Universidade do Estado do Rio de Janeiro

Instituto Politécnico

Julliany Sales Brandão

Aplicação de algoritmos genéticos para minimização do número de objetos processados e o *setup* num problema de corte unidimensional

Nova Friburgo

2009

Julliany Sales Brandão

Aplicação de algoritmos genéticos para minimização do número de objetos processados e o *setup* num problema de corte unidimensional

Dissertação apresentada como requisito parcial para obtenção do título de Mestre, ao Programa de Pós – Graduação em Modelagem Computacional do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Orientadores: Prof. João Flávio V. de Vasconcellos Dr.Eng. Mec.

Prof. Wagner Figueiro Sacco D.Sc.

Nova Friburgo

2009

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/E

B817 Brandão, Julliany Sales.
Aplicação de algoritmos genéticos para minimização do número de objetos processados e o *setup* num problema de corte unidimensional / Julliany Sales Brandão. – 2009.
99 f.: il.

Orientadores: João Flávio Vieira de Vasconcellos e Wagner Figueiredo Sacco.
Dissertação (Mestrado) - Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Problema do corte de estoque – Teses. 2. Algoritmos genéticos – Teses. 3. Otimização combinatória – Teses. 4. Pesquisa operacional – Teses. I. Vasconcellos, João Flávio Vieira de. II. Sacco, Wagner Figueiredo. III. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. IV. Título.

CDU 519.874

_____ _____

Assinatura Data

Julliany Sales Brandão

Aplicação de algoritmos genéticos para minimização do número de objetos processados e o *setup* num problema de corte unidimensional

Dissertação apresentada como requisito parcial para obtenção do título de Mestre, ao Programa de Pós – Graduação em Modelagem Computacional do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Aprovada em 22 de maio de 2009.

Banca examinadora:

_____

Prof. João Flávio Vieira Vasconcelos, Dr. Eng. Mec. (Orientador)
Universidade do Estado do Rio de Janeiro. Instituto Politécnico

_____

Prof. Wagner Figueiredo Sacco, DSc. (Orientador)
Universidade do Estado do Rio de Janeiro. Instituto Politécnico

_____

Prof. Gustavo Platt, DSc.
Universidade do Estado do Rio de Janeiro. Instituto Politécnico

_____

Prof. Luiz Biondi Neto, DSc.
Universidade do Estado do Rio de Janeiro. UERJ

_____

Prof. Luiz Leduíno de Salles Neto, DSc.
Universidade Federal de São Paulo. UNIFESP

Nova Friburgo
2009

DEDICATÓRIA

À Deus e Nossa Senhora Aparecida, aos meus pais: Valmir e Suelene, aos meus irmãos: Susany, Ivonaldo e Hellen Luiza e ao meu namorado e amigo Felipe Carmo. Dedico.

AGRADECIMENTOS

Agradeço a Deus e Nossa Senhora pela saúde, amparo e proteção.

À Santa Rita de Cássia por ser exemplo de que nada é impossível para Deus.

Aos meus pais por me amarem e confiarem em mim e nos meus sonhos.

À minha irmã Susany Brandão por todo amor, companheirismo, amizade e apoio constante.

Ao meu irmão Ivonaldo por seu carinho e apoio em todos os momentos.

À amiga-irmã que ganhei com o mestrado, Alessandra Martins Coelho. Alê, muito obrigada pelos conselhos, ensinamentos pessoais e profissionais, pelas discussões e correções linha a linha deste trabalho, que foram de fundamental importância.

Ao meu amigo Rodrigo Golfeto, pelas discussões e paciência indispensáveis para a realização deste trabalho.

Ao meu namorado e amigo Felipe Carmo por todo apoio, carinho, incentivo e paciência durante os períodos mais difíceis para a concretização deste trabalho, em que eu fui a pessoa mais "chata" do mundo.

Ao Leduíno, meu grande amigo e eterno mestre, por acreditar em mim desde o começo e ser um dos maiores responsáveis por meu desenvolvimento profissional.

À minha madrinha Dalva Rabelo, pelos conselhos e orações.

Aos meus tios adotivos no Rio de Janeiro Sandra Regina Carmo e Jorge Carmo pelas orações e cuidados.

À Neci Coutrim pelo acolhimento e torcida.

Aos meus amigos Idário Bomfim, Magda Groman e Maria José Dias, que sempre se fizeram presente, mesmo com a distância física.

Aos orientadores João Flávio Vasconcellos e Wagner Sacco por todo apoio e compreensão.

Aos meus colegas do CEFET-RJ, principalmente, ao Marcelo Duarte, Raimundo Moraes, Rodrigo Callado, Hermanse Chaves, Márcio Ferreira e David Rodrigues por todo apoio, compreensão e incentivo.

Ao pessoal da manutenção, portaria e limpeza do IPRJ e ao Erlon Cordeiro, que nos momentos de solidão e saudades da minha família, me acolheram, protegeram e apoiaram. Vocês são inesquecíveis, obrigada por tudo.

Aos amigos que conquistei no IPRJ em especial, o Welton Menezes, Agnaldo Esquincalha, Rodrigo Fernandes, Antônio Espósito, Edson Sousa, Wagner Telles, Vitor Cortines, Leandro Oliveira, Nelson Barbosa, Alexander e Julhane Alice Thomas.

Aos funcionários da secretaria e biblioteca do IPRJ, sempre disponíveis, principalmente, a Sandra Mueller, Teresa Silva e Ana Clara Rangel por todo apoio e orientações.

Aos professores do IPRJ pela contribuição no meu desenvolvimento acadêmico: Luiz Nélio Henderson, Luiz Biondi, Marcus Pacheco, Antonio José da Silva Neto, Francisco Moura Neto, Gil de carvalho, e de modo especial e inesquecível, ao professor e diretor Juarez Muylaert (*in memorian*).

À FAPERJ pelo período de apoio financeiro.

À todos vocês, meu muito obrigada.

“ Deus nos fez perfeitos e não escolhe os capacitados, capacita os escolhidos.”

Albert Einstein

RESUMO

BRANDÃO, Julliany Sales. Aplicação de algoritmos genéticos para minimização do número de objetos processados e o *setup* num problema de corte unidimensional. 2009. 99 f. Dissertação (Mestrado em Modelagem Computacional) - Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2009.

Esta dissertação apresenta a aplicação de uma nova abordagem utilizando Algoritmo Genético na resolução do Problema de Corte Unidimensional na minimização de dois objetivos, geralmente conflitantes, o número de objetos processados e o *setup*, simultaneamente. O problema de corte consiste, basicamente, em encontrar a melhor maneira de obter peças de tamanhos distintos (itens) a partir do corte de peças maiores (objetos) com o objetivo de minimizar alguma espécie de custo ou maximizar o lucro. A disposição dos itens no objeto para a realização de cortes durante sua produção é denominada padrão de corte. E o *setup* é o tempo de preparação de máquina. O modelo do problema, a função objetivo e o método proposto denominado SingleGA, bem como os passos utilizados para sua resolução, também são apresentados. Os resultados obtidos pelo SingleGA são comparados com os métodos SHP, Kombi234, ANLCP300 e Symbio, encontrados na literatura, a fim de verificar a capacidade de encontrar soluções viáveis e competitivas. Os resultados computacionais mostram que o método proposto, o qual utiliza apenas um algoritmo genético para resolver esses dois objetivos inversamente relacionados, proporciona bons resultados.

Palavras-chave: Problema de corte unidimensional; Algoritmos genéticos; *Setup*; Otimização combinatória.

ABSTRACT

This dissertation presents the application of the one new approach using Genetic Algorithm in solving One-Dimensional Cutting Problems, in order to minimize two objectives, usually conflicting, the number of processed objects and setup, simultaneously. The cutting problem basically consists in finding the best way to obtain parts of distinct sizes (items) from the cutting of larger parts (objects) with the purpose of minimizing an specific cost or maximizing the profit. The disposition of the items on the object for the making of cuts during its production is called the cutting pattern. The *setup* is the machine's preparation time. The model problem, the objective function, the method denominated SingleGA and the steps used to solve the problem are also presented. The obtained results of the SingleGA are compared to the following methods: SHP, Kombi234, ANLP300 and Symbio, found in literature, verifying its capacity to find feasible and competitive solutions. The computational results show that the proposed method, which only uses a genetic algorithm to solve these two objectives inversely related, provides good results.

Keywords: One-dimensional cutting problem; Genetic algorithm; Setup; Combinatorial optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

1.1 Justificativa

Problema de Corte é uma denominação genérica para uma classe de problemas combinatórios que consiste em encontrar a melhor disposição dos itens (padrão de corte) de tamanhos distintos (itens) a partir do corte de peças maiores (objetos), buscando um objetivo específico (Salles Neto, 2005). Os Problemas de Corte vêm sendo largamente estudado pela comunidade científica e se tornado tema relevante para a Pesquisa Operacional.

Kantorovich (1960) foi um dos pioneiros na área de problemas de corte, com a formulação para minimizar perdas; mas, o grande avanço da área se deu com os trabalhos de Gilmore e Gomory (1961, p.849-859; 1963, p.863-888), respectivamente, estudando o problema de corte através do processo de geração de colunas. Haessler (1975) foi o primeiro a tratar o problema de corte unidimensional considerando a minimização do número de objetos processados e o tempo de preparação de máquina (*setup*).

Existem diversas abordagens na busca de solucionar os mais variados objetivos, adotadas para resolver um problema de corte. Segundo Salles Neto (2005), alguns estudiosos como Kantorovich (1960), Paull e Water (1954), Metzger (1958) e Eilon (1960) objetivavam minimizar as perdas minimizando o desperdício. Foerster e Wascher (2000) buscavam minimizar o *setup* de uma solução inicial. Arenales et al. (2006) deram enfoque ao reaproveitamento das sobras e outros, como Haessler (1975), Salles Neto (2005) e Golfeto et al. (2007a), tinham como objetivo minimizar o número de objetos processados e o *setup*. Apesar do vasto número de pesquisadores trabalhando nesta área, a gama de possibilidades é tão grande que existem muitos problemas práticos sendo resolvidos de modo rudimentar, pois, ainda não foram estudados com maior profundidade.

Os estudos dos problemas de corte têm sido estimulados pela necessidade das empresas em aperfeiçoar seus processos devido a competitividade entre as empresas, redução de desperdícios, custos e a eficiência nas entregas. Com isso estudar este tema se tornou

importante e de grande relevância para o planejamento da produção em diversas indústrias como a de vidro, papel, têxtil, química, entre outras.

Segundo Salles Neto (2005), a grande quantidade de aplicações e dificuldade de resolução dos problemas de corte, uma vez que a maioria pertencem à classe *NP*-completo, têm levado vários pesquisadores em todo o mundo a concentrarem esforços no desenvolvimento de novos e eficientes métodos de resolução, em sua grande maioria heurísticos.

Ferreira (1999) define heurística como um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas, fornecendo, em geral, soluções satisfatórias, em um período de tempo razoável. E, de acordo com Osman e Laporte (1996), referenciado em Blum e Roli (2003) e Coelho (2006) uma metaheurística é formalmente definida como um processo de geração iterativo, que guia uma heurística subordinada, combinando inteligentemente diferentes conceitos para explorar o espaço de busca, o qual deve ser modelado para um problema específico.

O vetor que motivou este trabalho foi a implementação via programação orientada a objeto de uma nova abordagem utilizando um único algoritmo genético com o objetivo de resolver o mesmo problema inicialmente abordado por Haessler (1975), baseando-se em uma técnica recente apresentada por Golfeto et al. (2007), na qual dois algoritmos genéticos que se interagem em uma relação mutualística têm o objetivo de minimizar o número de objetos processados e o *setup*. O Symbio teve grande influência na construção e elaboração do método proposto. A abordagem aqui apresentada difere da abordagem de Golfeto et al. (2007) pelo fato de utilizar apenas um algoritmo genético na minimização desses dois objetivos. A contribuição do método proposto reside no fato de não se ter conhecimento na literatura, até o momento, da implementação de um único algoritmo genético utilizado na minimização desses dois objetivos em simultâneo.

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho tem como objetivo geral o estudo e a aplicação de um único algoritmo genético para solucionar o Problema de Corte Unidimensional para a minimização do número de objetos processados e o *setup*.

1.2.2 Objetivos específicos

- Geração de padrões e as respectivas soluções pelo método proposto.
- Testar o método proposto com os 1800 problemas-testes do gerador de problemas CUTGEN1 desenvolvido por Gau e Wascher (1995).
- Analisar os resultados obtidos com o método Symbio de Golfeto et al. (2007a) e outros diferentes métodos propostos na literatura, verificando e comparando, por diversos ângulos, a eficiência de cada um deles em relação à qualidade da solução final produzida, bem como o tempo gasto na sua obtenção.

1.3 Proposta de Desenvolvimento

Visando a atender aos objetivos apresentados, será realizada uma revisão sobre Problemas de Corte, focando o estudo em Problemas de Corte unidimensionais. Além disso, é realizado um estudo sobre metaheurísticas, enfatizando os Algoritmos Genéticos com o objetivo de melhor aplicá-los no problema em questão.

Os resultados fornecidos pelo algoritmo proposto, para cada classe de problemas-teste, serão comparados com alguns métodos propostos na literatura como o SHP (Haessler, 1975), Kombi234 (Foerster e Wascher, 2000), ANLCP300 (Salles Neto, 2005) e o Symbio (Golfeto et al., 2007a).

1.4 Organização do trabalho

O presente trabalho está organizado em 7 capítulos que representam de modo genérico as fases do trabalho desenvolvido para o tema que titula esta dissertação.

O capítulo 2 apresenta uma revisão de literatura sobre Problemas de Corte. A seção 2.1 traz uma introdução sobre o assunto tratado neste capítulo. A subseção 2.2 apresenta o problema de corte unidimensional. A subseção 2.3 versa sobre o *setup* no problema de corte. Algumas técnicas para a resolução dos problemas de corte são mostradas na seção 2.4.

O capítulo 3 realiza uma revisão sobre a metaheurística Algoritmo Genético. A seção 3.1 faz uma introdução dos conceitos de otimização, heurísticas e metaheurísticas. A subseção 3.2 versa a respeito dos Algoritmos Genéticos. Na subseção 3.2.1 são apresentadas as características gerais de um Algoritmo Genético. Enquanto seus operadores básicos são abordados na subseção 3.2.2. A seção 3.3 versa sobre o desempenho dos Algoritmos Genéticos. Encerrando o capítulo, são descritos, na seção 3.4, os parâmetros gerais de um Algoritmo Genético.

É apresentado no capítulo 4, a biblioteca dos componentes de algoritmos genéticos em C++, GALib. Uma introdução ao assunto é realizada na seção 4.1. A seção 4.2 versa a respeito do GALib. Nas seções 4.3, 4.4 e 4.5 são apresentadas, respectivamente, as funções utilizadas para testes: função quadrática, função de Rosenbrock, função de Goldstein-Price.

O capítulo 5 apresenta a formulação matemática e como a metaheurística Algoritmo Genético proposta foi aplicada ao problema de corte em estudo. Uma introdução foi realizada na seção 5.1. Uma formulação para o problema em questão é apresentada na subseção 5.2. A seção 5.3 apresenta o Algoritmo Genético proposto, SingleGA. A estrutura dos indivíduos das populações é exposta na seção 5.4. A seção 5.5 apresenta os indivíduos soluções e suas características. A seção 5.6 expõe os critérios de parada pré-estabelecidos. O pseudocódigo do método proposto está descrito na seção 5.7.

Os resultados computacionais são descritos no capítulo 6 e uma conclusão sobre este trabalho bem como propostas futuras são apresentadas no capítulo 7.

2 PROBLEMA DE CORTE E EMPACOTAMENTO

2.1 Introdução

Com os recentes avanços computacionais, tornou-se importante e de grande relevância estudar os problemas de corte e empacotamento bem como modelos de otimização para o controle e planejamento de sistemas produtivos os quais vêm sendo estimulados por diversas indústrias, como a de vidro, papel, têxtil, química, entre outras, que visam tornar seus processos mais eficientes devido a competitividade entre as empresas, redução dos desperdícios, custos e a eficiência na entrega.

Foi na década de 40 que surgiram os primeiros modelos matemáticos para os problemas de corte os quais eram adequados apenas para problemas pequenos. No entanto, nos anos 60, surgiram as principais pesquisas e houve um significativo avanço nesta área (Cherri et al., 2006). Trata-se de problemas facilmente representados por modelos matemáticos, mas, devido à sua complexidade, pertencem à classe *NP*-Completo, e são difíceis de serem resolvidos. Por isso, as pesquisas nesta área caminham para técnicas heurísticas, ou seja, métodos que, embora a exploração seja feita de maneira algorítmica, o progresso é obtido pela avaliação puramente empírica do resultado, uma vez que técnicas exatas como método de enumeração explícita e variantes (por exemplo *branch and cut*, *branch and price* (Savelsbergh (1997) e Belov e Scheithauer (2006))), demandam um elevado tempo computacional, inviabilizando a resolução de problemas práticos os quais envolvem dezenas de itens a serem produzidos.

O problema de corte consiste, basicamente, em encontrar a melhor maneira de obter peças de tamanhos distintos (itens) a partir do corte de peças maiores (objetos) com o objetivo de minimizar alguma espécie de custo ou maximizar o lucro.

Dentre os primeiros trabalhos publicados sobre o tema destacam-se Kantorovich (1960) que produziu em 1939 um primeiro trabalho, publicado em 1960, com uma formulação para minimizar as perdas (*trim loss*) num problema de corte unidimensional. Segundo Salles Neto e Moretti (2005), problemas semelhantes foram tratados Paull e Walter

(1954), Metzger (1958) e Eilon (1960). No entanto, todos os métodos propostos adequavam-se apenas a problemas pequenos (Doswland e Doswland, 1992). Foi com os trabalhos de Gilmore e Gomory (1961, 1963), respectivamente, estudando o problema de corte através do processo de geração de colunas que houve um considerável avanço nos estudos dos problemas de corte.

Os Problemas de Corte são considerados problemas difíceis de serem resolvidos mas importantes para o planejamento das indústrias sendo considerados um tema de relevância para a Pesquisa Operacional, haja vista que existem grupos nacionais e internacionais dedicados exclusivamente ao tema, como o ESICUP (*Euro Special Interest Group on Cutting and Packing*), o GPCE (Grupo Brasileiro de Corte, Empacotamento e Correlatos) e a Oficina Nacional de Corte, Empacotamento e Correlatos que ocorre anualmente reunindo pesquisadores brasileiros e estrangeiros.

Padrão de corte é a disposição dos itens no objeto para a realização de cortes durante sua produção. Nota-se, neste caso, a importância da geometria, uma vez que as formas e dimensões dos itens e objetos determinarão os possíveis padrões de corte. Quanto a dimensão o problema de corte está dividido em uni, bi, tri e *n*-dimensionais (Dyckhoff, 1990). Dentre os problemas de corte unidimensionais estão problemas de corte de vigas, estruturas metálicas, canos, esquadrias, papel, vidro e etc. Exemplos de problemas de corte bidimensionais aparecem no corte de chapas, divisórias, compensados. O corte de retalhos em fábrica de tecidos representa um exemplo de problema de corte tridimensional (Miyazawa, 2009).

Para solucionar um Problema de Corte são necessários os padrões de corte e as quantidades de vezes que estes serão executados, chamadas de freqüências dos padrões.

Embora o objetivo geral em um Problema de Corte seja a minimização de perdas, ele pode ser utilizado visando a maximização de lucros, redução de objetos utilizados, tempo de produção e/ou uma combinação desses (Salles Neto e Moretti, 2005).

A quantidade de padrões de corte viáveis é muito grande, sendo a escolha dos padrões a serem utilizados uma etapa decisiva na resolução dos problemas de cortes. Além disso, quando se é necessário atender uma grande demanda num curto espaço de tempo, o custo com cada troca de padrão de corte, apresenta uma substancial importância, uma vez que são

necessários ajustes de máquinas e consequentemente, tempo e trabalho. Ao tempo de preparação de máquina, dá-se o nome de *setup*. O custo de *setup* é outro fator relevante no processo de corte. Por isso, torna-se interessante avaliar os efeitos destes dois objetivos parcialmente conflitantes: minimizar do número de objetos processados (minimização da entrada) e a minimização do número de padrões de corte (*setups*) para uma avaliação mais geral de custo.

Devido à grande quantidade e diversidade dos trabalhos publicados na literatura, Dyckhoff (Dyckhoff, 1990) os separou por grupos com objetivo de concentrar os estudos e propôs uma classificação dos Problemas de Corte e Empacotamento, fixando quatro características principais segundo a lógica do problema: dimensão, modo de alocação das unidades, sortimento de unidades grandes e sortimento de unidades pequenas.

Essa classificação permitiu concentrar as pesquisas e melhor definir os problemas. Cada uma das características apresenta suas subdivisões representadas pelos símbolos entre parênteses.

1. Dimensão

(1) Unidimensional;

(2) Bidimensional;

(3) Tridimensional;

(N) N-dimensional, com N>3.

