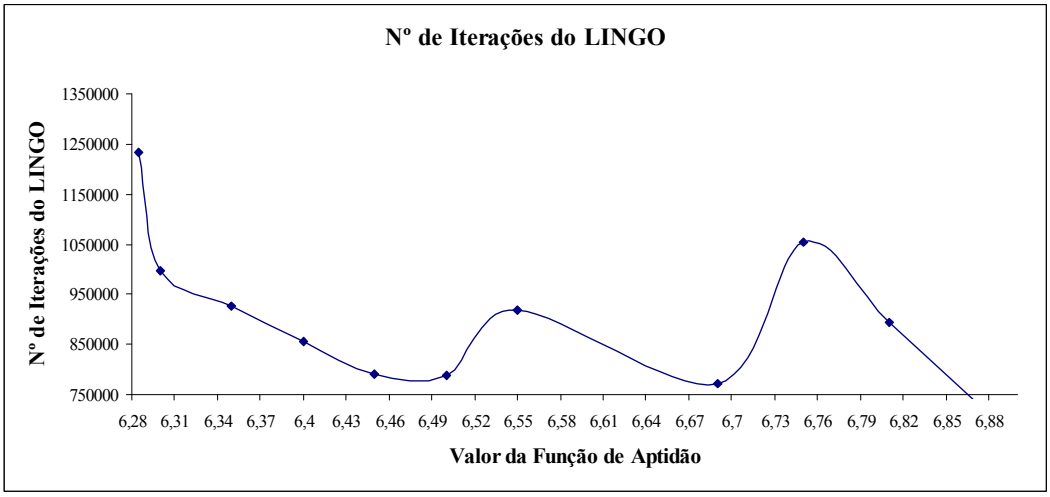

Figura 41: Número de Iterações do LINGO de acordo com o valor da função de aptidão

Finalmente, a Figura 42 ilustra o tempo computacional (em segundos) em relação ao valor de aptidão.

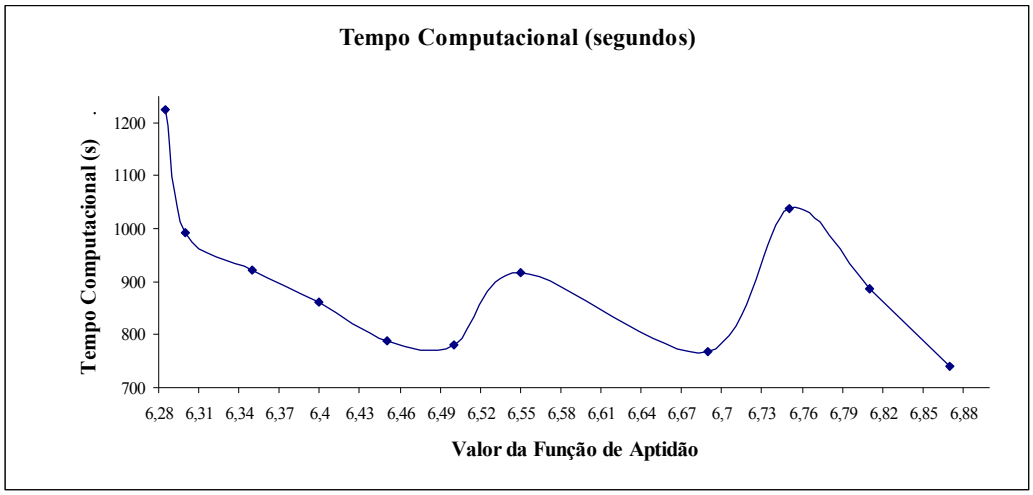

Figura 42: Tempo computacional de acordo com o valor da função de aptidão

Observando-se as Figuras 40, 41 e 42, percebe-se uma semelhança no comportamento dos resultados para número de iterações do AGEEH, número de iterações do LINGO e tempo computacional em relação ao valor da função de aptidão. Ainda, estes gráficos mostram uma acentuada variabilidade e são observados pontos de máximo local, próximos de 6,76; 6,55 e 6,28. Nestes pontos, quando o algoritmo alcançou mínimos locais, um maior número de iterações do AGEEH, maior número de iterações do LINGO e maior tempo computacional foram despendidos, gerando um esforço computacional maior para prosseguir a busca.

Uma análise foi efetuada com o objetivo de investigar a influência do tamanho da população e o número de iterações para hibridização sobre os resultados do número de iterações do AG, número de iterações do LINGO e tempo computacional quando se atinge o resultado ótimo.

Inicialmente, estimou-se os coeficientes de correlação de Pearson entre as variáveis número de iterações do AG, número de iterações do LINGO e tempo computacional, duas a duas e testou-se a hipótese nula de ausência de correlação versus a hipótese alternativa de existência de correlação. Os resultados indicaram significância estatística no nível de 0,05. Os coeficientes de correlação *r* evidenciaram uma forte associação entre número de iterações do AG e número de iterações do LINGO (*r = 0,9913*), entre número de iterações do AG e tempo computacional (*r = 0,9875*) e entre número de iterações do LINGO e tempo computacional (*r = 0,9978*). Sendo assim, a análise foi efetuada somente para o número de iterações do AG.

Considerando-se os níveis 40, 45 e 50 para o fator tamanho da população e os níveis 100, 115 e 125 para o fator número de iterações para hibridização, efetuou-se uma análise de variância para avaliar a influência dos fatores sobre o número médio de iterações do AG para

atingir o resultado ótimo. Inicialmente, testou-se a hipótese nula de inexistência de interação entre o tamanho da população e o número de iterações para hibridização. O resultado indicou que não há esta interação (*p = 0,2307*). Em seguida, ao testar-se a hipótese nula de médias iguais para os três níveis de iterações para hibridização, verificou-se que não existe diferença significativa entre eles (*p = 0,5992*). Da mesma forma, para os três níveis da população, não houve diferença significativa quanto ao número de iterações do AG (*p = 0,0553*). Entretanto, para o nível de significância de 0,05, pode-se afirmar que, para o tamanho da população há uma tendência à diferença estatisticamente significativa entre os níveis 40, 45 e 50 da população, em relação ao número médio de iterações do AG para atingir o resultado ótimo.

O comportamento semelhante do número de iterações do AG, número de iterações do LINGO e tempo computacional, para as combinações de tamanho da população e número de iterações para hibridização pode ser observado nos gráficos das Figuras 43, 44 e 45, construídos a partir das Tabelas 7, 8, e 9, das médias obtidas com todos os resultados para cada uma destas variáveis. Os gráficos confirmam o resultado do coeficiente de correlação encontrado. O melhor desempenho pode ser observado para tamanho da população igual a 45 e número de iterações para hibridização igual a 100 (Tabelas 7, 8 e 9 e Figuras 43, 44 e 45).

Tabela 7: Médias para o número de iterações do AGEEH de acordo com o número de iterações para hibridização e tamanho da população

	Número de iterações para hibridização		
Tamanho da População	100	115	125
40	834	805	710
45	640	746	696
50	661	797	901

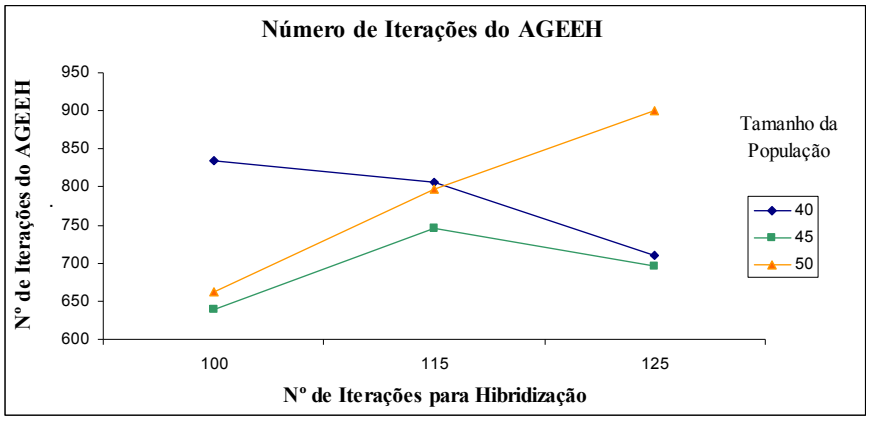

Figura 43: Número de iterações do AGEEH de acordo com o número de iterações para hibridização e tamanho da população

Tabela 8: Médias para o número de iterações do LINGO de acordo com o número de iterações para hibridização e tamanho da população

	Número de iterações para hibridização		
Tamanho da População	100	115	125
40	1142776	1094358	941442
45	907578	1013426	944055
50	941944	1085100	1200496

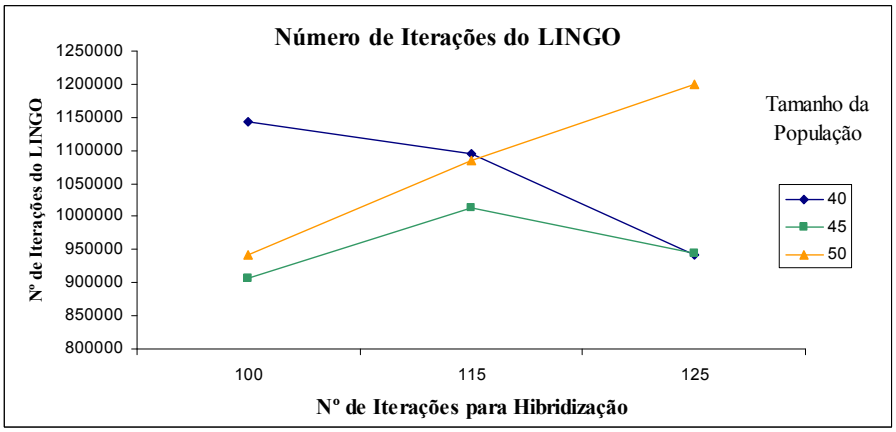

Figura 44: Número de iterações do LINGO de acordo com o número de iterações para hibridização e tamanho da população

Tabela 9: Médias para o tempo computacional de acordo com o número de iterações para hibridização e tamanho da população

	Número de iterações para hibridização		
Tamanho da População	100	115	125
40	1134	1079	932
45	901	1039	939
50	918	1069	1184

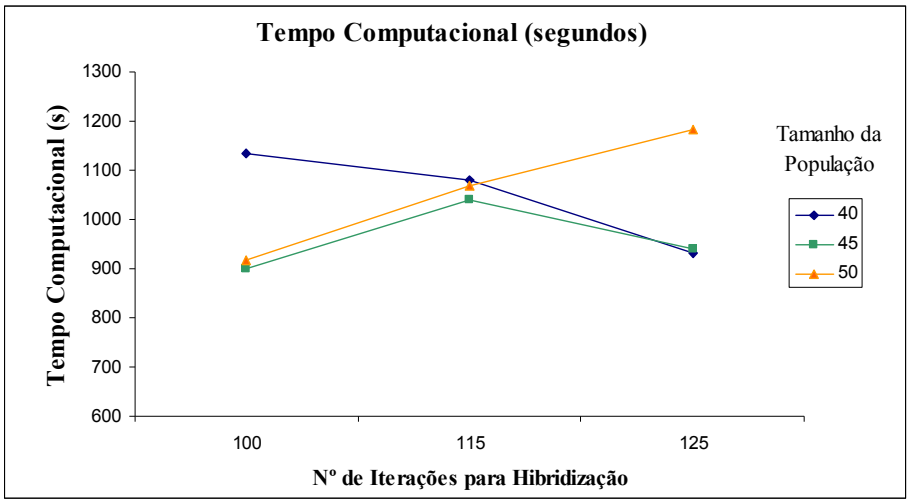

Figura 45: Tempo computacional de acordo com o número de iterações para hibridização e tamanho da população

6.2.3 Resultados obtidos com a metodologia algoritmo ProtoG – PLIM

Para a obtenção dos resultados para a metodologia algoritmo ProtoG - PLIM foram efetuadas 135 rodadas de testes, variando-se os parâmetros: tamanho da população, tamanho da sub-população e tamanho da partícula genética móvel. Da mesma forma que para a metodologia AGEEH – PLIM, os dados referentes ao número de tanques, clientes e intervalos de tempo devem ser também fornecidos pelo usuário. Também, estes dados devem estar em conformidade com o modelo em PL do LINGO 8.0, interligado ao programa. Os resultados gerados pelo programa para a fase de resolução do Algoritmo ProtoG são: número de iterações do algoritmo ProtoG, número de iterações do LINGO, tempo computacional e valor de aptidão (*fitness*). Para a fase da resolução do LINGO são obtidos os resultados: número de

iterações, tempo computacional e valor da função objetivo. Vale ressaltar que esta fase aplica o modelo discreto em PLIM no qual os resultados obtidos pelo algoritmo ProtoG são introduzidos como dados do problema (Seção 4.3). Da mesma forma como foi efetuado para a metodologia AGEEH – PLIM, foram salvos os resultados para todos os indivíduos encontrados com valor menor que 6,9, obtendo-se desta forma 405 resultados.

A Figura 46 ilustra a tela de interface com o usuário do programa computacional desenvolvido em VB para esta abordagem.

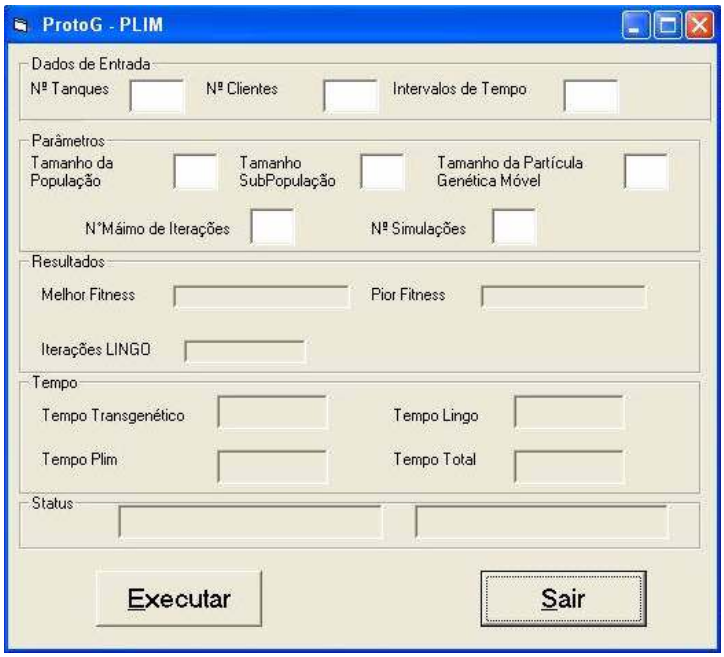

Figura 46: Formulário de interface do programa Algoritmo ProtoG - PLIM

Os resultados das rodadas de testes foram sintetizados para populações com tamanho igual a 20, 30 e 40 indivíduos. Os outros parâmetros variáveis que foram combinados com o tamanho da população são:

- Tamanho da Sub-população: 30% do tamanho da população;
- Tamanho da cadeia do vetor transgenético (partícula genética móvel): 5, 6 e 7;

Para todas as combinações dos parâmetros foram efetuadas 15 rodadas de testes. Estes resultados podem ser encontrados nas Tabelas do Anexo 2. Na Tabela 10 são apresentadas estatísticas descritivas do número de iterações do algoritmo ProtoG, do número de iterações do LINGO, do tempo computacional e da função objetivo resultantes dos 405 resultados obtidos nas 135 rodadas de testes.

Tabela 10: Estatísticas descritivas dos testes da metodologia algoritmo ProtoG – PLIM

Intervalos da função f	Nº de Iterações do ProtoG	Nº de Iterações do LINGO	Tempo Computacional (s)	Valor da Função de Aptidão
$f = 6,285$	147 ± 86,5	753340 ± 464587	888 ± 537	6,285 ± 0
$6,285 < f \leq 6,47$	108 ± 62	565021 ± 360228	660 ± 375	6,354 ± 0,053
$6,47 < f \leq 6,66$	85 ± 51	442470 ± 252712	515 ± 288	6,521 ± 0,025
$6,66 < f \leq 6,9$	74 ± 50	399271 ± 253122	463 ± 284	6,77 ± 0,069

Média ± desvio padrão

O resultado para a média do tempo computacional expresso em minutos foi de 11 minutos e 34 segundos com desvio padrão de 7 minutos e 26 segundos. Através das médias do número de iterações do LINGO e tempo computacional do programa desenvolvido calculou-se o desempenho de aproximadamente 854 iterações do LINGO por segundo.

A Tabela 11 mostra as médias do número de iterações do Algoritmo ProtoG, do número de iterações do LINGO e do tempo computacional para cada combinação dos parâmetros tamanho da população, tamanho da sub-população e tamanho da partícula genética móvel, quando o resultado ótimo igual a 6,285 foi encontrado. O melhor desempenho em relação a tempo computacional pôde ser observado para tamanho da população igual a 20, tamanho da sub-população igual a 6 e tamanho da partícula genética móvel igual a 6.

Tabela 11: Média para o resultado ótimo da metodologia Algoritmo ProtoG – PLIM

Tamanho População	Tamanho Sub-População	Tamanho da Partícula Genética Móvel	Nº de Iterações do ProtoG	Nº de Iterações do LINGO	Tempo Computacional (segundos)
20	6	5	207	706078	804,4
20	6	6	107	382817	467,8
20	6	7	123	445386	539,9
30	9	5	188	973764	1131,5
30	9	6	147	781346	929,7
30	9	7	111	616410	745,9
40	12	5	205	1414862	1614,7
40	12	6	119	837106	1006,5
40	12	7	73	535616	655,3

A Figura 47 ilustra o comportamento do número de iterações do Algoritmo ProtoG em relação ao valor da função de aptidão.

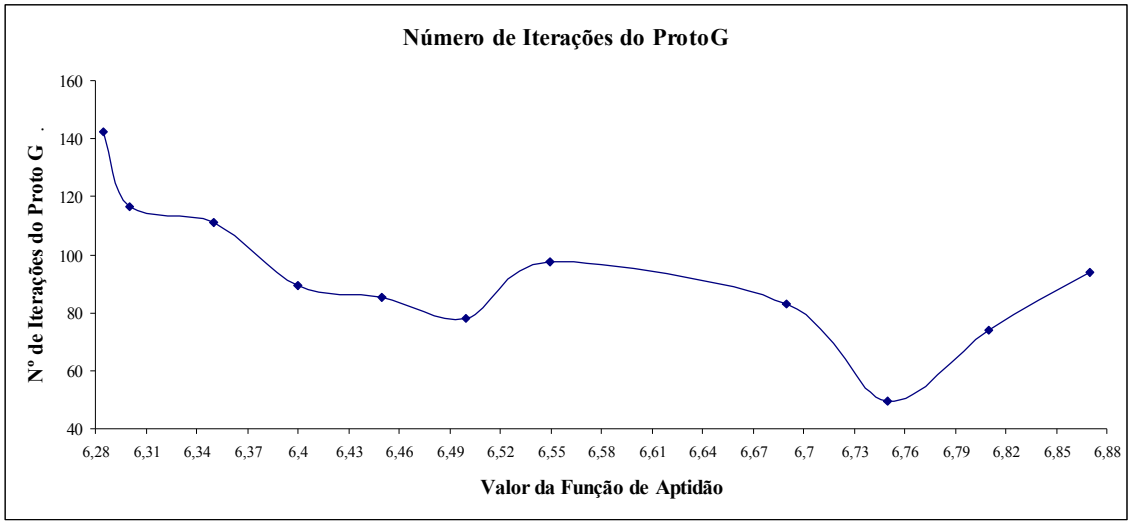

Figura 47: Número de Iterações do ProtoG de acordo com o valor da função de aptidão

Para ilustrar o número de iterações do LINGO de acordo com o valor da função de aptidão construiu-se o gráfico da Figura 48 a seguir.

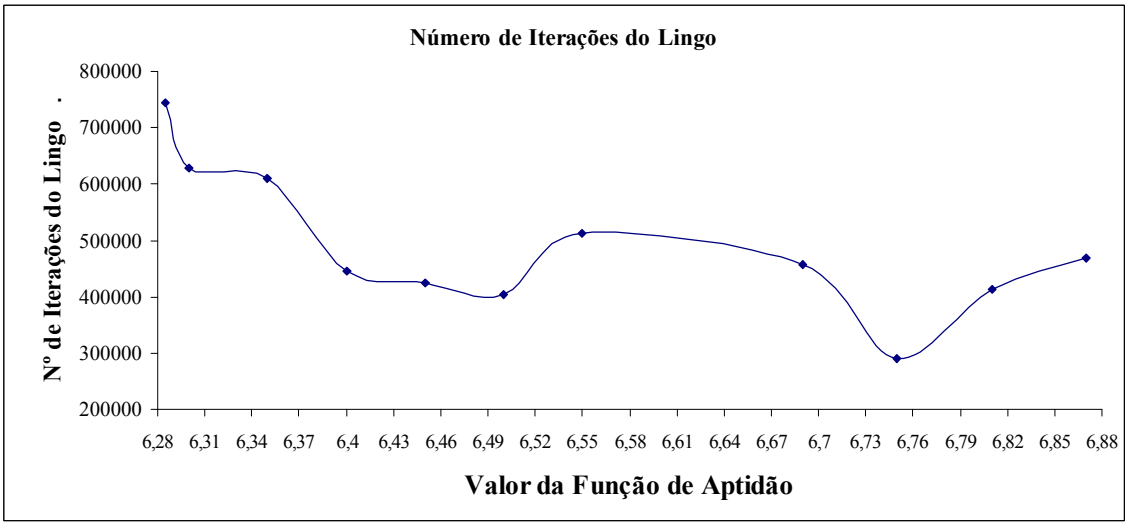

Figura 48: Número de Iterações do LINGO de acordo com o valor da função de aptidão

Finalmente, a Figura 49 ilustra o tempo computacional (em segundos) em relação ao valor de aptidão.

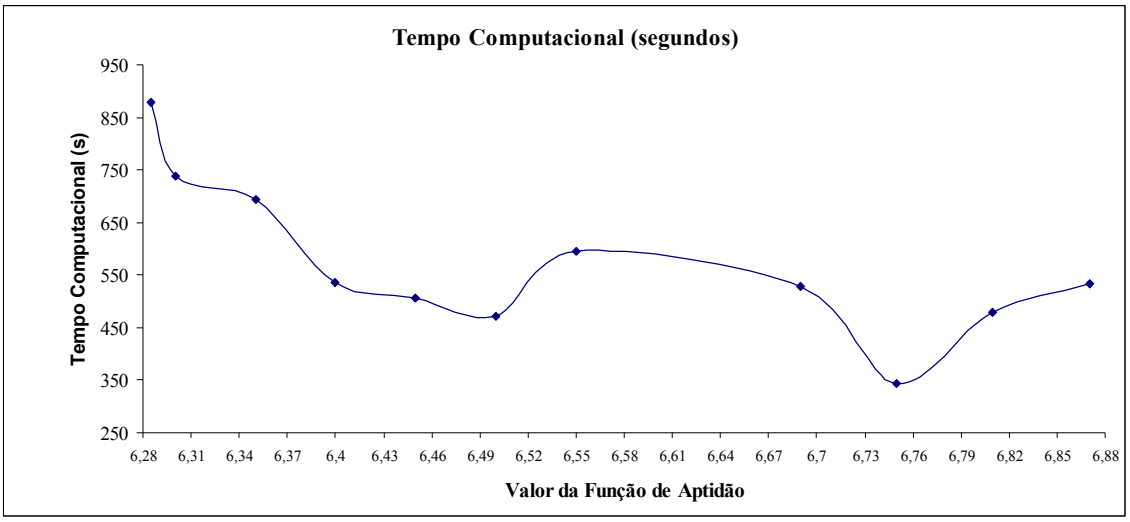

Figura 49: Tempo computacional de acordo com o valor da função de aptidão

Nas Figuras 47, 48 e 49 observa-se uma semelhança no comportamento dos resultados para número de iterações do ProtoG, número de iterações do LINGO e tempo computacional em relação ao valor da função de aptidão. Ainda, estes gráficos mostram uma pequena variabilidade e pode ser observada uma região com valores maiores entre os valores 6,52 e 6,7 e também para valores próximos de 6,28. Nestas regiões, quando o algoritmo alcançou mínimos locais, um maior número de iterações do algoritmo ProtoG, maior número de iterações do LINGO e maior tempo computacional foram despendidos, gerando um esforço computacional adicional para a busca pelo ponto mínimo.