2. Modo de Alocação das Unidades

(V) Seleção de unidades grandes

(B) Seleção de unidades pequenas

3. Sortimento de Unidades Grandes

(O) uma unidade;

(I) unidades de tamanhos iguais;

(D) unidades de tamanhos diferentes.

4. Sortimento de Unidades Pequenas

(F) poucas unidades de tamanhos diferentes;

(M) muitas unidades de muitos tamanhos diferentes;

(R) muitas unidades de poucos tamanhos diferentes;

(C) unidades de tamanhos iguais.

A combinação destas características resulta em 96 combinações diferentes e cada tipo pode apresentar restrições variadas.

A tipologia é indicada pela quádrupla $\alpha/\beta/\gamma/\delta$ onde α representa a dimensão do problema, β a forma de alocação, γ o sortimento de unidades grandes e δ o sortimento de unidades pequenas. Por exemplo, 3/V/D/R refere-se a um problema tridimensional, com seleção de unidades grandes, com sortimento de unidades grandes de tamanhos diferentes e muitas unidades pequenas de pouco tamanhos diferentes.

Quando não é fixada alguma especificação na tipologia do problema é porque esta não caracteriza ou não altera a sua lógica do problema, ou mesmo porque existem várias formulações e restrições diferentes referentes a especificação em branco para o mesmo problema. O problema da mochila 1/B/O, por exemplo, não especifica o tipo de sortimento para unidades pequenas, pois independente do tipo, a formulação do problema permanece inalterada. Em Salles Neto (2005) encontra-se um detalhamento sobre cada um destes problemas clássicos e sua respectiva classificação na tipologia de Dyckhoff (1990), como definição, exemplos e formulação.

O número de publicações cresceu consideravelmente nas últimas décadas e com isso, a tipologia de Dyckhoff (Dyckhoff, 1990) antes considerada como excelente instrumento para organizar as categorias dos problemas de corte e empacotamento passou a se tornar deficiente na categorização destes problemas. Para resolver esta deficiência Wascher et al. (2007) apresentou uma tipologia melhorada que está parcialmente baseada nas idéias de (Dyckhoff, 1990), mas introduz um novo critério de categorização e sugere um novo sistema consistente de nomes para estas categorias de problemas. Esta nova tipologia já está sendo adotada pelo ESICUP (*Euro Special Interest Group on Cutting and Packing*). A Figura 1, extraída de Wascher et al. (2007), representa os tipos de problemas básicos de acordo com a nova tipologia.

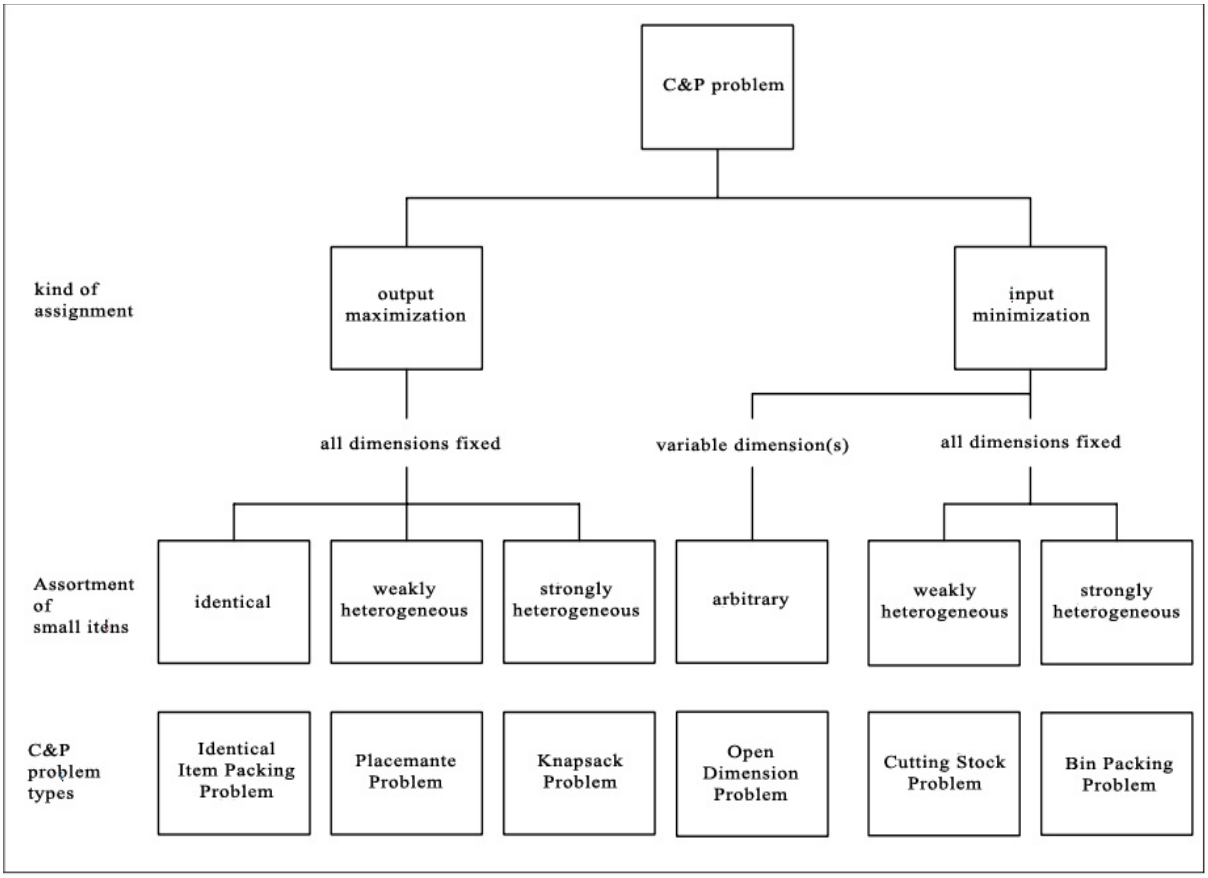


Figura 1 : Tipos básicos de problemas (extraída de Wascher et al. (2007))

Resumidamente, pode-se se classificar o problema de corte tratado neste trabalho, baseada na nova tipologia proposta por Wascher et al. (2007) quanto ao tipo de tarefa, sortimentos de itens pequenos e o tipo de problema. Seu tipo de tarefa é a minimização de entrada, quanto ao sortimentos de itens pequenos, possui todas as dimensões fixas cujos os itens podem ser fraco ou fortemente heterogêneos e o tipo de problema no qual está englobado é o *Cutting Stock Problem*.

2.2 Problema de Corte de Estoque Unidimensional

O problema de corte de estoque unidimensional consiste em decidir sobre a maneira de se cortar um conjunto de peças retangulares com demandas pré-estabelecidas, a partir de uma placa retangular (bobina) padrão com a finalidade de minimizar custo, tempo, perda de material envolvidos no processo de fabricação ou uma combinação destes (Carneiro, 1994).

Como o próprio nome sugere, o problema de corte unidimensional caracteriza-se pelo corte em apenas um sentido, ou seja, o corte é realizado em apenas uma dimensão. Mais especificamente, tem um número m de itens diferentes, com largura w_i, $i = 1,...,m$ que devem ser produzidos para atender cada demanda d_i partindo de uma bobina mestra de largura $W > w_i$ para todo i, através de cortes ao longo do seu comprimento (Salles Neto, 2005).

Segundo a tipologia de (Dyckhoff, 1990), o problema de corte unidimensional está classificado como (1/V/I/R) e na tipologia de Wascher et al. (2007) é identificado como SSSCSP (*Single Stock Size Cutting Stock Problem*) como mostra a Figura 2, extraída de Wascher et al. (2007). O SSSCSP inclui o clássico problema de corte unidimensional (Gilmore e Gomory, 1963; Wascher e Gau, 1996) cujos os objetos grandes são idênticos, ou seja, padrões de corte com mesmo tamanho geram itens de tamanhos distintos e menores considerados fracamente heterogêneos.

characteristics of large objects \ assortment of small items		weakly heterogeneous	strongly heterogeneous
all dimensions fixed	identical	Single Stock Size Cutting Stock Problem SSSCSP	Single Bin Size Bin Packin Problem SBSBPP
	weakly heterogeneous	Multiple Stock Size Cutting Stock Problem MSSCSP	Single Size Bin Packin Problem MBSBPP
	strongly heterogeneous	Residual Stock Size Cutting Stock Problem RCSP	Residual Bin Packin Problem RBPP
one large object variable dimension(s)		Open Dimension Problem ODP	

Figura 2: Tipos de problemas intermediários: minimização de entrada

À disposição dos itens na bobina mestre dá-se o nome de padrão de corte. A Figura 3 extraída de Salles Neto (2005) representa o Processo de Corte Unidimensional.

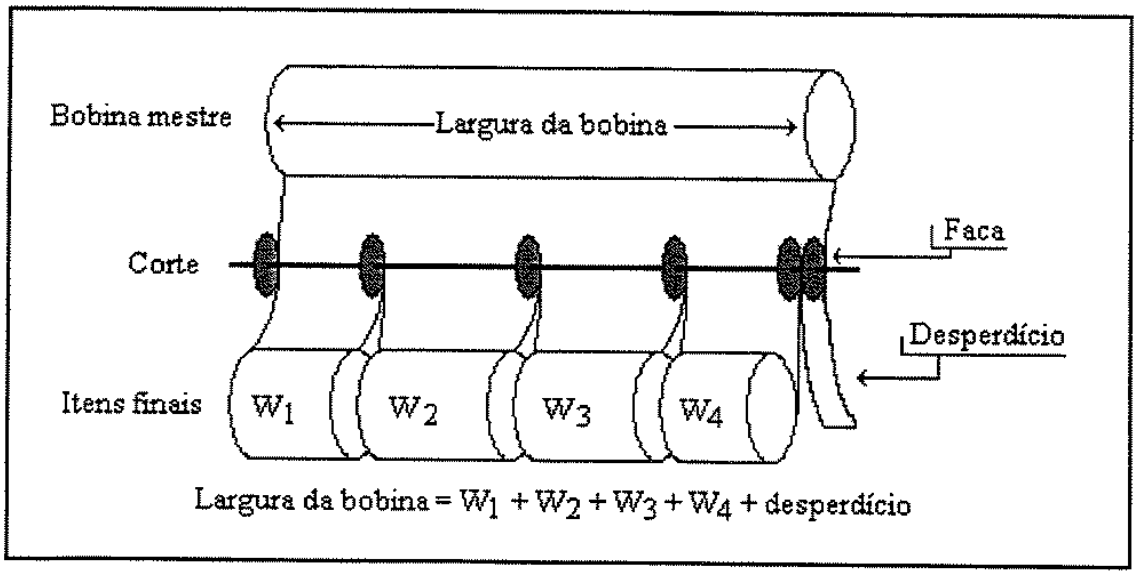


Figura 3 - Processo de Corte Unidimensional (extraída de Salles Neto, 2005)

Indústrias sejam elas têxtil, de papel, vidro entre outras, visam obter o maior lucro possível ao atender um pedido de um cliente, ou seja, buscam obter o menor custo possível para atender a demanda de produção, evitando o desperdício de matéria prima, bem como material em estoque, pois provavelmente haverá padrões de corte com sobras. A minimização do material desperdiçado torna-se importante para a lucratividade da empresa. Assim, para atender o propósito da indústria de minimizar os custos e verificando que o custo com desperdício é um fator relevante têm-se a seguinte formulação matemática para o Problema de Corte unidimensional para minimizar o desperdício (PCD), extraída de (Salles Neto, 2005), e representada em (2.1).

$$\begin{aligned} &\textit{Minimizar} \quad && c_1 \sum_{j=1}^{n} T_j x_j && \\ &\textit{Sujeito a:} \quad && \sum_{j=1}^{n} a_{ij} x_j = d_i \quad && i = 1,\ldots,m \\ &\qquad && x_j \in N \quad && j = 1,\ldots,n \end{aligned} \tag{2.1}$$

Onde:

$n =$ número de possíveis padrões de corte

m = número de itens diferentes

$c_1 =$ custo por metro desperdiçado

$x_j =$ número de bobinas processadas com o padrão de corte j

$$T_j = W - \sum_{i=1}^{m} a_{ij} w_i$$

W = tamanho da bobina mestre

$T_j =$ desperdício do padrão

$w_i =$ comprimento do item i

$a_{ij} =$ número de itens do tipo i no padrão j

$d_i =$ número de itens i demandados

O Problema de Corte para Minimizar o Número de Objetos Processados (PCOP) cuja formulação é apresentada em (2.2) pode ser considerada ao invés de minimizar o desperdício.

$$\textit{Minimizar} \quad c \sum_{j=1}^{n} x_j \tag{2.2}$$

$$\textit{Sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j = d_i \qquad i = 1,\ldots,m$$

$$x_j \in N \qquad\qquad j = 1,\ldots,n$$

onde c é o custo da bobina

Segundo Salles Neto (2005), quando se estima atender a demanda exata considerando qualquer excesso de produção como desperdício pode-se afirmar e também

demonstrar que a minimização de bobinas utilizadas é equivalente a minimizar o desperdício, conforme mostra a Proposição 2.1 extraída de Salles Neto (2005).

Proposição 2.1: Um vetor *x* é solução ótima do PCD se e somente se é também solução ótima do PCOP

Demonstração:

No PCD tem-se que minimizar a função $f(x) = c_1 \sum_{j=1}^{n} T_j \, x_j$.

Tem-se $T_j = W - \sum_{i=1}^{m} a_{ij} \, w_i$. Sem perda de generalidade pode-se assumir que $c_1 = 1$.

Assim $f(x) = \sum_{j=1}^{n} (W - \sum_{i=1}^{m} a_{ij} \, w_i) \, x_j \Rightarrow f(x) = \sum_{j=1}^{n} (W x_j - \sum_{i=1}^{m} a_{ij} \, w_i x_j)$.

Donde $f(x) = W \sum_{j=1}^{n} x_j - \sum_{i=1}^{m} w_i \sum_{j=1}^{n} a_{ij} \, x_j$.

Como $\sum_{j=1}^{n} a_{ij} \, x_j = d_i$ temos $f(x) = W \sum_{j=1}^{n} x_j - \sum_{i=1}^{m} w_i \, d_i$.

Logo, *x* minimiza $\sum_{j=1}^{n} T_j \, x_j$ se e somente se minimiza $\sum_{j=1}^{n} x_j$, como queria demonstrar.

Não é possível se trabalhar com todos os padrões de corte viáveis do ponto de vista computacional, uma vez que este número é elevado e tornaria qualquer método utilizado muito lento. Sendo assim, a escolha dos padrões avaliados é uma etapa difícil e decisiva na resolução de um problema de corte.

O Problema de Corte para Minimizar o Número de Objetos Processados (PCOP) pode apresentar algumas alterações na sua formulação básica a depender de outras características levadas em conta como, por exemplo, a limitação do número de bobinas na indústria, bem como os diferentes tamanhos das mesmas, e a tolerância à demanda. Esses exemplos são casos especiais do PCOP encontrados na literatura e que ocorre na prática a depender da necessidade e disponibilidade de determinadas indústrias. Mais detalhes podem ser encontrados em Salles Neto (2005).

2.3 O S*etup* no problema de corte

Na indústria, minimizar o número de objetos processados e/ou desperdício, na maioria das vezes, pode não ser suficiente. Quando uma demanda precisa ser atendida num curto espaço de tempo, o custo para trocas de padrões se torna relevante visto que estas trocas demandam um ajuste nas máquinas (facas para corte) e consequentemente, tempo e trabalho (Salles Neto e Moretti, 2005).

A minimização do custo de *setup* passa a ser mais um objetivo no processo de produção da indústria. Adicionando este objetivo ao PCOP, obtêm-se o Problema de Corte de Estoque para Minimizar o Número de Objetos Processados e o *Setup* (PCOPS), cuja formulação matemática geral, é indicada pela equação (2.3):

$$\textit{Minimizar } c_1 \sum_{j=1}^{n} x_j + c_2 \sum_{j=1}^{n} \delta(x_j) \qquad (2.3)$$

$$\textit{Sujeito a : } \sum_{j=1}^{n} a_{ij} x_j = d_i \qquad i = 1,...,m$$

$$x_j \in N \qquad j = 1,...,n$$

Onde c_2 é o custo de troca do padrão de corte

$$\delta(x_j) = \begin{cases} 1, \text{ se } x_j > 0 \\ 0, \text{ se } x_j = 0 \end{cases}$$

Conforme Teghem e seus colaboradores (Teghem et al.,1995), citado em Salles Neto e Moretti (2005), o número de padrões é que eleva o custo e não a freqüência com que ele é processado. O PCOPS apresenta dois objetivos conflitantes que é minimizar o número de objetos processados e o número de padrões distintos.

2.4 Aplicação de Heurísticas e Metaheurísticas para a resolução do PCOPS

Este tópico se propõe a apresentar alguns métodos utilizados para resolver o problema de corte com o objetivo de minimizar o número de objetos processados e o *setup.*

Haessler (1975) propôs o método objetivando a minimização do desperdício e o número de padrões utilizados, o qual é baseado numa técnica exaustiva (*Sequencial Heuristic Procedure* – SHP). São calculados a cada iteração, parâmetros definidos em Haessler (1975) e realizada um busca por padrões de corte que atendam a estes parâmetros. O primeiro parâmetro define o número de objetos que ainda devem ser processados para satisfazer as demandas residuais. O segundo define o número médio de itens obtidos em cada bobina. O processo é repetido até que todas as demandas sejam atendidas. Haessler (1975) e Salles Neto (2005) detalham esta técnica em seus respectivos trabalhos .

Em 2000, Foerster e Wascher (2000) desenvolveram uma heurística denominada Kombi que se baseia na combinação de padrões visando diminuir o *setup* de uma solução inicial. Ela tem por base o fato das somas das freqüências dos padrões resultantes serem as mesmas das freqüências dos padrões originais mantendo assim, o número de objetos processados constante, com uma possível redução de *setup*, ou seja, reduz-se o número de padrões para realização da mesma demanda original. O Kombi234, é uma variação deste método que se propõe a reduzir de quatro para três padrões, depois para dois e finalizar com apenas um padrão que atenda todas as restrições exigidas. No mesmo ano, Vanderbeck (2000) propôs o método exato *branch-cut-price* para modelar o problema de minimização de *setup* como um problema de programação inteira não-linear. O método consiste em inserir o método da geração de colunas (Gilmore e Gomory, 1961 e 1963), num algoritmo *branch-and-bound*. Este método resolve exatamente problemas de pequeno porte. No entanto, falha na obtenção de soluções ótimas em problemas de tamanhos médios e grandes (Salles Neto, 2005).

Umetami, Yagiura e Ibaraki (2003), desenvolveram o *Iterated Local Search algorithm with Adaptive Pattern Generations* (ILS-APG). Esta heurística baseia-se em fixar o número de padrões distintos (*setup*), buscando uma solução com desvio quadrático suficientemente pequeno dos elementos produzidos em relação à demanda. Utiliza-se uma heurística de

geração de padrões conhecida como (APG) devido a impossibilidade de tratar todos os padrões de corte viáveis.

Salles Neto (2005) desenvolveu o método ANLCP300, com o qual buscou-se minimizar o número de objetos processados e o *setup* através da aplicação do método lagrangiano aumentado num problema com função objetivo proposta por Haessler (1975), suavizada (Golfeto et al., (2007a)). Para o modelo não-linear o ANLCP300 adapta o método de geração de colunas de Gilmore e Gomory (1961,1963), e para a obtenção de uma solução viável utiliza uma heurística simples de arredondamento.

Recentemente, Golfeto e seus colaboradores (Golfeto et al.,(2007a)) desenvolveram o Symbio, baseada num processo simbiótico entre duas populações de espécies diferentes de algoritmos genéticos, gerando seus próprios padrões em conjunto com soluções para o problema. Este trabalho foi o pioneiro na utilização de um algoritmo genético que tratou a relação simbiótica entre espécies diferentes em problemas de otimização. Segundo Golfeto et al. (2007a), tal abordagem inédita, permite que duas populações, de soluções e de padrões, evoluam não só simultaneamente, mas de maneira simbiótica, isto é, um padrão que está presente em boas soluções tem grande possibilidade de sobreviver e gerar novos indivíduos que, por sua vez, podem ajudar a melhorar a população de soluções.

Neste capítulo foi realizado um estudo sobre problemas de corte, enfatizando o problema de corte unidimensional. Foi abordada a importância do *setup* na realização da produção de uma empresa. E para finalizar, apresentou-se alguns métodos para a resolução do problema de corte unidimensional com o objetivo de minimizar o número de objetos processados e o *setup*.

3 ALGORITMOS GENÉTICOS

3.1 Introdução

Segundo Deb (2001), referenciado em Coelho (2006) Otimização é o processo de encontrar e comparar soluções factíveis até que nenhuma solução melhor possa ser encontrada. Estas soluções são ditas boas ou ruins em termos de um objetivo.

Em geral, existe um grande número de problemas de otimização onde não é possível obter-se o ótimo global em tempo polinomial necessitando, portanto, de métodos heurísticos. Enquadram-se nesta categoria os problemas de corte, caixeiro viajante, problema de roteamento de veículos, entre outros.

Os métodos de resolução heurísticos objetivam encontrar soluções de boa qualidade, mesmo que essas não sejam necessariamente ótimas, em um tempo aceitável.

Heurísticas, em sua maioria, são utilizadas para tratar os problemas de maneira específica enquanto que metaheurísticas apresentam um comportamento mais geral que podem ser utilizadas por uma maior quantidade de problemas.

Segundo Glover e Kochenberger (2003) Metaheurísticas em sua definição original, são métodos de solução que realizam a interação entre melhorias locais e estratégias de alto nível a fim de criar um processo capaz de fugir da solução ótima local e buscar de maneira robusta um espaço solução. Ao longo do tempo, esses métodos também passaram a incluir quaisquer procedimentos que empregam estratégia para superar a armadilha da solução ótima local em espaços solução complexos, principalmente os procedimentos que utilizam uma ou mais estruturas de vizinhança como meio de se definir movimentos admissíveis de transição entre soluções, ou para construir ou destruir soluções em processos construtivos e destrutivos. Coelho (2006) descreve que as principais propriedades que uma metaheurística deve apresentar são: simplicidade, coerência, eficiência, robustez, facilidade de uso e inovação. No entanto, em geral, as metaheurísticas possuem apenas algumas destas propriedades. Para um

estudo mais aprofundado sobre o tema, consultar Blum e Roli (2003), Glover e Kochenberger (2003) e Coelho (2006).

Existem várias maneiras de classificar uma metaheurística. Quanto ao número de soluções simultâneas, as metaheurísticas podem ser classificadas como populacionais e não-populacionais. As metaheurísticas não-populacionais trabalham com apenas uma solução descrevendo uma trajetória num espaço de busca durante sua execução. Como exemplos desta categoria, estão: Busca Tabu (Glover, 1997), GRASP (Feo e Resende, 1995), *Iterated Local Search* (Lourenço, 2002) e *Simulated Annealing* (Kirkpatrick, 1983). Já as metaheurísticas populacionais descrevem a evolução de um conjunto de soluções no espaço de busca que, por sua vez, consistem em manter um conjunto de boas soluções e combiná-las na tentativa de produzir soluções ainda melhores (Coelho, 2006). Enquadram-se nesta última classe os Algoritmos Genéticos (Michalewicz, 1996 e 2000), assunto chave deste trabalho.

3.2 Algoritmos genéticos

Algoritmos genéticos são algoritmos de busca e otimização de soluções baseado nos mecanismos genéticos e evolucionários dos seres vivos, como a seleção natural e a sobrevivência do mais apto introduzidos por Charles Darwin no clássico "*The Origin of Species*" (1859).

Segundo o princípio básico da evolução natural de Darwin, os indivíduos mais aptos terão maiores chances de sobreviverem e transmitirem seus códigos genéticos para as gerações futuras. Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.

Nos anos 50 e 60, muitos biólogos começaram a desenvolver simulações computacionais de sistemas genéticos. No entanto, foi John Holland que começou a desenvolver as primeiras pesquisas sobre Algoritmos Genéticos e após refinar suas idéias, publicou em 1975 seu livro *Adaptation in Natural and Artificial Systems*, considerado hoje a "Bíblia" dos Algoritmos Genéticos.

Os Algoritmos Genéticos introduzidos rigorosamente por John Holland (Holland, 1975) trabalham com uma população de indivíduos, onde cada um representa uma solução

possível para um dado problema. A cada indivíduo é dado uma aptidão, isto é, um valor que quantifica a adaptabilidade do indivíduo ao meio (problema tratado). Os indivíduos com maiores aptidões possuem maiores oportunidades de serem escolhidos para a reprodução através da recombinação do material genético e com isso difundir suas características pelas gerações seguintes propiciando que áreas mais promissoras de busca sejam exploradas levando o algoritmo genético, na maioria dos casos, à convergência pela solução ótima do problema.

Os Algoritmos Genéticos utilizam um processo adaptativo e paralelo de busca de soluções em problemas complexos. É adaptativo, pois as soluções correntes influenciam na busca de soluções futuras. O paralelismo decorre do fato de que a cada momento um conjunto de soluções é considerado. Esta é uma técnica robusta e versátil necessitando apenas de informações sobre sua aptidão para encontrarem soluções do problema.

Segundo Lucas (2002), Holland (1975) inspirou-se na maneira como o darwinismo explica o processo de evolução das espécies para decompor o funcionamento dos Algoritmos Genéticos nas etapas de inicialização, avaliação, seleção, recombinação, mutação, atualização e finalização, conforme apresentado na Figura 4, adaptada de (Lucas, 2002).

O Algoritmo Genético, basicamente, cria uma população de possíveis respostas para o problema a ser tratado (inicialização) e depois a submete ao processo de evolução formado pelas etapas de avaliação, seleção, recombinação, mutação, atualização e finalização (Lucas, 2002).

Carvalho (2008) difere os Algoritmos Genéticos dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

1. Algoritmos Genéticos trabalham com uma codificação de conjunto de parâmetros e não com os próprios parâmetros.
2. Algoritmos Genéticos trabalham com uma população e não com um único ponto.
3. Algoritmos Genéticos utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar.
4. Algoritmos Genéticos utilizam regras de transição probabilísticas e não determinísticas.

Figura 4 – Estrutura do funcionamento de um algoritmo genético tradicional

Os Algoritmos Genéticos desde os conceitos básicos realizados por Holland (1962) vêm sendo aplicados em várias áreas de pesquisa e em situações do mundo real como: otimização de funções matemáticas, otimização combinatória, otimização de planejamento, problemas de corte e empacotamento, problema do caixeiro viajante, problema de otimização de rota de veículos, otimização de distribuição e logística, seleção de variáveis em mineração de dados, melhorias em telecomunicações, na evolução musical, otimização de plantão médico hospitalar entre outros, com bons resultados.

A seguir, será apresentada a Tabela 3.1, que faz uma analogia entre Algoritmos Genéticos e a natureza, adaptada de Pacheco (1999).

Tabela 3.1: Analogia dos algoritmos genéticos com a natureza

Natureza	**Algoritmos Genéticos**
Meio ambiente	Problema
Gene	Característica do problema
Cromossomo/ Genoma	- Coleção de genes - Estrutura de dados que codifica a solução de uma problema
Alelo	Valor da característica
Loco	Posição na palavra, vetor
Genótipo	Estrutura
Fenótipo	Cromossomo codificado
Indivíduo	Solução
Geração	Iteração completa do GA que gera uma nova população, ciclo
População	Conjunto de indivíduos no espaço de busca
Reprodução Sexual	Operador de recombinação
Mutação	Operador de mutação

3.2.1 Características gerais de um algoritmo genético

Algoritmos Genéticos são algoritmos de otimização global baseados nos mecanismos de solução natural e da genética. Apesar de aleatórios, exploram informações históricas. Cada iteração é chamada geração de indivíduos. Os indivíduos com maiores adaptações têm mais chances de se reproduzirem.

As representações são genotípicas e utilizam vetores de tamanho finito em um alfabeto finito (genótipo). Os elementos podem ser combinados formando as características reais do indivíduo (fenótipo).

De modo geral, Lucas (2002) destaca e explica algumas características gerais dos Algoritmos Genéticos, apresentadas abaixo:

- **Busca codificada:** a resolução de um problema necessita de um conjunto de soluções viáveis para ser codificado em uma população de indivíduos.
- **Generalidade:** garante a fácil adaptabilidade para problemas variados uma vez que apenas a representação e avaliação dos indivíduos necessitam de informações específicas do problema abordado.
- **Paralelismo explícito:** é justificado por cada indivíduo da população representar um ser isolado, avaliado de modo independente e selecionados de modo concorrente.
- **Paralelismo implícito:** consiste na capacidade de avaliar combinações ou esquemas mais ou menos desejáveis.
- **Busca estocástica:** o Algoritmo Genético não apresenta um comportamento determinístico. Mas também, não pode ser considerado um método totalmente aleatório pois ele guarda informações históricas das gerações anteriores.
- **Busca cega:** em nome da generalidade, a busca cega ignora o significado das estruturas que manipulam e como trabalhar com elas, o que por sua vez, afeta a eficiência.
- **Eficiência mediana:** é justificada pela busca cega. Na tentativa de minimizar o problema, técnicas de hibridação vêm sendo utilizadas.
- **Facilidade no uso de restrições:** As restrições podem apresentar diferentes graus de importância. Os indivíduos que violam uma restrição mais flexível são considerados mais aptos que aqueles que violam restrições mais graves.

3.2.2 Operadores básicos de um algoritmo genético

A seguir, serão apresentados os componentes principais que caracterizam e descrevem o funcionamento de um algoritmo genético, conforme ilustrado na Figura 4.

3.2.2.1 Definição da população inicial

A inicialização da população determina o processo de criação dos indivíduos para o primeiro ciclo do algoritmo. Tipicamente, utiliza-se para esta criação a inicialização aleatória.

Cada indivíduo representará uma possível solução para o problema, ou seja, um ponto no espaço de soluções.

Existem alternativas ao método aleatório para suprir deficiências provocadas pela criação aleatória de indivíduos de representação mais complexas, como por exemplo, a utilização de heurísticas para a geração da população inicial (Lucas, 2002).

3.2.2.2 Cálculo de aptidão

Após a inicialização da população, começa o processo de evolução com a avaliação da população com o cálculo de aptidão.

O cálculo de aptidão é a avaliação do indivíduo realizada pela função objetivo que determina quão bom e (quão) apto é o indivíduo para um dado problema, atribuindo um valor calculado como medida de qualidade (aptidão) para o problema analisado. Os indivíduos são ordenados conforme sua aptidão (Lindem, 2006).

3.2.2.3 Seleção

Nesta fase são escolhidos os indivíduos privilegiando os mais aptos para permanecer e se multiplicar na população. Uma maneira comum de seleção é aquela em que a probabilidade de um indivíduo ser selecionado na próxima geração é proporcional à sua aptidão (método da roleta viciada).

Indivíduos com maiores aptidões possuem um espaço maior na roleta e consequentemente têm maiores possibilidades de serem escolhidos para o processo de reprodução.

Assim, se f_i é a avaliação do *i*-ésimo indivíduo da população corrente, a probabilidade p_i (aptidão relativa) deste indivíduo ser selecionado é proporcional a (3.1):

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}, \qquad (3.1)$$

onde *n* é o número de indivíduos da população.

No método da roleta os indivíduos são organizados em ordem crescente de aptidão. Além deste método de seleção existem muitos outros, dentre os mais conhecidos estão o elitismo e o *Steady state* (Linden, 2006).

O elitismo consiste em selecionar os melhores indivíduos de cada geração garantindo suas presenças na geração seguinte, com o objetivo de garantir a preservação dos seus genomas. Isso garante que, no pior caso, o melhor indivíduo da geração *t*+1 é igual ao melhor indivíduo da geração *t* (quando nenhum indivíduo melhor é criado). Com isso, pode-se garantir que o gráfico de avaliação do melhor indivíduo como função do número de gerações decorridas é uma função crescente (Linden, 2006). Este procedimento, aparentemente simples, colabora notavelmente para a melhoria do desempenho da execução de um algoritmo genético. Esta melhoria é justificada pelo fato das melhores soluções serem mantidas, aumentando o componente de memória do algoritmo genético. Consequentemente, aumenta-se o índice de aproveitamento, sem prejudicar o componente de exploração do método que continua a atuar no restante da população. Essa técnica de seleção força, portanto, a cópia do melhor indivíduo a cada geração na próxima geração reduzindo o efeito aleatório do processo seletivo, garantindo o melhor membro de uma população na população seguinte.

Segundo Linden (2006), nos métodos de seleção citados acima, para a geração de uma nova população a anterior é destruída (“morre”). O autor também descreve que o *Steady state* tenta reproduzir a realidade biológica onde os indivíduos nascem aos poucos, os mais velhos morrem de modo lento e ambos interagem entre si. O *Steady state* mantêm o tamanho da população constante durante todo o processo de evolução. Os indivíduos criados são inseridos na população antes da exclusão para garantir que os mais fracos da totalidade sejam excluídos (Lucas, 2002). Logo, o *Steady state* gera *M* indivíduos ($M < N$), que substituem os piores indivíduos da população corrente, ao invés de criar toda a população de uma só vez. O *Steady state* sem duplicados, é similar ao *Steady state* com duplicados. O que os diferenciam

é o fato deste último permitir a presença de indivíduos duplicados, enquanto que o *Steady state* sem duplicados não permite.

A taxa de substituição é definida por uma constante que representa o percentual a ser substituído a cada geração.

3.2.2.4 Operadores genéticos

O princípio básico dos operadores genéticos consiste em transformar a população através de sucessivas gerações, estendendo a busca até alcançar um resultado satisfatório. A criação de novas soluções se dá através da combinação e refinamento das informações das soluções (indivíduos). Os operadores genéticos são necessários para a diversificação da população mantendo as características de adaptação adquiridas com as gerações anteriores.

As duas operações genéticas são: recombinação e mutação. Elas produzem novas soluções que formam uma nova população, denominada geração.

3.2.2.4.1 Operador de recombinação

O operador de recombinação é responsável por recombinar as características dos genitores durante a reprodução permitindo que os seus descendentes herdem essas características. Ele é considerado o operador genético predominante, o que justifica que sua taxa probabilística seja superior à taxa de mutação.

A utilização dos operadores genéticos normalmente varia em função do problema tratado, porém os três tipos de operadores de recombinação mais comuns, são:

Operador de recombinação de um ponto: recombina duas soluções a partir de um único ponto de corte aleatório, como mostrado na Figura 5 abaixo:

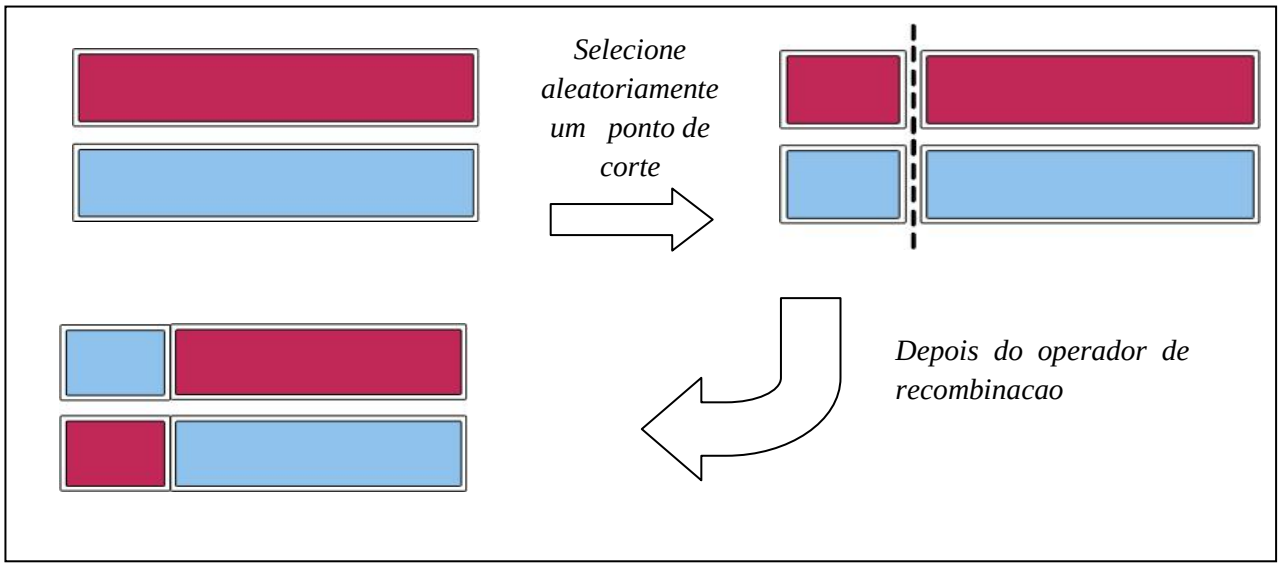


Figura 5 - Operador de recombinação de um ponto

Operador de recombinação de dois pontos: são escolhidos como referência dois pontos aleatórios para a troca de atributos. Veja o exemplo na Figura 6.

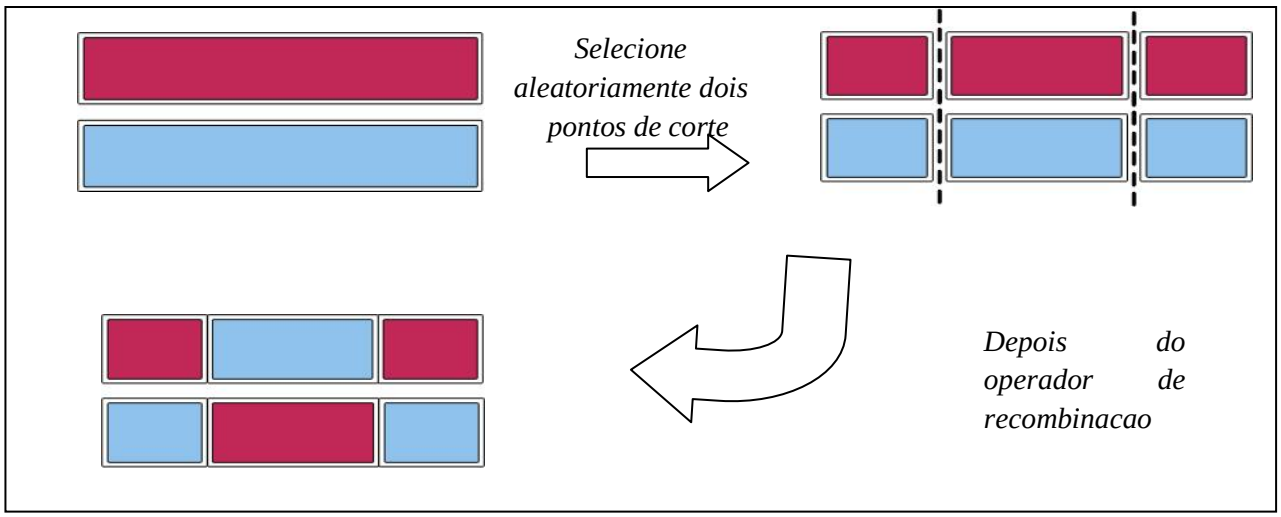


Figura 6 - Operador de recombinação de dois pontos

O primeiro filho é formado através da escolha do material genético do primeiro pai que está fora dos pontos de corte mais o material genético do segundo pai entre os pontos de corte. O segundo filho é formado com as outras partes restantes.

Operador de recombinação uniforme: é capaz de recombinar quaisquer posições entre dois genitores. Para tanto, utiliza uma palavra binária (0's e 1's) escolhida

aleatoriamente para designar os bits selecionados em cada genitor na criação dos descendentes. Este procedimento pode ser visualizado na Figura 7.

Para cada gene a ser preenchido nos cromossomos dos descendentes, o operador de recombinação uniforme sorteia de qual dos genitores este deve ser gerado.

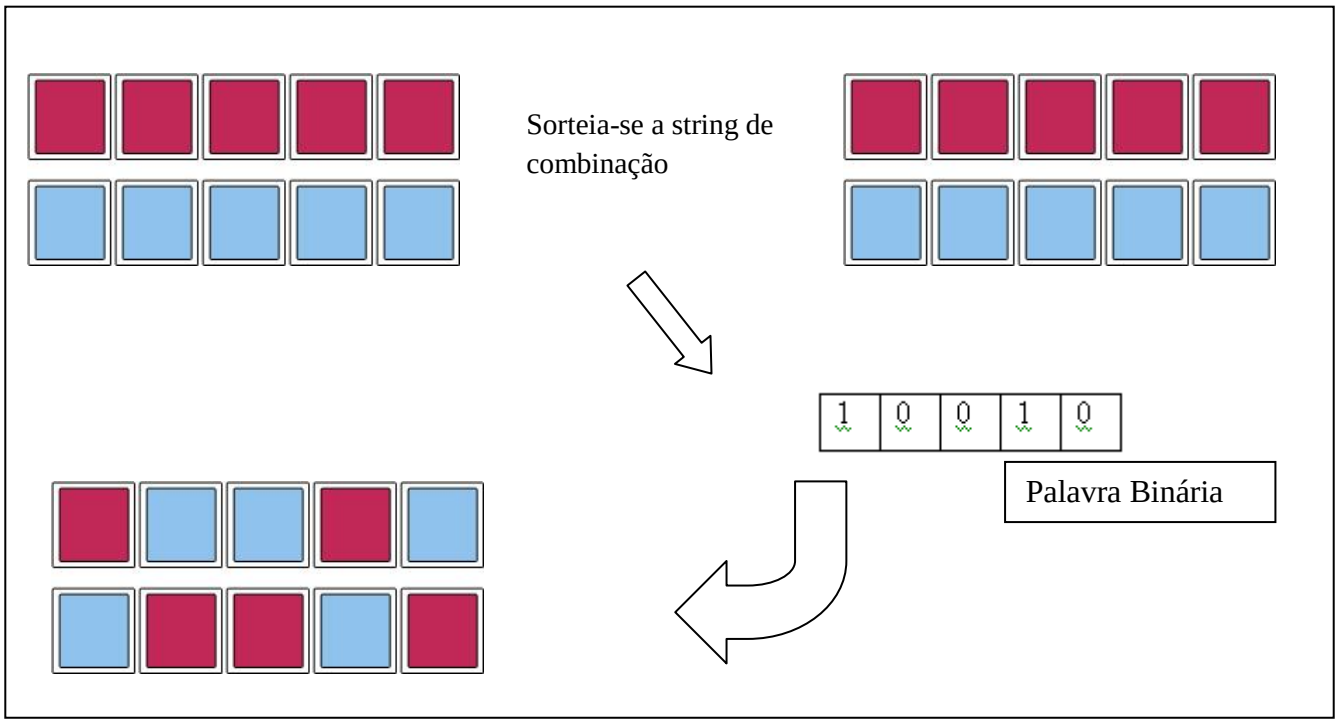


Figura 7 - Operador de recombinação uniforme

3.2.2.4.2 Operador de mutação

Segundo Lucas (2002), a mutação opera sobre os indivíduos resultantes do processo de recombinação e com uma probabilidade pré-determinada efetua algum tipo de alteração em sua estrutura.