Foi efetuada uma análise dos resultados com o objetivo de investigar a influência do tamanho da população e do tamanho da partícula genética móvel sobre os resultados do número de iterações do Algoritmo ProtoG, número de iterações do LINGO e tempo computacional, quando se atinge o resultado ótimo igual a 6,285.

Inicialmente, estimou-se os coeficientes de correlação de Pearson entre as variáveis número de iterações do Algoritmo ProtoG, número de iterações do LINGO e tempo computacional, duas a duas e testou-se a hipótese nula de ausência de correlação versus a hipótese alternativa de existência de correlação. Os resultados indicaram significância estatística no nível de 0,05. Os coeficientes de correlação *r* evidenciaram uma forte associação entre número de iterações do ProtoG e número de iterações do LINGO (*r = 0,8601*), entre número de iterações do ProtoG e tempo computacional (*r = 0,8552*) e entre número de iterações do LINGO e tempo computacional (*r = 0,9971*). Sendo assim, a análise foi efetuada somente para o número de iterações do Algoritmo ProtoG.

Considerando-se os níveis 20, 30 e 40 para o fator tamanho da população e os níveis 5, 6 e 7 para o tamanho da partícula genética móvel, efetuou-se uma análise de variância para avaliar a influência dos fatores sobre o número médio de iterações do ProtoG para atingir o resultado ótimo. Inicialmente, testou-se a hipótese nula de inexistência de interação entre o tamanho da população e o tamanho da partícula genética móvel. O resultado indicou que não há esta interação ($p = 0,5371$). Em seguida, ao testar-se a hipótese nula de médias iguais para os três níveis do tamanho da população, verificou-se que não existe diferença significativa entre eles ($p = 0,2618$). Para o tamanho da partícula genética móvel, houve diferença significativa entre os três níveis utilizados quanto ao número de iterações do ProtoG ($p < 0,0001$). Sendo assim, no nível de significância de 0,05, pode-se afirmar que existe diferença estatisticamente significativa entre os níveis 5, 6 e 7 do tamanho da partícula genética móvel, em relação ao número médio de iterações do ProtoG para atingir o resultado ótimo.

O comportamento semelhante para o número de iterações do Algoritmo ProtoG, número de iterações do LINGO e tempo computacional para as combinações de tamanho da população e tamanho da partícula genética móvel pode ser observado nos gráficos das Figuras 50, 51 e 52, construídos a partir das Tabelas 12, 13, e 14, das médias de todos os resultados para cada uma destas variáveis. Os gráficos confirmam o resultado do coeficiente de correlação encontrado. Para o número de iterações do Algoritmo ProtoG (Tabela 12 e Figura 50) observa-se que a combinação dos parâmetros tamanho da população igual a 40 e tamanho da partícula genética móvel igual a 7 foi a que mostrou melhor desempenho. Ainda, para número de iterações do LINGO e tempo computacional (Tabelas 13 e 14 e Figuras 51 e 52), o melhor desempenho pode ser observado para tamanho da população igual a 20 e tamanho da partícula genética móvel igual a 6.

Tabela 12: Média para o número de iterações do algoritmo ProtoG de acordo com o tamanho da população e o tamanho da partícula genética móvel

Tamanho da População	Tamanho da Partícula Genética Móvel		
	5	6	7
20	162	90	106
30	143	113	76
40	147	92	59

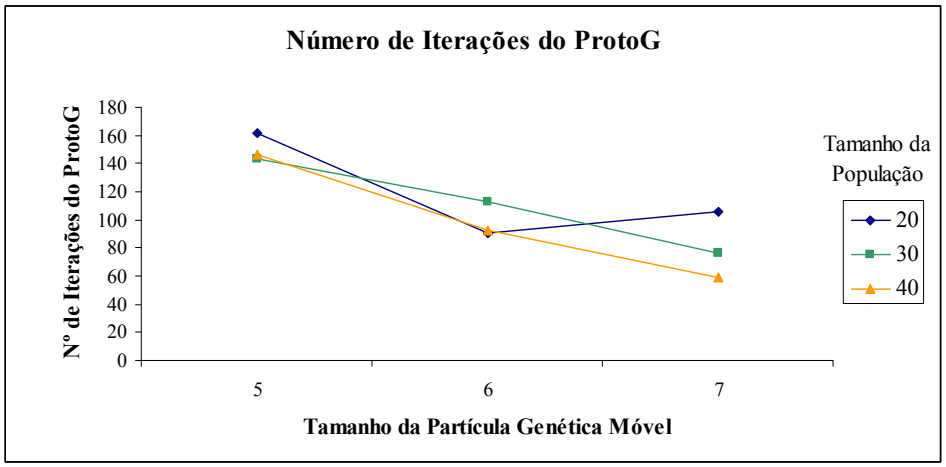

Figura 50: Número de iterações do ProtoG de acordo com o tamanho da população e tamanho da partícula genética móvel

Tabela 13: Média para o número de iterações do LINGO de acordo com o tamanho da população e o tamanho da partícula genética móvel

Tamanho da População	Tamanho da Partícula Genética Móvel		
	5	6	7
20	557217	321914	387885
30	746613	603928	425000
40	1039889	654927	436794

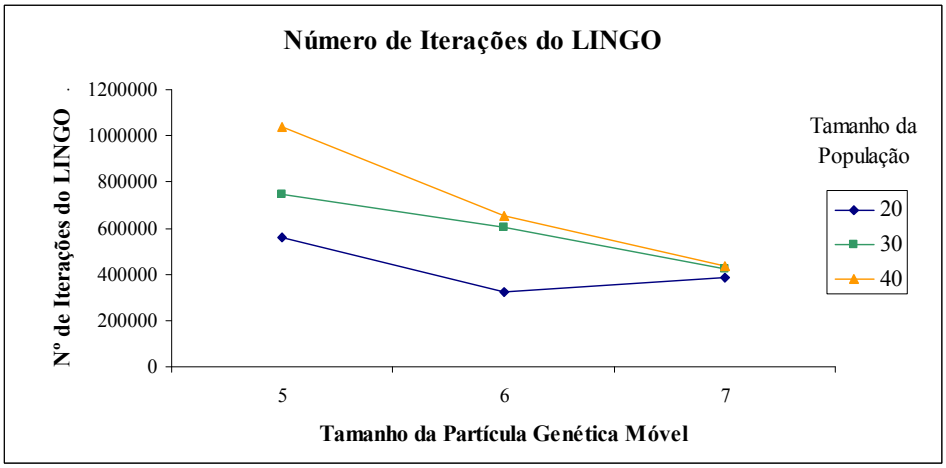

Figura 51: Número de iterações do LINGO de acordo com o tamanho da população e do tamanho partícula genética móvel

Tabela 14: Média para o tempo computacional de acordo com o tamanho da população e do tamanho partícula genética móvel

Tamanho da População	Tamanho da Partícula Genética Móvel		
	5	6	7
20	631,3	389,0	471,8
30	867,0	716,5	515,0
40	1166,2	784,8	537,5

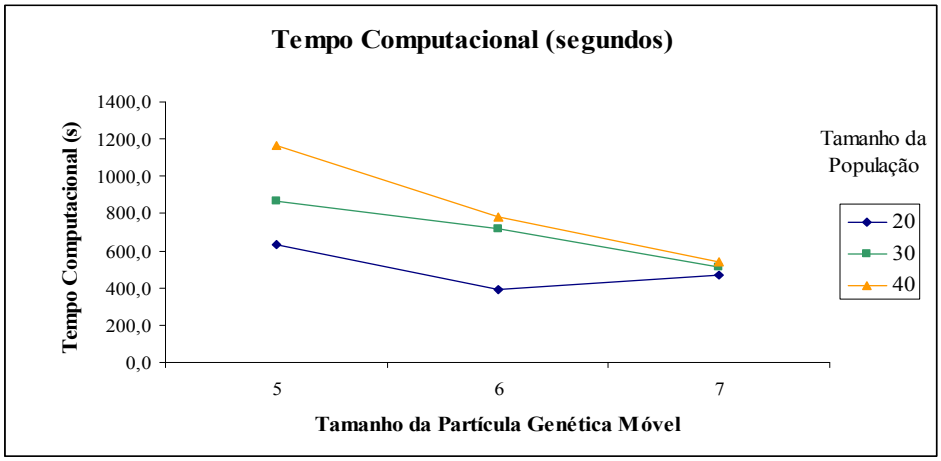

Figura 52: Tempo computacional de acordo com o tamanho da população e do tamanho partícula genética móvel

6.3 IMPLEMENTAÇÃO PARA O MODELO COM REPRESENTAÇÃO DE TEMPO CONTÍNUO

Para o modelo com representação de tempo contínuo, descrito na Seção 3.3 foram efetuados três grupos de testes distintos para a obtenção de resultados: resolução do modelo em PLIM utilizando o aplicativo LINGO 8.0, aplicação da metodologia AG baseado em simulação e algoritmo ProtoG através da simulação. Para a obtenção dos resultados foi utilizado um computador com processador 950 *MHz* e 128 *MB* de memória *RAM*.

6.3.1 Resultados para a modelagem PLIM com representação de tempo contínuo

Para a implementação da modelagem PLIM com representação de tempo contínuo foram utilizados os seguintes dados:

- Número de tanques: $TQ = 5$;
- Número de clientes: $C = 2$;
- Número de produtos: $D = 2$;
- Número de processos: $P = 2$;
- Número de eventos: $E = 6$;
- Tempo para certificação do produto após o enchimento do tanque: $TC = 4$h;
- Vazão de recebimento pelos tanques: $QR_1 = 0{,}7$ mil m^3/h, $QR_2 = 1{,}8$ mil m^3/h;
- Vazão de envio ao cliente c; $QE_1 = 1$ mil m^3/h, $QE_2 = 0{,}6$ mil m^3/h;
- Demanda de produto do tipo d do cliente c: $DEM_{1,1} = 30$ mil m^3, $DEM_{1,2} = 15$ mil m^3, $DEM_{2,1} = 15$ mil m^3 e $DEM_{2,2} = 15$ mil m^3;
- Volume mínimo permitido no tanque tq; $Volmin_1 = 1$ mil m^3, $Volmin_2 = 1$ mil m^3, $Volmin_3 = 1$ mil m^3, $Volmin_4 = 1$ mil m^3 e $Volmin_5 = 1$ mil m^3;
- Volume máximo permitido no tanque tq; $Volmax_1 = 16$ mil m^3, $Volmax_2 = 16$ mil m^3, $Volmax_3 = 16$ mil m^3, $Volmax_4 = 16$ mil m^3 e $Volmax_5 = 16$ mil m^3;
- Volume no tanque tq no início do processo; $Volini_1 = 7$ mil m^3, $Volini_2 = 1$ mil m^3, $Volini_3 = 1$ mil m^3, $Volini_4 = 16$ mil m^3 e $Volini_5 = 1$ mil m^3;
- Lote do produto d disponível do processo p. $LoteProc_{1,1} = 60$ mil m^3, $LoteProc_{1,2} = 15$ mil m^3, $LoteProc_{2,1} = 0$ mil m^3 e $LoteProc_{2,2} = 30$ mil m^3;
- Disponibilidade do tanque para determinado produto - assume valor um se o tanque recebe o produto e zero caso contrário: $TqProd_{1,1} = 1$, $TqProd_{1,2} = 0$, $TqProd_{2,1} = 1$, $TqProd_{2,2} = 0$, $TqProd_{3,1} = 1$, $TqProd_{3,2} = 0$, $TqProd_{4,1} = 0$, $TqProd_{4,2} = 1$, $TqProd_{5,1} = 0$ e $TqProd_{5,2} = 1$;

O modelo em PLIM descrito na Seção 3.3 usando os dados acima apresentou 611 variáveis contínuas, 162 inteiras e um total de 1192 restrições. Foi efetuada apenas uma rodada de teste com este modelo até encontrar-se o valor ótimo igual a 77,2857. O tempo computacional despendido para a resolução desta única rodada de teste pelo LINGO 8.0 foi 61 horas e 5 minutos e o número de iterações efetuadas foi de 144.677.965.

Para melhor compreensão do resultado encontrado foi elaborada uma carta de Gantt ilustrando as operações dos tanques, do processo e dos clientes em cada evento de tempo (Figura 53).

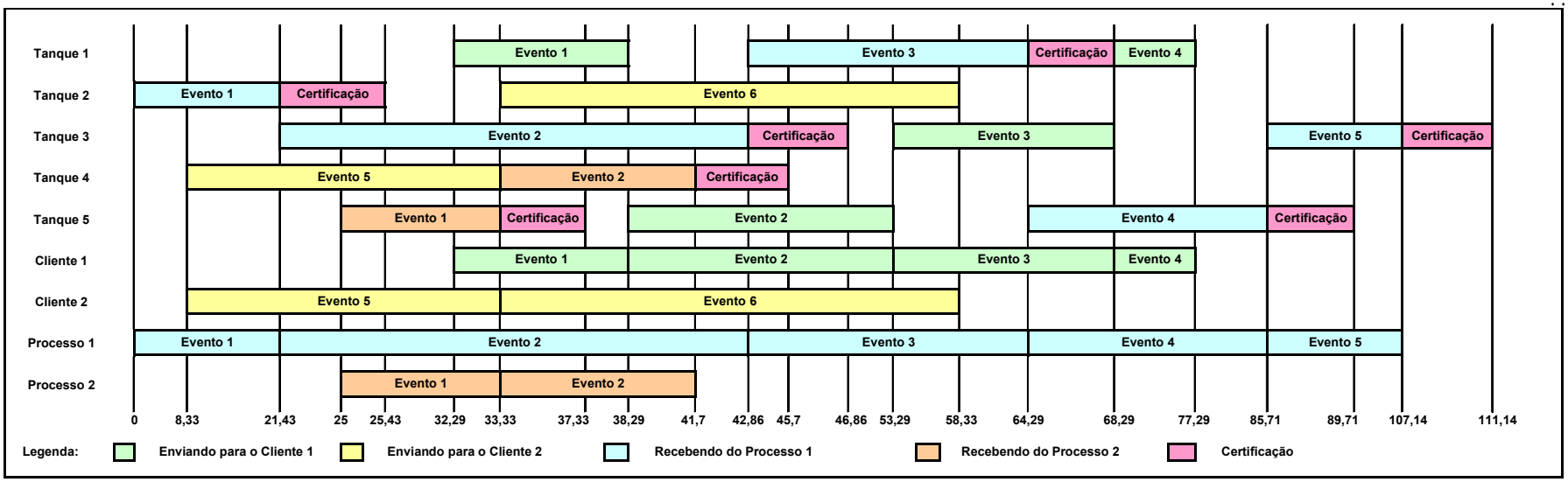

Figura 53: Carta de Gantt – resultado do modelo PLIM com representação contínua do tempo

As Figuras 54, 55, 56, 57 e 58 ilustram graficamente a variação do volume dos tanques.

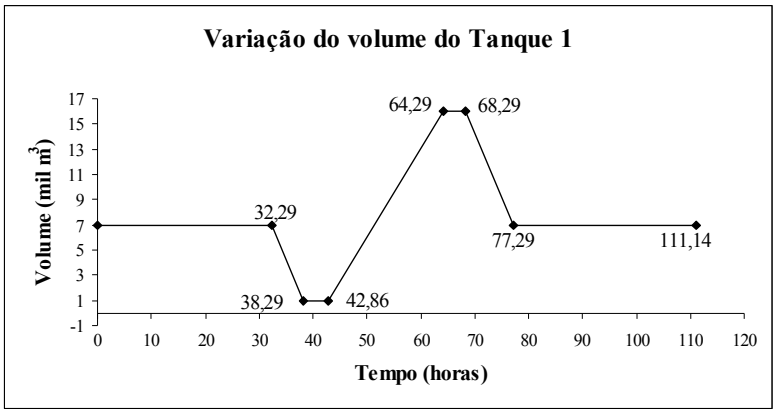

Figura 54: Variação do volume do tanque 1 (PLIM com tempo contínuo)

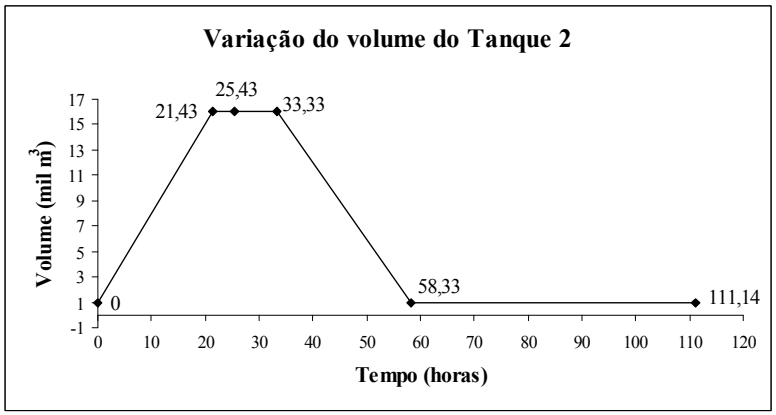

Figura 55: Variação do volume do tanque 2 (PLIM com tempo contínuo)

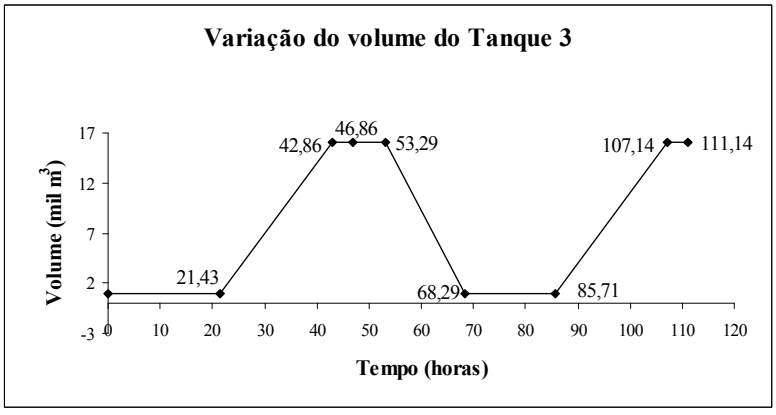

Figura 56: Variação do volume do tanque 3 (PLIM com tempo contínuo)

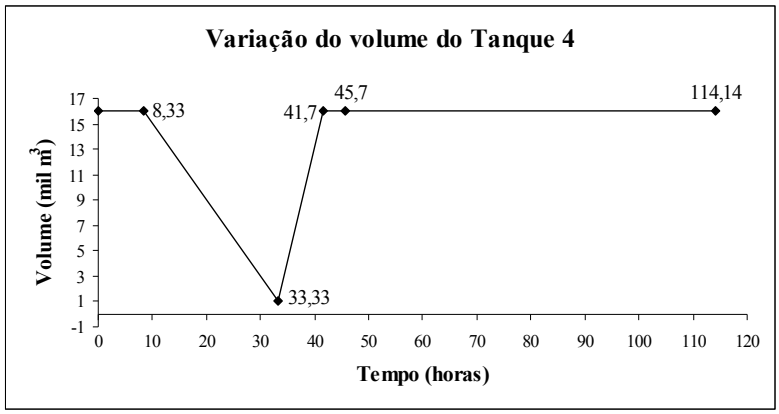

Figura 57: Variação do volume do tanque 4 (PLIM com tempo contínuo)

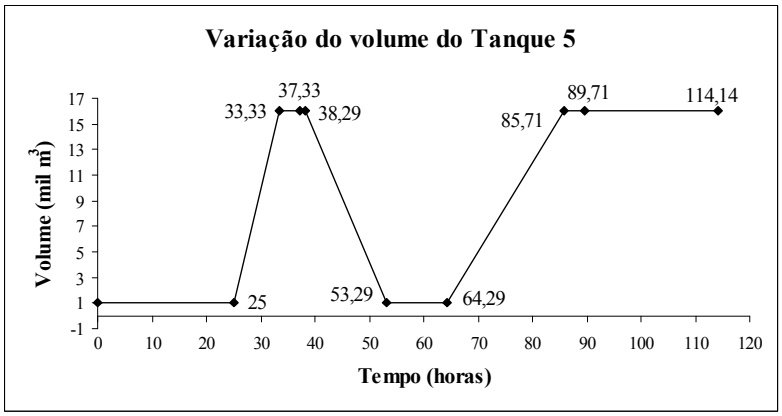

Figura 58: Variação do volume do tanque 5 (PLIM com tempo contínuo)

6.3.2 Resultados para a metodologia AG – Simulação

Para a abordagem AG – Simulação (AG baseado em Simulação) foram utilizados dados para duas instâncias distintas. As rodadas de testes foram efetuadas variando-se os parâmetros: tamanho da população, probabilidade de recombinação, probabilidade de mutação e número máximo de iterações para o AG. Os resultados encontrados ao final do processamento do programa são: valor de aptidão (*fitness*) e configuração do melhor individuo da população, tempo computacional total, tempo computacional para encontrar o resultado ótimo (se o mesmo for encontrado) e valores para as variáveis $XI_{tq,c,e}$ e $XF_{tq,c,e}$, lembrando que o índice c representa os clientes e os processos da seguinte forma: os índices de *1* a *C* são aplicados aos clientes e os índices de $C + 1$ a $C + P$ são utilizados para os processos.

A Figura 59 ilustra a tela de interface com o usuário do programa computacional desenvolvido em VB para esta abordagem.

Figura 59: Formulário de interface do programa AG - Simulação

6.3.2.1 Resultados para a 1ª instância da metodologia AG - Simulação

Esta instância foi aplicada com os mesmos dados utilizados para a modelagem PLIM com representação de tempo contínuo, detalhados na Seção 6.3.1, excetuando apenas o número de eventos que, nesta abordagem, é encontrado durante o teste. Foram efetuadas 90 rodadas de testes, variando-se o tamanho da população nos níveis 40, 50 e 60, fixando-se o número de iterações do AG em 100, a probabilidade de recombinação em 0,9 e a probabilidade de mutação em 0,1. Observou-se que, em todas as rodadas de testes, na geração da população, foi obtido o resultado ótimo igual a 77,286, não havendo necessidade de processamento do AG. Os tempos computacionais médios para 100 iterações apresentados na Tabela 15 indicam um acréscimo com o aumento do tamanho da população. Quanto ao tempo para atingir o resultado ótimo observa-se um acréscimo seguido de um decréscimo na média.