A operação da mutação é fundamental para garantir a biodiversidade proporcionando uma maior varredura do espaço de busca, e para evitar que o Algoritmo Genético tenha uma convergência prematura.

A probabilidade de ocorrência de mutação em um gene é denominada taxa de mutação. Embora a taxa de mutação seja pequena, alguns autores defendem o operador de

mutação como o operador mais importante para o processo evolutivo, o qual em alguns casos extremos, são utilizados como o único operador de transformação no curso de evolução de um Algoritmo Genético (Lucas, 2002). A idéia da utilização deste operador é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido com a busca.

A mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo altera-se levemente a direção de busca.

3.2.2.5 Atualização

Os indivíduos resultantes da ação dos operadores de recombinação e mutação são inseridos na população. Esta inserção pode ocorrer de várias maneiras. Todos os indivíduos criados substituem os antecessores (Algoritmo Genético simples) mantendo fixo o tamanho da população, ou os indivíduos gerados podem ser menor que sua população e os descendentes serão inseridos se melhores que os genitores, ou ainda, um conjunto de *n* melhores indivíduos pode ser sempre mantido (Lucas, 2002).

3.2.2.6 Finalização

Na etapa de finalização não existe operadores genéticos. Nesta fase realiza-se um teste para verificar se o processo de evolução será continuado ou finalizado por ter encontrado algum critério de parada pré-estabelecido.

Cada componente, de modo particular, é de fundamental importância para a otimização de uma função. Segundo Tanomaru (1995), se não houvesse o processo de seleção, os Algoritmos Genéticos além de perderem grande parte do processo evolutivo, seriam processos ineficientes parecidos com as buscas aleatórias. E sem recombinação, a busca seria aleatória partindo dos melhores indivíduos. A ausência da mutação realizaria a busca utilizando somente informações contidas na população e não teriam, de acordo com Tanomaru (1995), como repor material genético perdido durante o processo de seleção. Sem a atualização não seria possível renovar a população de indivíduos e por fim, a não definição

de critérios de parada, faria com que a execução do algoritmo genético nunca fosse finalizada.

3.3 Desempenho de algoritmos genéticos

Pacheco (1999) argumenta que o desempenho de um algoritmo genético é medido pelo grau de evolução obtido durante o processo evolucionário (experimento). Devido à natureza estocástica dos algoritmos genéticos faz-se necessário a avaliação do resultado médio de vários experimentos para se ter uma idéia de seu desempenho.

As principais medidas de desempenho segundo Pacheco (1999) são:

1 - Curva da média dos melhores indivíduos de cada geração em vários experimentos.

2 - Curva *on-line* da média da avaliação de todos os indivíduos até um determinado instante *t* em um experimento.

3 - Curva *off-line* da média da avaliação dos melhores indivíduos até o momento.

Goldberg (1989) afirma que a curva média dos melhores indivíduos em diversos experimentos representa o desempenho médio de um algoritmo genético e serve para o ajuste de parâmetros. E a curva *on-line* mede a velocidade com que os algoritmos genéticos conseguem produzir boas soluções para o consumo *on-line* das soluções, ou seja, ela tenta definir como está o comportamento médio dos indivíduos da população em relação às médias anteriores (Brun, 2009). A curva *off-line* mede o grau de convergência dos algoritmos genéticos na criação de soluções com indivíduos mais aptos gerados *off-line* em relação ao problema (Goldberg,1989) pois, muitas funções de desempenho podem ser simuladas e as melhores alternativas são utilizadas posteriormente para a determinação de algum critério de parada (Brun, 2009).

3.4 Parâmetros genéticos

A utilização dos parâmetros genéticos influencia o comportamento dos Algoritmos Genéticos e, portanto, faz-se necessário estudar quais os impactos causados pelos mesmos.

3.4.1 Tamanho da população

O tamanho da população afeta sensivelmente o desempenho do algoritmo genético, precisando ser um parâmetro bem estudado. Populações pequenas limitam o espaço de busca, não proporcionando uma variedade genética suficiente, levando o algoritmo a soluções não desejáveis. E, caso contrário, se a população for muito grande, corre-se o risco do método se aproximar e equiparar-se a uma busca exaustiva (Linden, 2006).

3.4.2 Taxa de recombinação

Esta taxa implica na velocidade com que novas estruturas são introduzidas na população. Se muito alta, estruturas com boas aptidões poderão ser rapidamente substituídas podendo perder estruturas de alta aptidão. Porém, se muito baixa, o algoritmo poderá ficar muito lento (Carvalho, 2008).

3.4.3 Taxa de Mutação

Segundo Carvalho (2008), a taxa de mutação baixa previne que uma dada posição fique estagnada em um ótimo local, possibilitando que se chegue a qualquer ponto do espaço de busca. Adotando uma taxa muito alta, a busca se torna essencialmente aleatória, aumentando a chance de uma boa solução ser destruída.

Segundo Brun (2009), a melhor taxa de mutação varia de acordo com o problema tratado, no entanto, acredita-se que o valor ideal está entre 0,001 e 0,1.

3.4.4 Intervalo de geração

Realiza o controle do percentual da população que será substituída durante a geração seguinte. Um valor muito alto provocará a substituição da maior parte da população, com isso, poderá ocorrer perda de estruturas de alta aptidão. Valores muito baixos poderão provocar a lentidão do algoritmo (Brun, 2009).

4 GALib: BIBLIOTECA DOS COMPONENTES DOS ALGORITMOS GENÉTICOS EM C++

4.1 Introdução

Neste capítulo será apresentado o GALib (Wall, 1996), biblioteca dos componentes dos algoritmos genéticos e um exemplo de um dos algoritmos pré-definidos do pacote com a inclusão de algumas funções clássicas da literatura como função objetivo a serem minimizadas: função quadrática, função de Rosenbrock e função de Goldstein-Price. Uma vez que estas funções possuem seus mínimos globais conhecidos, objetiva-se com os testes verificar se o algoritmo genético extraído do GALib, e denominado ex1, é capaz de encontrar o mínimo global de uma função e analisar sua eficácia e eficiência diante dos resultados encontrados. Para os mesmos, serão verificados os impactos das alterações no tamanho da população, taxa de substituição, e o número de genomas.

4.2 GALib

O GALib é a biblioteca dos componentes de algoritmos genéticos em C++ desenvolvida por Wall (1996), no Massachusetts Institute of Technology. A biblioteca inclui ferramentas para o uso de algoritmos genéticos realizando a otimização de qualquer programa em C++, e utilizando qualquer representação e operadores genéticos. É composta por várias funções pré-definidas de algoritmo genético que possibilita em alguns casos mais simples, apenas a inclusão da função objetivo e definição dos valores e tipos de operadores. Esta biblioteca proporciona também, a criação de um genoma próprio e definição de como os operadores e a função de inicialização devem se comportar.

A documentação do GALib disponível em seu site oficial (http://lancet.mit.edu/ga/) possui uma ampla visão de como implementar um algoritmo genético, a interface de programação de suas classes e exemplos personalizados que ilustram o funcionamento de diversos problemas-modelos já existentes na biblioteca (códigos-fontes implementados em C++), bem como, criar seu próprio genoma e demais particularidades.

O código fonte do GALib não é de domínio público mas está disponível sem custos para utilização sem fins lucrativos. Para fins lucrativos é necessário possuir uma licença entrando em contato com MIT Technology Licensing Office.

A biblioteca tem sido utilizada em várias plataformas como o DOS / Windows, o MacOS, e UNIX.

Quando se utiliza esta biblioteca trabalha-se principalmente com duas classes: um genoma e um algoritmo genético. O genoma representa a solução, e o algoritmo genético define o processo evolutivo. Ele usa a função de aptidão definida por cada um para determinar como realizar a sobrevivência dos indivíduos e usa os operadores incorporados no genoma para, através de estratégias de seleção, gerar novos indivíduos.

Nas seções seguintes serão apresentadas as funções-testes que utilizaram um algoritmo genético do GALib para a minimização de suas funções.

4.3 Função Quadrática

A função quadrática é uma função relativamente simples e que pode ser resolvida facilmente por métodos determinísticos. No entanto, como o objetivo aqui é aferir a qualidade do algoritmo genético, disponível no pacote do GALib, serão utilizadas funções simples e complexas não só para verificar se o método consegue encontrar o mínimo global, como também comprovar o porquê da literatura justificar que técnicas heurísticas são recomendadas para problemas que não conseguem obter suas soluções em tempo polinomial.

Os coeficientes da função quadrática utilizada têm as seguintes características: $a > 0$, logo a parábola possui concavidade para cima e b e $c = 0$. Isso significa dizer que a função é uma função incompleta do segundo grau possuindo duas raízes reais e iguais. O minimizador global desta função é zero.

A seguir a descrição da referida função, (4.1) e sua representação gráfica (Figura 8).

Função quadrática:

$$\boldsymbol{f(x) = x^2} \tag{4.1}$$

O critério de parada do algoritmo, o qual será adotado também nas demais funções, é que o número máximo de gerações seja atendido ou a diferença entre o maior e menor indivíduo da população dividido pelo maior for menor que 1×10^{-5}.

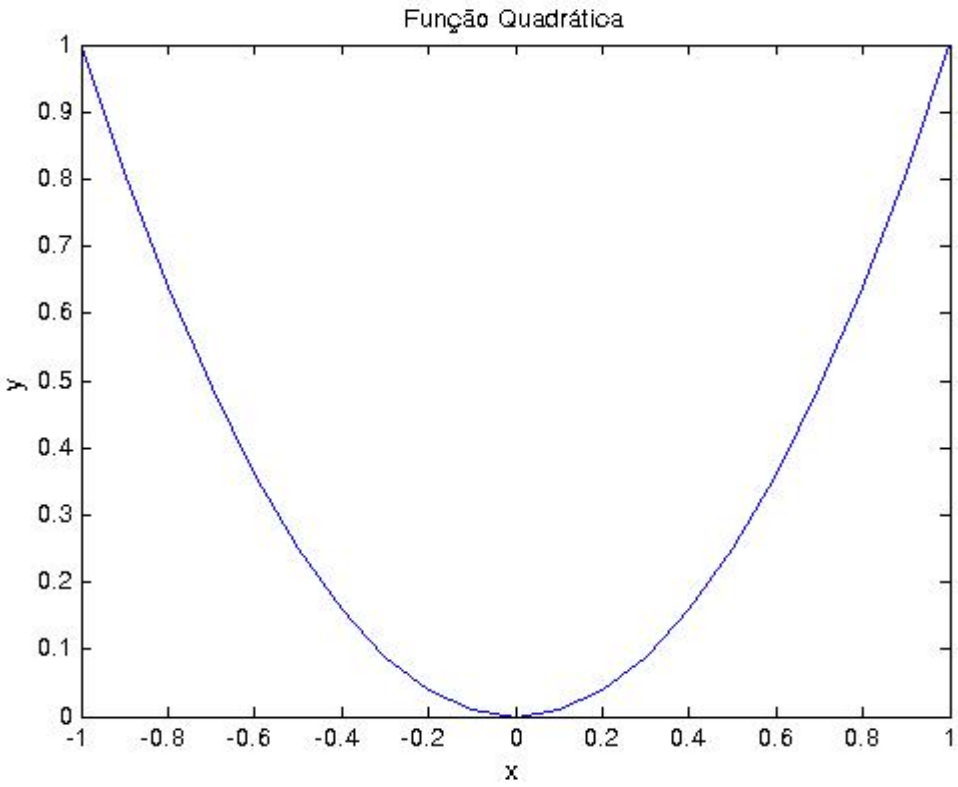


Figura 8: Representação gráfica da função $f(x) = x^2$ no intervalo [-1 1]

Para testar a qualidade do algoritmo genético testado, os parâmetros básicos utilizados no mesmo para a realização dos testes da função $\boldsymbol{f(x) = x^2}$ estão descritos na Tabela 4.1.

O número de genomas em todos os testes foi igual a 1.

Tabela 4.1 - Parâmetros básicos utilizados no Algoritmo Genético para a função $f(x) = x^2$

Teste	**Num_pop**	**Num_geracoes**	**Tx_subst**	**Tx_recomb**	**Tx_mutacao**	**Intervalo_busca**
1	500	1000	0,3	0,9	0,01	**[0 2]**
2	500	1000	0,3	0,9	0,01	**[-1 1]**
3	50	100	0,3	0,9	0,01	**[-1 1]**
4	50	100	0,3	0,9	0,01	**[0 2]**

Onde:

Num_pop: tamanho da população

Num_geracoes: número de gerações

Tx_subst: taxa de substituição

Tx_recomb: taxa de recombinação

Tx_mutação: taxa de mutação

Intervalo_busca: corresponde a região onde o método buscará a solução

Como mostra a Tabela 4.1, o que difere os parâmetros dos testes 1-2 é o intervalo de busca. Os testes 3-4 diferem dos outros também em número de população e gerações e entre eles pelo intervalo.

Para cada teste com os parâmetros apresentados acima, executou-se 10 vezes o método e em 100% dos casos o algoritmo genético conseguiu, considerando a margem de erro para cada indivíduo, encontrar a solução, ou seja, o mínimo global.

Vale ressaltar, no entanto, que para encontrar a solução de $\boldsymbol{f(x) = x^2}$, que é uma função extremamente simples, o método se mostrou demasiado lento. Isso pode ser comprovado na Tabela 4.2 que apresenta a quantidade de iterações necessárias para encontrar o mínimo da função a cada execução do método para todos os testes.

Tabela 4.2 - Análise da quantidade de iterações dos testes realizados

Execução	**Iteração_Teste1**	**Iteração_Teste2**	**Iteração_Teste3**	**Iteração_Teste4**
1	104	1000	100	**55**
2	108	1000	100	**48**
3	105	1000	100	**37**
4	1000	1000	100	**64**
5	145	1000	100	**47**
6	1000	1000	100	**43**
7	1000	1000	100	**51**
8	1000	1000	100	**54**
9	1000	1000	100	**68**
10	98	1000	100	**47**

Nota-se também na referida tabela que para o intervalo [-1 1] o método apresentou seus piores desempenhos. O melhor desempenho do método ocorreu com os parâmetros adotados com o Teste 4.

Estes resultados justificam a literatura que recomenda a utilização de métodos heurísticos apenas quando não houver um método determinístico capaz de resolver o problema (DENNIS e SCHNABEL, 1983).

4.4 Função de Rosenbrock

A função de Rosenbrock é uma função contínua, unimodal, com o ponto ótimo localizado em um vale parabólico íngreme com um fundo aproximadamente plano. Além disso, ela é uma função não-separável, isto é, suas variáveis interagem de forma não-linear (HERRERA;LOZANO, 2001).

A busca do ótimo da função de Rosenbrock é um problema difícil, pois o fundo do vale é aproximadamente plano, o que causa um lento progresso em vários algoritmos de busca. Essa característica faz com que esses algoritmos modifiquem, continuamente, a direção da busca para alcançar o ótimo e muitas vezes parem de modo prematuro, retornando a um ponto que não é um ótimo global. Além disso, essa função possui interações não-lineares (ou seja, é uma função não separável). Devido a essa característica, alguns autores a consideram um desafio para qualquer algoritmo de otimização contínua e por isso, foi selecionada para aferir a qualidade do algoritmo genético em questão.

O ponto mínimo da função de Rosenbrock é (1,1,1). A seguir a descrição da referida função, (4.2), e sua representação gráfica, Figura 9, são apresentadas.

Função Rosenbrock:

$$f(x) = \sum_{j=1}^{n} \alpha \ (x_j^2 - x_{j+1})^2 + (x_j - 1)^2 \quad \text{, onde } \alpha = 10$$

$$\forall\, x \in R^n \ \ com\ j = 1, \dots, n \qquad (4.2)$$

onde *n* = número de genomas (variáveis). Aqui adotou-se *n* = 3.

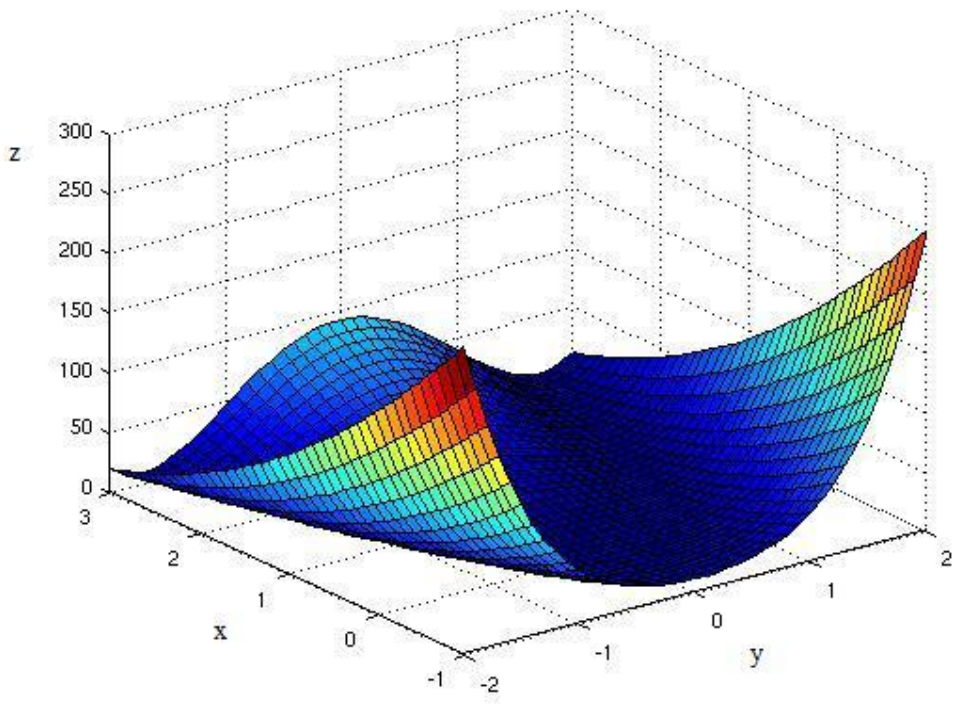


Figura 9 - Representação gráfica da função de Rosenbrock em [-2,2]x[-1,3]x[0,300]

A escolha do $\alpha = 10$ foi feita após testes no algoritmo genético em análise e pesquisas na literatura que garantiram à esta opção um melhor desempenho da função, quando comparados com $\alpha = 1$ ou $\alpha = 100$(DENNIS e SCHNABEL, 1983).

Portanto, utilizando a função de Rosenbrock com $\alpha = 10$ para aferir a qualidade do algoritmo genético testado, os parâmetros básicos utilizados no mesmo para a realização dos testes estão descritos na Tabela 4.3.

Tabela 4.3 - Parâmetros básicos utilizados no Algoritmo Genético para a função Rosenbrock

Teste	**Num_populacao**	**Num_geracoes**	**Tx_subst**	**Tx_recomb**	**Tx_mutacao**	**Num_genoma**
1	500	1000	0,3	0,9	0,01	3
2	500	1000	0,3	0,9	0,01	4
3	500	1000	0,3	0,9	0,01	10
4	500	1000	0,1	0,9	0,01	3
5	500	1000	0,5	0,9	0,01	3
6	200	500	0,3	0,9	0,01	3
7	500	500	0,3	0,9	0,01	3

Onde: Num_genoma: número de genomas

Como mostra a Tabela 4.3, o que difere os parâmetros dos testes 1-3 é o número de genomas. Similarmente, a variação ocorrida nos testes 4 e 5 consiste na mudança da taxa de substituição. Finalmente, os testes 6 e 7 diferem entre sim no quesito tamanho da população.

Para cada teste com os parâmetros apresentados acima, executou-se 10 vezes o método e em 100% dos casos o algoritmo genético conseguiu, considerando a margem de erro para cada indivíduo, encontrar a solução, ou seja, os mínimos globais.

Os valores dos parâmetros utilizados foram escolhidos empiricamente. Embora não se possa garantir que sejam os ideais, eles são parâmetros bons e que conseguiram em todos os

testes gerar uma diversidade das populações sem convergências prematuras para mínimos locais, garantindo os bons resultados encontrados.

Como já citado no capítulo anterior, não existe fórmula que determine estes parâmetros, os mesmos podem diferir de um problema para outro. É necessário, portanto, a realização de diversos testes para a escolha dos mesmos. A literatura apenas recomenda alguns caminhos e conceitos que ajudam a direcionar esta escolha, veja no capítulo 3.

As tabelas 4.4, 4.5 e 4.6 representam os resultados obtidos nas 10 execuções do método referente aos testes 1, 6 e 7, respectivamente.

Tabela 4.4 - Resultados obtidos utilizando os parâmetros adotados pelo Teste 1

Execução	x_1	x_2	x_3
1	0,99927	0,998568	0,997092
2	1,00084	1,0017	1,00334
3	1,0002	1,00039	1,00075
4	0,999869	0,999722	0,999439
5	1,0007	1,00139	1,0028
6	0,995982	0,991827	0,983593
7	0,99917	0,998373	0,996714
8	0,998764	0,997455	0,994884
9	1,00016	1,00031	1,00063
10	0,992762	0,985462	0,970894

Tabela 4.5 - Resultados obtidos utilizando os parâmetros adotados pelo Teste 6

Execução	x_1	x_2	x_3
1	1,00488	1,00981	1,01986
2	1,00568	1,01142	1,02304
3	0,990558	0,981269	0,96265
4	0,99431	0,989081	0,977598
5	0,988234	0,976302	0,951721
6	0,999543	0,99908	0,998137
7	0,987019	0,974041	0,947593
8	0,998689	0,997312	0,994572
9	0,999611	0,999224	0,998427
10	1,00488	1,00981	1,01986

Tabela 4.6 - Resultados obtidos utilizando os parâmetros adotados pelo Teste 7

Execução	x_1	x_2	x_3
1	0,998976	0,997914	0,995808
2	0,999842	0,999681	0,999394
3	0,992623	0,985656	0,970655
4	1,00082	1,00161	1,00325
5	1,00001	1,00001	1,00004
6	0,998787	0,997631	0,995195
7	0,963145	0,927467	0,859947
8	0,99413	0,988184	0,976317
9	0,989039	0,978136	0,956263
10	0,998976	0,997914	0,995808

O número de iterações foi igual ao número de gerações de cada teste. Na média, aproximadamente, 858 gerações (levando em consideração os testes realizados) e em todas as execuções de todos os testes, os mínimos globais, considerando a margem de erros, foram encontrados.

4.5 Função de Goldstein-Price

$$f(x_1, x_2) = [[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]] \quad (4.3)$$

Domínio de busca: $-2 \leq x_j \leq 2 \qquad j = 1, 2;$

A função possui quatro minimizadores locais; com minimizador global $x^* = (0, -1)$

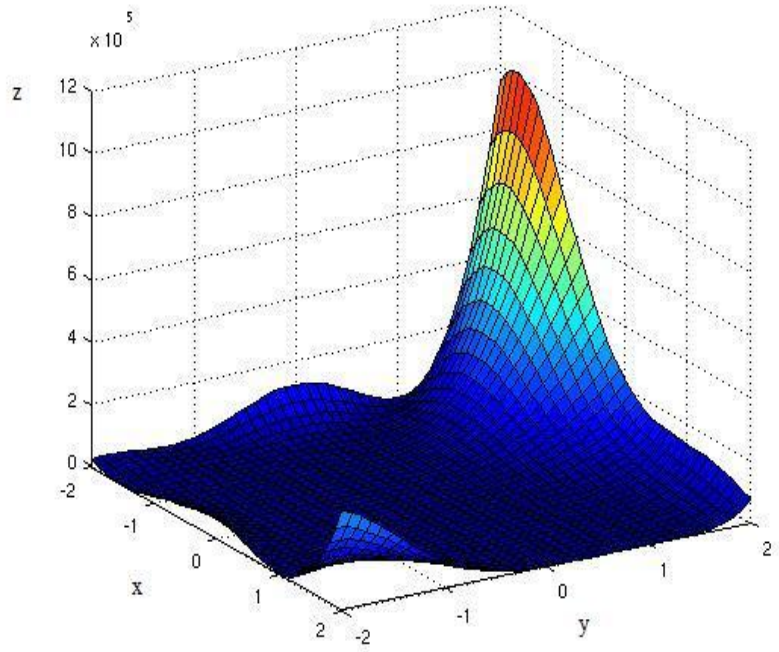


Figura 10: Representação gráfica da função de Goldstein-Price em [-2,2]x[-2,2]x[0,12]

Os parâmetros básicos utilizados pelo método para encontrar o mínimo da função Goldstein-Price estão descritos na Tabela 4.7.

O número de genomas em todos os testes foi igual a 2, e o intervalo adotado igual a [-2, 2], escolha esta baseada na literatura.

Tabela 4.7 - Parâmetros Básicos da função Goldstein-Price

Teste	**Num_populacao**	**Num_geracoes**	**Tx_subst**	**Tx_recomb**	**Tx_mutacao**
1	500	1000	0,3	0,9	0,01
2	500	1000	0,1	0,9	0,01
3	500	1000	0,5	0,9	0,01
4	200	500	0,3	0,9	0,01
5	500	500	0,3	0,9	0,01

As tabelas 4.8 - 4.12 mostram os resultados obtidos utilizando os parâmetros dos testes 1-5 (Tabela 4.7), para os testes de um a cinco, respectivamente.

Tabela 4.8 - Resultados do Teste 1

Execução	x_1	x_2	Num_Iteracoes
1	-5,48201e-006	-0,999987	175
2	1,5977e-005	-0,999988	202
3	-1,30239e-005	-0,99999	184
4	-6,18663e-006	-0,999995	180
5	4,38789e-006	-0,999991	191
6	-1,97564e-006	-1,00001	183
7	3,53787e-006	-0,999989	178
8	-8,77272e-006	-0,999998	179
9	9,61947e-006	-1,00001	186
10	-5,48201e-006	-0,999987	175

Tabela 4.9 - Resultados do Teste 2

Execução	x_1	x_2	Num_Iteracoes
1	-1,22291e-005	-1,00002	98
2	-2,15489e-005	-1	80
3	3,93816e-007	-1,00001	88
4	1,54738e-005	-0,99999	90
5	1,47257e-005	-1	87
6	-1,26136e-005	-0,999993	96
7	-1,00955e-005	-0,99999	85
8	1,81459e-005	-0,999999	108
9	-1,79934e-005	-1,00001	116
10	3,53617e-006	-0,999992	95

Tabela 4.10 - Resultados do Teste 3

Execução	x_1	x_2	Num_Iteracoes
1	-1,24099e-005	-0,999998	134
2	-1,24264e-006	-1,00001	143
3	-5,70108e-006	-1,00001	144
4	-6,72066e-006	-1,00001	79
5	1,75958e-005	-1	157
6	-9,7437e-006	-1	144
7	-1,07252e-005	-1	159
8	-1,87305e-005	-1,00001	184
9	-7,02433e-006	-1,00001	144
10	-1,13488e-005	-0,999994	114

Tabela 4.11 - Resultados do Teste 4

Execução	x_1	x_2	Num_Iteracoes
1	-4,78773e-006	-0,99999	85
2	2,17063e-005	-0,99999	75
3	1,32022e-005	-0,99998	73
4	1,22109e-005	-0,999984	60
5	2,40516e-005	-1,00002	62
6	-6,43783e-006	-1	62
7	-1,28145e-005	-0,999994	71
8	-7,3687e-006	-0,999989	63
9	-1,79702e-005	-1	70
10	-4,78773e-006	-0,99999	85

Tabela 4.12 - Resultados do Teste 5

Execução	x_1	x_2	Num_Iteracoes
1	-2,57333e-006	-0,999985	93
2	-6,94079e-006	-1,00001	80
3	-1,34051e-005	-0,999995	93
4	-4,67949e-006	-1,00001	76
5	-2,11092e-005	-1,00001	102
6	5,84765e-006	-0,999993	100
7	-1,10915e-006	-0,999996	83
8	-7,82841e-006	-1	74
9	1,17927e-005	-0,999992	71
10	2,63853e-006	-0,999999	75

Assim como nas funções testadas anteriormente, para cada teste com os parâmetros apresentados acima, executou-se 10 vezes o método e em 100% dos casos o algoritmo genético conseguiu encontrar a solução, ou seja, o mínimo global.

As mesmas conclusões sobre encontrar o mínimo global das funções anteriores, também são válidas para esta.

A Tabela 4.13 analisa o número de iterações (**Iter_Teste**) em cada execução do método para os parâmetros de testes apresentados na Tabela 4.7 e faz o cálculo da média. Nota-se que, os parâmetros utilizados pelo Teste 4, além de achar o mínimo da função como os demais, fez isso com o menor número de gerações (iterações).

Tabela 4.13 - Análise das médias das gerações

Execução	**Iter_Teste1**	**Iter_Teste2**	**Iter_Teste3**	**Iter_Teste4**	**Iter_Teste5**
1	98	175	134	85	93
2	80	202	143	75	80
3	88	184	144	73	93
4	90	180	79	60	76
5	87	191	157	62	102
6	96	183	144	62	100
7	85	178	159	71	83
8	108	179	184	63	74
9	116	186	144	70	71
10	95	175	114	85	75
Média	**94,3**	**183,3**	**140,2**	**70,6**	**84,7**

Este estudo preliminar da GALib foi importante para entender melhor a biblioteca e as ferramentas oferecidas para uma melhor utilização futura no problema proposto.

Todas as funções analisadas comprovam que o método utilizado é consistente e encontra o mínimo de uma função. É importante notar, que os parâmetros que possam apresentar um melhor desempenho para uma função, não necessariamente será o mais adequado para a outra sendo necessário, para cada problema, a realização de diversos testes para a escolha dos parâmetros visando uma solução ótima, ou com resultados de boa qualidade.

Pode-se concluir com este capítulo que os Algoritmos Genéticos são métodos eficazes para encontrar o mínimo e/ou máximo global de funções dos mais diversos comportamentos e serão utilizados na tentativa de minimizar um problema de corte unidimensional, bem como encontrar boas soluções.

5 FORMULAÇÃO MATEMÁTICA E METAHEURÍSTICA APLICADA AO PROBLEMA DE CORTE UNIDIMENSIONAL PROPOSTO

5.1 Introdução

A modelagem do problema de corte é bastante simples, no entanto, a resolução da mesma é complexa e não se tem conhecimento até o momento de nenhum método que a resolva em tempo polinomial.

Para avaliar e solucionar um problema de corte faz-se necessário gerar os padrões de corte e métodos para manipulá-los de tal maneira que a disposição final dos itens apresente uma boa solução, ou seja, com menor desperdício e redução de *setup*.

Neste capítulo, além da exposição da formulação matemática do problema, será apresentado em detalhes, como os Algoritmos Genéticos apresentados no capítulo 3 foram aplicados para a resolução do problema proposto, bem como as técnicas utilizadas para a sua codificação.

A seção 5.2 apresenta a formulação matemática do método proposto. As demais seções detalham os procedimentos heurísticos utilizados para resolver o problema.

5.2 Formulação do problema

Apresenta-se, a seguir, a modelagem para o Problema de Corte para Minimizar o Número de Objetos Processados e o *Setup* (PCOPS), baseada na formulação (2.3) com os custos, c_1 e α inclusos na função objetivo a ser minimizada (5.1). Na literatura apenas o ANLCP300 (Salles Neto e Moretti, 2005) e o Symbio (Golfeto et al., 2007a) utilizam estes custos diretamente na função objetivo.

$$\text{Minimizar } c_1 \sum_{j=1}^{n} x_j + \alpha c_1 \sum_{j=1}^{n} \delta(x_j)$$

$$\text{Sujeito a: } \sum_{j=1}^{n} a_{ij} x_j \geq d_i \quad i = 1,...,m \qquad (5.1)$$

$$x_j \in N \quad j = 1,...,n$$

onde:

$$\delta(x_j) = \begin{cases} 1, \text{ se } x_j > 0 \\ 0, \text{ se } x_j = 0 \end{cases}$$

n = número de possíveis padrões de corte

m = número de itens diferentes

c_1 = custo por metro desperdiçado

α = é o custo de troca do padrão de corte

x_j = número de bobinas processadas com o padrão de corte *j*

a_{ij} = número de itens do tipo *i* no padrão *j*

di = número de itens *i* demandados

O primeiro somatório da modelagem do problema, indica o número de objetos processados e o segundo, o número de *setup*, os quais devem ser minimizados simultaneamente.

A restrição adotada indica que a quantidade de itens produzidos (demanda atendida) devem ser igual ou superior a demanda pedida pelo cliente.

A restrição de desigualdade existente na demanda em $\sum_{j=1}^{n} a_{ij} x_j \geq d_i$ foi adotada pelos mesmos motivos dos pioneiros Salles Neto e Moretti (2005) que é a ampliação da possibilidade em se obter soluções com baixo *setup* sem aumentar muito o número de objetos processados.

Nas seções seguintes serão detalhados os procedimentos utilizados pelo método proposto para resolver o problema de minimização apresentado em (5.1).

5.3 Algoritmo genético SingleGA

O novo algoritmo genético tratado neste trabalho, denominado SingleGA, foi baseado no Symbio de Golfeto et al. (2007a). No entanto, o Symbio utiliza-se de conceitos de simbiose (Allaby, 1998), mais especificamente numa relação mutualística entre dois algoritmos genéticos (Pianka, 1994) que se evoluem de maneira benéfica, enquanto o método do presente estudo utiliza-se de apenas um algoritmo genético para resolver o mesmo objetivo.

Os resultados do trabalho de Golfelto et al. (2007.a e 2007.b) mostra um bom desempenho da metaheurística algoritmo genético para minimizar o número de objetos processados e o *setup*. E baseado nestes resultados, a proposta de desenvolver um algoritmo genético trabalhando individualmente para atingir os mesmos objetivos do Symbio tornou-se bastante atrativa.

O SingleGA utiliza-se de conceitos expostos no capítulo 3 como genoma, população, tipos de seleção, cruzamento e mutação para definir a estrutura, comportamento e particularidades do método. Para a implementação computacional, que será abordada no capítulo 6, contou-se os conceitos de orientação a objetos como classes, heranças, sobrecarga e polimorfismo (Deitel & Deitel, 2001).

No SingleGA existem duas espécies de população: a de soluções e a de padrões, detalhadas nas próximas seções. A população de padrões, gerada aleatoriamente, é estática, ou seja, não sofre qualquer espécie de evolução.

5.4 Estrutura dos indivíduos das populações

Neste trabalho, o gene da população de solução é representado por dois elementos, onde o primeiro elemento refere-se à quantidade de vezes (frequência) com que o padrão

indicado pelo segundo elemento será processado. O segundo elemento da população de solução serve apenas como um ponteiro para a população de padrões.

O gene da população de padrões é representado por um número real que indica o comprimento do item da lista de pedidos realizado pelo cliente.

Para modelar e implementar computacionalmente o gene solução criou-se, baseado em conceitos da orientação a objetos, um objeto gene que é formado por dois tipos inteiros representado pela freqüência e o índice do padrão, similarmente ao apresentado abaixo. O gene da população de padrões é representado por um número real (*float*), tipo primitivo de dados (Deitel & Deitel, 2001).

```
Classe Gene
        (...)
        Inteiro frequencia
        Inteiroiíndice_padrao
        (...)
Fim_Classe
```

Figura 11 – Construção computacional do gene solução

A seguir, a Figura 12 mostra os genes das populações: de padrões e soluções.

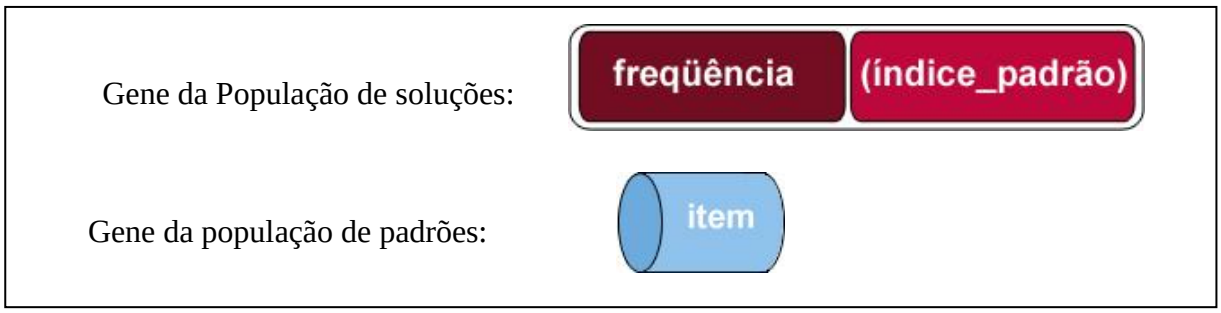


Figura 12 - Representação dos genes

A Figura 13 ilustra um exemplo da interpretação de um genoma do indivíduo da população de soluções.

Figura 13: Genoma da População de Soluções

Este exemplo (Figura 13) indica a existência de três padrões distintos sendo executados duas, quatro e cinco vezes, respectivamente. Ou seja, o padrão de índice 30, da população de padrões será executado duas vezes, o padrão de índice 20, quatro vezes e o padrão de índice 1, cinco vezes.

A soma das freqüências equivale ao número de objetos processados e a quantidade de padrões distintos, o número de *setup*. Neste exemplo (Figura 13), o número de objetos processados é igual a 11 e o número de *setups* igual a 3.

O número máximo de genes do indivíduo solução, ou seja, o tamanho máximo do genoma, será igual ao número de *setups* que é a quantidade de itens diferentes. No entanto, se o maior item a ser cortado apresentar um comprimento inferior ou igual a 50% do comprimento da bobina adota-se de modo experimental, o seguinte procedimento apresentado na Figura 14, que foi adaptado do utilizado por Golfeto et al. (2007a):

```
Se (Quantidade de itens > 30) Então
        GenesSolucoes = 16
Senao
        Se (Quantidade de itens > 15) Então
            GenesSolucoes = 12
        Senao
            GenesSolucoes = 8
        Fim-Se
Fim-Se
```

Figura 14 - Número de Genes do Indivíduo Solução

O tamanho do genoma do padrão é igual ao maior inteiro menor ou igual ao resultado da divisão entre o tamanho da bobina padrão pelo item de menor tamanho do padrão. Com isso, segundo Golfeto et al. (2007a), garante-se que é possível a formação de um padrão usando apenas o menor item, fazendo com que o algoritmo cubra todas as maneiras de resolver o problema. No entanto, isto traz implicações para o maior padrão, que não poderá utilizar todos os genes disponíveis, pois, o total de suas peças contidas no padrão ultrapassará o comprimento da bobina. Na tentativa de solucionar este problema, os itens são adicionados ao padrão da esquerda para a direita, se somente se, o padrão possuir espaço livre o suficiente para acomodar o item. Esta abordagem foi inspirada nos trabalhos de Golfeto et al. (2007a).

Os indivíduos formados por genes da população de soluções serão chamados de população de soluções e/ou indivíduos soluções, e os indivíduos formados por genes da população de padrões serão chamados de população de padrões e/ou indivíduos padrões.

A população de soluções foi formada por 600 indivíduos e a população de padrões por 400.

5.5 Indivíduos-soluções

Um indivíduo solução é formado por um, ou um conjunto de genes solução. O genoma solução, apresentado na Figura 13, é um exemplo de indivíduo solução. A população de soluções é formada por *n* indivíduos soluções.

As subseções seguintes apresentam as estruturas para gerar e formar novos indivíduos da população de soluções.

5.5.1 Geração da população inicial dos indivíduos soluções

Os indivíduos foram gerados aleatoriamente sem nenhum direcionamento, mas respeitando os limites dos elementos que compõem o gene solução. Ou seja, gera-se, aleatoriamente, um valor para a freqüência entre zero e a maior demanda. O índice padrão, possui os limites compreendidos entre zero e a quantidade de padrões – 1. E, assim, é gerado

o primeiro gene do genoma (freqüência,índice padrão). Este processo é repetido até que o tamanho do genoma seja atingido. Os limites são justificados na subseção 5.5.5.

O pseudocódigo para a geração da população inicial dos indivíduos soluções está representado pela Figura 15.