Tabela 15: Tempo computacional médio de acordo com os tamanhos da população

Tamanho da população	Tempo computacional médio (segundos)	Tempo computacional médio para atingir o resultado ótimo (segundos)
40	12,9	3,2
50	20,1	3,4
60	31,3	2,8

Uma das configurações encontradas para este resultado ótimo e os valores correspondentes para as variáveis $XI_{tq,c,e}$ e $XF_{tq,c,e}$, são:

C2t4p1t2p1C1t1p1t5p2t3p1t1p1

XI(4,2,1) = 0,43	e	XF(4,2,1) = 25,43
XI(2,3,2) = 0	e	XF(2,3,2) = 21,43
XI(2,2,3) = 25,43	e	XF(2,2,3) = 50,43
XI(1,1,4) = 32,29	e	XF(1,1,4) = 38,29
XI(5,4,5) = 25	e	XF(5,4,5) = 33,33
XI(5,1,6) = 38,29	e	XF(5,1,6) = 53,29
XI(3,3,7) = 21,43	e	XF(3,3,7) = 42,86
XI(3,1,8) = 53,29	e	XF(3,1,8) = 68,29
XI(1,3,9) = 42,86	e	XF(1,3,9) = 64,29
XI(1,1,10) = 68,29	e	XF(1,1,10) = 77,29
XI(4,4,11) = 33,33	e	XF(4,4,11) = 41,67
XI(2,3,12) = 64,29	e	XF(2,3,12) = 85,71
XI(5,3,13) = 85,71	e	XF(5,3,13) = 107,14

6.3.2.2 Resultados para a 2ª instância da metodologia AG - Simulação

Os dados para esta instância são:

- Número de tanques: $TQ = 13$;
- Número de clientes: $C = 4$;
- Número de produtos: $D = 2$;
- Número de processos: $P = 2$;
- Tempo para certificação do produto após o enchimento do tanque: $TC = 4$h;
- Vazão de recebimento pelos tanques: $QR_1 = 0,7$ mil m^3/h, $QR_2 = 1,8$ mil m^3/h;
- Vazão de envio ao cliente c; $QE_1 = 1$ mil m^3/h, $QE_2 = 0,6$ mil m^3/h, $QE_3 = 0,7$ mil m^3/h, $QE_4 = 0,8$ mil m^3/h;

- Demanda de produto do tipo *d* do cliente *c*: $DEM_{1,1}$ = 30 mil m^3, $DEM_{1,2}$ = 15 mil m^3, $DEM_{2,1}$ = 15 mil m^3, $DEM_{2,2}$ = 15 mil m^3, $DEM_{3,1}$ = 45 mil m^3, $DEM_{3,2}$ = 0 mil m^3, $DEM_{4,1}$ = 30 mil m^3, $DEM_{4,2}$ = 15 mil m^3;
- Volume mínimo permitido no tanque *tq*; $Volmin_{tq}$ = 1 mil m^3 $\forall$ *tq = 1,...,13*
- Volume máximo permitido no tanque *tq*; $Volmax_{tq}$ = 16 mil m^3, $\forall$ *tq = 1,...,13* ;
- Volume no tanque *tq* no início do processo; $Volini_1$ = 7 mil m^3, $Volini_2$ = 1 mil m^3, $Volini_3$ = 1 mil m^3, $Volini_4$ = 16 mil m^3, $Volini_5$ = 1 mil m^3, $Volini_6$ = 1 mil m^3, $Volini_7$ = 1 mil m^3, $Volini_8$ = 1 mil m^3, $Volini_9$ = 8 mil m^3, $Volini_{10}$ = 1 mil m^3, $Volini_{11}$ = 1 mil m^3, $Volini_{12}$ = 16 mil m^3 e $Volini_{13}$ = 16 mil m^3;
- Lote do produto *d* disponível do processo *p*. $LoteProc_{1,1}$ = 90 mil m^3, $LoteProc_{1,2}$ = 15 mil m^3, $LoteProc_{2,1}$ = 30 mil m^3 e $LoteProc_{2,2}$ = 60 mil m^3;

Disponibilidade do tanque para determinado produto - assume valor um se o tanque recebe o produto e zero caso contrário:

$TqProd_{i,1}$ = 1 com $TqProd_{i,2}$ = 0 para os tanques *i* = 1, 2, 3, 4, 7, 8, 13;

$TqProd_{i,1}$ = 0 com $TqProd_{i,2}$ = 1 para os tanques *i* = 5, 6, 9, 10, 11, 12;

As rodadas de testes foram efetuadas variando-se os parâmetros:

- tamanho da população: 40, 50 e 60;
- número de iterações do AG: 2000 e 5000;
- probabilidade de recombinação: 0,8, 0,85 e 0,9;
- probabilidade de mutação: 0,1, 0,15 e 0,2.

Cada uma das 54 combinações destes parâmetros foi rodada por 15 vezes gerando um total de 810 resultados que incluem o tempo computacional e o valor da função de aptidão. Na fase de testes, inicialmente, foram efetuadas rodadas preliminares para testar-se o funcionamento do programa desenvolvido para esta metodologia. Observou-se o resultado da função de aptidão igual a 120,143 como sendo o menor valor encontrado. Com o objetivo de observar a ocorrência deste valor, incluiu-se na programação desta metodologia uma condição de que o número de iterações e tempo computacional fossem salvos em arquivo de texto, quando este valor fosse encontrado pela primeira vez. A Tabela 16, mostra a média das 15 rodadas de testes efetuadas para as 54 combinações distintas destes parâmetros.

Tabela 16: Média dos resultados da metodologia AG - Simulação

Tamanho da População	Probabilidade de Recombinação	Probabilidade de Mutação	Nº de Iterações do AG	Tempo Computacional Médio (segundos)	Valor Médio da Função de Aptidão	Nº de Iterações Médio do AG para Resultado Mínimo	Tempo Computacional Médio para o resultado Mínimo
40	0,8	0,1	2000	55,1	126,694	1008	38,3
40	0,8	0,1	5000	96,1	124,683	1285	41,5
40	0,8	0,15	2000	67,8	126,564	-	-
40	0,8	0,15	5000	114,4	124,133	263	28,0
40	0,8	0,2	2000	66,1	125,498	-	-
40	0,8	0,2	5000	157,5	123,586	416	35,3
40	0,85	0,1	2000	44,9	127,664	1656	39,0
40	0,85	0,1	5000	102,5	127,271	520	32,0
40	0,85	0,15	2000	72,7	124,379	512	35,0
40	0,85	0,15	5000	128,3	124,936	2175	70,5
40	0,85	0,2	2000	66,1	125,498	-	-
40	0,85	0,2	5000	185,0	124,295	1255	51,0
40	0,9	0,1	2000	54,1	124,129	243	24,0
40	0,9	0,1	5000	99,6	123,229	1563	47,3
40	0,9	0,15	2000	60,1	123,186	1067	117,8
40	0,9	0,15	5000	112,9	124,550	1073	39,0
40	0,9	0,2	2000	80,4	125,286	-	-
40	0,9	0,2	5000	140,1	124,100	847	51,0
50	0,8	0,1	2000	58,0	125,921	-	-
50	0,8	0,1	5000	108,6	124,829	1715	54,7
50	0,8	0,15	2000	72,3	125,356	826	48,0
50	0,8	0,15	5000	131,3	123,071	937	48,8
50	0,8	0,2	2000	77,4	123,855	411	40,7
50	0,8	0,2	5000	150,5	123,855	-	-
50	0,85	0,1	2000	60,5	125,457	744	37,3
50	0,85	0,1	5000	109,1	125,412	320	34,0
50	0,85	0,15	2000	72,4	124,664	875	49,0
50	0,85	0,15	5000	134,9	123,691	2079	71,3
50	0,85	0,2	2000	77,1	124,133	480	36,0
50	0,85	0,2	5000	152,0	123,893	2296	81,5
50	0,9	0,1	2000	59,5	124,379	775	41,5
50	0,9	0,1	5000	90,1	124,157	154	27,5
50	0,9	0,15	2000	73,6	124,344	152	26,3
50	0,9	0,15	5000	138,3	124,379	2188	72,5
50	0,9	0,2	2000	70,8	124,343	621	46,3
50	0,9	0,2	5000	151,0	123,687	264	37,0
60	0,8	0,1	2000	58,3	123,498	1062	42,7
60	0,8	0,1	5000	113,6	124,045	356	42,0
60	0,8	0,15	2000	76,7	124,186	478	48,3
60	0,8	0,15	5000	123,7	124,405	894	63,0
60	0,8	0,2	2000	87,5	124,429	-	-
60	0,8	0,2	5000	160,5	127,612	1398	68,0
60	0,85	0,1	2000	63,4	124,019	1152	49,3
60	0,85	0,1	5000	118,5	124,336	547	46,3
60	0,85	0,15	2000	75,3	124,624	312	43,3
60	0,85	0,15	5000	133,1	123,033	1230	55,2
60	0,85	0,2	2000	88,5	124,500	-	-
60	0,85	0,2	5000	163,7	123,700	1974	85,0
60	0,9	0,1	2000	63,9	123,007	662	40,0
60	0,9	0,1	5000	108,0	125,543	777	46,5
60	0,9	0,15	2000	79,5	124,879	925	64,0
60	0,9	0,15	5000	128,9	123,600	126	36,0
60	0,9	0,2	2000	84,7	123,836	444	49,7
60	0,9	0,2	5000	173,2	123,993	4335	221,0

Para os 810 resultados foram calculadas estatísticas descritivas para o tempo computacional e para o valor da função de aptidão. A Tabela 17 mostra estes resultados.

Tabela 17: Estatística descritiva para todas os testes da metodologia AG - Simulação

	Tempo Computacional (segundos)	Valor da Função de Aptidão
Mínimo	38	120,143
Máximo	318	165
Média	99,857	124,562
Mediana	89	124
Desvio padrão	41,346	3,529

As estatísticas descritivas dos 114 (14,1%) testes que resultaram no valor mínimo foram calculadas para os resultados do tempo computacional, do número de iterações do AG e do tempo computacional para o resultado mínimo.

Tabela 18: Estatísticas descritivas para o melhor resultado da metodologia AG - Simulação

	Tempo Computacional (segundos)	Nº de Iterações do AG para melhor Resultado	Tempo Computacional (segundos) para melhor resultado
Mínimo	44,0	0	1,0
Máximo	263,0	4529	335,0
Média	99,1	1003	52,1
Mediana	94,5	612	43,0
Desvio padrão	40,0	1086	38,0

O gráfico da Figura 60 mostra o tempo computacional em relação ao valor da função de aptidão.

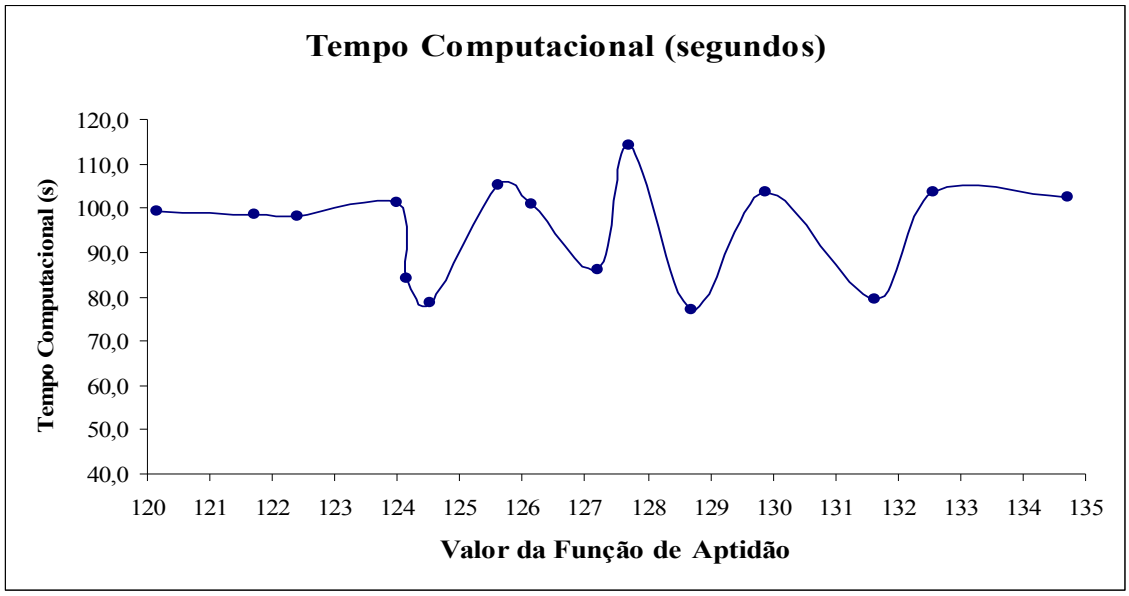

Figura 60: Tempo computacional de acordo com o valor da função de aptidão

O valor da função de aptidão de acordo com as probabilidades de recombinação e mutação e número de Iterações do AG pode ser analisado no gráfico da Figura 61.

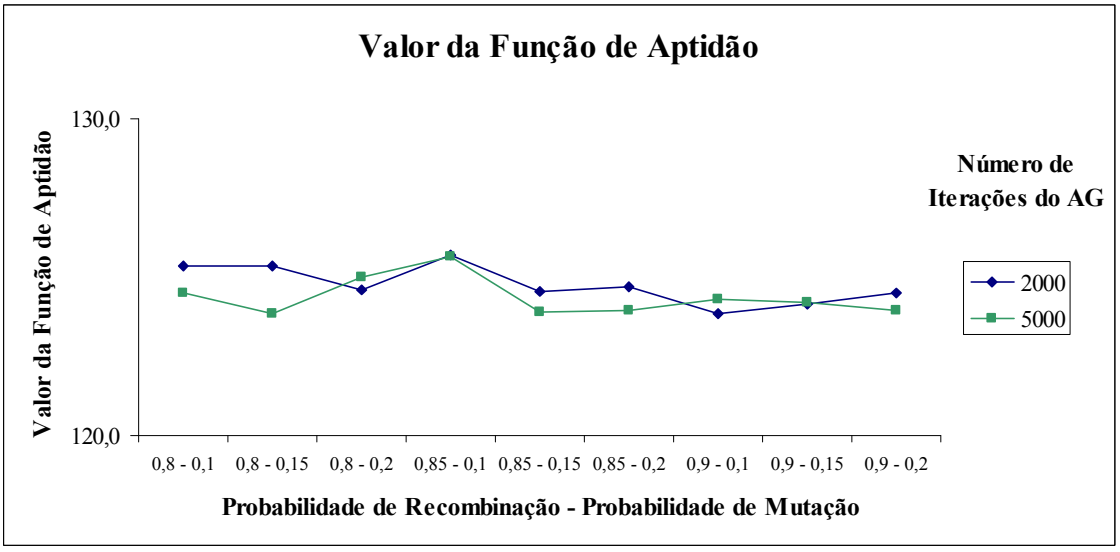

Figura 61: Valor da função de aptidão de acordo com as probabilidades de recombinação e mutação e número de iterações do AG

Ainda, na Figura 62 pode-se observar o valor da função de aptidão de acordo com as probabilidades de recombinação e mutação e tamanho da população.

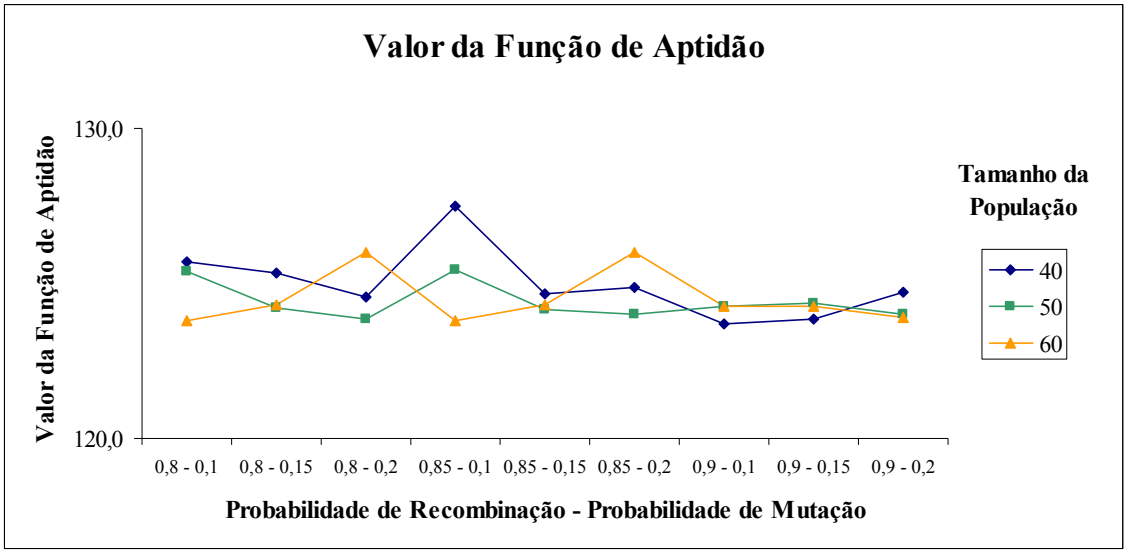

Figura 62: Valor da função de aptidão de acordo com as probabilidades de recombinação e mutação e tamanho da população

Com os 810 resultados obtidos investigou-se o efeito de cada um dos parâmetros tamanho da população, número de iterações do AG, probabilidades de recombinação e mutação e número de iterações para o AG sobre o tempo computacional. Para tanto, ajustou-se um modelo de regressão múltipla considerando os parâmetros como variáveis explicativas (independentes) e o tempo computacional como variável resposta (dependente).

O modelo ajustado apresentou um coeficiente de determinação igual a $R^2 = 0,7206$ indicando que os parâmetros considerados no modelo explicam 72,06% da variabilidade observada no tempo computacional. Quanto às variáveis explicativas o resultado do ajuste e os testes estatísticos efetuados com um nível de significância de 0,05 indicaram que são significantes os coeficientes de regressão das seguintes variáveis: tamanho da população ($p < 0,0001$), número de iterações do AG ($p < 0,0001$), probabilidade de mutação ($p < 0,0001$). A variável probabilidade de recombinação não se apresentou como sendo significativa ($p = 0,6645$). Desta forma variações nos parâmetros considerados (com exceção da probabilidade de recombinação) implicam significativamente em variações no tempo computacional.

As Tabelas 19 e 20 e os gráficos das Figuras 63 e 64 ilustram esta análise.

Tabela 19: Tempo computacional de acordo com a probabilidade de mutação e tamanho da população

	Probabilidade de Mutação		
Tamanho da População	0,1	0,15	0,2
40	75,4	92,7	115,9
50	81,0	103,8	113,1
60	87,6	102,9	126,4

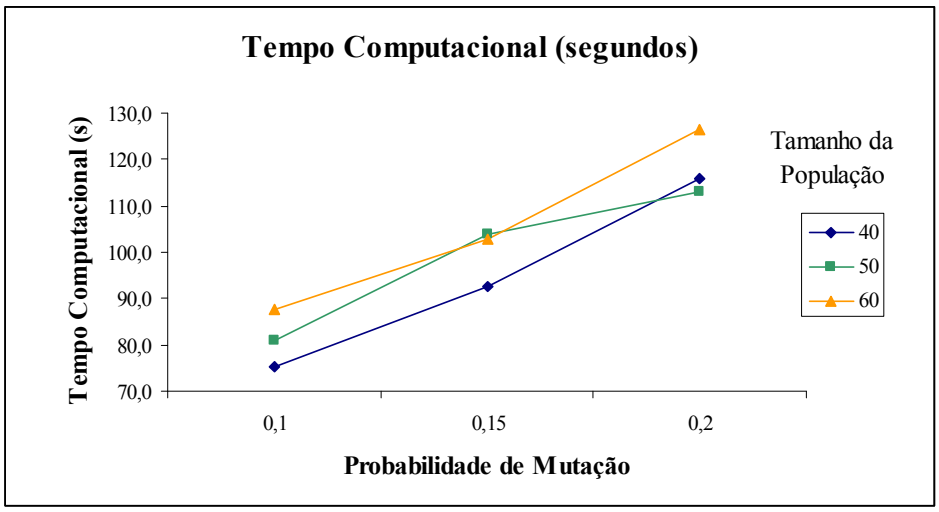

Figura 63: Tempo computacional de acordo com a probabilidade de mutação e tamanho da população

Tabela 20: Tempo computacional de acordo com o número de iterações do AG e a probabilidade de mutação

	Número de Iterações do AG	
Probabilidade de Mutação	2000	5000
0,1	57,5	105,1
0,15	72,3	127,3
0,2	77,6	159,3

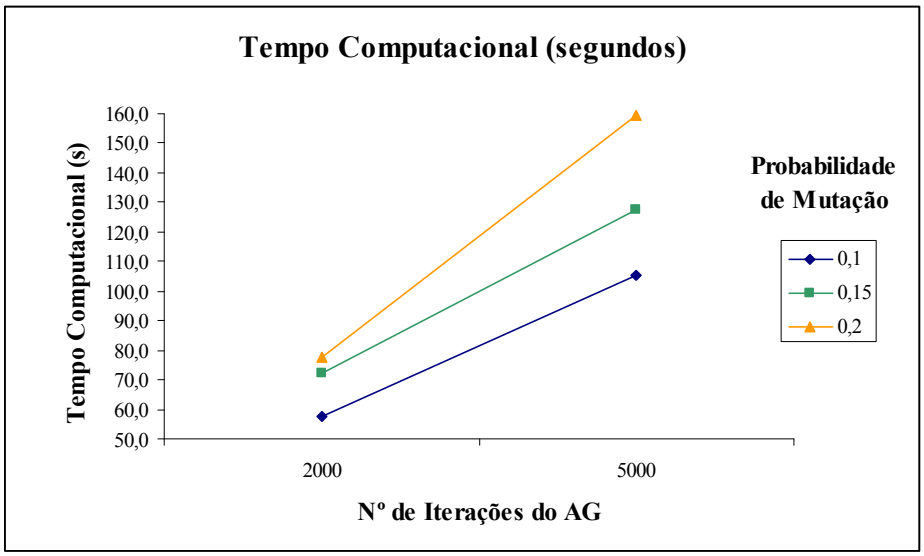

Figura 64: Tempo computacional de acordo com o número de iterações do AG e a probabilidade de mutação

Não se pode afirmar que o valor mínimo de 120,143 é o resultado ótimo, pois o problema não foi resolvido usando-se a modelagem PLIM com representação de tempo contínuo. Uma das configurações encontradas e os respectivos valores para as variáveis $XI_{tq,c,e}$ e $XF_{tq,c,e}$ para este resultado são:

LsInd(1)= C2t3p1t5p2C4t9p1t4p1t8p1t11p2C3t7p2t2p1t3p1C1t4p2t13p1t1p1t8p1

XI(3,5,1)= 0	e	XF(3,5,1)= 21,43
XI(3,2,2)= 25,43	e	XF(3,2,2)= 50,43
XI(5,6,3)= 25,00	e	XF(5,6,3)= 33,33
XI(5,2,4)= 50,43	e	XF(5,2,4)= 75,43
XI(9,4,5)= 19,36	e	XF(9,4,5)= 28,11
XI(4,4,6)= 28,11	e	XF(4,4,6)= 46,86
XI(8,5,7)= 21,43	e	XF(8,5,7)= 42,86
XI(8,4,8)= 46,86	e	XF(8,4,8)= 65,61
XI(11,6,9)= 33,33	e	XF(11,6,9)= 41,67
XI(11,4,10)= 65,61	e	XF(11,4,10)= 75,61
XI(7,6,11)= 41,67	e	XF(7,6,11)= 50,00
XI(7,3,12)= 54,00	e	XF(7,3,12)= 75,43
XI(2,5,13)= 42,86	e	XF(2,5,13)= 64,29
XI(2,3,14)= 75,43	e	XF(2,3,14)= 96,86
XI(3,5,15)= 64,29	e	XF(3,5,15)= 85,71
XI(3,3,16)= 96,86	e	XF(3,3,16)= 118,29
XI(4,6,17)= 50,00	e	XF(4,6,17)= 58,33
XI(4,1,18)= 75,14	e	XF(4,1,18)= 90,14
XI(13,1,19)= 90,14	e	XF(13,1,19)= 105,14

XI(1,1,20)= 105,14	e	XF(1,1,20)= 111,14
XI(8,5,21)= 85,71	e	XF(8,5,21)= 107,14
XI(8,1,22)= 111,14	e	XF(8,1,22)= 120,14
XI(6,6,23)= 58,33	e	XF(6,6,23)= 66,67
XI(9,6,24)= 66,67	e	XF(9,6,24)= 75,00
XI(2,5,25)= 107,14	e	XF(2,5,25)= 128,57
XI(5,5,26)= 128,57	e	XF(5,5,26)= 150,00

A Figura 65 mostra a carta de Gantt correspondente a esta configuração.