Procedimento Gera_Populacao_Solucao

1. **Para** i = 0 ate tamanho do genoma faca
2. Gere um numero inteiro entre 0 e o tamanho da demanda maxima
3. Atribua este numero para o elemento frequencia
4. Gere um numero inteiro entre 0 e quantidades de padroes-1
5. Atribua este numero ao indice padrao
6. Atribua estes numeros para formar o gene (frequencia, índice padrao)
7. **Fim_Para**

Fim Procedimento Gera_Populacao_Solucao

Figura 15 – Pseudocódigo para a geração da população inicial dos indivíduos soluções

5.5.2 Estrutura de seleção

A estrutura de seleção adotada foi o elitismo com *steady state*. A idéia básica do elitismo já foi introduzida no capítulo 3.

A escolha do elitismo se deve ao fato deste recurso aumentar o desempenho do algoritmo genético, uma vez que ele garante que a melhor solução encontrada até o momento será mantida nas próximas gerações.

O *steady state* utilizado foi o disponível no pacote do GALib (Wall, 1996) da classe de algoritmos genéticos GASteadyStateGA. Segundo o tutorial do GALib (Wall, 1996), este método gera, a cada geração, uma população temporária de indivíduos por clonagem e os inserem na população da geração atual, removendo os piores indivíduos. A taxa de substituição adotada foi de 25%, ou seja, os 75% melhores indivíduos da população de

soluções foram selecionados para permanecer na população. Este valor foi ajustado de modo experimental.

5.5.3 Função de avaliação (aptidão)

O cálculo da aptidão (F_a) do indivíduo solução para atender o objetivo de minimizar ao mesmo tempo, o número de objetos processados e o *setup* (5.1), baseado e descrito em Golfeto et al. (2007a) é realizado do seguinte modo:

$$F_a = c_1 \sum_{j=1}^{n} x_j + \alpha c_1 \sum_{j=1}^{n} \delta\left(x_j\right) + \sum_{j=1}^{n} \tau\left(x_j\right) + \rho \qquad (5.2)$$

onde:

τ: é a perda relativa e pode ser calculada por $\tau(x_j) = \frac{t_j}{W \sum_{j=1}^{n} x_j}$ sendo t_j o desperdício do padrão *j*.

ρ: são penalidades caso a solução seja inviável (não atenda a demanda pedida pelo cliente). É proporcional a soma das infactibilidades de todos os itens, ou seja, o valor que falta para atingir a demanda do item (s), multiplicado por 1000. Este valor 1000 foi escolhido de modo experimental.

Os valores c_1 e α são os custos de objetos processados e *setup*, respectivamente, tratados de maneira explícita na função objetivo, não necessitando quaisquer outras modificações para alcançar objetivos distintos.

Assim como em Golfeto et al. (2007a), no SingleGA o τ tem a função de verificar se o algoritmo encontrou um ótimo local, uma vez que uma pequena alteração na perda relativa provocará uma melhora da função objetivo, alongando sua execução.

Durante o cálculo da função objetivo faz-se necessário uma interpretação dos padrões, ou seja, verificar quais os genes ativos do padrão. Os genes ativos serão aqueles que cabem na bobina-mestre sem ultrapassar o seu tamanho.

O pseudocódigo da função de avaliação está representado pela Figura 16.

Procedimento Func_Aptidao

1. Some as frequencias do genoma //(objetos processados)
2. Some os índices distintos dos padrões //(*setup*)
3. Interprete os padrões
4. Calcule a perda relativa
5. **Se** a solução é inviável
6. Realize o cálculo da penalidade
7. **Senao**
8. Atribua zero a penalidade
9. Fim-Se
10. Calcule a função de aptidão

Fim Procedimento Func_Aptidao

Figura 16 – Pseudocódigo da função de avaliação (aptidão)

5.5.4 Operador de recombinação

Optou-se por utilizar neste trabalho o operador de recombinação uniforme, o qual consiste em sortear aleatoriamente um valor entre 0 e 1. No SingleGA, se o número sorteado for menor ou igual a 0,7 a recombinação é realizada com o indivíduo que possuir maior avaliação na função objetivo, caso contrário, realizará com o indivíduo de menor avaliação.

A taxa de recombinação adotada foi de 30%.

Apresenta-se na Figura 17, o pseudocódigo do operador de recombinação dos indivíduos soluções.

Procedimento RecombinacaoS (Genitor1, Genitor2, Descendente)

1 Calcule a funcao de aptidão do Genitor1
2 Calcule a funcao de aptidão do Genitor2
3 Atribua 70% de chances ao indivíduo com maior avaliacao na funcao de aptidao
4 **Se** numero_aleaorio (0,1) ≤ 0,7
5 Descendente = Herde caracteristicas do indivíduo com maior valor na funcao de aptidao
6 **Senao**
7 Descendente = Herde caracteristicas do individuo com menor valor na funcao de aptidão
8 **Fim-se**

Fim-Procedimento RecombinacaoS

Figura 17- Pseudocódigo do operador de recombinação dos indivíduos soluções.

5.5.5 Operador de mutação

O operador de mutação consiste em sortear aleatoriamente a posição no genoma do gene a ser mutado e em seguida determinar aleatoriamente qual dos seus elementos (freqüência ou indice_padrao) sofrerá mutação.

Caso o elemento escolhido do gene seja a freqüência, será sorteado um valor entre os limites mínimo e máximo. Caso contrário, será sorteado um indivíduo da população de padrão e um ponteiro será criado (Golfeto et al., 2007a).

Os limites máximo e mínimo para a frequência são necessários para restringir a região de soluções do problema. Ao limite inferior é atribuído o valor zero e para o limite máximo, o valor da maior demanda para um único item, ou seja, a maior demanda do pedido. Isso possibilita que um item esteja contido em apenas um padrão e tenha toda a sua demanda atendida.

Para o índice do padrão, elemento que compõe o gene da população de solução, os limites ficam entre zero e a quantidade de padrões -1. Segundo Golfeto et al. (2007a), este

valor permite a obtenção do valor ótimo de objetos processados no caso do problema de restrição de integralidade.

A taxa de mutação é o quociente entre 1 e número diferentes de itens (1/*m).*

A seguir, a Figura 18 apresenta o pseudocódigo do operador de mutação.

```
Procedimento MutacaoS ( Genoma)
    1  Gere aleatoriamente a posicao do Genoma a ser mutado
    2  Gere aleatoriamente um elemento do gene solução (frequencia ou indice)
    3  Se elemento frequencia
    4  Gere um número entre 0 e a maior demanda do pedido
    5   Atribua este valor gerado ao elemento freqüência do Genoma a ser mutado
    6  Senao
    7  Gere um numero entre 0 e a quantidade de padroes -1
    8   Atribua este valor gerado ao elemento índice_padrao do Genoma a ser mutado
    9   Fim-Se
Fim Procedimento MutacaoS
```

Figura 18 - Pseudocódigo do operador de mutação.do indivíduo solução

5.6 Critérios de parada do SingleGA

Os critérios de parada como já visto no capítulo 3, são condições que enquanto não forem verdadeiras o método continua a ser executado.

O SingleGA utilizou três critérios de parada:

- Número máximo de gerações: 1000
- Tempo máximo de execução: 500s
- Convergência: 500 gerações, ou seja, se o algoritmo não melhorar a solução por 500 gerações ele pára.

Estes critérios foram utilizados de modo experimental.

5.7 Pseudocódigo SingleGA

Nesta seção, após a apresentação dos pseudocódigos utilizados para a execução de cada etapa do algoritmo genético, será exposto, resumidamente, na Figura 19, o procedimento utilizado pelo algoritmo proposto, SingleGA, que, conforme a modelagem apresentada em (5.1) tem por objetivo minimizar o número de objetos processados e o *setup.*

Procedimento SingleGA

1 Gere os individuos da populacao de padroes aleatoriamente
2 Gere os individuos da populacao de solucoes aleatoriamente
3 Calcule a função de aptidao dos individuos solucoes
4 Selecione os individuos solucoes genitores
5 Utilize operadores de recombinacao e mutacao para gerar novos indivíduos soluções (descendentes)
6 Atualize a população
7 **Se** algum criterio de parada for satisfeito PARE a execucao do algoritmo
8 **Senao** retorne ao 3.

Fim-Procedimento SingleGA

Figura 19 - Pseudocódigo do SingleGA

O método, SingleGA, foi implementado na linguagem de programação C++ utilizando os paradigmas da orientação objetos e, em alguns casos, funções pré-definidas do pacote da GALib.

6 RESULTADOS COMPUTACIONAIS

6.1 Testes computacionais

Para a realização dos testes computacionais foram utilizadas instâncias geradas aleatoriamente pelo gerador de problemas de corte unidimensional, CUTGEN1, desenvolvido por Gau e Wascher (1995). Gerou-se 18 classes caracterizadas pelos diferentes valores dos parâmetros de entrada do CUTGEN1, apresentados a seguir, contendo cada uma 100 problemas, totalizando 1800 testes para aferir a qualidade do método proposto.

Os parâmetros de entrada do CUTGEN1 são:

m = número de itens demandados

W = tamanho do padrão

v_1 e v_2= limites superior e inferior dos comprimentos dos itens

$\bar{d}$ = demanda média

Os valores assumidos por cada um destes parâmetros foram:

- v_1 assumiu os valores 0,01 e 0,2
- v_2 assumiu os valores 0,2 e 0,8
- O número de itens, m, foi fixado em 10, 20 ou 40
- A demanda média, $\bar{d}$, dos itens foi de 10 ou 100
- W assumiu em todas classes o valor 1000

Foram geradas, deste modo, seis classes com itens pequenos (v_1 = 0,01 e v_2 = 0,2), seis classes com itens de larguras variadas (v_1 = 0,01 e v_2 = 0,8) e outras seis classes com itens grandes (v_1 = 0,2 e v_2 = 0,8), mostrados na Tabela 6.1.

Os parâmetros e a semente usados para gerar os 1800 problemas no CUTGEN1 foram os mesmos utilizados no Kombi234 por Foerster e Wascher (2000), Salles Neto e Moretti (2005) e Golfeto et al. (2007a).

Tabela 6.1 - Classes Geradas

Classe	v_1	v_2	m	$\bar{d}$
1	0,01	0,2	10	10
2	0,01	0,2	10	100
3	0,01	0,2	20	10
4	0,01	0,2	20	100
5	0,01	0,2	40	10
6	0,01	0,2	40	100
7	0,01	0,8	10	10
8	0,01	0,8	10	100
9	0,01	0,8	20	10
10	0,01	0,8	20	100
11	0,01	0,8	40	10
12	0,01	0,8	40	100
13	0,2	0,8	10	10
14	0,2	0,8	10	100
15	0,2	0,8	20	10
16	0,2	0,8	20	100
17	0,2	0,8	40	10
18	0,2	0,8	40	100

O CUTGEN1 foi desenvolvido devido a uma pequena quantidade de exemplos disponíveis na literatura de problemas-testes para os problemas de corte unidimensionais. Ele é uma ferramenta que gera um conjunto de exemplos com propriedades específicas definidas acima, e dados padrão para serem utilizados pelos problemas de corte unidimensional objetivando uma comparação mais aproximada e justa entre os métodos avaliados e para comparações com produções futuras Gau e Wascher (1995).

A escolha do tamanho das populações dos indivíduos, número de gerações, operadores genéticos e suas respectivas taxas de seleção, recombinação e mutação foi realizada empiricamente pois não existem na literatura valores pré-definidos.

Para um melhor entendimento dos resultados, conceitos como padrão de corte, *setup*, freqüência e objetos processados devem ser reforçados. Padrão de corte é a disposição dos itens no objeto para a realização de cortes durante sua produção. E freqüência de padrões, é a quantidades de vezes que os padrões de corte serão executados. O *Setup* é o tempo de preparação de máquina, e a quantidade de padrões distintos define a quantidade de *setup*. Por fim, define-se número de objetos processados como a soma das freqüências dos padrões.

Os resultados obtidos com o método proposto foram comparados com aqueles encontrados pela heurística Kombi234 proposta por Foerster e Wascher (2000), a heurística SHP proposta por Haessler (1975) e implementada por Salles Neto e Moretti (2005), a heurística ANLCP300 desenvolvida por Salles Neto e Moretti (2005) e a metaheurística Symbio proposta por Golfeto et al. (2007a), abordadas no capítulo 2 na seção 2.4.

O método proposto SingleGA foi implementado na linguagem C++ no compilador Bloodshed DevC++ num microcomputador Pentium Dual Core 1.73 533MHz, 2G de memória RAM. Os métodos ANLCP300 e SHP, implementados por Salles Neto (2005) foram programados na linguagem FORTRAN, utilizando o compilador g77 para Linux, num microcomputador AMD AthlonXP 1800 MHz com 512 de memória RAM. O Symbio também foi implementado na linguagem FORTRAN por Golfeto et al. (2007a), porém, utilizou o compilador Microsoft FORTRAN Power Station num micro-computador AMD SEMPRON 2300+ 1533MHz com 640 MB de memória RAM. Os resultados da heurística Kombi234, implementados pelos próprios autores Foerster e Wascher (2000), foram obtidos num código em MODULA-2 em MS-DOS 6.0 utilizando um IBM 486/66.

Os resultados serão apresentados com três valores para o custo do *setup*: $\alpha = 1$, que foi denominado SingleGA01, $\alpha = 5$, chamado de SingleGA05 e $\alpha = 10$, que denominou-se SingleGA10. Em todos os casos, o custo do número de objetos processados foi fixado em uma unidade $(c_1 = 1)$. O valor de α visa penalizar o número de padrões diferentes.

6.2 Comparação e análise do número de objetos processados e o S*etup*

A Tabela 6.2 apresenta a média dos números de objetos processados e do *setup* para os 100 problemas de cada uma das classes.

Para uma análise mais detalhada, calculou-se a variação percentual do *setup* e dos números de objetos processados do método proposto, SingleGA, em relação aos demais métodos. A expressão utilizada para o cálculo da variação do *setup* está representada pela equação 6.1, abaixo:

$$\frac{100x(Setup_{SingleGA} - Setup_{método\ comparado})}{Setup_{SingleGA}} \qquad (6.1)$$

Analogamente, a expressão para determinar a variação percentual do número de objetos processados é igual a equação (6.2).

$$\frac{100x(ObjProc_{SingleGA} - ObjProc_{método\ comparado})}{ObjProc_{SingleGA}} \quad (6.2)$$

Para uma melhor visualização do desempenho do método proposto, nas tabelas de variações percentuais utilizou-se um esquema de cores cujo significado está descrito na Tabela 6.3.

Tabela 6.3 - Legenda de cores

COR	SIGNIFICADO
	O método proposto é melhor
	As diferenças percentuais são pequenas
	A diferença é nula ou praticamente nula
	O método proposto é pior
	Quantidade de classes em que o método proposto foi melhor
	O valor das diferenças percentuais

Tabela 6.2 - Média dos números de objetos processados e do *setup*

	SHP		**Kombi234**		**ANLCP300**		**Symbio01**		**Symbio05**		**Symbio10**		**SingleGA01**		**SingleGA05**		**SingleGA10**	
classe	obj	set	obj	set	obj	set	obj	set	obj	set	obj	set	obj	set	obj	set	obj	set
1	14,17	3,95	11,49	3,4	14,3	3,02	12,59	3,09	14,19	2,02	14,84	1,85	13,48	3,07	15,15	2,19	15,56	1,97
2	116,47	5,94	110,25	7,81	121,48	4,5	115,17	6,11	116,26	5,28	117,8	4,68	121,05	5,5	125,33	4,17	127,36	3,78
3	25,29	5	22,13	4,84	25,14	4,84	25,96	5,74	27,43	4,8	28,23	4,47	27,55	6,22	30,33	4,57	32,5	4,18
4	255,33	7,31	215,93	7,28	224,86	7,28	235,93	10,59	236,89	9,86	239,72	9,36	260,55	9,38	268,02	8	269,73	7,1
5	46,89	6,87	42,96	10,75	45,66	7,02	57,17	9,89	60,85	8,44	61,42	8,23	63,47	10,89	66,57	8,69	71,14	7,79
6	433,59	10,81	424,71	25,44	432,72	10,92	472,95	30,07	518,81	14,16	518,85	14,08	613,42	14,71	618,21	13,39	623,58	12,36
7	55,84	8,84	50,21	7,9	51,54	5,54	51,58	6,36	53,04	5,48	53,75	5,21	52,25	5,87	53,75	5,46	54,23	5,35
8	515,76	9,76	499,52	9,96	495,94	8	510,5	8,51	512,92	8,16	514,65	7,76	512,94	6,8	515,62	6,4	518,21	6,07
9	108,54	17,19	93,67	15,03	114,52	10,58	99,36	10,94	99,41	10,47	101,62	10,01	101,22	10,68	103,72	9,74	104,63	9,39
10	1001,59	19,37	932,32	19,28	969,2	13,96	970,57	16,7	975,56	16,28	975,72	16,04	1019,5	14,49	1018,37	15,06	1032	13,34
11	202,8	32,2	176,97	28,74	231,84	20	198,26	23,04	201,43	22,24	201,44	21,56	231,12	29,8	249,5	26,42	249,01	24,02
12	1873,05	37,25	1766,2	37,31	1861,2	23,68	1932,16	32,93	1932,79	31,74	1917,4	31,52	2428,78	35,62	2342,27	35,33	2462,6	34,58
13	69,97	9,38	63,27	8,97	70,54	5,64	65,32	7,28	66,51	6,7	67,32	6,68	65,65	7,05	66,97	6,64	67,63	6,56
14	643,55	9,85	632,12	10,32	634,02	7,92	646,77	8,62	646,31	8,3	650,52	8,21	648,37	7,78	650,43	7,48	652,43	7,22
15	136,05	18,03	119,43	16,88	127,38	9,92	127,39	13,66	128,43	13,18	128,87	12,99	128,03	13,77	129,7	13,1	130,16	12,9
16	1253,55	19,63	1191,8	19,91	1194,74	13,88	1254,69	16,68	1253,79	16,57	1251,9	16,34	1294,64	18,16	1303,2	16,39	1301,9	15,33
17	256,01	34,39	224,68	31,46	297,58	22,6	244,49	27,22	248,67	25,82	247,62	25,62	266,34	30,12	271	26,83	270	25,86
18	2381,54	38,23	2342,4	38,28	2430,04	27,44	2414,07	32,7	2419,34	32,03	2422,41	32,18	2761,4	36,75	2753,89	35,82	2751,5	33,76

As tabelas 6.4, 6.5 e 6.6 apresentam a variação percentual do *setup* para o SingleGA com custo de *setup* $\alpha = 1$, 5 e 10 respectivamente, em relação ao demais métodos, encontrada após o cálculo da expressão (6.1). Os valores negativos encontrados nas tabelas de variação percentual indicam que o método proposto é melhor que o método comparado.

Tabela 6.4 - Variação percentual do *setup* com $c_1 = 1$ e $\alpha = 1$ em relação ao SingleGA com os demais métodos

Classe	%SHP	%Kombi234	%ANLCP	%Symbio01
1	-28,665	-10,7492	1,628664495	-0,6514658
2	-8	-42	18,18181818	-11,090909
3	19,614	22,1865	22,18649518	7,7170418
4	22,068	22,38806	22,3880597	-12,899787
5	36,915	1,285583	35,53719008	9,1827365
6	26,513	-72,9436	25,76478586	-104,41876
7	-50,596	-34,5826	5,621805792	-8,3475298
8	-43,529	-46,4706	-17,64705882	-25,147059
9	-60,955	-40,7303	0,936329588	-2,4344569
10	-33,678	-33,0573	3,657694962	-15,251898
11	-8,0537	3,557047	32,88590604	22,684564
12	-4,5761	-4,74453	33,5204941	7,5519371
13	-33,05	-27,234	20	-3,2624114
14	-26,607	-32,6478	-1,799485861	-10,796915
15	-30,937	-22,5853	27,95933188	0,7988381
16	-8,0947	-9,63656	23,56828194	8,1497797
17	-14,177	-4,44887	24,96679947	9,6281541
18	-4,0272	-4,16327	25,33333333	11,020408
Supera em:	14	14	2	10

A variação percentual do *setup* do SingleGA com $\alpha = 1$, (Tabela 6.4), apresentou melhores resultados em 14 classes em relação ao SHP e o Kombi234. Em relação ao ANLCP300 foi melhor em 2 classes e obteve em 10 classes melhores resultados que o Symbio01.