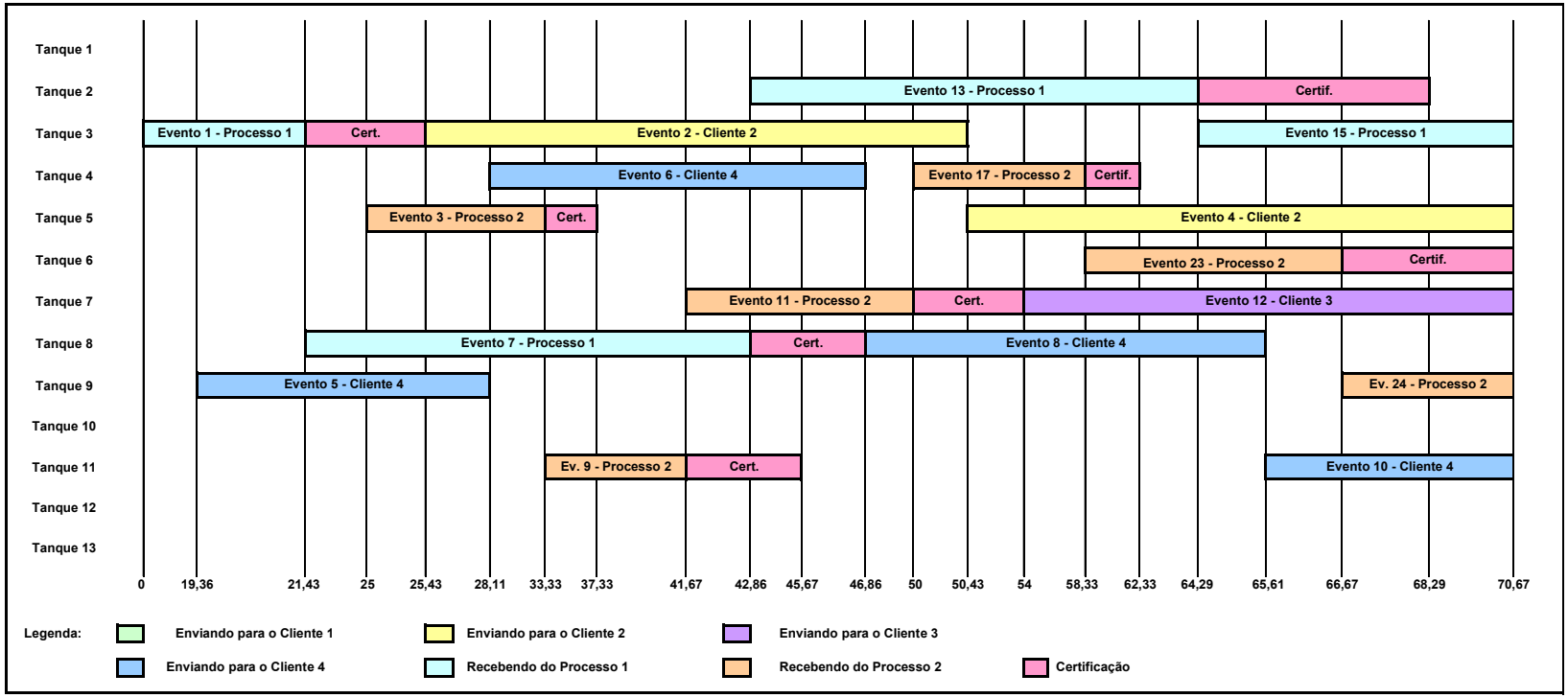

Figura 65: Carta de Gantt – resultado do modelo PLIM com representação contínua do tempo - 1ª parte

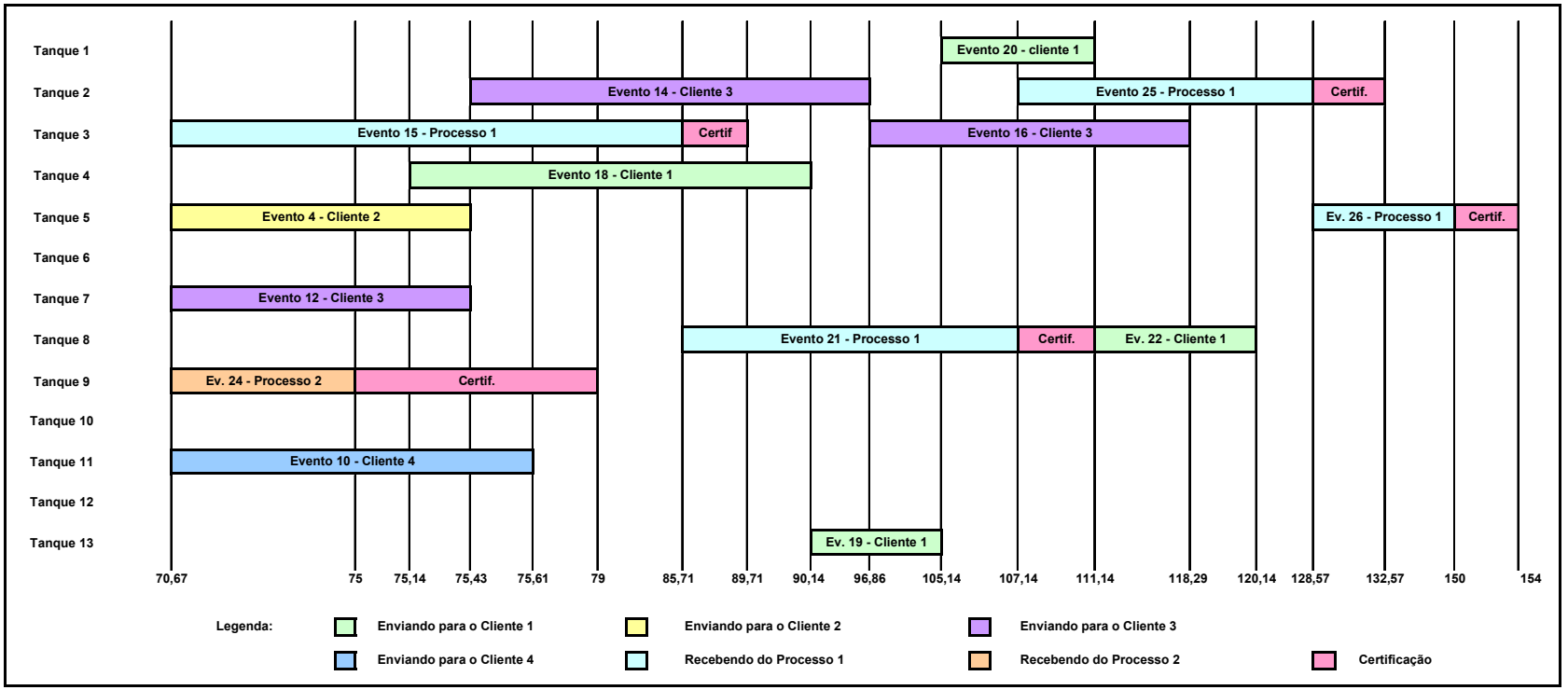

Figura 65: Carta de Gantt – resultado do modelo PLIM com representação contínua do tempo - 2ª parte

6.3.3 Resultados para a metodologia ProtoG – Simulação

Para a abordagem ProtoG – Simulação (Algoritmo ProtoG baseado em Simulação) foram utilizados dados para duas instâncias distintas. As rodadas de testes foram efetuadas variando-se os parâmetros: tamanho da população, tamanho da sub-população, número de partículas genéticas móveis por cliente e número de iterações do ProtoG. Os resultados encontrados ao final do processamento do programa são: valor de aptidão (*fitness*) e configuração do melhor indivíduo da população, tempo computacional total, tempo computacional para encontrar o resultado ótimo (se o mesmo for encontrado) e valores para as variáveis $XI_{tq,c,e}$ e $XF_{tq,c,e}$, lembrando que o índice c representa os processos e os clientes da seguinte forma: os índices de 1 a C são aplicados aos clientes e os índices de $C + 1$ a $C + P$ são utilizados para os processos.

A Figura 66 ilustra a tela de interface com o usuário do programa computacional desenvolvido em VB para esta abordagem.

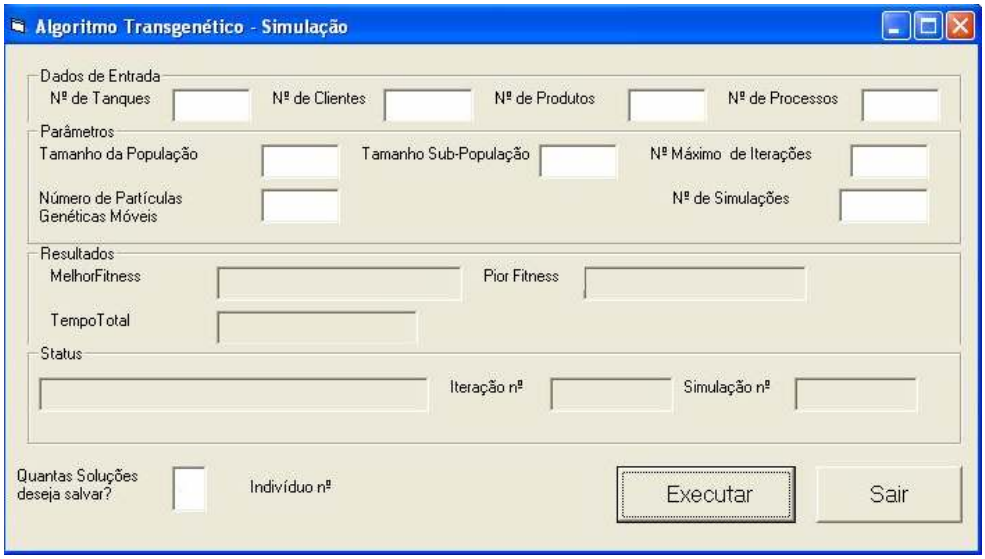

Figura 66: Formulário de interface do programa ProtoG - Simulação

6.3.3.1 Resultados para a 1ª instância da metodologia ProtoG - Simulação

Esta instância foi rodada da mesma forma que a primeira instância para a metodologia AG – Simulação. Foram efetuadas 90 rodadas de testes, variando-se o tamanho da população

nos níveis 40, 50 e 60, tamanho da sub-população igual a 70% do tamanho da população (28, 35 e 42, respectivamente), número de partículas genéticas móveis por cliente igual a 4 e número de iterações do ProtoG igual a 100. Observou-se que em todos os testes, na geração da população, foi obtido o resultado ótimo igual a 77,286, não havendo necessidade de processamento do ProtoG. Os tempos computacionais médios para 100 iterações apresentados na Tabela 21 indicam valores crescentes acompanhando o aumento do tamanho da população.

Tabela 21: Tempo computacional médio de acordo com os tamanhos da população

Tamanho da população	Tempo computacional médio (segundos)	Tempo computacional médio para atingir o resultado ótimo
40	29,7	3,4
50	40,0	3,9
60	48,3	4,0

Como um exemplo entre as configurações existentes para o resultado ótimo desta metodologia pode-se assumir a mesma configuração detalhada na Seção 6.3.2.1 para a metodologia AG – Simulação.

6.3.3.2 Resultados para a 2ª instância da metodologia ProtoG - Simulação

Os dados para esta instância são os mesmos especificados na Seção 6.3.2.2.

As rodadas de testes foram efetuadas variando-se os parâmetros:

- tamanho da população: 40, 50 e 60;
- tamanho da sub-população: 70% e 80% do tamanho da população (28 e 32; 35 e 40; 42 e 48, respectivamente)
- número de iterações do ProtoG: 100 e 200;
- número de partículas genéticas móveis por cliente: 4 e 6.

Cada uma das 24 combinações destes parâmetros foi aplicada por 15 vezes gerando um total de 360 rodadas de testes. Os resultados obtidos foram o tempo computacional e o valor da função de aptidão. Com base nos resultados da metodologia AG – Simulação, confirmou-se o resultado da função de aptidão igual a 120,143 como sendo o menor valor encontrado. Com o objetivo de observar a ocorrência deste valor, incluiu-se na programação

desta metodologia uma condição de que o número de iterações e tempo computacional fossem salvos em arquivo de texto, quando este valor fosse encontrado pela primeira vez.

A Tabela 22, mostra a média das 15 rodadas de testes efetuadas para as 24 combinações distintas destes parâmetros.

Tabela 22: Média dos resultados da metodologia ProtoG - Simulação

Tamanho População	Tamanho Sub-População	Número Partículas Genéticas Móveis	Nº de Iterações do ProtoG	Tempo Computacional Médio (segundos)	Valor Médio da Função de Aptidão	Nº médio de Iterações do AG p/Melhor Res	Tempo Computacional (s) médio para melhor resultado
40	28	4	100	148,0	124,386	17	47,0
40	32	4	100	162,9	124,443	36	74,7
40	28	6	100	188,9	122,793	62	116,3
40	32	6	100	224,3	123,193	40	108,0
40	28	4	200	272,0	123,229	54	92,7
40	32	4	200	207,8	123,300	58	72,7
40	28	6	200	371,9	123,048	17	57,5
40	32	6	200	380,2	122,383	38	96,2
50	35	4	100	179,3	124,057	5	40,0
50	35	6	100	228,6	122,829	62	131,5
50	40	4	100	188,9	124,586	53	118,0
50	40	6	100	296,3	123,657	31	92,5
50	35	6	200	417,1	122,800	14	62,0
50	40	4	200	347,9	122,071	47	109,2
50	40	6	200	334,3	123,471	17	48,0
50	35	4	200	344,5	122,907	73	147,0
60	42	4	100	176,1	123,336	60	110,3
60	48	4	100	163,4	123,972	56	100,6
60	42	6	100	252,9	124,345	8	80,0
60	48	6	100	295,7	122,679	20	90,0
60	42	4	200	260,7	123,586	51	88,0
60	48	4	200	434,9	122,271	47	132,0
60	42	6	200	500,0	122,855	72	204,3
60	48	6	200	396,1	122,421	70	155,2

Para os 360 resultados foram calculadas estatísticas descritivas para o tempo computacional e para o valor da função de aptidão. A Tabela 23 mostra estes resultados.

Tabela 23: Estatísticas descritivas para o tempo computacional e o valor da função de aptidão

	Tempo Computacional (segundos)	Valor da Função de Aptidão
Mínimo	140	120,143
Máximo	522	132,710
Média	282,2	123,276
Mediana	266,0	124,000
Desvio padrão	99,7	2,570

As estatísticas descritivas dos 82 (22,8%) resultados que encontraram o valor mínimo foram calculadas em relação ao tempo computacional, ao número de iterações do ProtoG e ao tempo computacional para o resultado mínimo (Tabela 24).

Tabela 24: Estatísticas descritivas para os melhores resultados da metodologia ProtoG - Simulação

	Tempo Computacional (segundos)	Nº de Iterações do AG para melhor Resultado	Tempo Computacional (segundos) para melhor resultado
Mínimo	149,0	2	30,0
Máximo	511,0	175	363,0
Média	299,2	46,7	108,1
Mediana	303,0	33	86,0
Desvio padrão	105,3	39,9	70,3

O gráfico da Figura 67 mostra o tempo computacional em relação ao valor da função de aptidão.

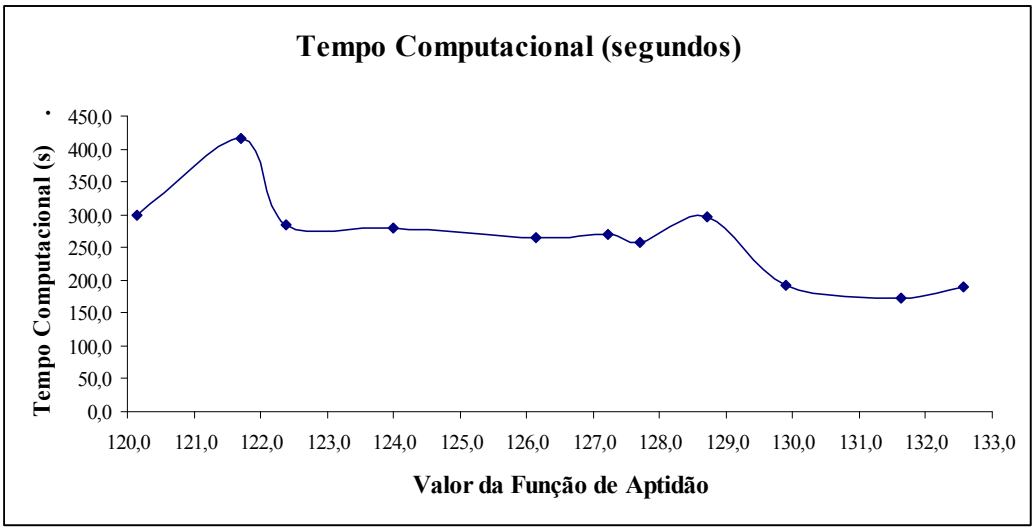

Figura 67: Tempo computacional de acordo com o valor da função de aptidão

O valor da função de aptidão de acordo com os tamanhos da população e da sub-população e número de iterações do ProtoG pode ser analisado no gráfico da Figura 68.

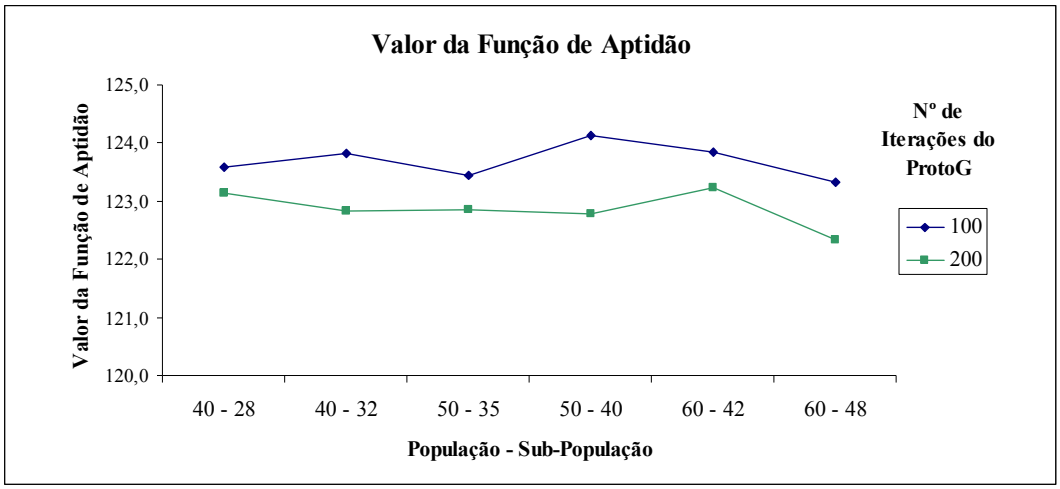

Figura 68: Valor da função de aptidão de acordo com os tamanhos da população e da sub-população e número de iterações do ProtoG

Ainda, na Figura 69 pode-se observar o valor da função de aptidão de acordo com os tamanhos da população e da sub-população e número de partículas genéticas móveis.

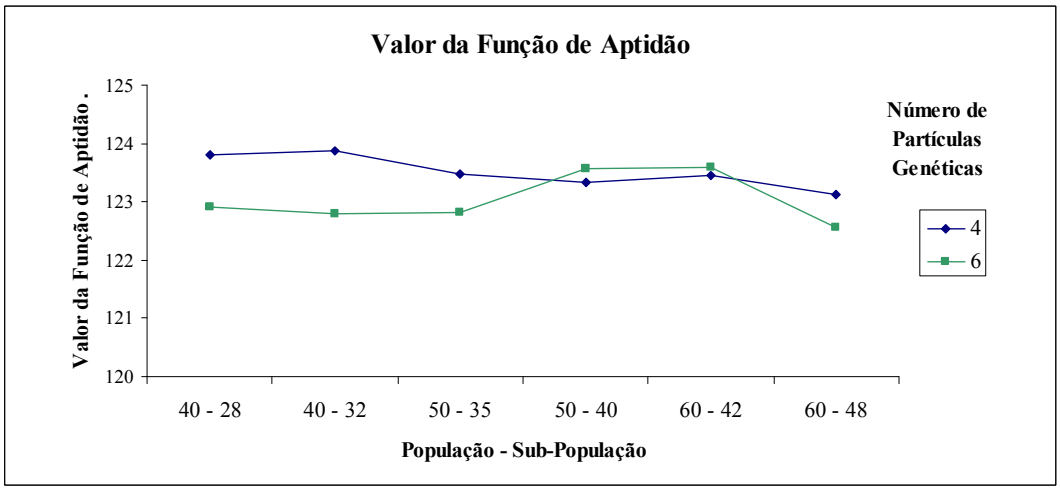

Figura 69: Valor da função de aptidão de acordo com os tamanhos da população e da sub-população e número de partículas genéticas móveis

Com os 360 resultados obtidos, investigou-se o efeito de cada um dos parâmetros (tamanho da população, tamanho da sub-população, número de partículas genéticas móveis e número de iterações do ProtoG) sobre o tempo computacional. Para tanto, ajustou-se um modelo de regressão múltipla considerando os parâmetros como variáveis explicativas (independentes) e o tempo computacional como variável resposta (dependente).

O modelo ajustado apresentou um coeficiente de determinação igual a $R^2 = 0,7970$ indicando que os parâmetros considerados no modelo explicam 79,70% da variabilidade observada no tempo computacional. Quanto às variáveis explicativas o resultado do ajuste e os testes estatísticos efetuados com um nível de significância de 0,05 indicaram que são significantes os coeficientes de regressão das seguintes variáveis: tamanho da população ($p = 0,0243$), sub-população ($p = 0,0348$), número de iterações do ProtoG ($p < 0,0001$) e número de partículas genéticas móveis ($p < 0,0001$). Desta forma variações nos parâmetros considerados implicam significativamente em variações no tempo computacional.

As Tabelas 25, 26, 27, 28, 29 e 30 e os gráficos das Figuras 70, 71, 72, 73, 74 e 75 ilustram esta análise.