Tabela 6.5 - Variação percentual do *setup* com $c_1 = 1$ e $\alpha = 5$ em relação ao SingleGA com os demais métodos

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio05
1	-80,3653	-55,2511	-37,8995	7,762557078
2	-42,446	-87,2902	-7,91367	-26,618705
3	-9,40919	-5,9081	-5,9081	-5,03282276
4	8,625	9	9	-23,25
5	20,9436	-23,7054	19,21749	2,876869965
6	19,2681	-89,9925	18,4466	-5,75056012
7	-61,9048	-44,6886	-1,4652	-0,36630037
8	-52,5	-55,625	-25	-27,5
9	-76,4887	-54,3121	-8,62423	-7,49486653
10	-28,6189	-28,0212	7,304117	-8,10092962
11	-21,8774	-8,78123	24,29977	15,82134746
12	-5,43447	-5,6043	32,97481	10,16133598
13	-41,2651	-35,0904	15,06024	-0,90361446
14	-31,6845	-37,9679	-5,88235	-10,9625668
15	-37,6336	-28,855	24,27481	-0,61068702
16	-19,7682	-21,4765	15,31422	-1,09823063
17	-28,1774	-17,2568	15,76593	3,764442788
18	-6,72808	-6,86767	23,39475	10,58068118
Supera em:	**15**	**17**	**7**	**12**

Na Tabela 6.5 em que o custo de *setup* foi $\alpha = 5$, o SingleGA apresentou uma variação percentual dos resultados melhores que com o custo 1. Foi melhor que o SHP em 15 classes, ganhou do Kombi234 em 17 classes, e superou o ANLCP300 em 7 classes, e obteve melhores resultados 12 classes em relação ao Symbio05.

Tabela 6.6 - Variação percentual do *setup* com $c_1 = 1$ e $\alpha = 10$ em relação ao SingleGA com demais métodos

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio10
1	-100,508	-72,5888	-53,2995	6,091370558
2	-57,1429	-106,614	-19,0476	-23,8095238
3	-19,6172	-15,7895	-15,7895	-6,93779904
4	-2,95775	-2,53521	-2,53521	-31,8309859
5	11,81	-37,9974	9,884467	-5,64826701
6	12,5405	-105,825	11,65049	-13,9158576
7	-65,2336	-47,6636	-3,5514	2,61682243
8	-60,7908	-64,0857	-31,7957	-27,8418451
9	-83,0671	-60,0639	-12,6731	-6,6027689
10	-45,2024	-44,5277	-4,64768	-20,2398801
11	-34,055	-19,6503	16,73605	10,24146545
12	-7,72123	-7,89474	31,52111	8,849045691
13	-42,9878	-36,7378	14,02439	-1,82926829
14	-36,4266	-42,9363	-9,69529	-13,7119114
15	-39,7674	-30,8527	23,10078	-0,69767442
16	-28,0496	-29,8761	9,458578	-6,58838878
17	-32,9853	-21,6551	12,60634	0,928074246
18	-13,2405	-13,3886	18,72038	4,680094787
Supera em:	**16**	**18**	**9**	**12**

Já com o custo de *setup* $\alpha = 10$, (Tabela 6.6), foi que o SingleGA obteve uma melhor variação percentual do *setup* e apresentou seus melhores resultados (em relação ao custo 1 e 5). Foi melhor que todos os métodos na maioria das classes, exceto em relação ao ANLCP300 com o qual empatou em número de classes. Superou o SHP em 16 classes, obteve melhores resultados em todas as classes quando comparado com o Kombi234, foi superior ao ANLCP300 em 9 classes. Obteve melhores resultados em 12 quando comparado com o Symbio10.

Similarmente ao *setup*, as tabelas 6.7, 6.8 e 6.9 mostra a variação percentual do número de objetos processados do SingleGA com os três custos citados, em comparação com os outros métodos.

Tabela 6.7 - Variação percentual do número de objetos processados com $c_1 = 1$ e $\alpha = 1$ em relação ao SingleGA com os demais métodos

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio01
1	-5,1187	14,7626113	-6,08308605	6,60237389
2	3,78356	8,92193309	-0,35522511	4,8574969
3	8,20327	19,6733212	8,7477314	5,77132486
4	2,00345	17,1253118	13,6979467	9,44924199
5	26,1226	32,3144793	28,060501	9,92594927
6	29,316	30,7635878	29,457794	22,8994816
7	-6,8708	3,90430622	1,35885167	1,28229665
8	-0,5498	2,6162904	3,31422778	0,47568916
9	-7,2318	7,4590002	-13,1396957	1,83758151
10	1,75674	8,55125061	4,93379107	4,79941148
11	12,2534	23,4293873	-0,31152648	14,2177224
12	22,881	27,280363	23,3689342	20,4473028
13	-6,5804	3,62528561	-7,44859101	0,50266565
14	0,7434	2,50628499	2,21324244	0,24677268
15	-6,2642	6,71717566	0,50769351	0,49988284
16	3,17386	7,94352098	7,71643082	3,08579991
17	3,8785	15,641661	-11,7293685	8,20379966
18	13,7561	15,1734627	11,9997103	12,5780401
Supera em:	6	0	6	0
Dif % =	(4) 3,3%	(4) 3,9%	(4) 3,3%	(7) 3%

Para os valores $c_1 = 1$ e $\alpha = 1$, (Tabela 6.7), a variação percentual do número de objetos processados do SingleGA apresentou seus melhores resultados. Foi superior ao SHP e ANLCP300 em 6 classes, sendo que a diferença em outras 4 classes destes dois métodos em relação ao SingleGA foi inferior a 3,5%. Não superou os demais métodos em nenhuma outra classe. No entanto, a diferença em 4 classes em relação ao Kombi234 ficou por volta de 3,9% e, quando confrontado com o Symbio01 a diferença em 7 classes não foi superior aos 3% .

Tabela 6.8 - Variação percentual do número de objetos processados com $c_1 = 1$ e $\alpha = 5$ em relação ao SingleGA com os demais métodos

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio05
1	6,46865	24,15842	5,610561	6,3366337
2	7,06934	12,03223	3,07189	7,2368946
3	16,6172	27,03594	17,11177	9,5614903
4	4,73472	19,43512	16,10328	11,614805
5	29,5629	35,46643	31,41055	8,5924591
6	29,8636	31,30004	30,00437	16,078679
7	-3,8884	6,586047	4,111628	1,3209302
8	-0,0272	3,122455	3,816764	0,5236414
9	-4,6471	9,689549	-10,4126	4,1554184
10	1,64773	8,449778	4,828304	4,2037766
11	18,7174	29,07014	7,078156	19,266533
12	20,0327	24,59452	20,53862	17,482186
13	-4,4796	5,524862	-5,33075	0,6868747
14	1,05776	2,815061	2,522946	0,6334271
15	-4,8959	7,918273	1,788743	0,9791827
16	3,80985	8,548189	8,322591	3,7914365
17	5,53137	17,09225	-9,80812	8,2398524
18	13,5209	14,94214	11,75973	12,14827
Supera em:	5	0	3	0
Dif % =	(2) 1,6%	(1) 2,8%		(5) 1,3%

A variação percentual do número de objetos processados com custos $c_1 = 1$ e $\alpha = 5$, (Tabela 6.8), mostrou a superioridade do método em 5 classes sobre o SHP e diferença percentual de menos de 1,6% em mais duas classes. Em relação ao ANLCP300 foi melhor em 3 classes. Não obteve melhores resultados quando comparado com o Kombi234 e o Symbio05. Porém a diferença percentual em 5 classes em relação ao Symbio05 caiu para 1,3% .

Tabela 6.9 - Variação percentual do número de objetos processados com $c_1 = 1$ e $\alpha = 10$ em relação ao SingleGA com os demais métodos

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio10
1	8,93316	26,15681	8,097686	4,62724936
2	8,55057	13,43436	4,616834	7,50628141
3	22,1846	31,90769	22,64615	13,1384615
4	5,33867	19,94587	16,63515	11,1259408
5	34,0877	39,61203	35,8167	13,6631993
6	30,4676	31,89166	30,60714	16,7949581
7	-2,9688	7,412871	4,960354	0,88511894
8	0,47278	3,606646	4,297486	0,68698018
9	-3,737	10,47501	-9,45236	2,87680398
10	2,94483	9,657164	6,083451	5,45165604
11	18,5575	28,93057	6,895305	19,1036505
12	23,9389	28,27789	24,42012	22,1379378
13	-3,46	6,446843	-4,30282	0,45837646
14	1,36107	3,112978	2,821759	0,29275171
15	-4,5252	8,2437	2,135833	0,99108789
16	3,71676	8,459683	8,233866	3,84349509
17	5,18148	16,78519	-10,2148	8,28888889
18	13,4454	14,86794	11,68276	11,9600653
Supera em:	4	0	3	0
Dif % =	(4) 3,7%		(2) 2,8%	(5) 1%

Com custos $c_1 = 1$ e $\alpha = 10$, a variação percentual do número de objetos processados (Tabela 6.9), apresentou os seguintes resultados: superioridade em 4 classes do SHP e em mais 4 a diferença percentual ficou por volta dos 3,7%. Superou o ANLCP300 em 3 classes e em mais 2 classes diferenciou-se em 2,8%. Nos demais, métodos o SingleGA não apresentou melhores resultados em nenhuma classe, no entanto, comparados com o Symbio10, obteve diferenças muitos pequenas apresentando uma diferença em 5 classes inferior a 1%.

A variação percentual do número de objetos processados mostra que apesar do desempenho do método proposto ser inferior aos demais, as diferenças percentuais (%) na maioria das médias das classes do SingleGA, quando comparados com os outros métodos, não é muito grande.

O Kombi234 mantém constante o número de objetos processados e é o melhor método para minimizar este objetivo. Sendo assim, é esperado que ele seja melhor, na média, que todos os outros métodos comparados.

Em relação ao número de objetos processados o desempenho do SingleGA caiu quando comparado com a média dos *setups*, apresentando resultados piores à medida que o custo de *setup* aumentava, isso se justifica por se tratar de objetivos inversamente relacionados.

6.3 Comparação e análise do custo total

O custo total é o custo da função objetivo e seu cálculo é realizado pela seguinte expressão representada pela equação 6.3:

$$custo_{total} = c_1 \times obj_{media} + \alpha \times setup_{media} \quad (6.3)$$

Sendo:

$obj_{media} =$ é a média do número de objetos processados entre os 100 problemas de cada classe.

$setup_{media} =$ é a média do número de *setup* entre os 100 problemas de cada classe.

A expressão utilizada para o cálculo da variação do custo total está representada pela equação 6.4, abaixo:

$$100 \times \left(\frac{CustoTotal_{SingleGA} - CustoTotal_{metcomparado}}{CustoTotal_{SingleGA}} \right) \quad (6.4)$$

O custo total médio foi calculado para aferir a qualidade e desempenho do método proposto tratando, ao mesmo tempo, os dois objetivos para efeito de comparação do valor da função objetivo com os demais métodos. A Tabela 6.10 traz a média do custo total para $c_1 = 1$ e $\alpha = 1$. E, para um melhor compreender estes valores, a Tabela 6.11 apresenta a variação percentual do custo total do método proposto SingleGA01, em relação aos métodos SHP, Kombi234, ANLCP300 e Symbio01 o que torna mais fácil a análise de desempenho do método.

Tabela 6.10 - Médias do custo total para $c_1 = 1$ e $\alpha = 1$

CLASSE	SHP	Kombi234	ANLCP300	Symbio01	SingleGA01
1	18,12	14,89	17,32	15,68	**16,55**
2	122,41	118,06	125,98	121,28	**126,55**
3	30,29	26,97	29,98	31,7	**33,77**
4	262,64	223,21	232,14	246,52	**269,93**
5	53,76	53,71	52,68	67,06	**74,36**
6	444,4	450,15	443,64	503,02	**628,13**
7	64,68	58,11	57,08	57,94	**58,12**
8	525,52	509,48	503,94	519,01	**519,74**
9	125,73	108,7	125,1	110,3	**111,9**
10	1020,96	951,6	983,16	987,27	**1033,99**
11	235	205,71	251,84	221,3	**260,92**
12	1910,3	1803,51	1884,88	1965,09	**2464,4**
13	79,35	72,24	76,18	72,6	**72,7**
14	653,4	642,44	641,94	655,39	**656,15**
15	154,08	136,31	137,3	141,05	**141,8**
16	1273,18	1211,71	1208,62	1271,37	**1312,8**
17	290,4	256,14	320,18	271,71	**296,46**
18	2419,77	2380,68	2457,48	2446,77	**2798,15**

Tabela 6.11 - Variação percentual do custo total do SingleGA01 em relação aos demais métodos para $c_1 = 1$ e $\alpha = 1$

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio01
1	**-9,4864**	**10,03021**	**-4,65257**	**5,25679758**
2	**3,271434**	**6,708811**	**0,450415**	**4,16436191**
3	**10,305**	**20,13622**	**11,22298**	**6,12970092**
4	**2,7007**	**17,30819**	**13,99993**	**8,67261883**
5	**27,70307**	**27,77031**	**29,15546**	**9,81710597**
6	**29,25031**	**28,3349**	**29,37131**	**19,9178514**
7	**-11,287**	**0,017206**	**1,789401**	**0,30970406**
8	**-1,11209**	**1,974064**	**3,039982**	**0,14045484**
9	**-12,3592**	**2,859696**	**-11,7962**	**1,42984808**
10	**1,260167**	**7,968162**	**4,915908**	**4,51841894**
11	**9,934079**	**21,15974**	**3,479994**	**15,184731**
12	**22,48417**	**26,81748**	**23,51566**	**20,2609154**
13	**-9,14718**	**0,632737**	**-4,7868**	**0,13755158**
14	**0,419111**	**2,089461**	**2,165663**	**0,11582717**
15	**-8,66008**	**3,87165**	**3,173484**	**0,52891396**
16	**3,017977**	**7,700335**	**7,93571**	**3,15585009**
17	**2,044121**	**13,60049**	**-8,00108**	**8,34851245**
18	**13,52251**	**14,9195**	**12,17483**	**12,5575827**
Supera em:	**6**	**0**	**4**	**0**
Dif % =	**(6) 3,5%**	**(5) 3%**	**(6) 3,5%**	**(6) 1,5%**

Neste caso, ($c_1 = 1$ e $\alpha = 1$), o SingleGA01 foi melhor em 6 classes que o SHP e em outras 6 teve uma diferença inferior a 3,5%, quanto ao ANLCP300 foi melhor em 4 classes e em 6 apresentou diferença menor que 3,5%. Embora não tenha apresentado melhores resultados quando comparado com o Kombi234 e o Symbio01, a diferença percentual do primeiro em 5 classes foi de 3%, enquanto o último esta diferença caiu para 1,5% em 6 classes.

A Tabela 6.12 apresenta a média do custo total para ($c_1 = 1$ e $\alpha = 5$), e com os mesmos objetivos acima, a Tabela 6.13 traz a variação percentual destas médias. Aqui o método proposto analisado é o SingleGA05.

Tabela 6.12 - Médias do custo total para $c_1 = 1$ e $\alpha = 5$

CLASSE	**SHP**	**Kombi234**	**ANLCP300**	**Symbio05**	**SingleGA05**
1	33,92	28,49	29,4	24,29	**26,1**
2	146,17	149,3	143,98	142,66	**146,18**
3	50,29	46,33	49,34	51,43	**53,18**
4	291,88	252,33	261,26	286,19	**308,02**
5	81,24	96,71	80,76	103,05	**110,02**
6	487,64	551,91	487,32	589,61	**685,16**
7	100,04	89,71	79,24	80,44	**81,05**
8	564,56	549,32	535,94	553,72	**547,62**
9	194,49	168,82	167,42	151,76	**152,42**
10	1098,44	1028,72	1039	1056,96	**1093,67**
11	363,8	320,67	331,84	312,63	**381,6**
12	2059,3	1952,75	1979,6	2091,49	**2518,92**
13	116,87	108,12	98,74	100,01	**100,17**
14	692,8	683,72	673,62	687,81	**687,83**
15	226,2	203,83	176,98	194,33	**195,2**
16	1351,7	1291,35	1264,14	1336,64	**1385,15**
17	427,96	381,98	410,58	377,77	**405,15**
18	2572,69	2533,8	2567,24	2579,49	**2932,99**

Tabela 6.13 - Variação percentual do custo total do SingleGA05 em relação aos demais métodos para $c_1 = 1$ e $\alpha = 5$

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio05
1	-29,9617	-9,1570881	-12,6436782	6,9348659
2	0,006841	-2,1343549	1,504993843	2,4079901
3	5,434374	12,8807823	7,220759684	3,2907108
4	5,239919	18,0799948	15,18083241	7,0872021
5	26,15888	12,0978004	26,59516452	6,3352118
6	28,8283	19,4480121	28,8750073	13,945648
7	-23,43	-10,684762	2,233189389	0,7526218
8	-3,09339	-0,3104342	2,132865856	-1,1139111
9	-27,6014	-10,759743	-9,84122819	0,433014
10	-0,43615	5,93872009	4,998765624	3,3565884
11	4,66457	15,9669811	13,03983229	18,073899
12	18,24671	22,4766964	21,41076334	16,96878
13	-16,6717	-7,9365079	1,427573126	0,1597285
14	-0,72256	0,59753137	2,065917451	0,0029077
15	-15,8811	-4,4211066	9,334016393	0,4456967
16	2,414901	6,77182977	8,736237953	3,5021478
17	-5,63001	5,71886955	-1,34024435	6,7579909
18	12,28439	13,610343	12,47020958	12,052547
Supera em:	9	7	3	1
Dif % =	1 =		(5) 2,3%	(3) 0,7% 2=

Para os custos $c_1 = 1$ e $\alpha = 5$, o custo total do SingleGA05 apresentou melhores resultados em 9 classes do SHP e na classe 2 obteve praticamente o mesmo resultado (0,006841%). Já no Kombi234 ela foi superior em 7 classes e no ANLCP300 em 3 classes e em outras 5 classes a diferença percentual foi de 2,3%. Finalmente, ao ser confrontado com o Symbio05, foi melhor em 1 classe, em 2 classes (13 e 14) teve resultados praticamente iguais e em mais 3 apresentou uma diferença de apenas 0,7%.

A Tabela 6.14 traz a média do custo total para $c_1 = 1$ e $\alpha = 10$, e com os mesmos objetivos acima, e a Tabela 6.15 apresenta a variação percentual destas médias. Aqui, o método proposto analisado é o SingleGA10.

Tabela 6.14 - Médias do custo total para $c_1 = 1$ e $\alpha = 10$

CLASSE	SHP	Kombi234	ANLCP300	Symbio10	SingleGA10
1	53,67	45,49	44,5	33,34	**35,26**
2	175,87	188,35	166,48	164,6	**165,16**
3	75,29	70,53	73,54	72,93	**74,3**
4	328,43	288,73	297,66	333,32	**340,73**
5	115,59	150,46	115,86	143,72	**149,04**
6	541,69	679,11	541,92	659,65	**747,18**
7	144,24	129,21	106,94	105,85	**107,73**
8	613,36	599,12	575,94	592,25	**578,91**
9	280,44	243,97	220,32	201,72	**198,53**
10	1195,29	1125,12	1108,8	1136,12	**1165,38**
11	524,8	464,37	431,84	417,04	**489,21**
12	2245,55	2139,3	2098	2232,6	**2808,36**
13	163,77	152,97	126,94	134,12	**133,23**
14	742,05	735,32	713,22	732,62	**724,63**
15	316,35	288,23	226,58	258,77	**259,16**
16	1449,85	1390,9	1333,54	1415,3	**1455,24**
17	599,91	539,28	523,58	503,82	**528,6**
18	2763,84	2725,2	2704,44	2744,21	**3089,09**

Tabela 6.15 - Variação percentual do custo total do SingleGA10 em relação aos demais métodos para $c_1 = 1$ e $\alpha = 10$

Classe	%SHP	%Kombi234	%ANLCP300	%Symbio10
1	**-52,2121**	**-29,013046**	**-26,2053318**	**5,44526376**
2	**-6,48462**	**-14,04093**	**-0,79922499**	**0,33906515**
3	**-1,33244**	**5,07402423**	**1,022880215**	**1,84387618**
4	**3,609896**	**15,2613506**	**12,64050715**	**2,17474247**
5	**22,44364**	**-0,9527644**	**22,26247987**	**3,56951154**
6	**27,50207**	**9,11025456**	**27,47129206**	**11,714714**
7	**-33,8903**	**-19,938736**	**0,733314768**	**1,7451035**
8	**-5,95084**	**-3,4910435**	**0,513033114**	**-2,3043306**
9	**-41,2582**	**-22,888228**	**-10,9756712**	**-1,6068101**
10	**-2,56654**	**3,45466715**	**4,855068733**	**2,51076902**
11	**-7,27499**	**5,07757405**	**11,72707017**	**14,7523558**
12	**20,04052**	**23,8238687**	**25,29447792**	**20,5016451**
13	**-22,9228**	**-14,816483**	**4,721158898**	**-0,6680177**
14	**-2,40399**	**-1,4752356**	**1,574596691**	**-1,1026317**
15	**-22,0674**	**-11,217009**	**12,57138447**	**0,15048619**
16	**0,370386**	**4,42126385**	**8,362881724**	**2,74456447**
17	**-13,4904**	**-2,0204313**	**0,949678396**	**4,68785471**
18	**10,52899**	**11,7798446**	**12,45188713**	**11,164453**
Supera em:	**12**	**10**	**3**	**4**
Dif % =	**(1) 0,4%**		**(5) 2%**	**(6) 2,8% 1=**

A média do custo total do SingleGA10 para $c_1 = 1$ e $\alpha = 10$, representada pela Tabela 6.14, obteve os melhores resultados quando comparados com custos de *setup* 1 e 5. A Tabela 6.15 apresenta a variação percentual destas médias.