Tabela 25: Tempo computacional de acordo com o tamanho da sub-população e tamanho da população

	Sub-População (% da Pop.)	
Tamanho da População	70%	80%
40	245,2	243,8
50	292,4	291,9
60	297,4	322,5

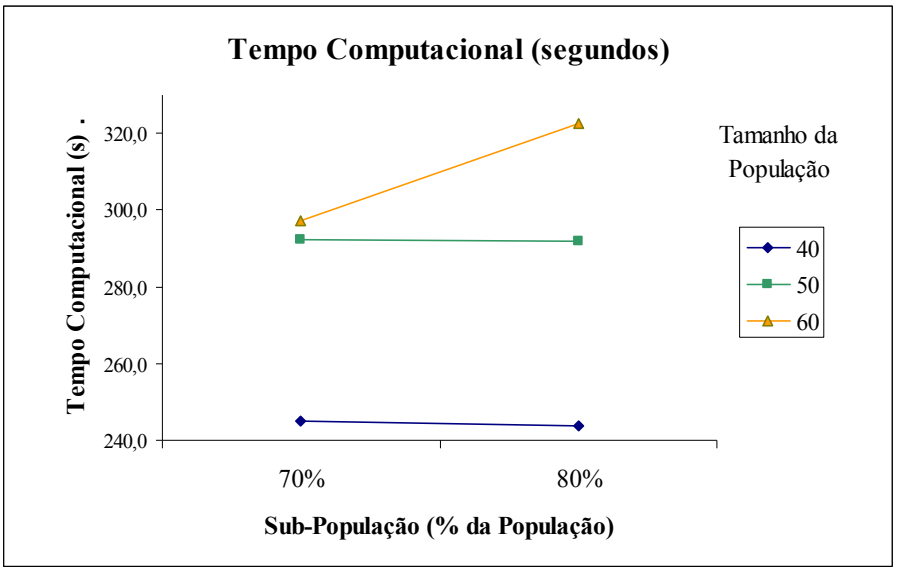

Figura 70: Tempo computacional de acordo com o tamanho da sub-população e tamanho da população

Tabela 26: Tempo computacional de acordo com o número de partículas genéticas móveis e tamanho da população

	Nº de Partículas Genéticas	
Tamanho da População	4	6
40	197,7	291,3
50	265,2	319,1
60	258,8	361,2

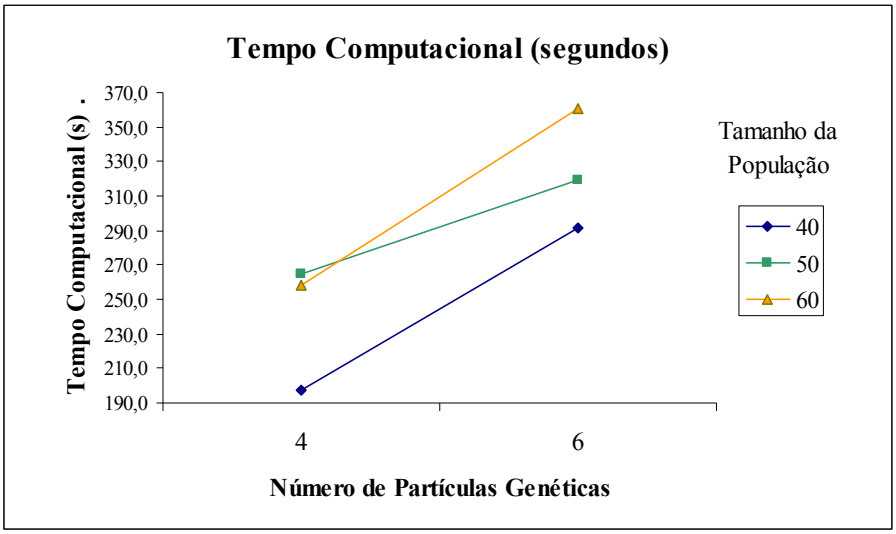

Figura 71: Tempo computacional de acordo com o número de partículas genéticas móveis e tamanho da população

Tabela 27: Tempo computacional de acordo com o número de iterações do ProtoG e o tamanho da população

	Nº de Iterações do ProtoG	
Tamanho da População	100	200
40	181,0	308,0
50	223,3	361,0
60	222,0	397,9

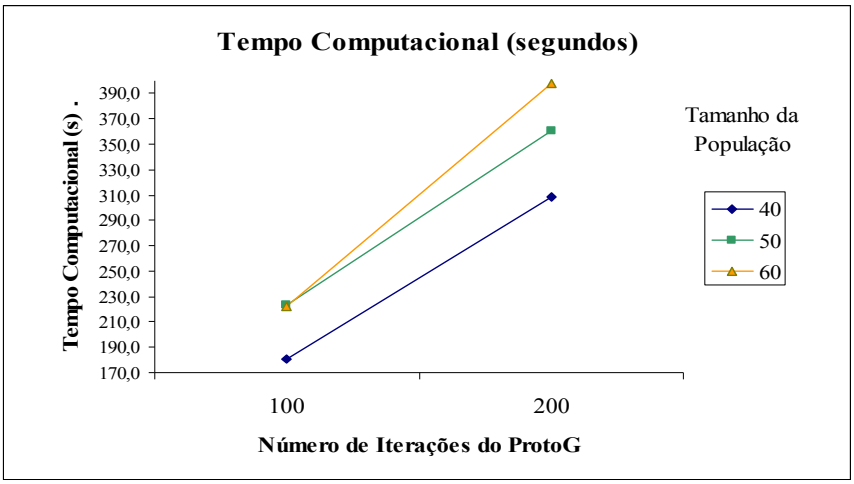

Figura 72: Tempo computacional de acordo com o número de iterações do ProtoG e o tamanho da população

Tabela 28: Tempo computacional de acordo com o número de iterações do ProtoG e o número de partículas genéticas móveis

	Nº de Iterações do ProtoG	
Número de Partículas Genéticas	100	200
4	169,8	311,3
6	247,8	399,9

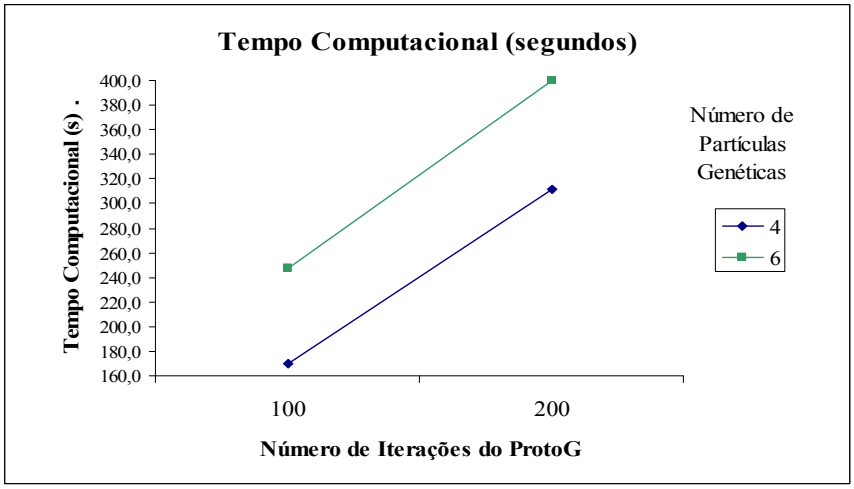

Figura 73: Tempo computacional de acordo com o número de iterações do ProtoG e o número de partículas genéticas móveis

Tabela 29: Tempo computacional de acordo com a sub-população e o número de iterações do ProtoG

	Nº de Iterações do ProtoG	
Sub-População (% da População)	100	200
70%	195,6	361,0
80%	221,9	350,2

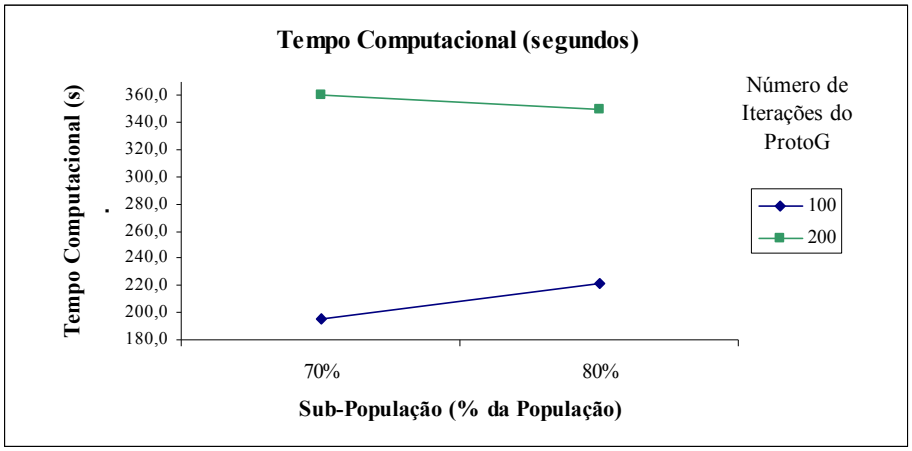

Figura 74: Tempo computacional de acordo com a sub-população e o número de iterações do ProtoG

Tabela 30: Tempo computacional de acordo com o tamanho da sub-população e o número de partículas genéticas móveis

	Sub-População (% da População)	
Número de Partículas Genéticas	70%	80%
4	204,6	251,0
6	326,6	321,2

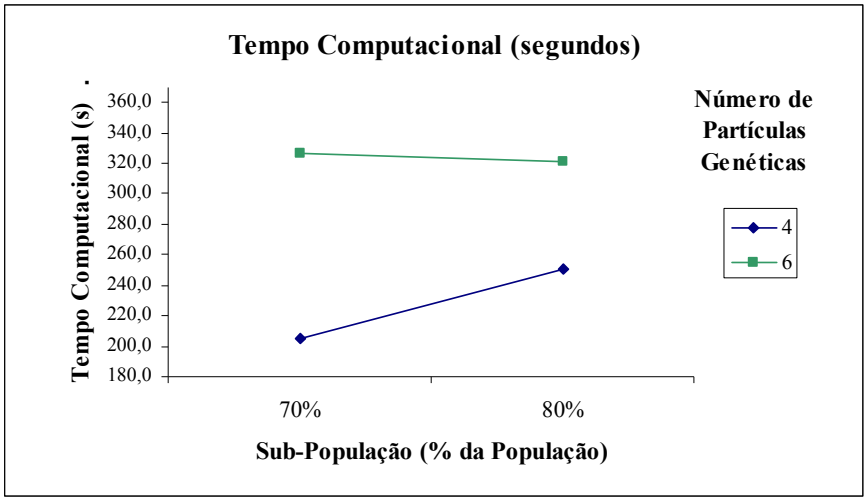

Figura 75: Tempo computacional de acordo com o tamanho da sub-população e o número de partículas genéticas móveis

CAPÍTULO 7

CONCLUSÕES

7.1 CONSIDERAÇÕES INICIAIS

Neste trabalho foram abordadas aplicações para otimização da programação de produção (*scheduling*) em uma refinaria de petróleo. O problema tratado foi o de transferência e estocagem de diesel, que inclui o envio de diesel aos tanques e posterior distribuição deste produto a clientes. O desenvolvimento e implementação de novas metodologias visaram à obtenção de soluções eficientes que priorizassem o retorno financeiro, de forma a tornar economicamente viáveis seus resultados práticos. Em particular, os seguintes tópicos foram trabalhados:

- A importância e a necessidade da utilização de técnicas de otimização para problemas de transferência e estocagem em refinarias de petróleo;
- Modelagem matemática do problema. Foram elaborados modelos utilizando representações de tempo discreto e contínuo;
- Descrição do modelo em PLIM com discretização uniforme de tempo. Para a resolução do problema modelado com representação de tempo discreto foram utilizadas as seguintes metodologias:
  - O algoritmo *branch and bound* com o uso do aplicativo computacional LINGO 8.0 para a resolução do modelo em PLIM;
  - A metodologia AGEEH – PLIM que integra um algoritmo genético híbrido com o modelo em PLIM relaxado;
  - A metodologia algoritmo ProtoG – PLIM que utiliza a transgenética computacional (Algoritmo ProtoG) integrada ao modelo PLIM relaxado;
- Descrição do modelo em PLIM com representação de tempo contínuo. Este modelo foi resolvido utilizando-se as seguintes abordagens:
  - O algoritmo *branch and bound* com o uso do aplicativo LINGO 8.0 para a resolução do modelo em PLIM;
  - A metodologia AG – Simulação que aplica um algoritmo genético para a otimização baseada em simulação;

- A metodologia ProtoG – Simulação que aplica o algoritmo transgenético ProtoG para a otimização baseada em simulação;
- Obtenção de resultados através de rodadas de testes com dados reais. Apresentação dos resultados em tabelas e gráficos para facilitar a comparação em termos de tempo computacional, número de iterações e qualidade de soluções (função objetivo ou função de aptidão).

7.2 SUMÁRIO DO TRABALHO

O Capítulo 1 descreve os motivos que levaram ao desenvolvimento desta pesquisa, a justificativa de sua importância e seus objetivos.

O Capítulo 2 desenvolve a fundamentação teórica baseada em livros e artigos científicos relacionados com os seguintes assuntos: programação matemática com enfoque em PLIM, cadeias de suprimento, planejamento e programação de produção, técnicas da computação evolucionária tais como AG e transgenética computacional e técnicas de simulação e otimização baseada em simulação.

As modelagens em PLIM construídas para o problema com representação de tempo discreto e contínuo são mostradas no Capítulo 3.

As metodologias desenvolvidas com o uso de técnicas da computação evolucionária integradas ao modelo PLIM com representação de tempo discreto estão descritas no Capítulo 4. Primeiramente foi abordada a técnica AGEEH – PLIM. Os passos para sua utilização foram organizados em módulos para facilitar o entendimento. Em seguida, o algoritmo ProtoG – PLIM foi detalhado também com a utilização de módulos.

As abordagens AG –Simulação e ProtoG - Simulação estão detalhadas no Capítulo 5 onde primeiramente estão descritos os elementos do modelo de simulação. A seguir, a descrição do funcionamento destas metodologias foi organizado em módulos, com o intuito de facilitar a compreensão.

Os resultados obtidos das rodadas de testes efetuadas com as diversas abordagens já citadas são apresentados em tabelas e gráficos no Capítulo 6. São também comentadas as análises estatísticas obtidas através dos resultados.

7.3 DISCUSSÃO DOS RESULTADOS

A análise dos resultados é feita para as modelagens com representação discreta e contínua do tempo. Inicialmente, os resultados de cada abordagem são discutidos separadamente e em seguida efetuadas comparações entre as diferentes metodologias aplicadas.

7.3.1 Representação discreta do tempo

O processo de transferência e estocagem de óleo diesel da refinaria de petróleo estudado foi analisado de forma a obter-se um modelo com representação discreta no tempo, com características lineares. Por ser um modelo em PLIM com discretização do tempo, um problema que surge na resolução através do aplicativo LINGO 8.0 é que a redução do tamanho do intervalo de tempo implica em um número de intervalos (T) maior e conseqüentemente um número maior de variáveis binárias, fazendo com que o tempo computacional aumente. Este acréscimo no tempo se deve ao fato do problema ser do tipo combinatorial. Neste caso, aumentando-se o número de variáveis inteiras, aumenta-se o número de combinações possíveis para o vetor solução, o que pode tornar o problema inviável, em termos de tempo computacional, na resolução com o algoritmo *branch and bound*.

Os testes preliminares para a representação discreta no tempo foram efetuados para apenas uma instância. Os dados para os parâmetros desta instância estão exibidos na Seção 6.2.

Para o modelo em PLIM com a aplicação do LINGO 8.0 foram efetuados 25 testes que resultaram no valor ótimo de 6,285 unidades monetárias para a função objetivo. A Tabela 4 mostra os resultados obtidos para o número de iterações e o tempo computacional (segundos). A média para o número de iterações foi 743124,8 com um desvio padrão de 273218,69. Para o tempo, a média foi de 22 minutos e 39 segundos com desvio padrão de 8 minutos e 52 segundos.

A Tabela 31 exibe as médias das iterações do LINGO 8.0 e do tempo computacional de cada uma das metodologias de tempo discreto de acordo com os intervalos da função de aptidão (f). Com base nestes dados foram efetuadas comparações entre as três metodologias em relação ao desempenho.

Tabela 31: Resumo das médias para as metodologias PLIM, AGEEH – PLIM e ProtoG - PLIM

Intervalos da função f	Metodologia	Iterações do LINGO	Tempo Computacional (segundos)
$f = 6,285$ (resultado ótimo)	PLIM	743125	1359,4
	AGEEH – PLIM	1229718	1219
	ProtoG – PLIM	753340	888
$6,285 < f \leq 6,47$	AGEEH – PLIM	988961	921
	ProtoG – PLIM	565021	660
$6,47 < f \leq 6,66$	AGEEH – PLIM	860666	856
	ProtoG – PLIM	442470	515
$6,66 < f \leq 6,9$	AGEEH – PLIM	837905	834
	ProtoG – PLIM	399271	463

Para o resultado ótimo na metodologia AGEEH – PLIM, tem-se um número de iterações 65,5% maior com tempo computacional 10,3% menor do que na metodologia PLIM. Se $6,285 < f \leq 6,47$, a metodologia AGEEH – PLIM resulta num número de iterações 33,1% maior com tempo computacional 32,2% menor. Se $6,47 < f \leq 6,66$, a metodologia AGEEH – PLIM resulta num número de iterações 15,8% maior com tempo computacional 37% menor. Ainda, Se $6,66 < f \leq 6,9$, a metodologia AGEEH – PLIM resulta num número de iterações 12,8% maior com tempo computacional 38,6% menor. Em resumo, para a metodologia AGEEH – PLIM, comparada com a metodologia PLIM, quanto mais afastado do valor ótimo, menor é a perda em relação ao número de iterações e maior é o ganho em relação ao tempo computacional.

De modo geral, observa-se que o desempenho da metodologia AGEEH – PLIM, com relação ao tempo computacional, foi melhor. Conclui-se, portanto, que na aplicação da metodologia AGEEH – PLIM, para a instância utilizada, deve-se ponderar a vantagem de um tempo computacional menor para a obtenção de um bom resultado, que pode não ser o ótimo.

Os resultados obtidos para a metodologia AGEEH - PLIM evidenciam que o desempenho depende dos parâmetros utilizados. A análise estatística permitiu afirmar que o número de iterações do AG influencia mais no desempenho do que o número de iterações para hibridização. Os parâmetros utilizados neste trabalho foram escolhidos com base em rodadas de testes prévias.

Deve-se destacar que o uso de probabilidades de recombinação e mutação adaptativas faz com que quanto mais afastado um indivíduo está do primeiro da população, maiores estas probabilidades. Desta forma, este indivíduo acaba sendo induzido a fazer as operações de

recombinação e de mutação. Por outro lado, os melhores indivíduos da população terão sempre probabilidades menores para que a chance de realização destas operações fique reduzida com o objetivo de preservar estes indivíduos.

Para os testes realizados neste trabalho, analisando-se os resultados da Tabela 6, observa-se que o melhor desempenho em termos de número de iterações do AG, número de iterações do LINGO e tempo computacional ao atingir o resultado ótimo, foi observado para tamanho de população igual a 45 e número de iterações para hibridização igual a 125. Considerando todos os resultados obtidos, o melhor desempenho ocorreu para o mesmo tamanho da população e número de iterações para hibridização igual a 100 (Tabelas 7, 8 e 9).

A variabilidade dos valores da função de aptidão analisada nas Figuras 40, 41 e 42 apresenta um comportamento semelhante quando se observa o número de iterações do AGEEH, o número de iterações do LINGO e o tempo computacional. À medida que o valor da função se aproxima do valor ótimo (6,285), esta variabilidade diminui e observa-se um crescimento acentuado que gera um custo em relação ao desempenho do algoritmo AG. Isto ocorre porque o algoritmo atinge um mínimo local e acaba por exigir um esforço adicional para sair deste mínimo e se direcionar para o resultado ótimo.

Ainda, a partir dos dados resumidos na Tabela 31 pode-se comparar os resultados das metodologias PLIM e ProtoG - PLIM, da seguinte forma: se o resultado ótimo para o valor da função de aptidão (*f*) for alcançado na metodologia ProtoG – PLIM, tem-se um número de iterações apenas 1,4% maior com tempo computacional 34,7% menor. Se $6{,}285 < f \leq 6{,}47$, a metodologia ProtoG – PLIM resulta num número de iterações 24% menor com tempo computacional 51,4% menor. Se $6{,}47 < f \leq 6{,}66$, a metodologia ProtoG – PLIM resulta num número de iterações 40,4% menor com tempo computacional 62,1% menor. Ainda, se $6{,}66 < f \leq 6{,}9$, a metodologia ProtoG – PLIM resulta num número de iterações 46,3% menor com tempo computacional 65,9% menor. Em resumo, comparando-se a metodologia ProtoG – PLIM com a metodologia PLIM, observa-se que, quanto mais afastado o resultado do valor ótimo, maior é o ganho em relação ao número de iterações e maior é o ganho em relação ao tempo computacional.

De modo geral observa-se que o desempenho da metodologia ProtoG – PLIM com relação ao tempo computacional foi melhor. Conclui-se, portanto, que deve ser analisada a vantagem em reduzir-se o número de iterações do LINGO e tempo computacional em detrimento do aumento no valor da função de aptidão.

Os resultados obtidos para a metodologia ProtoG - PLIM evidenciam que o desempenho depende dos parâmetros utilizados, destacando-se que o número de iterações do ProtoG tem influência estatisticamente significativa sobre o desempenho do algoritmo.

Para os testes realizados neste trabalho, analisando-se os resultados da Tabela 11 observa-se que o melhor desempenho em termos de número de iterações do LINGO e tempo computacional ao atingir o resultado ótimo, foi observado para tamanho de população igual a 20, tamanho da sup-população igual a 6 e tamanho da partícula genética móvel igual a 6. Considerando todos os resultados obtidos, o melhor desempenho ocorreu para a mesma combinação de parâmetros (Tabelas 12, 13 e 14).

A variabilidade dos valores da função de aptidão apresentada nas Figuras 47, 48 e 49 apresenta um comportamento semelhante quando se observa o número de iterações do ProtoG, o número de iterações do LINGO e o tempo computacional. À medida que o valor da função se aproxima do valor ótimo (6,285), esta variabilidade diminui e observa-se um crescimento que gera um custo em relação ao desempenho do algoritmo ProtoG. Isto ocorre porque o algoritmo atinge um mínimo local e acaba por exigir um esforço adicional para sair deste mínimo e se direcionar para o resultado ótimo.

Finalmente, comparando-se as metodologias AGEEH – PLIM e ProtoG – PLIM, se o resultado ótimo para o valor da função de aptidão (f) for alcançado na metodologia ProtoG – PLIM, tem-se um número de iterações 38,7% menor, com tempo computacional 27,2% menor. Se $6{,}285 < f \leq 6{,}47$, a metodologia ProtoG – PLIM resulta num número de iterações 42,9% menor, com tempo computacional 28,3% menor. Se $6{,}47 < f \leq 6{,}66$, a metodologia ProtoG – PLIM resulta num número de iterações 48,6% menor, com tempo computacional 39,8% menor. Ainda, Se $6{,}66 < f \leq 6{,}9$, a metodologia ProtoG – PLIM resulta num número de iterações 52,3% menor, com tempo computacional 44,5% menor.

Com esta análise comparativa entre as três metodologias com representação de tempo discreto, pode-se inferir que, para valores da função de aptidão maiores do que o valor ótimo, a metodologia ProtoG – PLIM tem desempenho melhor quando comparado com as metodologias PLIM e AGEEH-PLIM no que refere ao número de iterações do LINGO e ao tempo computacional. No resultado ótimo, o número de iterações do LINGO é discretamente maior e o tempo computacional é menor. Esta vantagem em número de iterações do LINGO 8.0 na resolução do modelo PLIM não implica em ganho de tempo computacional, pois o desempenho de número de iterações por segundo é menor se comparado às duas outras metodologias (igual a 546,7 para o PLIM, 1008,8 para o AGEEH-PLIM e 848,4 para o ProtoG-PLIM).

7.3.2 Representação contínua do tempo

O modelo com representação de tempo contínuo, para o problema de transferência e estocagem de diesel abordado é linear e foi construído usando técnicas de PLIM. Este modelo manipula intervalos contínuos do tempo usando eventos. A vantagem para este tipo de representação é conseguir-se o momento exato do início e término de operações que, na representação de tempo discreto, fica limitado ao início e término dos intervalos de tempo determinados na modelagem.

As três metodologias para representação de tempo contínuo desenvolvidas neste trabalho foram aplicadas para uma primeira instância, de acordo com os dados mostrados da Seção 6.3.1.

O único teste para a modelagem em PLIM com representação de tempo contínuo encontrando o valor ótimo igual 77,2857, despendeu um tempo computacional de 61 horas e 29 minutos, mostrando a inviabilidade do uso deste modelo para esta aplicação prática.

Em contrapartida, a aplicação das metodologias que utilizam AG e ProtoG baseadas em simulação surpreendeu ao mostrar um desempenho de aproximadamente 3 segundos para a obtenção do mesmo resultado ótimo (77,2857), para as duas abordagens.

Como todas as rodadas de testes desta instância para as duas metodologias obtiveram o resultado ótimo já na geração da população, estas foram aplicadas para uma segunda instância, de forma que fosse possível discutir o desempenho em termos computacionais. A metodologia PLIM não pôde ser aplicada por ter se mostrado inviável em relação ao tempo computacional, já na primeira instância.

Para a segunda instância, o número de tanques e de clientes foi aumentado com o intuito de incrementar a complexidade do problema. O valor mínimo da função de aptidão encontrado nas rodadas de testes das duas metodologias foi de 120,143. Para a abordagem AG-simulação o tempo computacional médio para os 810 resultados foi de 99,9 segundos, com valor médio para a função de aptidão igual a 124,562. Já para a abordagem ProtoG-simulação, o tempo computacional médio para os 360 resultados foi de 282,2 segundos com valor médio para a função de aptidão igual a 123,76.

A abordagem AG-simulação obteve o resultado mínimo em 14,1% dos testes realizados, correspondendo ao número médio de iterações igual a 1003 e tempo computacional médio de 52,1 segundos. Já para a metodologia ProtoG-simulação, o valor mínimo foi obtido em 22,8% dos testes, com tempo computacional médio de 108,1 segundos, com número médio de iterações igual a 46,7.

Com isso, é possível afirmar que as duas metodologias obtiveram um desempenho adequado. Não é possível a comparação entre estas metodologias a partir dos resultados obtidos, pois as rodadas de testes ocorreram para um número pré-fixado de iterações em função do não conhecimento do resultado ótimo.

O comportamento do tempo computacional da metodologia AG-simulação, de acordo com o valor da função de aptidão (Figura 60), reflete grande variabilidade antes de atingir o resultado mínimo. Este fato indica que por diversas vezes o algoritmo atingiu pontos de mínimo local despendendo tempo para dar continuidade à busca.

A análise estatística dos resultados obtidos nos testes indicou que o tempo computacional sofre influência significativa do tamanho da população, número de iterações do AG e probabilidade de mutação. Na Figura 63 percebe-se que quanto menor a probabilidade de mutação e o tamanho da população, menor é o tempo computacional.

Nas Figuras 61 e 62 observa-se que as combinações dos parâmetros probabilidade de recombinação e de mutação nos níveis 0,9 - 0,1 e 0,9 - 0,15 foram as que resultaram em valores da função de aptidão mais próximos do mínimo.

O comportamento do tempo computacional da metodologia ProtoG-simulação, de acordo com o valor da função de aptidão (Figura 67), mostrou pequena variabilidade antes de atingir o resultado mínimo, tendo ocorrido um pico nas proximidades deste resultado. Este fato indica que, possivelmente, neste ponto o algoritmo atingiu um ponto de mínimo local, acabando por despender um tempo computacional maior para sair deste ponto. Ainda, em algumas rodadas de testes o algoritmo não conseguiu, com o número de iterações fixado, sair deste ponto.

A análise estatística dos resultados obtidos nas rodadas de testes indicou que o tempo computacional sofre influência significativa do tamanho da população, tamanho da sub-população, número de iterações do ProtoG e número de partículas genéticas móveis. Nas Figuras 68 e 69 percebe-se que a combinação dos parâmetros tamanho da população igual a 60, tamanho da sub-população igual a 48, número de partículas genéticas igual a 6 e número de iterações igual a 200 destacou-se com relação ao desempenho para encontrar os menores valores da função de aptidão.

Um acréscimo no tamanho da população implica em aumento do tempo computacional. O mesmo ocorre para os parâmetros número de partículas genéticas móveis e tamanho da sub-população (Figuras 70, 71, 72, 73 , 74 e 75).

7.4 CONTRIBUIÇÕES DO TRABALHO

Este trabalho produziu como contribuições, modelos e metodologias com especificidades para resolução de problemas de transferência e estocagem de produtos em refinarias de petróleo. O estudo de caso abordado tratou os problemas relacionados ao setor do diesel. Tanto os modelos como as metodologias envolvem originalidade na sua formulação, tendo em vista, após ampla pesquisa bibliográfica não terem sido encontradas referências a grande parte do que foi desenvolvido. A seguir, estão enunciadas as principais contribuições relacionadas a este trabalho:

1. Formulação de modelo em PLIM com discretização uniforme do tempo para o problema de transferência e estocagem de diesel, envolvendo tanques, clientes, um tipo de produto e um tipo de recebimento.
2. Formulação de modelo em PLIM com representação de tempo contínuo para o problema de transferência e estocagem de diesel, envolvendo tanques, clientes, mais de um tipo de produto e mais de um tipo de recebimento.
3. Desenvolvimento de uma metodologia em substituição ao método *branch and bound*, utilizado para a resolução do modelo em PLIM com discretização uniforme do tempo. Esta metodologia apresentou originalidade na utilização de AG como técnica alternativa de busca da solução inteira para o modelo em PLIM, usando a resolução de problemas relaxados em PL para o cálculo da função de aptidão. Os resultados obtidos mostraram que a metodologia é adequada, levando-se em conta a redução do tempo computacional, sem grande perda na qualidade da solução.
4. Desenvolvimento de uma segunda metodologia com o mesmo propósito de substituição do método *branch and bound*. Esta metodologia fez uso do algoritmo transgenético ProtoG, técnica desenvolvida recentemente e ainda com poucas aplicações. Da mesma forma que a metodologia descrita no item 3, o algoritmo ProtoG foi utilizado para encontrar a solução para o problema modelado em PLIM. O algoritmo faz a busca da solução utilizando o modelo relaxado em PL para calcular o valor da função de aptidão. Esta técnica apresentou grandes vantagens como alternativa em substituição ao algoritmo *branch and bound* para o problema abordado.
5. Criação de um modelo de simulação para o problema com representação de tempo contínuo. Para a otimização deste modelo foram utilizadas duas técnicas da CE. A otimização baseada em simulação tem apresentado um imenso potencial e uma aceitação

crescente por pesquisadores e especialistas das indústrias. Uma ampla busca por artigos referentes ao assunto foi efetuada e concluiu-se que há muito ainda a ser desenvolvido nesta área. Como já analisado anteriormente, os resultados gerados por estas metodologias neste trabalho foram surpreendentes. A utilização de AG com simulação não é propriamente original, uma vez que há referências em artigos e livros sobre a aplicação desta técnica para a otimização de problemas. No entanto, a abordagem proposta neste trabalho para a utilização de AG integrado a um simulador do processo de transferência e estocagem pode ser considerada como original, e sua aplicação não se limita apenas ao problema considerado. Como inovação, a segunda técnica da CE acoplada ao simulador foi a abordagem da transgenética computacional, mais especificamente, o algoritmo ProtoG. A integração deste algoritmo com o simulador mostrou-se eficiente para a aplicação prática abordada. Desta forma, conclui-se que as duas metodologias, usando otimização baseada em simulação, podem contribuir significativamente para o problema de transferência e estocagem de produtos em refinarias de petróleo.

7.5 SUGESTÕES PARA TRABALHOS FUTUROS

Novas pesquisas podem ser desenvolvidas a partir deste trabalho. Algumas das propostas sugeridas são:

- O desenvolvimento de um modelo em PLIM com discretização uniforme de tempo que inclua mais de um tipo de produto e recebimento;
- A aplicação das técnicas descritas em outras áreas de planejamento e programação de operações de produção e distribuição de produtos nas refinarias de petróleo. Entre estas áreas podem ser destacadas a programação de suprimento de petróleo, as operações de transporte por dutos, a programação da produção propriamente dita e a distribuição de produtos de uma forma geral;
- Aplicação de outras técnicas tais como *Simulated Annealing*, Busca Tabu, evolução diferencial em substituição às técnicas de AG e Algoritmo Transgenético ProtoG nas metodologias AGEEH-PLIM e algoritmo ProtoG – PLIM;
- Desenvolvimento de metodologias de otimização baseada em simulação usando outras heurísticas tais como: *Simulated Annealing*, Busca Tabu, evolução diferencial e outras;
- Integração das metodologias abordadas com aplicativos para aquisição de dados e geração automática de respostas. A demanda de produtos em refinarias de petróleo sofre alterações

diariamente o que faz com que a programação da transferência e estocagem seja alterada constantemente. Ainda, pode ser implementada uma interface gráfica para facilitar o entendimento sobre as operações a serem efetuadas pelos usuários do sistema desenvolvido;

- Aplicação das metodologias desenvolvidas para problemas semelhantes em outras indústrias, podendo-se citar entre outras, as indústrias químicas, de alimentos e farmacêuticas.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLAOUI, H., ARTIBA, A. Integrating Simulation and Optimization to Schedule a Hybrid Flow Shop with Maintenance Constraints. **Computers & Industrial Engineering**, v. 47, p. 431-450, 2004.

ALOULOU, M. A., KOVALYOUV, M. Y., PORTMANN, M. C. Maximization Problems in Single Machine Scheduling. **Annals of Operations Research**, v.129, p.21-32, 2004.

ANDRADÓTTIR, S. **Simulation optimization. Chapter 9 in Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. ed. J. Banks. New York: John Wiley & Sons, 1998.

AOKI, T., NAKAYAMA, S., YAMAMOTO, M., HASHIMOTO, M., TANAKA, J. I. Combinatorial Scheduler: Simulation & Optimization Algorithm. **Proceedings of the 1991 Winter Simulation Conference**, p. 280-288, 1991.

AOMAR, R. A. A Robust Simulation-Based Multicliteria Optimization Methodology. **Proceedings of the 2002 Winter Simulation Conference**, v. 2, p. 1931-1939, 2002.

ARMENTANO, V. A., RONCONI, D. P. Minimização do Tempo Total de Atraso no Problema de FlowShop com Buffer Zero através de Busca Tabu. **Gestão & Produção**. v. 7, n. 3, p. 352-363, 2000.

BABU, P., PERIDY, L., PINSON, E. A Branch and Bound Algorithm to Minimize Total Weighted Tardiness on a Single Processor. **Annals of Operations Research**, v. 129, p.33-46, 2004.

BÄCK, T., SCHWEFEL, H. P. An Overview of Evolucionary Algorithms for Parametrer Optimization. **Evolucionary Computation**, v. 1, n. 1, p. 1-23. The MIT Press, 1993.

BÄCK, T., FOGEL, D. B., MICHALEWICZ, Z. **Evolucionary Computation 2: Advanced Algorithms and Operators,** Bristol, UK, Institute of Physics, 2000.

BAESLER, F., MORAGA, M. RAMIS, F. J. Productivity Improvement in the Wook Industry using Simulation and Artificial Intelligence. **Proceedings of the 2002 Winter Simulation Conference**, v.2, p. 1095-1098, 2002.

BAKER, K. R. **Introduction to Sequencing and Scheduling**. John Wiley & Sons, Inc. United States of America, 1974.

BANKS, J., CARSON II, J. S., NELSON, B. L., NICOL, D. M. **Discret-Event System Simulation**. Prentice Hall, Upper Saddle River, New Jersey, 2000.

BAPAT, V., STURROCK, D. T. The ARENA Product Family: enterprise modeling solutions. **Proceedings of the 2003 Winter Simulation Conference**, v.1, p. 210-217, 2003.

BARBOSA, H. J. C. **Introdução aos Algoritmos Genéticos**. Minicurso, XX Congresso Nacional de Matemática Aplicada e Computacional – CNMAC, Gramado, RS, 1997.

BATEMAN, R. E., BOWDEN, R. G., GOGG, T. J., HARRELL, C. R., MOTT, J. R. A. **A Simulação – Aprimorando Sistemas**. Belge – Engenharia e Sistemas Ltda, São Paulo, Brasil, 1999.

BAUDET, P., AZZARO-PANTEL, C., DOMENECH, S. PIBOULEAU, L. Discret-Event Simulation Approach for Scheduling Batch Processes, **Chemical Engineering**, v. 19, p. 633-638, 1995.

BERNAL-HARO, L,. AZZARO-PANTEL, C. DOMENECH, S. PIBOULEUAU, L. Design of Multipurpose Batch Chemical Plants using a Genetic Algorithm. Computers & Chemical Engineering, v. 22, p. 777-780 , 1998.

BIREWAR, D. B. **Design, planning and scheduling of multiproduct batch plants**. Ph.d. Thesis 194p. Departamente of Chemical Engineering. Carnegie Mellon University, Pittsburgh (PA), 1989.

BLAZEWICZ, J., MACHOWIAK, M., WEGLARZ, J., KOVALYOV, M. Y., TRYSTRAM, D. Scheduling Malleable Tasks on Parallel Processors to Minimize the Makespan. **Annals of Operations Research**, v. 129, p.65-80, 2004.

BLOECHLE, W. K., LAUGHERY JR, K. R. Simulation Interoperability Using Micro Saint Simulation Software. **Proceedings of the 1999 Winter Simulation Conference**, v. 1, p. 286-288, 1999.

BLOECHLE, W. K., SCHUNK, D. Micro Saint Sharp simulation software. **Proceedings of the 2003 Winter Simulation Conference**, v. 1, p. 182-187, 2003.

BONNELLE, P., BOS FELDMAN, M. Automating the Scheduling Process. **NPRA Computer Conference**, p. 99-132, Kansas City, EUA, 1999.

BRADLEY, S. P., HAX, A. C., MAGNANTI, T. L. **Applied Mathematical Programming**. Addison-Wesley Publishing Company, United States of America, 1977.

BROWN, E. C., SUMICHRAST, R. T. Evaluating Performance Advantages of Grouping Genetic Algorithms. **Engineering Applications of Artificial Intelligence**, v. 18, p. 1-12, 2005.

BRUZZONE, A. ORSONI, A. MOSCA, R. REVETRIA, R. Ai-Based Optimization for Fleet Management in Maritime Logistics. **Proceedings of the 2002 Winter Simulation Conference**, v.2, p. 1174-1182, 2002.

BUZZO, W. R., MOCCELLIN, J. V. Programação da Produção em Sistemas *Flow Shop* utilizando um Método Heurístico Híbrido Algoritmo Genético - *Simulated Annealing*. **Gestão & Produção**. V. 7, n. 3, p. 364-377, 2000.

CASTRO, P., BARBOSA-PÓVOA, A. P. F. D., MATOS, H. Short-Term Scheduling of a Polymer Compounding Plant. **European Symposium on Computer Aided Process Engineering**, v.12, p.649-654, 2002.

CASTRO, R. E. **Otimização de Estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto**. Tese de Doutorado. Programa de Engenharia Civil COPPE/UFRJ.2001.

CAVE, A. NAHAVANDI, S., KOUZANI, A. Simulation Optimization for Process Scheduling Through Simulated Annealing. **Proceedings of the 2002 Winter Simulation Conference**, v. 2, p. 1909-1913, 2002.

CHEN, Z. L. Simultaneous Job Scheduling and Resource Allocation on Parallel Machines. **Annals of Operations Research**, v. 129, p.135-153, 2004.

CHENG, R., GEN, M. Parallel Machine Scheduling Problems using Memetic Algorithms. **Computers & Industrial Engineering**, v. 33, (3-4), p. 761-764, 1997.

CHENG, T. M., FENG, C. W., CHEN, Y. L. A Hybrid Mechanism for Optimizing Construction Simulation Models. **Automation in Construction**, v. 14, p. 85-98, 2005.

CHEUNG, W, ZHOU, H. Using Genetic Algorithms and Heuristics for Job Shop Scheduling with Sequence-Dependent Setup Times. **Annals of Operations Research**, v. 107, p. 65-81, 2001.

CHWIF, L., BARRETTO, M. R. P., SALIBY, E. (2002). Supply chain Analysis: Spreadsheet or Simulation? **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 59-66, 2002.

COELHO, L. S., COELHO, A. A. R. Algoritmos Evolutivos em Identificação e Controle de Processos: Uma Visão Integrada e Perspectivas. **SBA Controle & Automação**. V.10, n.01, p.13-30, 1999.

CONCANNON, K. H., HUNTER, K. I., TREMBLE, J. M. SIMUL8 - Planner simulation-based planning and scheduling. **Proceedings of the 2003 Winter Simulation Conference**, v. 2, p. 1488-1493, 2003.

CORREA, E. S., STEINER, M. T. A., FREITAS, A. A., CARNIERI, C. A Genetic Algorithm for solving a Capacitated P-Median Problem. **Numerical Algorithms**. Netherlands, v. 35, p. 373-388, 2004.

COSTA, W. E., GOLDBARG, M. C., GOLDBARG, E. F. G. Algoritmo Transgenético para o Problema de Flow-Shop de Permutação. **Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional**, v. 1, p. 1487-1497, 2003.

DANTZIG, G. B. **Linear Programming and Extensions**. Princeton University Press, Princenton, New Jersey, United States of America, 1963.

DARWIN, C. **The Origin of Species**. Fac-Simile da edição original – Charles W. Eliot, L. L. D., 1981.

DEB, K., PAL, K. Efficiently Solving: A Large-Scale Integer Linear Programs using a Customized Genetic Algorithm. **Genetic and Evolutionary Computation - GECCO**, v. 3102, p. 1054-1065, 2004

DONALD, D. L. Tutorial on Ergonomic and Process Modeling Using Quest and UGRIP. **Proceedings of the 1998 Winter Simulation Conference**, v. 1, p 297-302, 1998.

ESPEJO, L. G. A., GALVÃO, R. D. O uso de relaxações Lagrangeana e Surrogate em problemas de programação inteira. **Pesquisa Operacional,** v. 22, n. 3, p. 387-402, 2002.

FALCÃO, A. O., BORGES, J. G. Designing an Evolution Program for Solving Integer Forest Management Scheduling Models: an Application in Portugal. **Forest Science**, v. 47, n. 2, p. 158-168, 2001.

FARO, M. **Programação e Controle de Produção de Unidades em Batelada com Agregação de Tempo e Modelos de Desempenho**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo. São Paulo, 2001.

FINKE, D. A., TRABAND, M. T. Shop Scheduling using Tabu Search and Simulation. **Proceedings of the 2002 Winter Simulation Conference**, v.1, p. 1013-1017, 2002.

FREITAS FILHO, P. J. **Introdução à Modelagem e Simulação de Sistemas – com Aplicações em ARENA**. Visual Books Ltda. Florianópolis, SC, 2001.

FU, M. C. Optimization via Simulation: a Review. **Annals of Operations Research**, v. 53, p. 199-248, 1994.

FU, M. C., ANDRADÓTTIR, S., CARSON, J. S., GLOVER, F., HARRELL, C. R., HO, Y. C., KELLY, J. P. ROBINSON, S. M. Integrating Optimization and Simulation: Research and Practice, **Proceedings of the 2000 Winter Simulation Conference**, v. 1, p. 610-616, 2000.

GAREY, M. R., JOHNSON, D. S. . **Computers and Intractability – A Guide to the Theory of NP-Completeness**. Bell Laboratories, Morray Hill, New Jersey, W. H. Freeman and Company, New York, 1979.

GIANNELOS, N. F., GEORGIADIS, M. C. A New Event-Driven MILP Formulation for Short-Term Scheduling of Continuous Production Facilities. **European Symposium on Computer Aided Process Engineering**, v.12, p.667-672, 2002.

GIGLIO, A. L. **Técnicas de Otimização Mista-Inteira para o Planejamento de Produção em Plantas Petroquímicas**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo. São Paulo, 2001.

GLOBE, J. Introduction to SIMFACTORY 11.5. **Proceedings of the 1991 Winter Simulation Conference**. p. 77-80, 1991.

GOLDBARG, E. F. G., CASTRO, M. P., GOLDBARG, M. C. A Transgenetic Algorithm for the Gas Network Pipe Sizing Problem. **Proceedings of International Conference on Computational Methods**, p. 80-84, 2004

GOLDBARG, M. C., GOLDBARG, E. F. G. Transgenética Computacional: uma Aplicação ao Problema Quadrático de Alocação. **Pesquisa Operacional**, v.22, n.3, p.359-386, 2002.

GOLDBARG, M. C., GOLDBARG, E. F. G., MEDEIROS NETO, F. D. Algoritmos Evolucionários na Determinação da Configuração Ótima de Sistemas de Co-geração de Energia com Base em Gás Natural. **Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional**, v. 1, 2003.

GOLDBARG, M. C., GOLDBARG, E. F. G., MEDEIROS NETO, F. D. An Evolutionary Approach for the Piston Pump Mobile Unit Problem. **Proceedings of MCO'04 Fifth International Conference on Computer Sciences**. V. 1, p. 281-290, 2004.

GOLDBARG, M. C., GOLDBARG, E. F. G., MEDEIROS NETO, F. D. Algoritmos evolucionários na determinação da configuração de custo mínimo de sistemas de co-geração de energia com base no gás natural. **Pesquisa Operacional**. v. 25, n. 2. p. 231-253, 2005.

GOLDBARG, M. C., GOUVÊA, E. F. Extra-Intracelular Trnasgenetic Algorithm Applied to the Graph Coloring Problem. **Annals of 4th Metaheuristics International Conference**, v. 1, p. 321-326, 2001.

GOLDBARG, M. C., LUNA, H. P. L. **Otimização Combinatória e Programação Linear**. **Modelos e Algoritmos**. Editora Campus, Rio de Janeiro, Brasil, 2000.

GOLDBERG, D.E. Optimal initial population size for binary-coded genetic algorithms. **TCGA Report**, University of Alabama, Tuscaloosa, n. 85001, 1985.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Addison-Wesley Publishing Company, INC. United States of America, 1989.

GOUVÊA, E. F., GOLDBARG, M. C. An Intracellular Transgenetic Algorithm applied to the Quadratic Assignment Problem. **In: IV Congresso Chileno de Investigacíon Operativa – anais do IV Congresso Chileno de Investigacíon Operativa**, v. 1 p. 39-45, 2001.

GRAUPNER, T. D., RICHTER, H., SIHN, W. Configuration, simulation and animation of manufacturing systems via the internet. **Proceedings of the 2002 Winter Simulation Conference**, v.1, p. 825-831, 2002.

GUPTA, J. N. D., KOULAMA, C. P., KYPARISIS, G. J., POTTS, C. N., STRUSEVICH, V. A. Scheduling Three-Operation Jobs in a Two-Machine Flow Shop to Minimize Makespan. **Annals of Operations Research**. v.129, p. 171-185, 2004.

HARRELL, C. R., PRICE, R. N. Simulation modeling using ProModel technology. **Proceedings of the 2003 Winter Simulation Conference**, v.1, p. 175-181, 2003.

HARTMANN, S. Project Scheduling with Multiple Modes: A Genetic Algorithm. **Annals of Operations Research**, v. 102, p. 111-135, 2001.

HIGUCHI, T., TROUTT, M. D. Dynamic simulation of the supply chain for a short life cycle product – lessons from the Tamagochi case. **Computers & Operations Research**, v.31, p. 1097-1114, 2004.

HO, W., JI, P., Component Scheduling for Chip Shooter Machines. A Hybrid Genetic Algorithm Aproach. **Computers and Operations Research**, v. 30, p. 2175-2189, 2003.

HOLLAND, J.H. **Adaptation in Natural and Artificial Systems**, 2 ed., The University of Michigan Press, MI, 1975.

HULLINGER, D. R. Taylor Enterprise Dynamics. Proceedings of the 1999 **Winter Simulation Conference**, v.1, p. 227-229, 1999.

ILKYEONG, M., JIEOM, LEE. Genetic Algorithm Application to the Job Shop Scheduling Problem with Alternative Routings. **Brain Korea 21 Logistics Team**, p.1-19, 2000.

INGALLS, R. G. Introduction to Simulation. **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 7-16, 2002.

JENG, A. A. K., LIN, B. M. T. Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs. **Computers & Operations Research**, v. 32, p. 521-536, 2005.

JOINES, A. J., GUPTA, D., GOKCE, M. A., KING, R. E., KAY, M. G. Supply Chain Multi-Objective Simulation Optimization. **Proceedings of the 2002 Winter Simulation Conference**, v. 2, p. 1306-1314, 2002.

JOLY, M. **Técnicas e Otimização Mista-Inteira para o Scheduling e Gerenciamento da Produção em Refinarias de Petróleo**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 1999.

KELTON, W. D., SADOWSKI, R. P., SADOWSKI, D. A. **Simulation with ARENA**. WCB/ McGraw-Hill, United States of America, 1998

KHEIR, N. A. **Systems Modeling and Computer Simulation**. Marcel Dekker, Inc, 2nd ed., United States of America, 1996.

KRAHL, D. Modeling with Extend. **Proceedings of the 1999 Winter Simulation Conference**, v.1, p. 188-195, 1999.

KRAHL, D. An Interactive Simulation Tool. **Proceedings of the 2003 Winter Simulation Conference**, v.1, p. 188-196, 2003.

KREIPL, S., PINEDO, M. Planning and Scheduling in Supply Chains: An Overview of Issues in Practice. **Production and Operations Management**, v. 13, Iss. 1, p 77-92, 2004.

LAW, A. M., KELTON, W. D. **Simulation Modeling and Analysis**, 3 ed. McGraw-Hill, Nova York, 2000.

LAW, A. M., McCOMAS, M. G. Simulation-Based Optimization. **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 41-44, 2002.

LIAW, C. F. A Hybrid Genetic Algorithm for the Open Shop Scheduling Problem. **European Journal of Operational Research**, v. 124, p. 24-42, 2000.

LILEGDON, W. R. Manufacturing decision making with FACTOR. **Proceedings of the 1993 Winter Simulation Conference**, p. 159-164, 1993.

LIMA, M. J., BARBOSA, G. A., BEAL, C. R. Otimização da Transferência e Estocagem utilizando Ferramentas de Modelagem e Simulação. **Boletim Técnico da Petrobrás**, v. 46 (3/4) p. 301-307, 2003.

MAGALHÃES, M. V. O., MORO, L. F. L., SMANIA, P., HASSIMOTO, M. K., PINTO, J. M, ABADIA, SIPP, G. J. A Solution for Refinery Scheduling. **NPRA Computer Conference**, San Antonio (EUA), 1998.

MAGATÃO, L. **Programação Matemática Aplicada à Otimização da Operações de um Poliduto.** Dissertação de Mestrado. Centro Federal de Educação Tecnológica do Paraná, Curitiba, Brasil, 2001.

MÁS, R. **Otimização da Programação de Suprimento de Petróleo**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2001.

MAYERLE, S. F. Um Algoritmo Genético para Solução do Problema do Caixeiro Viajante. **Artigo de circulação interna do departamento de Engenharia de Produção e sistemas da UFSC**, 1994.

MEHTA, A., RAWLES, I. Business Solutions Using WITNESS. **Proceedings of the 1999 Winter Simulation Conference**, v. 1, p. 230-233, 1999.

MERZ, P., FREISLEBEN, B. Fitness Landscape Analysis and Memetic Algorithm for the Quadratic Assignmente Problem, **IEEE Transactions on Evolutionary Computation**, v. 4, p. 337-352, 2000.

MIRANDA, M. N. **Algoritmos Genéticos: Fundamentos e Aplicações**. [Online]. [http://www.gta.ufrj.br/~marcio/genetic.html] , 03 de março de 2005.

MITCHELL, M. **An Introduction to Genetic Algorithms**. MIT Press, Massachusets, London, England, 1996.

MOKOTOFF, E., JIMENO, J. L. Heuristics Based on Parcial Enumeration for the Unrelated Parallel Processor Scheduling Problem. **Annals of Operations Research**, v. 117, p.133-150, 2002.

MORO, L. F. L. **Técnicas de Otimização Mista-Inteira para o Planejamento e Programação de Produção em Refinarias de Petróleo**. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo, SP, Brasil, 2000.

MOROWITZ, H. **Beginning of Cellular Life**. New Haven, Conn, Yale University Press, 1992.

MORTON, T. E., PENTICO, D. W. **Heuristic Scheduling Systems with Applications to Production Systems and Project Management**. John Wiley & Sons, Inc. United States of America, 1993.

MUSSELMAN, K., O'REILLY, J. DUKET, S. The role of simulation in advanced planning and scheduling. **Proceedings of the 2002 Winter Simulation Conference**, v. 2, p. 1825-1830, 2002.

NAWAZ, J. E., ENSCORE, E. E., HAM, I. A heuristic algorithm for the m-machine, n-job sequencing problem. **Omega**, v. 11, p. 91- 95, 1983.

NEMHAUSER, G. L. WOLSEY, L. A. **Integer and Combinatorial Optimization**. John Wiley & Sons, Inc. United States of America, 1998.

NORDGREN, W. B. FlexSim simulation environment. **Proceedings of the 2003 Winter Simulation Conference**, v. 1, p. 197-200, 2003

ÓLAFSSON, S., KIM, J. Simulation Optimization. **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 79-84, 2002.

PAOLUCCI, M., SACILE, R., BOCCALATTE, A. Allocating crude oil supply to port and refinery tanks: a simulation-based decision support system. **Decision Support Systems**, v.33, p. 39-54, 2002.

PASUPATHY, T., RAJENDRAN, C., SURESH, R. K. A Multi-Objective Genetic Algorithm for Scheduling in Flow Shops to Minimize the Makespan and Total Flow Time of Jobs. **International Journal of Advanced Manufacturing Technology**, p. 1-12, published on line, 12 January 2005.

PEGDEN, C. D., SHANNON,R. E., SADOWSKI, R. P. **Introduction to Simulation using Siman**. McGraw-Hill, Inc, 2 ed., United States os America, 1995.

PEKNY, J. F., ZENTNER, M. G. Learning to Solve Process Scheduling Problems: the Role of Rigorous Knowledge Acquisition Frameworks. **Foundations of Computer Aided Process Operations**. Rippin, D. W. T., Hale J. C., Davis, J. F., Eds: Cache. Austin (TX), p 275-309, 1994.

PICHITLAMKEN, J., NELSON, B. L. A Combined Procedure for Optimization via Simulation. **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 292-300, 2002.

PIERA, M. A., NARCISO, M., GUASCH, A., RIERA, D. Optimization of Logistic and Manufacturing Systems through Simulation: A Colored Petri Net – Based Methodology, **Simulation**, v. 80, p. 121-129, 2004.

PINTO, J. M. **Planejamento e Programação de Operações de Produção e Distribuição em Refinarias de Petróleo**. Tese para a obtenção do título de Livre Docente. Universidade de São Paulo, São Paulo, 2000.

PINTO, J. M., GROSSMANN, I. E. Assignment and Sequencing Models for the Scheduling of Process Systems. **Annals of Operations Research**, v. 81, p. 443-466, 1998.

PRICE, R.N., HARREL, C. R. Simulation Modeling and Optimization Using ProModel. **Proceedings of the 1999 Winter Simulation Conference**, v. 1, p. 176-181, 1999.

RAMOS, I. C. O., GOLDBARG, M. C., GOLDBARG, E. F. G., DÓRIA NETO, A. D., FARIAS, J. P. F. Algoritmo ProtoG na Solução do Problema do Caixeiro Viajante. **Anais do XXXV Seminário Brasileiro de Pesquisa Operacional**, v. 1, p. 1590-1601, 2003a.

RAMOS, I. C. O., GOLDBARG, M. C., GOLDBARG, E. F. G., DÓRIA NETO, A. D., FARIAS, J. P. F. A ProtoG Algorithm Applie to the Traveling Salesman Problem. **Proceedings of XXIII International Conference of The Chilean Computer Science Society**, v. 1, p. 23-30, 2003b.

REEVES, C. R. **Modern Heuristic Techniques for Combinatorial Problems**, McGraw-Hill Book Company, London, 1995.

REJOWSKI JUNIOR, R. **Programação de Distribuição Dutoviária de Derivados de Petróleo**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo. São Paulo, 2001.

REKLAITIS, G. Overview of Scheduling and Planning of Batch Process Operations. **In Proceedings of the NATO Advanced Study Institute – Batch Process Systems Engineering**, Antalya, Turkey, p. 660-705, 1992.

RIBEIRO FILHO, G. LORENA, L. A. N. Algoritmo Genético Construtivo e Geração de Colunas: uma Aplicação para Coloração de Grafos, **Anais do XXXII Simpósio Brasileiro de Pesquisa Operacional**, Viçosa – MG, 2000.

ROHRER, M. AutoMod Product Suite Tutorial (AutoMod, Simulator, AutoStar) by AutoSimulation. **Proceedings of the 1999 Winter Simulation Conference**, v.1, p. 220-226, 1999.

ROHRER, M. Maximizing simulation ROI with AutoMod. **Proceedings of the 2003 Winter Simulation Conference**, v.1, p. 201-209, 2003.

SADOWSKI, D., BAPAT, V. The ARENA Product Family: Enterprise Modeling Solutions. **Proceedings of the 1999 Winter Simulation Conference**, v.1, p. 159-166, 1999.

SALHI, S., GAMAL, M. D. H. A Genetic Algorithm Base Approach for the Uncapacitated Continuous Location-Allocation Problem. **Annals of Operations Research**, v. 123, p. 203-222, 2003.

SALIBY, E. **Repensando a Simulação: A Amostragem Descritiva**. Editora Atlas S. A., São Paulo, 1989.

SHAH, N., PANTELIDES, C. C., SARGENT, R. W. H. Optimal Periodic Scheduling of Multipurporse Bathc Plants. **Annals of Operations Research**, v. 42, p. 193-228, 1993.

SHAH, N. Mathematical Programming Techniques for crude Oil Scheduling. **Computers Chemical Engineering**, v.20, p.1227-1232, 1996.

SHERALI, H. D., DRISCOLL, P. J. Evolution an State-of-the-Art in Integer Programming. **Journal of Computational and Applied Mathematics**, v. 124, p. 319-340, 2000.

SOUZA, D. O. **Algoritmos Genéticos Aplicados ao Planejamento do Transporte Principal de Madeira**. Dissertação de Mestrado. Universidade Federal do Paraná, Curitiba, Brasil, 2004.

SOUZANI, A. B. **Otimização da Produção de Plantas em Batelada com Operação por Turnos Intermitentes**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo. São Paulo, 1999.

SRINIVAS, M., PATNAIK, L. M. Adaptative Probabilities of Crossover and Mutation in Genetic Algorithms. **IEE Transactions on Systems, Man an Cybernetics**, v. 24, n. 4, p. 656-666, 1994.

STEBEL, S. L. **Modelagem da estocagem e Distribuição de GLP de uma Refinaria de Petróleo**. Dissertação de Mestrado. Centro Federal de Educação Tecnológica do Paraná, Curitiba, Brasil, 2001.

TAHA, H. A. **Integer Programming. Theory, Applications, and Computations**. Academic Press, Inc. Orlando, Florida, United States of America, 1975.

TAHA, H. A. **Operations Research. An Introduction**. Prentice Hall, Inc. Upper Saddle River, New Jersey, 1997.

TANOMARU, J. Motivação, Fundamentos e Aplicações de Algoritmos Genéticos. II **Congresso Brasileiro de Redes Neurais – III Escola de Redes Neurais**, p. 373-403, 1995.

TEWOLDEBERHAN, T. W., VERBRAECK, A., VALENTIN, E., BARDONNET, G. An Evaluation and Selection Methodology for Discrete-Event Simulation Software. **Proceedings of the 2002 Winter Simulation Conference**, v. 1, p. 67-75, 2002.

THONEY, K. A., JOINES, J. A., MANNINAGARAJAN, P., HODGSON, T. J. Rolling Horizon Scheduling in Large Job Shops. **Proceedings of the 2002 Winter Simulation Conference**, v. 2, p. 1891-1896, 2002.

TORABI, S. A., FATEMI GHOMI, S. M. T., KARIMI, B. A Hybrid Genetic Algorithm for the Finite Horizon Economic Lot and Delivety Scheduling in Supply Chains. **European Journal of Operational Research**, p. 1-16, In Press, avaible online 8 January 2005.

VAKHANIA, N. Single-Machine Scheduling with Release Times and Tails. **Annals of Operations Research**, v. 129, p.253-271, 2004.

VAMANAN, M., WANG, Q., BATTA, R., SZCZERBA, R. J. Integration of COTS software products ARENA & CPLEX for an inventory/logistics problem. **Computers & Operations Research**, v. 31q, p. 533-547, 2004.

VELLOSO, M. L. F., MENDONÇA, J. M., PACHECO, M. A. C., VELLASCO, M. M. B. R. Otimização e Planejamento de Horários por Algoritmos Genéticos, Anais do **II Congresso Brasileiro de Redes Neurais**, v.1, p. 35-39, 1995.

WANG, H. F., WU, K. Y. Modeling and Analysis for Multi-Period, Multi-Product and Multi-Resource Production Scheduling. **Journal of Intelligent Manufacturing**, v. 14, p. 297-309, 2003.

WANG, H. F., WU, K. Y. Hybrid Genetic Algorithm for Optimization Problems with Permutation Property. **Computers and Operations Research**, v. 31, n. 14, p. 2453-2471, 2004.

WANG, L., ZHANG, L., ZHENG, D. Z. A Class of Order-Based Genetic Algorithm for Flow Shop Scheduling. **Int. J. Adv. Manuf. Technol.** V. 22, p. 828-835, 2003.

WEI, L., ZHAO, M. A Niche Hybrid Genetic Algorithm for Global Optimization of Continuous Multimodal Functions. **Applied Mathematics and Computation**, v. 160, p. 649-661, 2005.

WILLIAMS, E. J., GUNAL, A. Supply Chain Simulation and Analysis with SimFlex^{TM}. **Proceedings of the 2003 Winter Simulation Conference**, v.1, p. 231-237, 2003.

WILLIAMS, H. P. **Model Building in Mathematical Programming**. John Wiley & Sons Ltd., England, 1999.

ZIONTS, S. **Linear and Integer Programming**. Prentice-Hall, Inc, Englewood Cliffs, New Jersey, United States of America, 1974.

ANEXO 1

Resultados para a metodologia AGEEH – PLIM (1ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
1	40	100	401	577980	577	6,45
2	40	100	490	657736	653	6,35
3	40	100	529	738167	731	6,285
4	40	100	222	324488	333	6,87
5	40	100	297	401542	407	6,81
6	40	100	301	437031	443	6,45
7	40	100	383	508083	511	6,4
8	40	100	881	1110925	1119	6,35
9	40	100	1515	1885585	1931	6,285
10	40	100	1301	1859595	1942	6,87
11	40	100	1401	2023706	2111	6,285
12	40	100	601	1026342	1029	6,35
13	40	100	801	1374802	1363	6,285
14	40	100	701	969130	934	6,75
15	40	100	721	989392	952	6,5
16	40	100	769	1033791	1001	6,45
17	40	100	801	1108579	1065	6,285
18	40	100	274	397427	399	6,87
19	40	100	595	875515	872	6,81
20	40	100	601	924663	917	6,3
21	40	100	818	1210462	1196	6,285
22	40	100	751	1018798	986	6,3
23	40	100	2079	2632667	2554	6,285
24	40	100	401	583570	567	6,3
25	40	100	801	1177097	1136	6,285
26	40	100	501	680098	642	6,3
27	40	100	2723	3361953	3242	6,285
28	40	100	443	653173	662	6,35
29	40	100	488	717703	721	6,285
30	40	100	501	786785	762	6,285
31	40	100	801	1103732	1097	6,4
32	40	100	826	1120679	1117	6,35
33	40	100	844	1144044	1137	6,3
34	40	100	3878	4783120	4704	6,285
35	40	100	360	512368	537	6,3
36	40	100	414	605464	638	6,285
37	40	100	901	1265872	1271	6,3
38	40	100	928	1298765	1305	6,285
39	40	100	487	687427	676	6,285
40	40	115	416	547627	521	6,87
41	40	115	636	851218	814	6,81
42	40	115	691	940886	901	6,285
43	40	115	440	578691	567	6,55
44	40	115	473	655524	635	6,45
45	40	115	576	806848	775	6,285
46	40	115	787	982073	962	6,285

Resultados para a metodologia AGEEH – PLIM (2ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
47	40	115	921	1251073	1238	6,81
48	40	115	1151	1628548	1641	6,45
49	40	115	1165	1655999	1667	6,3
50	40	115	1272	1820737	1834	6,285
51	40	115	1381	2020045	2073	6,3
52	40	115	1684	2424533	2494	6,285
53	40	115	691	962259	921	6,4
54	40	115	806	1140835	1096	6,285
55	40	115	473	718242	716	6,87
56	40	115	590	872765	881	6,81
57	40	115	813	1164481	1178	6,55
58	40	115	968	1366867	1378	6,4
59	40	115	1112	1553446	1567	6,3
60	40	115	1162	1633242	1654	6,285
61	40	115	691	879869	831	6,5
62	40	115	801	1000042	944	6,4
63	40	115	806	1043226	983	6,3
64	40	115	921	1217194	1146	6,285
65	40	115	713	983430	984	6,55
66	40	115	721	1009772	1007	6,285
67	40	115	462	644492	610	6,4
68	40	115	489	678140	642	6,3
69	40	115	1333	1611545	1563	6,285
70	40	115	691	853681	846	6,69
71	40	115	806	993041	984	6,45
72	40	115	1023	1258033	1238	6,3
73	40	115	1792	2151573	2139	6,285
74	40	115	369	490395	467	6,35
75	40	115	382	502007	479	6,3
76	40	115	572	709792	692	6,285
77	40	115	303	414036	415	6,285
78	40	115	346	512854	489	6,4
79	40	115	656	1043607	1006	6,3
80	40	115	798	1237898	1198	6,285
81	40	115	691	915426	899	6,69
82	40	115	715	935206	921	6,55
83	40	115	1151	1490566	1499	6,285
84	40	125	626	844355	854	6,5
85	40	125	637	859048	867	6,35
86	40	125	693	916679	927	6,3
87	40	125	1016	1351599	1370	6,285
88	40	125	376	506570	502	6,45
89	40	125	581	771148	756	6,285
90	40	125	376	516619	518	6,55
91	40	125	501	688496	690	6,45
92	40	125	626	847013	848	6,35
93	40	125	642	859275	861	6,285
94	40	125	376	502332	497	6,4
95	40	125	710	893849	888	6,285
96	40	125	626	904774	902	6,69
97	40	125	667	949793	950	6,35
98	40	125	749	1058044	1056	6,3
99	40	125	814	1169293	1169	6,285
100	40	125	501	691531	677	6,5
101	40	125	530	713769	697	6,45
102	40	125	533	715321	698	6,4
103	40	125	572	763314	742	6,285
104	40	125	376	480523	477	6,87
105	40	125	626	814068	796	6,5
106	40	125	832	1061972	1035	6,4
107	40	125	865	1097896	1068	6,3

Resultados para a metodologia AGEEH – PLIM (3ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
108	40	125	979	1259991	1230	6,285
109	40	125	592	705284	741	6,3
110	40	125	1098	1329345	1373	6,285
111	40	125	501	654036	644	6,87
112	40	125	532	688280	679	6,55
113	40	125	543	705503	694	6,45
114	40	125	558	719363	709	6,3
115	40	125	899	1155387	1156	6,285
116	40	125	626	832058	815	6,5
117	40	125	751	989270	969	6,35
118	40	125	1537	1853298	1827	6,3
119	40	125	2133	2490467	2482	6,285
120	40	125	843	1127708	1058	6,285
121	40	125	373	485945	494	6,87
122	40	125	527	755477	767	6,81
123	40	125	544	773287	784	6,55
124	40	125	615	852839	860	6,4
125	40	125	795	1087226	1089	6,285
126	40	125	501	1045644	1005	6,87
127	40	125	981	1309066	1285	6,81
128	40	125	1126	1569205	1546	6,3
129	40	125	1146	1589680	1566	6,285
130	40	125	376	481100	478	6,45
131	40	125	501	651198	651	6,3
132	40	125	1251	1614755	1570	6,285
133	40	125	501	660318	630	6,4
134	40	125	601	763165	729	6,3
135	40	125	626	828831	798	6,285
136	45	100	512	702355	683	6,87
137	45	100	801	1134688	1117	6,81
138	45	100	1001	1416873	1402	6,285
139	45	100	501	761350	720	6,285
140	45	100	301	462226	462	6,3
141	45	100	551	785919	785	6,285
142	45	100	336	459807	443	6,5
143	45	100	601	862762	842	6,3
144	45	100	1331	1785173	1766	6,285
145	45	100	501	689807	664	6,69
146	45	100	634	851953	830	6,285
147	45	100	701	963480	959	6,5
148	45	100	901	1273784	1344	6,285
149	45	100	201	296484	303	6,35
150	45	100	301	428488	431	6,285
151	45	100	401	589507	579	6,3
152	45	100	1201	1679334	1675	6,285
153	45	100	699	1025290	1011	6,3
154	45	100	721	1097798	1077	6,285
155	45	100	503	772939	806	6,69
156	45	100	512	778646	815	6,5
157	45	100	514	781843	820	6,45
158	45	100	604	917130	955	6,3
159	45	100	890	1263357	1310	6,285
160	45	100	501	666224	646	6,5
161	45	100	601	819997	806	6,4
162	45	100	622	847320	833	6,285
163	45	100	401	551438	528	6,3
164	45	100	533	725607	693	6,285
165	45	100	501	696747	696	6,3
166	45	100	701	975511	977	6,285
167	45	100	755	1110976	1093	6,87
168	45	100	801	1201952	1187	6,35

Resultados para a metodologia AGEEH – PLIM (4ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
169	45	100	919	1367362	1357	6,285
170	45	100	501	684305	646	6,3
171	45	100	976	1244383	1173	6,285
172	45	115	1036	1483851	1646	6,285
173	45	115	461	624357	618	6,45
174	45	115	617	818767	812	6,4
175	45	115	627	828414	822	6,285
176	45	115	806	1100797	1075	6,3
177	45	115	1216	1687962	1619	6,285
178	45	115	576	836942	893	6,35
179	45	115	700	999642	1067	6,3
180	45	115	768	1061616	1132	6,285
181	45	115	806	1021919	1027	6,285
182	45	115	401	523969	601	6,285
183	45	115	921	1360120	1466	6,87
184	45	115	1151	1768191	1868	6,285
185	45	115	576	771402	790	6,81
186	45	115	639	839302	857	6,75
187	45	115	801	1077795	1104	6,69
188	45	115	806	1118851	1149	6,285
189	45	115	314	405099	411	6,81
190	45	115	346	473692	483	6,55
191	45	115	570	717502	741	6,3
192	45	115	761	949145	978	6,285
193	45	115	231	328516	337	6,69
194	45	115	993	1311192	1368	6,55
195	45	115	1144	1494924	1573	6,45
196	45	115	1290	1721994	1822	6,35
197	45	115	1600	2041896	2148	6,285
198	45	115	674	909952	941	6,69
199	45	115	680	910642	946	6,5
200	45	115	691	959303	993	6,35
201	45	115	717	990891	1024	6,285
202	45	115	395	535894	523	6,69
203	45	115	411	542596	533	6,35
204	45	115	420	550517	541	6,285
205	45	115	806	1150471	1171	6,285
206	45	115	820	1127942	1088	6,81
207	45	115	857	1167223	1127	6,55
208	45	115	889	1207301	1163	6,4
209	45	115	915	1231806	1190	6,285
210	45	115	652	871201	864	6,285
211	45	125	738	1033338	1053	6,3
212	45	125	845	1181736	1202	6,285
213	45	125	693	998225	1001	6,87
214	45	125	948	1393451	1398	6,285
215	45	125	426	608118	599	6,3
216	45	125	514	735985	727	6,285
217	45	125	923	1350774	1335	6,3
218	45	125	964	1444550	1398	6,285
219	45	125	419	565797	529	6,285
220	45	125	364	448659	445	6,3
221	45	125	382	516270	511	6,285
222	45	125	869	1108540	1060	6,285
223	45	125	751	1050393	1008	6,75
224	45	125	870	1160435	1114	6,55
225	45	125	887	1218626	1170	6,4
226	45	125	934	1261394	1216	6,35
227	45	125	955	1287791	1234	6,3
228	45	125	1336	1745957	1676	6,285
229	45	125	376	505700	490	6,285

Resultados para a metodologia AGEEH – PLIM (5ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
230	45	125	561	782456	740	6,4
231	45	125	577	764391	761	6,3
232	45	125	704	930799	931	6,285
233	45	125	614	761083	718	6,4
234	45	125	670	877222	831	6,35
235	45	125	787	1019592	966	6,3
236	45	125	855	1071424	1022	6,285
237	45	125	342	431814	413	6,285
238	45	125	631	841229	827	6,3
239	45	125	923	1240619	1264	6,285
240	45	125	525	725650	736	6,81
241	45	125	532	731562	743	6,55
242	45	125	608	815756	830	6,45
243	45	125	610	823015	1256	6,3
244	45	125	629	895928	912	6,285
245	45	125	501	643954	651	6,35
246	45	125	626	820764	826	6,3
247	45	125	868	1137025	1146	6,285
248	50	100	486	677286	651	6,45
249	50	100	501	729784	700	6,4
250	50	100	564	793743	758	6,285
251	50	100	405	572699	567	6,87
252	50	100	596	815760	823	6,3
253	50	100	788	1096931	1118	6,285
254	50	100	601	865010	868	6,4
255	50	100	701	1009769	1013	6,3
256	50	100	902	1277454	1276	6,285
257	50	100	533	748157	701	6,35
258	50	100	564	789536	727	6,3
259	50	100	734	1019228	963	6,285
260	50	100	494	731623	738	6,285
261	50	100	401	628670	624	6,5
262	50	100	801	1195281	1183	6,285
263	50	100	401	594485	579	6,285
264	50	100	701	985545	931	6,3
265	50	100	1747	2413736	2355	6,285
266	50	100	401	620782	609	6,75
267	50	100	424	643788	634	6,285
268	50	100	584	824134	805	6,45
269	50	100	632	918610	897	6,35
270	50	100	641	935394	911	6,3
271	50	100	651	948725	924	6,285
272	50	100	401	592842	578	6,75
273	50	100	418	611459	601	6,285
274	50	100	401	602068	581	6,55
275	50	100	598	867894	854	6,285
276	50	100	701	1069746	1067	6,285
277	50	100	701	1082325	1007	6,285
278	50	100	901	1188392	1138	6,75
279	50	100	1001	1336006	1305	6,45
280	50	100	1011	1343044	1288	6,4
281	50	100	1101	1496189	1436	6,285
282	50	115	231	328836	311	6,87
283	50	115	889	1224197	1203	6,285
284	50	115	921	1240906	1191	6,69
285	50	115	1266	1720177	1681	6,285
286	50	115	231	355465	333	6,285
287	50	115	1036	1420233	1357	6,35
288	50	115	1266	1776109	1716	6,285
289	50	115	691	1005769	979	6,4
290	50	115	806	1176638	1139	6,285

Resultados para a metodologia AGEEH – PLIM (6ª parte)

Nº	Tamanho da População	Nº de Iterações para Hibridização	Nº de Iterações do AG	Nº de Iterações do Lingo	Tempo Computacional (segundos)	Valor da Função de Aptidão
291	50	115	346	492347	486	6,69
292	50	115	503	705703	708	6,45
293	50	115	671	925166	940	6,3
294	50	115	714	1012298	1048	6,285
295	50	115	406	538687	525	6,45
296	50	115	408	540007	525	6,35
297	50	115	861	1049690	1028	6,3
298	50	115	1916	2220939	2222	6,285
299	50	115	367	533241	522	6,69
300	50	115	573	756399	754	6,55
301	50	115	576	800382	801	6,3
302	50	115	1052	1407984	1507	6,285
303	50	115	323	424133	397	6,285
304	50	115	461	666176	702	6,35
305	50	115	576	823092	869	6,285
306	50	115	116	171403	175	6,45
307	50	115	586	905964	877	6,4
308	50	115	702	1069420	1030	6,3
309	50	115	1618	2140387	2119	6,285
310	50	115	691	999137	970	6,35
311	50	115	1036	1436586	1394	6,3
312	50	115	1873	2483081	2397	6,285
313	50	115	691	968777	929	6,4
314	50	115	806	1112214	1071	6,35
315	50	115	874	1184729	1142	6,3
316	50	115	1878	2440596	2453	6,285
317	50	115	576	820556	818	6,4
318	50	115	760	1039682	1032	6,35
319	50	115	856	1183284	1174	6,285
320	50	115	921	1218504	1176	6,285
321	50	125	126	183379	179	6,87
322	50	125	850	1125303	1122	6,81
323	50	125	1039	1402865	1409	6,75
324	50	125	1251	1669260	1682	6,35
325	50	125	1626	2150412	2167	6,3
326	50	125	1658	2183839	2202	6,285
327	50	125	417	613483	587	6,285
328	50	125	501	711158	684	6,3
329	50	125	837	1140490	1090	6,285
330	50	125	367	487119	468	6,35
331	50	125	466	637304	607	6,285
332	50	125	626	852079	854	6,35
333	50	125	751	1033320	989	6,285
334	50	125	376	545936	563	6,285
335	50	125	441	593770	580	6,285
336	50	125	709	922337	865	6,4
337	50	125	764	1016632	956	6,3
338	50	125	1203	1513500	1460	6,285
339	50	125	876	1213109	1176	6,35
340	50	125	884	1218767	1184	6,285
341	50	125	751	1064856	1015	6,285
342	50	125	844	1151438	1148	6,285
343	50	125	626	804525	773	6,285
344	50	125	626	892245	847	6,285
345	50	125	1376	1761510	1775	6,75
346	50	125	1501	1936680	1956	6,55
347	50	125	1533	1980054	2028	6,3
348	50	125	1989	2615609	2661	6,285
349	50	125	751	988553	963	6,45
350	50	125	944	1218693	1176	6,3
351	50	125	1222	1587158	1537	6,285

ANEXO 2

Resultados da metodologia Algoritmo ProtoG – PLIM (1ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
1	20	6	5	153	525243	636	6,3
2	20	6	5	164	558470	675	6,285
3	20	6	5	91	296723	342	6,285
4	20	6	5	154	517223	625	6,285
5	20	6	5	60	225118	256	6,87
6	20	6	5	70	258669	292	6,69
7	20	6	5	72	262920	298	6,535
8	20	6	5	128	446967	503	6,285
9	20	6	5	198	671253	775	6,87
10	20	6	5	203	689824	794	6,69
11	20	6	5	226	763643	878	6,45
12	20	6	5	230	775549	893	6,35
13	20	6	5	320	1080106	1241	6,285
14	20	6	5	87	307066	365	6,35
15	20	6	5	120	415124	485	6,285
16	20	6	5	24	97407	113	6,81
17	20	6	5	43	164649	194	6,69
18	20	6	5	49	182730	217	6,35
19	20	6	5	56	212606	248	6,3
20	20	6	5	342	1171209	1329	6,285
21	20	6	5	71	248741	278	6,55
22	20	6	5	118	402143	447	6,5
23	20	6	5	119	405372	450	6,35
24	20	6	5	168	572953	634	6,3
25	20	6	5	192	656710	736	6,285
26	20	6	5	179	621599	716	6,3
27	20	6	5	338	1149873	1322	6,285
28	20	6	5	45	170913	186	6,87
29	20	6	5	70	246856	272	6,5
30	20	6	5	74	259746	287	6,35
31	20	6	5	82	292820	323	6,285
32	20	6	5	211	716669	789	6,32
33	20	6	5	257	871405	963	6,285
34	20	6	5	183	647288	735	6,285
35	20	6	5	132	462405	523	6,35
36	20	6	5	145	513317	577	6,3
37	20	6	5	172	599936	674	6,285
38	20	6	5	62	231057	260	6,3
39	20	6	5	233	802043	913	6,285
40	20	6	5	203	690699	764	6,87
41	20	6	5	244	826180	913	6,55
42	20	6	5	249	842678	930	6,285
43	20	6	5	314	1057320	1170	6,35
44	20	6	5	318	1078264	1189	6,3
45	20	6	5	322	1085268	1200	6,285

Resultados da metodologia Algoritmo ProtoG – PLIM (2ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
46	20	6	6	42	167944	209	6,285
47	20	6	6	46	168427	198	6,75
48	20	6	6	63	223852	263	6,45
49	20	6	6	91	317877	380	6,285
50	20	6	6	128	444053	543	6,4
51	20	6	6	141	494948	603	6,3
52	20	6	6	152	531130	657	6,285
53	20	6	6	41	153442	188	6,5
54	20	6	6	57	206828	253	6,285
55	20	6	6	39	148542	181	6,35
56	20	6	6	136	480124	593	6,285
57	20	6	6	41	159786	195	6,5
58	20	6	6	54	202994	244	6,4
59	20	6	6	63	231858	278	6,35
60	20	6	6	78	288032	338	6,3
61	20	6	6	85	353264	392	6,285
62	20	6	6	70	253961	286	6,285
63	20	6	6	118	412433	492	6,81
64	20	6	6	149	514744	617	6,4
65	20	6	6	154	530370	635	6,35
66	20	6	6	155	540007	642	6,3
67	20	6	6	227	775574	925	6,285
68	20	6	6	98	346946	407	6,285
69	20	6	6	22	91929	105	6,4
70	20	6	6	44	166224	187	6,285
71	20	6	6	151	530701	644	6,3
72	20	6	6	205	709910	1029	6,285
73	20	6	6	79	273526	318	6,3
74	20	6	6	142	496117	585	6,285
75	20	6	6	15	67827	76	6,75
76	20	6	6	37	143726	162	6,69
77	20	6	6	63	229717	262	6,4
78	20	6	6	76	286065	321	6,285
79	20	6	6	74	266622	344	6,4
80	20	6	6	112	397336	494	6,285
81	20	6	6	38	143120	164	6,5
82	20	6	6	64	229833	272	6,45
83	20	6	6	71	252953	299	6,285
84	20	6	7	63	231672	266	6,75
85	20	6	7	73	266940	312	6,35
86	20	6	7	192	674086	782	6,3
87	20	6	7	211	727708	846	6,285
88	20	6	7	54	205941	241	6,285
89	20	6	7	58	211623	245	6,55
90	20	6	7	91	334898	380	6,285
91	20	6	7	83	304759	411	6,45
92	20	6	7	95	349024	463	6,4
93	20	6	7	103	372508	499	6,35
94	20	6	7	125	489625	644	6,285
95	20	6	7	102	377197	443	6,285
96	20	6	7	52	193078	235	6,45
97	20	6	7	53	209051	246	6,3
98	20	6	7	195	687975	813	6,285
99	20	6	7	69	268466	360	6,4
100	20	6	7	115	425031	549	6,285
101	20	6	7	65	245169	311	6,3
102	20	6	7	105	376779	477	6,285
103	20	6	7	68	244869	292	6,3
104	20	6	7	96	334886	400	6,285
105	20	6	7	81	288656	386	6,35

Resultados da metodologia Algoritmo ProtoG – PLIM (3ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
106	20	6	7	151	528811	672	6,35
107	20	6	7	167	583013	744	6,285
108	20	6	7	18	75445	95	6,45
109	20	6	7	21	86282	108	6,285
110	20	6	7	74	272116	351	6,55
111	20	6	7	91	333319	430	6,5
112	20	6	7	97	353939	455	6,4
113	20	6	7	103	374753	482	6,35
114	20	6	7	104	382600	487	6,3
115	20	6	7	108	399962	510	6,285
116	20	6	7	54	195574	244	6,35
117	20	6	7	105	389628	475	6,285
118	20	6	7	19	81686	100	6,81
119	20	6	7	24	100810	122	6,75
120	20	6	7	44	168625	204	6,3
121	20	6	7	112	414937	489	6,285
122	20	6	7	141	525865	598	6,87
123	20	6	7	159	590994	672	6,69
124	20	6	7	164	604593	690	6,45
125	20	6	7	182	669421	769	6,4
126	20	6	7	224	819212	985	6,4
127	20	6	7	231	836350	968	6,35
128	20	6	7	234	846931	980	6,285
129	30	9	5	155	795838	923	6,55
130	30	9	5	159	815800	944	6,5
131	30	9	5	182	930162	1076	6,45
132	30	9	5	215	1091154	1256	6,35
133	30	9	5	226	1145101	1316	6,285
134	30	9	5	100	520471	600	6,5
135	30	9	5	111	574914	659	6,4
136	30	9	5	148	762544	871	6,3
137	30	9	5	198	998694	1140	6,285
138	30	9	5	205	1041245	1192	6,5
139	30	9	5	216	1102502	1258	6,285
140	30	9	5	141	725650	869	6,45
141	30	9	5	178	913499	1093	6,3
142	30	9	5	285	1451748	1730	6,285
143	30	9	5	92	475826	569	6,35
144	30	9	5	128	654528	772	6,285
145	30	9	5	117	617320	718	6,87
146	30	9	5	118	618260	723	6,81
147	30	9	5	125	652752	763	6,55
148	30	9	5	162	841097	990	6,3
149	30	9	5	197	1015126	1200	6,285
150	30	9	5	112	586638	694	6,81
151	30	9	5	196	1003017	1185	6,3
152	30	9	5	283	1477012	1742	6,285
153	30	9	5	83	444277	506	6,5
154	30	9	5	107	564397	645	6,4
155	30	9	5	112	590697	675	6,285
156	30	9	5	29	166505	190	6,87
157	30	9	5	61	362446	402	6,81
158	30	9	5	75	474116	507	6,69
159	30	9	5	136	767208	849	6,35
160	30	9	5	190	1043989	1155	6,285
161	30	9	5	83	438398	491	6,75
162	30	9	5	106	549349	615	6,45
163	30	9	5	187	949210	1067	6,3
164	30	9	5	282	1420392	1598	6,285
165	30	9	5	111	578854	678	6,5

Resultados da metodologia Algoritmo ProtoG – PLIM (4ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
166	30	9	5	166	854479	1005	6,45
167	30	9	5	171	879948	1036	6,35
168	30	9	5	228	1207168	1404	6,3
169	30	9	5	319	1662413	1945	6,285
170	30	9	5	118	598845	710	6,45
171	30	9	5	153	772422	919	6,4
172	30	9	5	169	861315	1020	6,285
173	30	9	5	29	173549	207	6,285
174	30	9	5	26	153497	188	6,4
175	30	9	5	36	209342	252	6,3
176	30	9	5	67	357352	435	6,285
177	30	9	5	37	208621	249	6,45
178	30	9	5	89	468406	557	6,4
179	30	9	5	115	603721	716	6,3
180	30	9	5	125	652040	779	6,285
181	30	9	6	95	508317	608	6,285
182	30	9	6	95	500861	645	6,45
183	30	9	6	106	557054	712	6,4
184	30	9	6	131	691269	883	6,285
185	30	9	6	44	244142	302	6,45
186	30	9	6	98	535393	673	6,3
187	30	9	6	127	674044	844	6,285
188	30	9	6	67	363544	436	6,4
189	30	9	6	104	550932	645	6,35
190	30	9	6	160	846768	965	6,285
191	30	9	6	39	221885	276	6,4
192	30	9	6	81	431724	537	6,285
193	30	9	6	28	165090	199	6,5
194	30	9	6	54	302255	364	6,35
195	30	9	6	89	483830	577	6,3
196	30	9	6	162	867181	1063	6,285
197	30	9	6	103	531683	637	6,35
198	30	9	6	106	554102	660	6,3
199	30	9	6	124	634705	763	6,285
200	30	9	6	68	366092	444	6,35
201	30	9	6	174	918827	1127	6,3
202	30	9	6	344	1777875	2211	6,285
203	30	9	6	41	237508	269	6,3
204	30	9	6	76	413724	471	6,285
205	30	9	6	24	146688	167	6,285
206	30	9	6	40	223666	257	6,5
207	30	9	6	42	233537	269	6,4
208	30	9	6	59	318805	363	6,285
209	30	9	6	88	507423	603	6,45
210	30	9	6	90	519373	616	6,35
211	30	9	6	129	715808	847	6,3
212	30	9	6	142	786016	930	6,285
213	30	9	6	104	552237	641	6,45
214	30	9	6	155	813432	936	6,35
215	30	9	6	190	985941	1141	6,3
216	30	9	6	211	1102428	1277	6,285
217	30	9	6	132	700156	810	6,35
218	30	9	6	215	1130647	1290	6,3
219	30	9	6	316	1688855	1931	6,285
220	30	9	6	43	245557	274	6,69
221	30	9	6	73	388998	434	6,35
222	30	9	6	133	698738	780	6,3
223	30	9	6	160	831794	933	6,285
224	30	9	7	68	377860	449	6,35
225	30	9	7	97	524314	627	6,285

Resultados da metodologia Algoritmo ProtoG – PLIM (5ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
226	30	9	7	46	257250	315	6,75
227	30	9	7	47	261389	321	6,55
228	30	9	7	52	285560	349	6,35
229	30	9	7	84	489925	607	6,3
230	30	9	7	108	612003	760	6,285
231	30	9	7	19	117220	141	6,5
232	30	9	7	26	165046	191	6,285
233	30	9	7	33	191663	235	6,3
234	30	9	7	81	445085	543	6,285
235	30	9	7	57	312792	386	6,75
236	30	9	7	85	455238	567	6,35
237	30	9	7	104	552619	716	6,3
238	30	9	7	201	1114589	1397	6,285
239	30	9	7	10	70171	82	6,45
240	30	9	7	20	129277	151	6,3
241	30	9	7	31	180405	219	6,285
242	30	9	7	26	161590	201	6,81
243	30	9	7	40	231632	285	6,75
244	30	9	7	46	260662	324	6,45
245	30	9	7	48	277665	340	6,35
246	30	9	7	114	609718	742	6,285
247	30	9	7	34	196168	226	6,3
248	30	9	7	59	339793	407	6,285
249	30	9	7	50	292368	358	6,55
250	30	9	7	56	323900	395	6,4
251	30	9	7	80	444111	534	6,35
252	30	9	7	128	775596	907	6,3
253	30	9	7	167	975910	1156	6,285
254	30	9	7	84	462889	563	6,75
255	30	9	7	100	543044	669	6,45
256	30	9	7	103	558369	686	6,35
257	30	9	7	112	610356	747	6,3
258	30	9	7	160	855908	1074	6,285
259	30	9	7	43	251797	310	6,69
260	30	9	7	45	254067	322	6,35
261	30	9	7	79	432048	543	6,285
262	30	9	7	42	236861	285	6,45
263	30	9	7	124	673847	781	6,3
264	30	9	7	238	1332069	1548	6,285
265	30	9	7	50	278992	322	6,5
266	30	9	7	59	325218	381	6,45
267	30	9	7	67	366901	429	6,4
268	30	9	7	82	441029	514	6,3
269	30	9	7	85	461428	537	6,285
270	30	9	7	34	197618	239	6,285
271	30	9	7	42	240827	300	6,75
272	30	9	7	52	294970	362	6,3
273	30	9	7	191	1000223	1206	6,285
274	40	12	5	50	360841	410	6,75
275	40	12	5	86	599740	676	6,55
276	40	12	5	135	922011	1037	6,285
277	40	12	5	110	757861	864	6,75
278	40	12	5	117	807284	920	6,69
279	40	12	5	133	907824	1046	6,55
280	40	12	5	175	1225154	1409	6,4
281	40	12	5	234	1650698	1887	6,35
282	40	12	5	268	2983165	2136	6,35
283	40	12	5	277	1939808	2210	6,285
284	40	12	5	74	539885	605	6,3
285	40	12	5	161	1163747	1322	6,285

Resultados da metodologia Algoritmo ProtoG – PLIM (6ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
286	40	12	5	100	691822	782	6,87
287	40	12	5	127	864174	977	6,3
288	40	12	5	301	2016857	2278	6,285
289	40	12	5	82	587672	666	6,87
290	40	12	5	84	601436	682	6,69
291	40	12	5	89	632491	724	6,5
292	40	12	5	92	660243	752	6,285
293	40	12	5	122	838486	925	6,3
294	40	12	5	149	1015438	1123	6,285
295	40	12	5	100	711215	810	6,3
296	40	12	5	115	811098	924	6,285
297	40	12	5	90	627137	700	6,35
298	40	12	5	150	1034441	1171	6,3
299	40	12	5	188	1279943	1457	6,285
300	40	12	5	141	965747	1089	6,87
301	40	12	5	150	1023518	1155	6,69
302	40	12	5	172	1165885	1322	6,35
303	40	12	5	212	1468369	1657	6,3
304	40	12	5	216	1490971	1684	6,285
305	40	12	5	107	742030	817	6,35
306	40	12	5	164	1128811	1259	6,3
307	40	12	5	319	2211828	2498	6,285
308	40	12	5	131	985439	1096	6,81
309	40	12	5	167	1227254	1371	6,35
310	40	12	5	177	1296055	1446	6,285
311	40	12	5	21	169324	200	6,87
312	40	12	5	72	501347	571	6,55
313	40	12	5	177	1208400	1376	6,35
314	40	12	5	201	1377853	1562	6,3
315	40	12	5	348	2368215	2709	6,285
316	40	12	5	78	544867	635	6,35
317	40	12	5	140	956559	1106	6,285
318	40	12	5	57	414450	488	6,81
319	40	12	5	91	635211	755	6,69
320	40	12	5	103	710472	843	6,5
321	40	12	5	108	746851	883	6,4
322	40	12	5	142	969562	1151	6,35
323	40	12	5	144	986952	1169	6,3
324	40	12	5	266	1796972	2149	6,3
325	40	12	5	306	2058526	2466	6,285
326	40	12	5	57	417974	489	6,75
327	40	12	5	99	694210	820	6,55
328	40	12	5	101	708031	838	6,35
329	40	12	5	150	1031633	1209	6,285
330	40	12	6	13	113360	135	6,45
331	40	12	6	95	668946	792	6,35
332	40	12	6	110	769133	907	6,285
333	40	12	6	71	509985	592	6,87
334	40	12	6	76	537007	627	6,35
335	40	12	6	127	890748	1028	6,285
336	40	12	6	16	133790	169	6,4
337	40	12	6	70	498795	590	6,3
338	40	12	6	151	1038494	1227	6,285
339	40	12	6	139	968525	1158	6,4
340	40	12	6	180	1247730	1520	6,285
341	40	12	6	148	1050005	1223	6,35
342	40	12	6	157	1105570	1284	6,3
343	40	12	6	207	1450591	1684	6,285
344	40	12	6	54	395275	469	6,55
345	40	12	6	56	408955	484	6,35

Resultados da metodologia Algoritmo ProtoG – PLIM (7ª parte)

Nº	Tamanho População	Tamanho Sub-População	Tamanho Partícula Genética	Nº de Iterações do ProtoG	Nº de Iterações do Lingo	Tempo Computacional (s)	Valor da Função de Aptidão
346	40	12	6	116	813850	971	6,285
347	40	12	6	63	461403	554	6,435
348	40	12	6	83	590837	725	6,35
349	40	12	6	98	699692	857	6,3
350	40	12	6	268	1827422	2271	6,285
351	40	12	6	53	387198	479	6,285
352	40	12	6	96	672695	827	6,45
353	40	12	6	106	742509	912	6,285
354	40	12	6	27	206050	253	6,35
355	40	12	6	51	376490	459	6,285
356	40	12	6	67	489898	606	6,3
357	40	12	6	120	830450	1015	6,285
358	40	12	6	35	273341	345	6,285
359	40	12	6	45	338245	398	6,3
360	40	12	6	79	572239	661	6,285
361	40	12	6	29	227311	297	6,3
362	40	12	6	98	683620	858	6,285
363	40	12	6	52	386599	461	6,5
364	40	12	6	77	557857	653	6,3
365	40	12	6	91	652769	761	6,285
366	40	12	7	27	208793	251	6,45
367	40	12	7	59	430029	518	6,4
368	40	12	7	66	475859	571	6,35
369	40	12	7	89	629753	758	6,285
370	40	12	7	17	140626	172	6,69
371	40	12	7	24	184744	224	6,4
372	40	12	7	62	455288	553	6,3
373	40	12	7	64	469507	573	6,285
374	40	12	7	16	153504	185	6,285
375	40	12	7	82	590703	741	6,3
376	40	12	7	138	963372	1182	6,285
377	40	12	7	130	918333	1103	6,285
378	40	12	7	55	441479	547	6,35
379	40	12	7	57	455025	566	6,285
380	40	12	7	20	167076	217	6,75
381	40	12	7	27	210679	273	6,35
382	40	12	7	36	278394	351	6,285
383	40	12	7	23	192387	217	6,285
384	40	12	7	43	325347	398	6,5
385	40	12	7	71	520115	633	6,45
386	40	12	7	75	546188	667	6,35
387	40	12	7	86	660637	809	6,3
388	40	12	7	92	696826	854	6,285
389	40	12	7	10	100803	127	6,87
390	40	12	7	16	140058	180	6,75
391	40	12	7	26	210861	274	6,69
392	40	12	7	29	235427	307	6,285
393	40	12	7	78	562655	693	6,35
394	40	12	7	88	636111	780	6,285
395	40	12	7	38	295561	352	6,3
396	40	12	7	72	536253	651	6,285
397	40	12	7	36	279437	353	6,75
398	40	12	7	47	351301	443	6,35
399	40	12	7	70	513836	634	6,3
400	40	12	7	72	523065	660	6,285
401	40	12	7	84	610887	749	6,69
402	40	12	7	87	636448	776	6,3
403	40	12	7	112	798819	984	6,285
404	40	12	7	55	401124	519	6,4
405	40	12	7	72	524447	655	6,285