Os resultados obtidos pelo SingleGA10 foram melhores que o SHP em 12 classes e em mais uma, na classe 16, apresentou um diferença de apenas 0,4%. Superou o Kombi234 em 10 classes, o ANLCP300 em 3 classes, e neste último, em mais 5 classes apresentou uma diferença inferior a 2%. E em relação ao Symbio10 foi superior em 4 classes e em 7 a diferença apresentada foi inferior a 3%.

O SingleGA10 foi melhor que o Kombi234 e o SHP a maioria das classes enquanto em relação ao ANLCP300 e Symbio10 o mesmo não ocorreu. Porém, devido às pequenas diferenças das variações percentuais, analisou-se graficamente o comportamento do SingleGA10 com o ANLCP300 (Gráfico 6.1) e o Symbio10 (Gráfico 6.2), e notou-se que o desempenho praticamente se igualam em 15 classes com o ANLCP300 e em 16 quando comparadas com o Symbio10. As maiores diferenças apresentadas pelo método proposto em relação a todos os outros se encontram na classe 12 e 18.

Gráfico 6.1 - Representação gráfica das médias do custo total do SingleGA10 com o ANLCP300

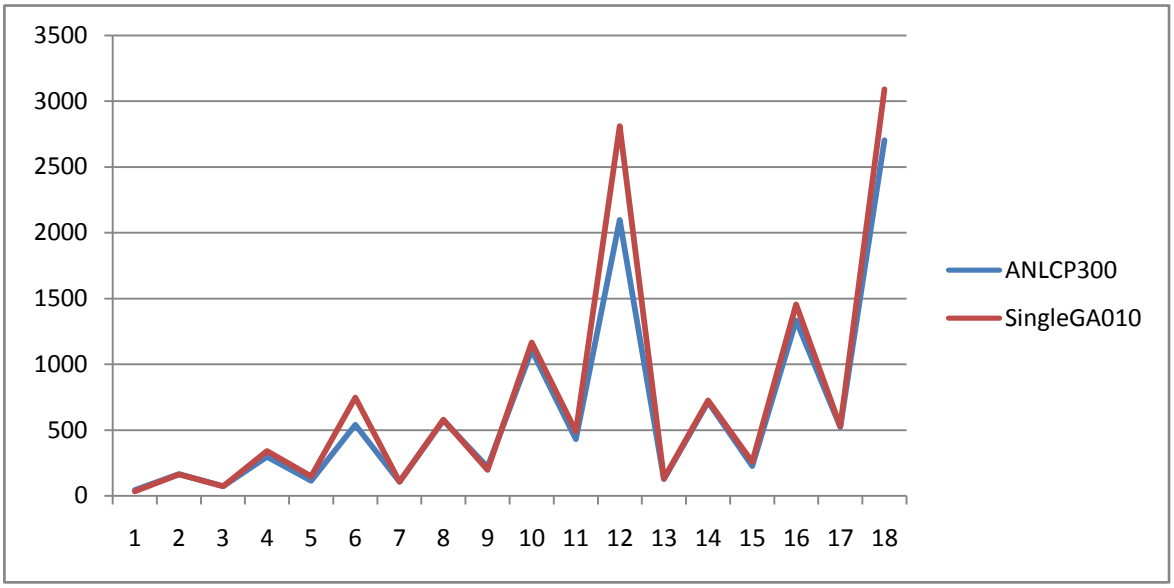
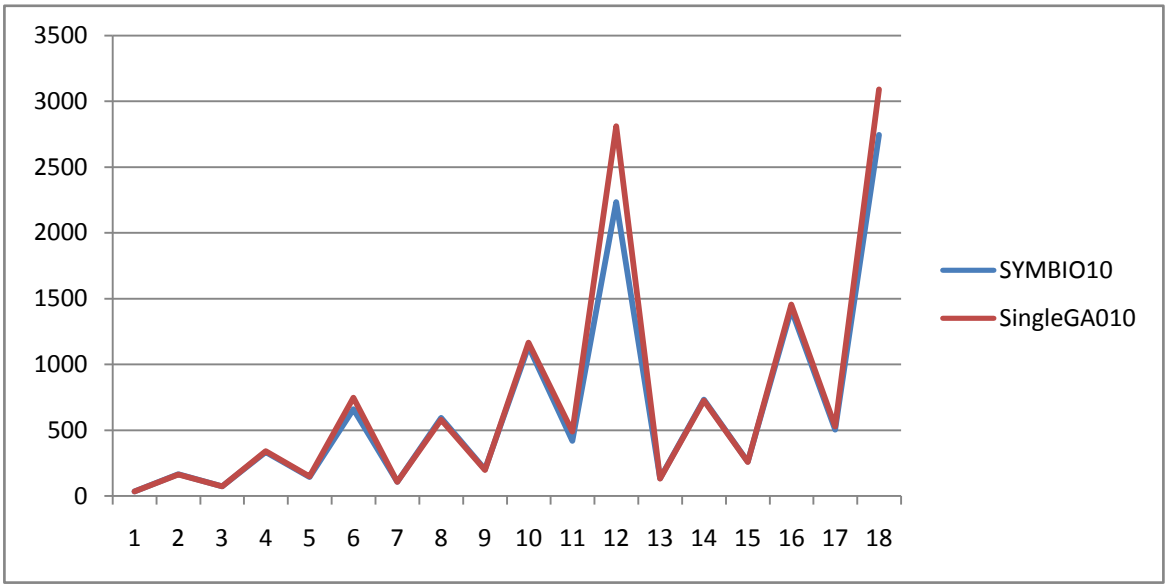


Gráfico 6.2 - Representação gráfica das médias do custo total do SingleGA10 com o Symbio10



Segundo Salles Neto e Moretti (2005), quando o *setup* é o primeiro objetivo a ser atendido o custo de *setup* deve ser superior ao número de objetos processados. E quanto maior o custo de *setup* maior o desempenho do método proposto, SingleGA. A definição deste valor, também abordado e exemplificado por Salles Neto e Moretti (2005), bem como o custo do número de objetos processados levam em consideração, diversas variáveis como tempo de entrega, demanda exigida, custo de mão de obra, entre outros fatores.

Assim, notou-se que para maiores custos de *setup*, o método proposto apresentou seus melhores resultados, comparados com os demais métodos em relação ao custo total. Visto que o número de objetos processados e *setup* são objetivos, geralmente, conflitantes, o número de objetos processados foi maior no SingleGA com $\alpha = 10$ na maioria das classes que os comparados com suas demais variações com custos de *setups* inferiores.

O tempo computacional médio dos métodos, em segundos, de cada uma das 18 classes encontra-se na Tabela 6.16. Este tempo tem caráter apenas ilustrativo e informativo. Ele não pode ser utilizado para efeito comparativo pois, cada método utiliza plataformas e linguagens de implementação diferentes.

Tabela 6.16 - Tempo médio em segundos dos métodos

CLASSE	SHP	Kombi234	ANLCP300	Symbio05	SingleGA05
1	0,01	0,14	1,12	18,54	27,8
2	0,08	1,14	3,17	37,88	54,02
3	0,17	1,74	0,89	33,25	162,06
4	0,21	16	1,15	68,11	202,95
5	0,27	38,03	0,64	58,29	432,39
6	0,31	379,17	0,91	158,04	478,96
7	0,01	0,07	18,1	19,62	33,54
8	0,02	0,2	11,78	48,48	60,93
9	0,04	3,37	105,49	38,75	251,31
10	0,06	3,25	106,59	127,25	232,5
11	0,22	36,26	216,97	117,25	494,78
12	0,32	76,31	376,04	426,08	477,83
13	0,01	0,08	7,43	17,66	13,4
14	0,02	0,13	2,02	31,19	19,33
15	0,03	1,81	60,02	41,12	47,17
16	0,04	2,6	39,87	133,9	63,58
17	0,16	50,93	267	153,45	129,13
18	0,24	70,94	538,92	388,89	206,79

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 Conclusões Gerais

Este trabalho realizou o estudo de uma nova abordagem via algoritmo genético para solucionar o Problema de Corte unidimensional com o objetivo de minimizar o número de objetos processados e o *setup*. Apresentou-se uma revisão sobre o tema, descrevendo suas características e apontando o conflito entre estes dois objetivos. Estudou-se também a metaheurística Algoritmos Genéticos para a elaboração e aplicação de um novo algoritmo genético que sozinho conseguiu resolver estes objetivos conflitantes.

A contribuição do método proposto reside no fato de não se ter conhecimento na literatura, até o momento, da implementação de um único algoritmo genético utilizado na minimização desses dois objetivos simultaneamente. Além da possibilidade de trabalhar com os dois custos, (c_1 e α), diretamente na função objetivo, que é uma vantagem do método sobre o Kombi234 e o SHP.

Para validar o algoritmo proposto foram utilizados 1800 problemas-teste gerados aleatoriamente pelo gerador de problemas de corte unidimensional, CUTGEN1 através de uma análise comparativa do seu desempenho com os métodos SHP, Kombi234, ANLCP300 e o Symbio. Verificou-se que o método proposto foi superior ao Kombi234 e SHP na maioria das classes testadas, e obteve bons resultados em algumas classes quando comparados com o ANLCP300 e o Symbio10. À medida que se aumentou o custo de *setup*, o método apresentou melhores resultados.

Quando o *setup* é o principal objetivo, o método proposto apresentou seu melhor desempenho em relação a sua minimização em praticamente todas as classes de todos os métodos, exceto ao ANLCP300 com o qual empatou em número de classes.

O excelente desempenho em minimizar o número de *setups*, prejudicou a eficiência em minimizar o número de objetos processados na maioria das classes, fato justificado por se tratarem de objetivos geralmente conflitantes.

As análises gráficas realizadas mostraram o bom desempenho do método proposto e deixam mais evidente que são pequenas as variações percentuais para a maioria das classes quando comparado com os outros métodos, validando assim, o novo

método. No entanto, sua grande desvantagem em relação aos demais é o tempo computacional. Acredita-se que a escolha dos parâmetros e os operadores genéticos adotados possam ser os responsáveis por este alto custo computacional, principalmente o operador de seleção que seleciona sempre os mais aptos, prejudicando assim, a diversidade da população.

O método atendeu aos objetivos pretendidos mas notou-se que é possível refiná-lo e apresentar resultados superiores ao encontrados, com um estudo ainda mais aprofundado dos parâmetros como tamanho da população, operadores genéticos e suas respectivas taxas.

E, baseado nos resultados computacionais apresentados, pode-se afirmar que o novo método é competitivo e promissor no ramo de Problemas de Corte unidimensionais.

7.2 Trabalhos futuros

Como trabalho futuro visar-se-á o estudo do comportamento de outros operadores genéticos, principalmente o de seleção e o de recombinação para obter melhores resultados num menor espaço de tempo e acrescentar uma restrição à demanda para evitar que os números de objetos processados sejam muito elevados. Depois disso, buscar-se-á adaptar o novo método para tratar o problema de corte unidimensional proposto como um problema multiobjetivo.

Por fim, pretende-se utilizar os conceitos de simbiose entre populações diferentes para a implementação de um método que desenvolva uma relação mútua entre indivíduos de espécies distintas. Este estudo é motivado pelo fato de só existir um trabalho na literatura que trata o problema por este ângulo e o mesmo, obteve bons resultados. Para as propostas futuras citadas, serão utilizados também os conceitos e técnicas de paralelização com o objetivo de torná-los mais eficientes.

REFERÊNCIAS BIBLIOGRÁFICAS

ARENALLES, M.N.; MORABITO, R.; YANASSE, H.H. O problema de corte e empacotamento e aplicações industriais. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 20., 1997, Gramado-RS. [Anais...] . Gramado: [SBMAC], 1997.

ALLABY, M. Dictionary of ecology. New York: Oxford University Press, 1998.

BÄCK, T. Evolutionary algorithms in theory and practice. New York: Oxford University Press, 1996.

BÄCK, T., FOGEL, D.B. MICHALEWICZ, Z. (eds.). Handbook of evolutionary computation. New York: Institute of Physics Publishing and Oxford University Press, 1997.

_____. Evolutionary computation 1: basic algorithms and operators. New York: Institute of Physics Publishing, 2000a.

_____. Evolutionary computation 2: advanced algorithms and operators. New York: Institute of Physics Publishing, 2000b.

BELOV, G. E SCHEITHAUER, G. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research, v. 171, n.1, p. 85–106, 2006

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, v. 35, n. 3, p. 268–308, 2003.

BRUN, A. L. Algoritmos genéticos. Disponível em: <http://www.fag.edu.br/professores/suzan/.../MC01(AG).PDF>. Acesso em: 03/03/2009. Apostila preparada para o Encontro Paranaense de Computação.

CARNEIRO, S. A. Problema de corte via algoritmo genético. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, 1994.

CARVALHO, A.P.L. F. Algoritmos genéticos: notas de aula. Disponível em: <http://www.icmc.usp.br/~andre/research/genetic/index.htm>. Acesso em: 21/12/2008

CHERRI, A.C. Heurística para o problema de corte com o reaproveitamento das sobras de material. 2006. Dissertação. (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos-SP, 2006.

COELHO, A.M. Uma abordagem via algoritmos meméticos para a solução do problema de horário escolar. 2006. Dissertação (Mestrado em Modelagem Matemática e Computacional) - Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2006.

DARWIN, C. The origin of species. London: John Murray, 1859. (Penguin Classics, 1985).

DEB, K. Multi-objective optimization using evolutionary algorithms. New York: Wiley, 2001.

DEITEL, H.M.; DEITEL, P.J. C++: como programar. 3.ed. Porto Alegre: Bookman, 2001.

DENNIS, J. E.; SCHNABEL, R. B. Numerical methods for unconstrained optimization and nonlinear equations. Englewood Cliffs: Prentice-Hall, 1983.

DOWSLAND, K.; DOWSLAND, W. Packing problems. European Journal of Operational Research, v. 56, p 2-14, 1992.

DYCKHOFF, H. A typology of cutting and packing problems. European Journal of Operation Research, v. 444, p 145-159, 1990.

EGUCHI, T.; HIRASAWA, K.; HU, J. Symbiotic evolutional models in multiagent systems. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2003, v. 2, p. 739-746, 2003.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. Journal of Global Optimization, v. 6, p. 109–133, 1995.

FERREIRA, A. B. H. Novo dicionário Aurélio Século XXI. 3 ed. São Paulo: Nova Fronteira, 1999.

FOERSTER, H.; WASCHER, G. Pattern reduction in one-dimensional cutting-stock problem. International Journal of Prod. Res., v. 38, p.1657-1676, 2000.

GAU, T.; WASCHER, G. CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem. European Journal of Operational Research, v. 84, p. 572-579, 1995.

GILMORE, P.C.; GOMORY, R.E. A linear programming approach to the cuting stock problem I. Operations Research, v. 9, p. 849-859, 1961.

_____. A linear programming approach to the cutting stock problem II. Operations Research, v. 11, p. 863-888, 1963.

GLOVER F.; LAGUNA M. Tabu search. Massachussets: Kluwer, 1997.

GLOVER, F.; KOCHENBERGER, G.A. (eds.). Handbook of metaheuristics. Dordrecth: Kluwer, 2003.

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

GOLFETO, R. R.; MORETTI, A. C.; SALLES NETO, L. L. Algoritmo genético simbiótico aplicado ao problema de corte unidimensional. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007. Anais... 2007a .

GOLFETO, R. R.; MORETTI, A. C.; SALLES NETO, L. L. Algoritmo genético simbiótico aplicado ao problema de corte unidimensional multi-objetivo. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007. Anais... 2007b.

HAESSLER, R. Controlling cutting pattern changes in one-dimensional trim problems. Operations Research, v. 23, p. 483-493,1975.

_____. A note on computacional modifications to the Gilmore-Gomory cutting stock algorithm. Operations Research, v. 28, p. 1001-1005, 1980

HERRERA, F.; M. LOZANO. Adaptative genetic operators based on coevolution with fuzzy behaviors. IEEE Transactions on Evolutionary Computation, v.5, n.2, p.149-165, 2001.

HIRASAWA, K.; ISHIKAWA, I.; HU, J.; JIN, C.; MURATA J. Genetic symbiosis algorithm. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2000. Proceedings … V. 2, p. 02-xxvi, 2000.

HOLLAND, J.H. Outline for a logical theory of adaptive systems. J. Assoc. Comput. Mach., v. 3, p. 297-314, 1962.

_____. Adaptation in natural and artificial systems. Michigan: University of Michigan Press, 1975.

_____. Adaptation in natural and artificial systems. 2nd ed. Massachussets: MIT Press, 1992.

KANTOROVICH, L.V. Mathematical methods of organizing and planning production. Management Science, v. 6, p 366-422, 1960.

KIRKPATRICK, S.; GELATTI JR., C.D.; VECCHI, M. P. Optimization by simulated annealing. Science, v. 220, p. 671–680, 1983

LINDEN, R. Algoritmos genéticos. Rio de Janeiro: Brasport , 2006.

LOURENÇO, H. R.; MARTIN, O.; STUETZLE, T. Iterated local search. In: Handbook of metaheuristics. Norwell: Kluwer, 2002. p. 321–353.

LUCAS, D.C. Algoritmos genéticos: um estudo de seus conceitos fundamentais e aplicação no problema de grade de horária. 2000. Monografia (Graduação em Informática) - Instituto de Física e Matemática, Universidade Federal de Pelotas, Pelotas, 2000.

MAO, J.; HIRASAWA, K.; HU, J.; MURATA, J. Genetic symbiosis algorithm for multiobjective optimization problem. In: IEEE Interenational Workshop on Robot and Human Interactive Communication, 2000. Proceedings… p.137-142, 2000.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation, v. 4, n. 1, p. 1-32, 1996.

MICHALEWICZ, Z.; FOGEL, D. B. How to solve it: modern heuristics. [S.l.]: Springer, 2000.

MIYAZAWA, F. K. Problemas de corte e empacotamento. Disponível em: http://www.ic.unicamp.br/~fkm/problems/empacotamento.html. Acesso em: 10/01/2009.

MIRANDA, M. N. de. Algoritmos genéticos: fundamentos e aplicações. Notas de aula. Disponível em: http://www.gta.ufrj.br/~marcio/genetic.html. Acesso em: 21/12/2008.

PACHECO, M. A. Algoritmos genéticos: princípios e aplicações, versão 1. Notas de Aula em 14/07/1999. Disponível em: <www.ica.ele.puc-rio.br>. Acesso em: 22/12/2008.

PIANKA, E. R. Evolutionary ecology. New York: HarperCollins,1994.

POLDI, K. C.; ARENALES, M. N. Sobre uma heurística de redução de padrões de corte para o problema de corte de estoque. Tema - Tendências em Matemática Aplicada e Computacional, v. 4, n. 2, p. 227-236, 2003.

_____. Heurísticas para o problema de corte unidimensional inteiro. Pesquisa Operacional, v. 26, p. 473-492, 2006.

SALLES NETO, L. L.; MORETTI, A. C. Modelo não-linear para minimizar o número de objetos processados e o setup num problema de corte unidimensional. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 37., 2005. Anais... p: 1679-1688, 2005.

SALLES NETO, L.L. Modelo não linear para minimizar o número de objetos processados e o setup num problema de corte unidimensional. 2005. Tese (Doutorado) -

Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Campinas, 2005.

SAVELSBERGH, M. A branch-and-price algorithm for the generalized assignment problem. Operations Research, v. 45, p. 831–841, 1997.

TANOMARU, J. Motivação, fundamentos e aplicações de algoritmos genéticos. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 2., 1995. Anais... [S.l.]: [s.n.], 1995.

TEGHEM, J., PIRLOT, M. E ANOTONIADIS, C. Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. Journal of Computacional and Applied Mathematics, v. 64, p. 91-102, 1995.

UMETAMI, S.; YAGIURA, M.; IBARAKI, T. One dimensional cutting stock problem to minimize the number of different patterns. European Journal of Operational Research, v. 146, n. 2, p. 388-402, 2003.

WÄSCHER, G.; GAU, T. Heuristics for the integer one-dimensional cutting stock problem: a computational study. OR Spektrum, v. 18, p. 131-144. 1996.

WÄSCHER, G.; HAUBNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. European Journal of Operational Research, v. 183, p. 1109-1130, 2007

WATSON, R. A.; POLLACK, J. B. How symbiosis can guide evolution. In: EUROPEAN CONFERENCE ON ADVANCES IN ARTIFICIAL LIFE, 5., Proceedings… . [S.l.]: Springer, 1999.

WALL, M. A C++ library of genetic algorithm components. Massachusetts Institute of Technology. Mechanical Engineering Department, 1996

VANDERBERCK, F. Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem. Operations Research, v. 48, p. 915-926, 2000.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo