Universidade Federal de Juiz de Fora
Programa de Pós-Graduação em Modelagem Computacional

Hibridização de Algoritmos Genéticos e Sistemas Imunológicos Artificiais para Problemas de Otimização com Restrições em Engenharia

Por

Heder Soares Bernardino

Juiz de Fora, MG - BRASIL
Fevereiro de 2008

Bernardino, Heder Soares

Hibridização de algoritmos genéticos com sistemas imunológicos artificiais para problemas de otimização com restrições em engenharia / Heder Soares Bernardino; orientadores: Prof. Dr. Helio José Corrêa Barbosa e Prof. Dr. Afonso Celso de Castro Lemonge. - 2008.

181 f.

Dissertação (Mestrado em Modelagem Computacional) – Universidade Federal de Juiz de Fora, Juiz de Fora, 2008.

1. Otimização-Matemática. 2. Algoritmo Genético. 3. Meta-heurística. I. Barbosa, Helio José Corrêa, orientador. II. Lemonge, Afonso Celso de Castro, co-orientador. III. Título.

CDU 519.863

HIBRIDIZAÇÃO DE ALGORITMOS GENÉTICOS E SISTEMAS IMUNOLÓGICOS ARTIFICIAIS PARA PROBLEMAS DE OTIMIZAÇÃO COM RESTRIÇÕES EM ENGENHARIA

Heder Soares Bernardino

DISSERTAÇÃO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.SC.) EM MODELAGEM COMPUTACIONAL.

Aprovada por:

Prof. Helio José Corrêa Barbosa, D.Sc.
(Orientador)

Prof. Afonso Celso de Castro Lemonge, D.Sc.
(Co-orientador)

Prof. Wilhelm Passarella Freire, D.Sc.

Prof[a]. Fernanda Maria Pereira Raupp, D.Sc.

JUIZ DE FORA, MG - BRASIL
FEVEREIRO DE 2008

AGRADECIMENTOS

Meus sinceros agradecimentos à minha amada Ana Paula pela paciência, companheirismo e ajuda. Sempre ao meu lado em todos os momentos, mesmo nos mais difíceis.

Um agradecimento especial aos meus orientadores Helio José Corrêa Barbosa e Afonso Celso de Castro Lemonge. Mais que mestres, considero-os grandes amigos. Responsáveis diretos por todo conhecimento empregado nessa obra.

Aos amigos do dia-a-dia pela confiança e cumplicidade.

Agradeço de coração aos meus pais, Hélio e Joana, e ao meu irmão, Hebert, por toda a ajuda e amor.

Meus agradecimentos a todos os professores do Mestrado em Modelagem Computacional da Universidade Federal de Juiz de Fora. Sem esquecer dos alunos que, juntamente comigo, formaram a primeira turma para esse curso.

Por fim, meus agradecimentos a todos aqueles que, direta ou indiretamente, contribuíram para o desenvolvimento desse trabalho.

Resumo da Dissertação apresentada à UFJF como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

HIBRIDIZAÇÃO DE ALGORITMOS GENÉTICOS E SISTEMAS IMUNOLÓGICOS ARTIFICIAIS PARA PROBLEMAS DE OTIMIZAÇÃO COM RESTRIÇÕES EM ENGENHARIA

Heder Soares Bernardino

Fevereiro/2008

Orientador : Helio José Corrêa Barbosa
Co-orientador : Afonso Celso de Castro Lemonge

Neste trabalho são propostas, implementadas e testadas computacionalmente algumas possibilidades de hibridização de algoritmos genéticos e sistemas imunológicos artificiais visando a solução de problemas de otimização com restrições.

Tais problemas aparecem frequentemente na modelagem computacional de problemas científicos e tecnológicos relevantes.

Após breve formulação e caracterização da solução de um problema de otimização com restrições em $\Re^n$ é feita uma introdução à modelagem estrutural e aos problemas de otimização desta área, mostrando sua importância e as dificuldades no desenvolvimento de métodos de resolução aproximados.

São apresentadas então diversas meta-heurísticas bio-inspiradas para a resolução destes problemas de otimização, em especial os algoritmos genéticos (AGs) e os sistemas imunológicos artificiais (SIAs). Em seguida, são propostas algumas possibilidades de hibridização de AGs e SIAs para a otimização restrita.

Experimentos computacionais visando avaliar o desempenho dos procedimentos propostos são realizados considerando vários problemas-teste da literatura e, finalmente, são apresentadas conclusões e propostas para trabalhos futuros.

Abstract of Dissertation presented to UFJF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

HYBRIDIZATION OF GENETIC ALGORITHMS AND ARTIFICIAL IMMUNE SYSTEMS FOR CONSTRAINED OPTIMIZATION PROBLEMS IN ENGINEERING

Heder Soares Bernardino
February/2008

Advisor: Helio José Corrêa Barbosa
Co-advisor: Afonso Celso de Castro Lemonge

In this work some possibilities of hibridization of genetic algorithms and artificial immune systems are proposed, computationally implemented, and tested in order to solve constrained optimization problems.

Such problems arise frequently in the computational modeling of relevant scientific and technological problems.

After a brief formulation and solution characterization for a constrained optimization problem in $\Re^n$, an introduction is made to structural modeling and the optimization problems in this area, showing their relevance and the difficulties arising in the development of approximate solution methods.

Several bio-inspired meta-heuristics for solving such optimization problems are then presented with emphasis on genetic algorithms (GAs) and artificial immune systems (AISs). Some possible GA-SIA hybrids are then proposed.

In order to assess the performance of the proposed procedures, computational experiments are conducted considering several test-problems from the literature and, finally, conclusions as well as future work proposals are presented.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Antes de mostrar os principais processos para se obter uma solução ótima para um dado problema, serão discutidos os conceitos que envolvem a otimização.

Do ponto de vista prático, a otimização trata de problemas e de um conjunto de métodos que tem como objetivo buscar os melhores valores possíveis para solucionar sistemas de interesse para os seres humanos. Esses métodos são uma forma genérica para resolver muitos dos problemas que imagina-se modelar.

Destacamos alguns dos problemas que podem ser resolvidos através de um modelo de otimização:

- um engenheiro procura o melhor projeto possível para uma estrutura ou para um problema mecânico;
- um engenheiro de controle e automação procura o melhor ajuste possível para os controles de um determinado processo industrial;
- um engenheiro de produção busca a melhor configuração possível para encadear as etapas de fabricação de um produto;
- um matemático computacional estuda modelos quantitativos de epidemias, procurando determinar as melhores políticas de vacinação;
- um cientista da computação estuda o desempenho de uma rede de computadores, e tenta estabelecer uma melhor estratégia de tráfego de informação possível, visando maximizar o fluxo global de informação nessa rede;

- um economista procura a melhor carteira de investimentos, visando maximizar o retorno financeiro;
- um pecuarista procura determinar a melhor política de compras e vendas das cabeças de um rebanho de gado.

Apesar de formulados em escopos totalmente distintos, os problemas apresentados anteriormente, e muitos outros, podem ser entendidos como problemas de otimização. Os mesmos devem ser formulados matematicamente e submetidos a algum método ou técnica de otimização.

Um processo de otimização irrestrita busca encontrar um ponto de máximo (ou de mínimo) de um objetivo modelado matematicamente. Essa "função objetivo" representa o problema a ser resolvido. Além disso, o problema a ser otimizado pode estar sujeito a restrições que limitam o espaço de busca. As soluções candidadas viáveis são aquelas que não violam nenhuma das restrições. As restrições podem ser definidas em equações e inequações de restrição, e limitações nos limites das variáveis que compõem o problema.

A otimização dessa função objetivo pode ser feita, em alguns problemas, através da aplicação pura e simples de recursos matemáticos que permitem encontrar analíticamente, sem aproximações, soluções diretas e exatas. Entretanto, a modelagem de algumas funções objetivo pode acarretar dificuldades, tornando impossível a obtenção de uma solução de forma analítica.

Além disso, em alguns problemas de otimização estrutural (um dos focos desse trabalho) nem sempre se tem acesso direto à função que se deseja otimizar. Normalmente, há um processo que, dado um certo conjunto de variáveis, ditas de projeto ou decisão, obtém valores numéricos da função objetivo e das violações das restrições, caso haja alguma.

Uma alternativa seria testar todas as soluções viáveis do problema nessa função objetivo e tomar o melhor resultado. Isso não pode ser feito, pois os problemas que envolvem modelagem computacional são, muitas vezes, muito complexos, ou seja, difíceis de serem computados. Em muitos casos, pode-se levar horas e até mesmo dias para avaliar uma função num dado ponto. É o caso, por exemplo, de um modelo detalhado em aerodinâmica para uma asa de avião. Dessa forma, não é interessante

utilizar métodos que requerem calcular a função objetivo milhares ou milhões de vezes.

Outra forma de resolver problemas de otimização restrita é por meio de métodos numéricos, que podem ser classificados em determinísticos e probabilísticos. Alguns métodos determinísticos são diretos e, a partir de uma solução inicial e uma direção de melhoria da função (gradiente, por exemplo), buscam o melhor resultado para o problema de otimização. Normalmente, encontram uma boa solução aproximada. Todavia, a necessidade de um "chute inicial" pode fazer o método convergir para uma solução ruim ou nem convergir. Dentre os métodos probabilísticos estão os algoritmos genéticos (AG) [5–7], os sistemas imunológicos artificiais (SIA) [8], *Simulated Annealing* (Recozimento Simulado) [9], entre outros. A grande vantagem dos métodos probabilísticos é que, por serem muitas vezes populacionais, não requerem uma solução inicial adequada.

Algumas meta-heurísticas, como os AGs, tem como objetivo resolver esses problemas de otimização com um número reduzido de cálculos da função objetivo do problema. Apesar de serem eficazes na obtenção de boas soluções, os AGs devem ser adaptados para problemas em que há restrições ao espaço de busca. Muitas técnicas foram, e estão sendo, elaboradas com o objetivo de melhorar o desempenho na resolução de problemas de otimização com restrições. Algumas dessas soluções são apresentadas nesse trabalho. Mais especificamente, serão estudados algoritmos que combinem a eficiência comprovada de busca dos algoritmos genéticos com a capacidade de criação de diversidade dos sistemas imunológicos artificiais. Esta técnica para resolução de problemas de otimização, inspirada em trabalhos da literatura [10–15], segue sendo estudada e tem mostrado bons resultados. Pode ser aplicada a problemas de otimização lineares e não-lineares, com variáveis contínuas, discretas ou mistas, nas mais diversas áreas e sem a necessidade de uma boa solução inicial para a convergência do método.

O presente trabalho inicia formalizando a classe de problemas a serem resolvidos, mostrando métodos analíticos clássicos e apresentando uma formulação simplificada para o caso de otimização estrutural. Posteriormente, são estudadas metaheurísticas bio-inspiradas, em especial, os algoritmos genéticos e os sistemas imunológicos artificiais, que são utilizados no conjunto de algoritmos propostos. No Capítulo 4 as técnicas propostas por essa dissertação são apresentadas em suas dife-

rentes versões. A evolução do algoritmo híbrido, desde a inspiração até os métodos atuais, também será mostrada. Em seguida, os resultados obtidos nos experimentos computacionais são apresentados e comparados com os obtidos na literatura. Por fim, são apresentadas as conclusões e propostas para trabalhos futuros.

Capítulo 2

Otimização

A otimização é uma área da matemática que vem sendo aplicada em diversas situações nas mais variadas áreas, como: Engenharia, Economia, Administração, Química, Física, Biologia, etc. O presente trabalho foca a otimização em engenharia.

A origem dos problemas de otimização não é clara. [16] sugere que o estudo desses problemas começou há 25 séculos. Segundo [17], um dos primeiros relatos sobre a otimização pode ser encontrada na Eneida, de Virgílio, poeta romano que viveu de 70 a.C. a 19 d.C..

Diz a lenda que a cidade de Cartago foi fundada por Dido, uma princesa fenícia que teve que abandonar Tiro, sua terra natal. Procurando um lugar para ficar, conseguiu convencer um chefe africano que lhe concedeu a terra que conseguisse delimitar com o couro de um touro. Primeiro, o couro foi cortado em tiras bem finas. Depois, atou os pedaços para poder traçar a área que iria possuir. Para terminar, traçou um semi-círculo no chão, à beira do mar Mediterrâneo. Esta era a máxima área costeira que ela poderia envolver. Naquele local foi construída a famosa cidade de Cartago. Assim, Dido teria resolvido o primeiro problema de otimização de que se tem notícia, mesmo sendo apenas uma história mitológica.

Apesar do estudo de problemas de otimização existir há muitos anos, os primeiros métodos matemáticos de caráter geral foram criados apenas por volta do século XVII. Pierre de Fermat representava curvas por meio de relações algébricas entre coordenadas. Assim nascia a geometria cartesiana, aproximadamente em 1630. O nome “cartesiana” é dada em honra a Descartes, que a divulgou em 1637. Através

dessas representações, Fermat iniciou sua investigação no traçado de tangentes e, em decorrência disso, de pontos extremos de curvas. Através disso, criou um método para encontrar máximos e mínimos de uma função.

O procedimento era semelhante ao que segue [17]:

- De posse de uma expressão algébrica de uma curva (por exemplo, $f(x) = ax^2 + bx + c$), encontra-se o valor de $f(x)$ para um ponto próximo ($f(x + \varepsilon)$;
- Expande-se a expressão, separando os termos no deslocamento. Ou seja, para o exemplo, $a\varepsilon^2 + 2ax\varepsilon + b\varepsilon$;
- Como o deslocamento considerado deve ser muito pequeno, então $\varepsilon^2 \rightarrow 0$. Logo, tem-se $2ax\varepsilon + b\varepsilon = \varepsilon(2ax + b)$;
- Investigando onde a expressão se anula, e como $\varepsilon \neq 0$, então chega-se a $2ax + b = 0 \Rightarrow x = -b/2a$;
- Neste ponto, a tangente à curva é horizontal, assim este pode ser um ponto extremo.

O processo seguido por Fermat é muito semelhante ao que se tem hoje, sem o conceito de função, derivada ou diferencial. Mais de um século depois, Laplace dá créditos a Fermat pela investigação do cálculo diferencial [17].

Este capítulo segue com a formulação dos problemas de otimização, definições matemáticas para encontrar extremos de funções de várias variáveis e conceitos de otimização estrutural, que é um dos focos de aplicação desse trabalho. Os conceitos que se seguem, com exceção da Seção 2.5, foram baseados nas referências [1, 16, 18], onde podem ser obtidas informações mais detalhadas sobre o assunto.

2.1 Formulação

Para cada problema, um especialista na área deve traduzir a realidade da situação para um modelo matemático. O conjunto de informações relevantes ao problema varia em cada um dos casos. Consideraremos, inicialmente, o caso em que as

variáveis envolvidas são contínuas. Assim, depois de modelados matematicamente, pode-se escrever os problemas de otimização contínua como

Otimizar

$$f(x) \tag{2.1}$$

Sujeito a

$$g(x) \leq 0 \tag{2.2}$$

$$h(x) = 0 \tag{2.3}$$

$$x_{i_{Min}} \leq x_i \leq x_{i_{Max}} \tag{2.4}$$

onde, otimizar significa encontrar uma solução para $f(x)$ que faça com que a função modelada seja máxima ou mínima, dependendo do objetivo do problema, $f : \Re^n \rightarrow \Re, g : \Re^n \rightarrow \Re^a, h : \Re^n \rightarrow \Re^b$, a e b são o número de inequações e equações de restrição, respectivamente.

Uma observação a ser considerada é que as restrições (2.2) e (2.3) não são obrigatórias na modelagem, ou seja, a estrutura apresentada anteriormente não obriga que o problema tenha algum tipo de restrição (sejam elas equações ou inequações), apesar disso ser comum.

O vetor x representa o conjunto de variáveis conhecidas como de decisão ou de projeto. No decorrer do processo, esse vetor pode assumir qualquer valor válido, isto é, que satisfaça as restrições do problema. A melhor solução é encontrada quando se consegue um x válido que otimize a função proposta. Como x é um vetor contendo os valores das variáveis do problema, pode ser descrito da seguinte forma:

$$x = (x_1, x_2, \ldots, x_n) \tag{2.5}$$

onde n é o número de variáveis consideradas no problema e x_i o valor de cada uma das variáveis, com $i = 1, 2, \ldots, n$. Ou seja, x é um vetor de variáveis do conjunto dos números reais ($x \in \Re^n$). Vamos agora caracterizar o conjunto de soluções adminissíveis ou válidas do problema de otimização definido por $S = \{x \in \Re^n | g_j(x) \leq 0, h_k(x) = 0, x_{i_{Min}} \leq x_i \leq x_{i_{Max}}\}$, com $j = 1, \ldots, a$, $k = 1, \ldots, b$ e $i = 1, \ldots, n$.

O próximo ponto a ser considerado é a função objetivo $f(x)$. Seu valor é que deve ser otimizado (maximizado ou minimizado). Por exemplo, minimizar o custo em

uma construção, nesse caso a função objetivo é modelada com os custos associados às variáveis de projeto. Em linguagem matemática, pode-se escrever da seguinte forma:

$$f : \Re^n \rightarrow \Re \tag{2.6}$$

onde n é o número de variáveis a serem consideradas no problema.

Chega-se, então, às seguintes definições para as soluções de um problema de minimização. Daqui em diante, considera-se o problema de otimização como sendo apenas de minimização. A extensão das definições para um problema de maximização é simples, visto que minimizar uma função f equivale a maximizar $-f$.

Definição 2.1 *Um ponto $x^* \in S$ é um minimizador local de f em S se e somente se existe $\varepsilon > 0$ tal que $f(x) \geq f(x^*)$ para todo $x \in S$ tal que $\|x - x^*\| < \varepsilon$. Se $f(x) > f(x^*)$ para todo $x \in S$ tal que $x \neq x^*$ e $\|x - x^*\| < \varepsilon$, diz-se que se trata de um minimizador local estrito em S.*

Definição 2.2 *Um ponto $x^* \in S$ é um minimizador global de f em S se e somente se $f(x) \geq f(x^*)$ para todo $x \in S$. Se $f(x) > f(x^*)$ para todo $x \in S$ tal que $x \neq x^*$, diz-se que se trata de um minimizador global estrito em S.*

2.2 Otimização sem Restrições

Nesta seção serão brevemente apresentadas as idéias básicas que permitem a caracterização dos extremos de uma função f que estejam no interior do conjunto S.

Proposição 2.1 (Condições necessárias de primeira ordem)
Seja $f : S \subset \Re^n \rightarrow \Re$, $f \in C^1$. Se $x^* \in S$ é um minimizador local de f em S, então $\nabla f(x^*) = 0$.

Definição 2.3 *Chama-se ponto crítico o ponto que satisfaz a condição necessária de primeira ordem.*

É importante observar que a recíproca da Proposição 2.1 é falsa. Ou seja, tendo um ponto a tal que $\nabla f(a) = 0$ não significa que a seja extremo local de f. Como

exemplo, a Função 2.7. Nela, tem-se $\nabla f(a) = 0$ em $a = (0, 0)$. Todavia, o ponto a é conhecido como um Ponto de Sela, ou seja, um ponto crítico da função que não é nem máximo nem mínimo local.

$$f(x) = x_1^2 - x_2^2 \tag{2.7}$$

Proposição 2.2 (Condições necessárias de segunda ordem)
Seja $f : S \subset \Re^n \to \Re$, $f \in C^2$. Se $x^* \in S$ é um minimizador local de f em S, então: i) $\nabla f(x^*) = 0$; ii) $\nabla^2 f(x^*)$ é semidefinida positiva.

Voltando ao exemplo da Função 2.7, vê-se que sua matriz Hessiana (2.8) não é semidefinida positiva. Assim, o ponto crítico $(0, 0)$ não pode ser um ponto de mínimo.

$$\nabla^2 f(x^*) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \tag{2.8}$$

Proposição 2.3 (Condições suficientes de segunda ordem)
Seja $f : S \subset \Re^n \to \Re$, $f \in C^2$. Para $x^* \in S$, se $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*) > 0$, então x^* é um minimizador local estrito de f em $\Re^n$.

É importante lembrar que trata-se de uma condição suficiente e que exige mais regularidade da função objetivo.

As condições de segunda ordem (Proposições 2.2 e 2.3) estabelecem critérios para verificar se um ponto crítico é um ponto de mínimo local. Nada é dito sobre sua globalidade. Para isso precisa-se verificar se f pertence a uma classe específica de funções para as quais pode-se garantir a globalidade de pontos críticos. As funções convexas, como serão vistas na seção seguinte, podem garantir que um mínimo local é, também, global.

Definição 2.4 *Mínimo é o valor da função objetivo no ponto de mínimo ou minimizador.*

2.3 Convexidade

Definição 2.5 *Um subconjunto $S \subset \Re^n$ é convexo se e somente se para todo $x, y \in S$ e para todo $\lambda \in [0,1]$, se verifica que $\lambda x + (1-\lambda)y \in S$.*

Em outras palavras, a Definição 2.5 quer dizer que, para quaisquer dois pontos no subconjunto S, todos os pontos no segmento de reta que une esses pontos também devem pertencer a S para que o subconjunto seja dito convexo. A Figura 2.1 mostra exemplos de um subconjunto convexo e outro (não convexo).

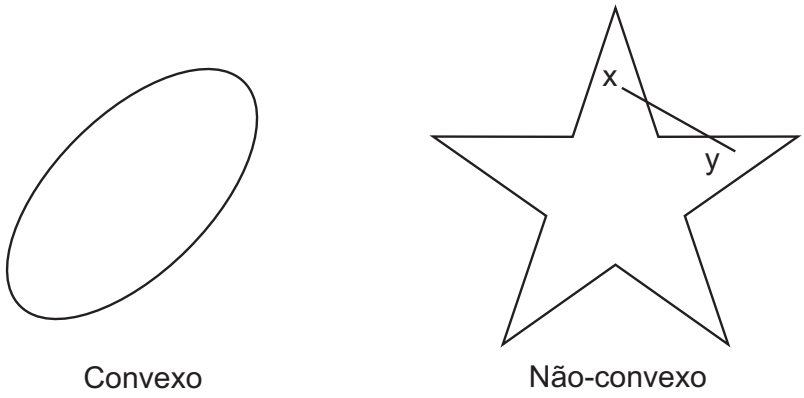

Figura 2.1: Exemplos de subconjunto convexo e não-convexo.

Definição 2.6 *Uma função $f : S \subset \Re^n \to \Re$ é dita convexa em S se e somente se*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{2.9}$$

para todo $x, y \in S$, $\lambda \in [0,1]$.

f é dita estritamente convexa se para todo $x, y \in S$ e $\lambda \in (0,1)$, com $x \neq y$ e se verifica

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y) \tag{2.10}$$

A Figura 2.2 exemplifica a Definição 2.6 com três funções, sendo uma delas estritamente convexa, outra convexa e uma terceira não-convexa.

Uma caracterização útil de uma função convexa é dada a seguir.

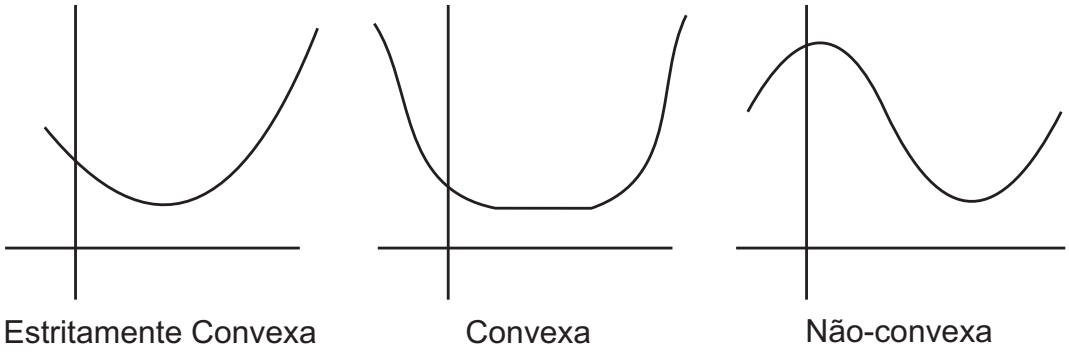

Figura 2.2: Exemplos de função estritamente convexa, convexa e não-convexa.

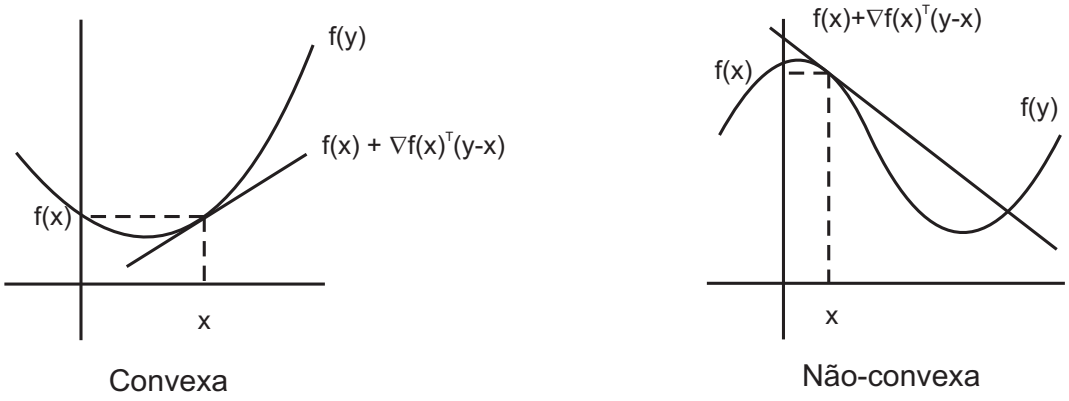

Figura 2.3: Exemplos de função convexa e não-convexa em funções de classe C^1.

Proposição 2.4 *Seja $f : S \subset \Re^n \to \Re$, $f \in C^1$, então f será convexa em S convexo se e somente se para todo $x, y \in S$ se verifica que*

$$f(y) \geq f(x) + \nabla^t f(x)(y - x)$$

A Proposição 2.4 é ilustrada na Figura 2.3.

Proposição 2.5 *Seja $f : \Re^n \to \Re$ uma função convexa definida em S convexo, então:*

- *O conjunto $\Gamma \subset S$ onde f toma seu valor mínimo é convexo;*
- *Qualquer minimizador local de f é também minimizador global da função.*

Proposição 2.6 *Seja $f : \Re^n \to \Re$, $f \in C^1$ convexa definida em S convexo. Se existe $x^* \in S$ tal que se verifica*

$$\nabla^t f(x^*)(y - x^*) \geq 0 \qquad \forall y \in S \tag{2.11}$$

então x^* é um minimizador global de f em S.

Através da Proposição 2.6, observamos que um problema de otimização pode ser interpretado como um problema variacional. Essa nova abordagem permite desenvolver ou usar algoritmos bem diferentes dos utilizados em otimização.

2.4 Otimização com Restrições

Nesta seção serão vistos casos em que o espaço de busca está limitado por restrições. O problema é formulado como apresentado na Seção 2.1. Segundo [1], a dificuldade dos problemas de minimização com restrições depende fortemente da complexidade destas.

2.4.1 Restrições de Igualdade

Para um problema de minimização com restrições de igualdade, a seguinte forma será considerada:

$$\text{minimizar } f(x) \tag{2.12}$$

$$\text{sujeito a } h(x) = 0 \tag{2.13}$$

onde $f, h \in C^1, f : \Re^n \to \Re, h : \Re^n \to \Re^m$ e $m < n$. A função h pode ser não-linear.

Definimos a região factível do problema (2.12)-(2.13) por $S = \{x \in \Re^n | h(x) = 0\}$.

Teorema 2.1 Teorema dos Multiplicadores de Lagrange.
Sejam $f, h_1, \ldots, h_m$ funções de classe C^1 de n variáveis e seja x^* um extremo local de f no conjunto admissível S. Suponha que x^* satisfaça a seguinte condição de regularidade: o posto da matriz jacobiana

$$\begin{bmatrix} \nabla h_1(x^*) \\ \vdots \\ \nabla h_m(x^*) \end{bmatrix} = \begin{bmatrix} \frac{\partial h_1(x^*)}{\partial x_1} & \cdots & \frac{\partial h_1(x^*)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m(x^*)}{\partial x_1} & \cdots & \frac{\partial h_m(x^*)}{\partial x_n} \end{bmatrix} \in \Re^{m \times n}$$

é igual a m, ou seja, o número de restrições. Então, existem números reais $\lambda_1^*, \ldots, \lambda_m^*$ *(conhecidos como multiplicadores de Lagrange) tais que*

$$\begin{cases} \nabla f(x^*) &= \lambda_1^* \nabla h_1(x^*) + \cdots + \lambda_m^* \nabla h_m(x^*) = \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*), \\ h_1(x^*) &= 0, \\ &\vdots \\ h_m(x^*) &= 0 \end{cases}$$

isto é, o vetor gradiente $\nabla f(x^*)$ *é uma combinação linear de* $\nabla h_1(x^*), \ldots, \nabla h_m(x^*)$. *O sistema acima reune as condições de primeira ordem para o problema de otimização com restrições de igualdade. De forma equivalente, o ponto*

$$(x^*, \lambda^*) = (x_1^*, \ldots, x_n^*, \lambda_1^*, \ldots, \lambda_m^*)$$

é ponto crítico do lagrangeano definido por

$$L(x, \lambda) = f(x) - \lambda_1 h_1(x) - \cdots - \lambda_m h_m(x)$$

Teorema 2.2 Condições de segunda ordem
Sejam x^* *um ponto regular, o plano tangente* $T = \{y \in \Re^n | J_h(x^*)y = 0\}$ *e supondo* $f, h \in C^2$. *Se* x^k *é um minimizador local de* f, *então existe* $\lambda^* \in \Re^m$ *tal que*

$$\nabla f(x^*) + \sum_{j=1}^{m} \lambda_j^* \nabla h_j(x^*) = 0$$

e

$$y^T \nabla_x^2 L(x^*, \lambda^*) y \geq 0 \quad \forall y \in T \tag{2.14}$$

Para o Teorema 2.2, se a desigualdade é satisfeita estritamente em (2.14), então x^* passa a ser minimizador local estrito de f.

$$y \nabla_x^2 L(x^*, \lambda^*) y > 0 \quad \forall y \in T \tag{2.15}$$

2.4.2 Restrições de Desigualdade

Na Seção 2.4.1 foram estudados problemas de otimização nos quais o espaço de busca factível é formado pela interseção de várias superfícies. Para encontrar os

pontos críticos naqueles problemas é necessário fazê-lo no Lagrangeano correspondente, desde que a condição de regularidade seja satisfeita ($\nabla h(x^*) \neq 0$, onde x^* é ponto crítico de f). Todavia, a grande maioria dos problemas de otimização, em especial os de engenharia, são constituídos por restrições de desigualdades. Assim, o espaço de busca factível fica sendo $S = \{x \in \Re^n | g_1(x) \leq 0, \ldots, g_k(x) \leq 0\}$. As restrições podem estar ativas (quando $g_k(x^*) = 0$) ou não (se $g_k(x^*) < 0$).

Teorema 2.3 Condições de primeira ordem.
Sejam $f, g_1, \ldots, g_k$ funções de classe C^1 de n variáveis definidas em um aberto de $\Re^n$ e seja $x^* \in \Re^n$ um mínimo local de f no conjunto admissível S. Caso algumas restrições sejam ativas em x^*, estas serão renomeadas de forma que sejam as l primeiras: $g_1, \ldots, g_l$. Suponha que x^* satisfaça a seguinte condição de regularidade: o posto da matriz jacobiana

$$\begin{bmatrix} \nabla g_1(x^*) \\ \vdots \\ \nabla g_l(x^*) \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1(x^*)}{\partial x_1} & \cdots & \frac{\partial g_1(x^*)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_l(x^*)}{\partial x_1} & \cdots & \frac{\partial g_l(x^*)}{\partial x_n} \end{bmatrix} \in \Re^{l \times n}$$

é igual a l, ou seja, igual ao número de restrições ativas no ponto x^*. Então, existem multiplicadores $\mu_1^*, \ldots, \mu_k^*$ tais que

$$\begin{cases} \nabla f(x^*) = \mu_1^* \nabla g_1(x^*) + \cdots + \mu_k^* \nabla g_k(x^*) = \sum_{j=1}^{k} \mu_j^* \nabla g_j(x^*), \\ \mu_1 \cdot g_1(x^*) = 0, \\ \vdots \\ \mu_k \cdot g_k(x^*) = 0, \\ \mu_1^* \geq 0, \\ \vdots \\ \mu_k^* \geq 0, \\ g_1(x^*) \leq 0, \\ \vdots \\ g_k(x^*) \leq 0, \end{cases}$$

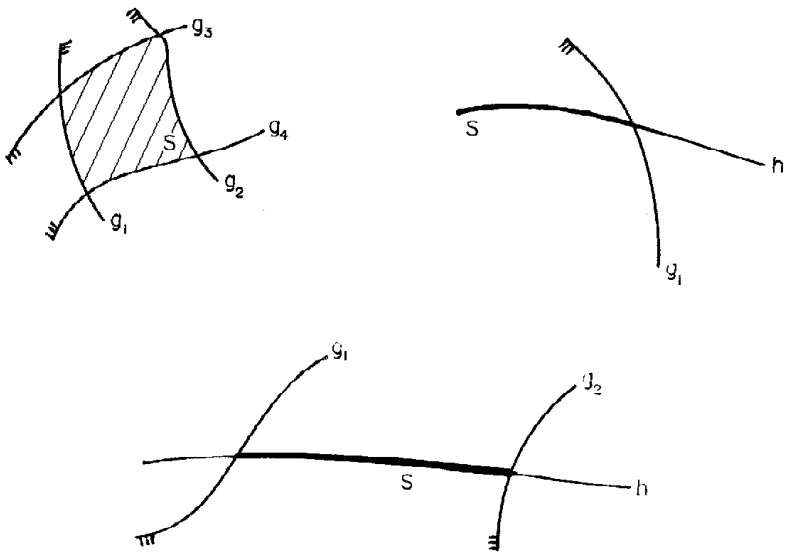

Figura 2.4: Possibilidades para um problema de otimização com restrições [1].

2.4.3 Restrições Mistas

Combinando o que foi visto nas Seções 2.4.1 e 2.4.2, pode-se então resolver o problema proposto na Seção 2.1. A Figura 2.4 ilustra algumas possibilidades para o problema de otimização com restrições mistas, para $n = 2$. Com isso, tem-se que o espaço de busca factível passa a ser $S = \{x \in \Re^n | h(x) = 0 \text{ e } g(x) \leq 0\}$ e pode-se obter um teorema geral para este caso.

Teorema 2.4 Teorema de Karush-Kuhn-Tucker
Sejam $f, h_1, \ldots, h_m, g_1, \ldots, g_k$ funções de classe C^1 de n variáveis definidas em um aberto de $\Re^n$ e seja $x^* \in \Re^n$ um mínimo local de f no conjunto admissível S. Caso algumas restrições de desigualdade estejam ativas em x^*, estas serão renomeadas de forma que sejam as l primeiras: $g_1, \ldots, g_l$. Suponha que x^* satisfaça a seguinte

condição de regularidade: o posto da matriz jacobiana estendida

$$\begin{bmatrix} \nabla h_1(x^*) \\ \vdots \\ \nabla h_m(x^*) \\ \nabla g_1(x^*) \\ \vdots \\ \nabla g_l(x^*) \end{bmatrix}_{(m+l)\times n} = \begin{bmatrix} \frac{\partial h_1(x^*)}{\partial x_1} & \cdots & \frac{\partial h_1(x^*)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m(x^*)}{\partial x_1} & \cdots & \frac{\partial h_m(x^*)}{\partial x_n} \\ \frac{\partial g_1(x^*)}{\partial x_1} & \cdots & \frac{\partial g_1(x^*)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_l(x^*)}{\partial x_1} & \cdots & \frac{\partial g_l(x^*)}{\partial x_n} \end{bmatrix}_{(m+l)\times n}$$

é igual a $m + l$, ou seja, é igual ao número de restrições ativas no ponto x^*. Considerando o lagrangeano associado ao problema de otimização com restrições de igualdade e desigualdade

$$L(x, \lambda, \mu) = f(x) - \lambda_1 h_1(x) - \cdots - \lambda_m h_m(x) - \mu_1 g_1(x) - \cdots - \mu_k g_k(x)$$

com $\lambda \in \Re^m$ e $\mu \in \Re^k$. Então, existem multiplicadores $\lambda^* \in \Re^m$ e $\mu^* \in \Re^k$ tais que

$$\begin{cases} \nabla f(x^*) &= \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) + \sum_{j=1}^{k} \mu_j^* \nabla g_j(x^*), \\ h_1(x^*) &= 0, \\ &\vdots \\ h_m(x^*) &= 0 \\ \mu_1^* \cdot g_1(x^*) &= 0, \\ &\vdots \\ \mu_k^* \cdot g_k(x^*) &= 0, \\ \mu_1^* &\geq 0, \\ &\vdots \\ \mu_k^* &\geq 0, \\ g_1(x^*) &\leq 0, \\ &\vdots \\ g_k(x^*) &\leq 0, \end{cases}$$

O Teorema 2.4 é também denominado como condições de primeira ordem.

Como pode ser visto, a busca analítica por pontos extremos em problemas diferenciáveis é uma tarefa muito difícil. Primeiramente, a função a ser otimizada f e

suas restrições devem ser explícitas e, como será visto na Seção 2.5, isso não é sempre possível. Um outro problema seria classificar esses pontos entre maximizadores, minimizadores ou de sela. Para que essa determinação seja feita há a necessidade de se limitar ainda mais as as funções que podem ser trabalhadas (classe C^2). Finalizando, para se saber quais desses pontos são maximizadores ou minimizadores globais (normalmente, objetivo do problema), é necessário garantir a convexidade do lagrangeano.

2.5 Otimização Estrutural

Um dos fortes interesses nas áreas da engenharia mecânica e estrutural, por exemplo, é a identficação de mecanismos capazes de encontrar soluções econômicas e seguras para os projetos desejados. Dessa forma, em um projeto mecânico busca-se uma utilização adequada de materiais a serem dispostos em seus vários elementos ou componentes capazes de propocionar um perfeito funcionamento ao serem submetidos às solicitações impostas.

Na concepção de um projeto mecânico/estrutural são necessárias definições sobre topologia, forma e dimensões da estrutura (tamanho das peças estruturais), tipos de materiais e assim por diante. Além de tudo, é preciso escolher de forma criteriosa uma configuração estrutural que seja capaz de explorar todos os recursos disponíveis fornecidos por todos esses quesitos inerentes ao projeto. Pode-se dizer que o foco maior concentra-se, então, na determinação de proporções adequadas de uma variedade de tipos de componentes estruturais, que concebem um projeto mecânico/estrutural, que sejam capazes de proporcionar um ótimo desempenho global quando submetido às solicitações dos carregamentos aplicados.

As soluções ótimas almejadas nem sempre estão disponíveis de forma imediata ou, às vezes são impossíveis de serem detectadas. Não raramente, para um dado projeto, a determinação de uma configuração eficiente é inerente a um processo baseado na "tentativa, erro e acerto", que nem sempre garante a solução ótima esperada.

A otimização estrutural pode ser vista como uma fusão de áreas da engenharia e matemática capaz de adicionar ingredientes ao projeto mecânico/estrutural que vão além da experiência dos projetistas adquiridas ao longo dos tempos que visa a

busca das soluções ótimas desejadas.

Como nos problemas de otimização usuais, em um problema de otimização estrutural o interesse está em modelar matematicamente ou mesmo experimentalmente uma ou mais funções objetivo que sejam capazes de representar os aspectos almejados na busca da solução ótima. Na sua grande maioria os problemas de otimização estrutural apresentam restrições, que podem ser mais ou menos complexas.

É muito comum também que uma função objetivo não represente o objetivo a ser atingido, e sim uma série de funções objetivo envolvidas no problema, e assim tem-se uma otimização multiobjetivo.

Entre várias possibilidades de objetivos, alguns são mais freqüentes: escolha do material (custo); a melhor disposição dos componentes estruturais; definição dos tipos de elementos a serem usados e que estejam disponíveis no mercado (perfis metálicos para uma treliça); aspectos ligados à topologia do projeto; durabilidade; confiabilidade; funcionabilidade; eficiência; capacidade e tempo de execução; etc [19].

Um exemplo muito simples que justifica a otimização de um projeto estrutural é a busca de soluções ótimas para treliças metálicas como as torres de transmissões que devem ser fabricadas centenas de vezes para cobrir as distâncias esperadas nos fornecimentos de energia. Um exemplo dessas torres é mostrado na Figura 2.5. A economia, por menor que seja, no projeto de uma unidade dessas torres será significativa quando multiplicada por essas possíveis centenas de unidades.

Também, na indústria aeronáutica, a minimização de peso dos componentes de uma aeronave é extremamente atraente já que o menor peso possível da aeronave é o quesito mais cobiçado.

Dentre um expressivo número de problemas de otimização estrutural citam-se os seguintes [19]: otimização da forma da asa de uma aeronave, otimização da seção transversal de uma viga de uma ponte, otimização da espessura de uma placa ou de um vaso de pressão, otimização de estruturas tridimensionais de edificações, localização ótima da posição de pilares na estrutura de uma edificação, maximização das freqüências de vibração da estrutura de uma ponte e assim por diante. Algumas aplicações reais e acadêmicas estão disponíveis em [20].

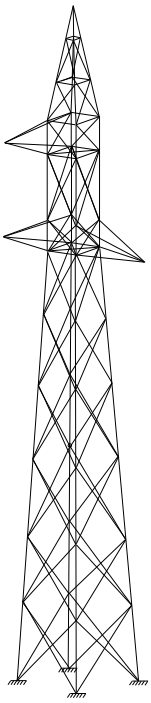

Figura 2.5: Torre de transmissão contendo 160 barras.

A grande maioria dos problemas de otimização em engenharia estão sujeitos a restrições e essas podem ser mais ou menos complexas e conseqüentemente facilitando ou não a busca da melhor solução. Usualmente, estas restrições referem-se às tensões internas nos elementos da estrutura, deslocamentos, freqüências de vibração, cargas críticas de flambagem, consumo máximo de material, etc. Também, é muito comum as restrições não serem escritas explicitamente como funções das variáveis de projeto envolvidas na função objetivo e isso pode acarretar dificuldades na avaliação de uma solução candidata quanto a sua factibilidade.

2.5.1 Um exemplo simples de otimização estrutural

A estrutura mostrada na Figura 2.6 é uma representação esquemática da estrutura de uma treliça formada por três barras de eixos retos. Em geral, estas estruturas são metálicas e usadas largamente em coberturas, hangares, edifícios, etc. Nas extremidades de cada barra são definidos nós sendo, neste caso, os de número 1, 2, 3 e 4. Sobre o nó 2 é aplicado um carregamento externo constituído por duas cargas

concentradas de intensidade P_1 e P_2 nas direções dos eixos x e y, respectivamente.

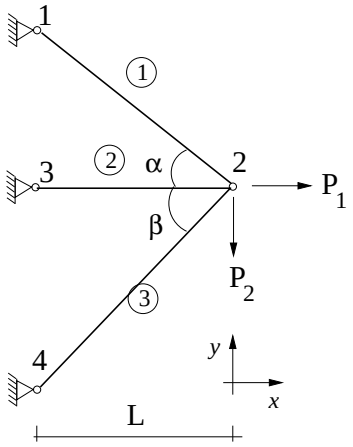

Figura 2.6: Treliça formada por três barras.

Pretende-se definir para esta treliça a formulação de um problema de otimização com o objetivo de minimizar o volume da mesma. Isto visa a diminuição do custo do material a ser empregado na sua execução, satisfazendo as condições de segurança com o máximo de economia.

Designando-se por A_i, $i = 1, \ldots, 3$, as áreas das barras das seções transversais, definidas como variáveis de projeto, o volume $V(A_i)$ a ser minimizado é escrito como:

$$V(A_i) = A_1 L_1 + A_2 L_2 + A_3 L_3 \tag{2.16}$$

ou

$$V(A_i) = \sum_{i=1}^{3} A_i L_i \tag{2.17}$$

onde, L_i, $i = 1, 2, 3$, são os comprimentos das barras.

Claramente, devem ser introduzidas restrições ao problema de otimização, a fim de evitar a solução inconcebível que acarretaria em áreas com valores nulos e um volume $V(A_i) = 0$.

Como visto anteriormente, a grande maioria dos problemas de otimização em engenharia estão sujeitos a restrições e essas podem ser mais ou menos complexas e conseqüentemente facilitando ou não a busca da melhor solução. Usualmente, estas restrições referem-se às tensões internas nos elementos da estrutura, deslocamentos,

freqüencias de vibração, cargas críticas de flambagem, consumo máximo de material, etc. Também, é muito comum as restrições não serem escritas explicitamente como funções das variáveis de projeto envolvidas na função objetivo e isso pode acarretar dificuldades na avaliação de uma solução candidata quanto à sua factibilidade.

Determinação das restrições

Em virtude da aplicação das cargas P_1 e P_2 sobre o nó 2 da estrutura, o mesmo sofrerá deslocamentos (translações) nas direções dos eixos x e y. A Figura 2.7 ilustra uma provável configuração geométrica da treliça após a aplicação do carregamento onde L^* representa o valor modificado da dimensão L da Treliça na configuração indeformada mostrada na Figura 2.6. As barras 1, 2 e 3 sofrerão deformações e estarão submetidas a tensões internas. Estas tensões internas podem ser representadas como sendo oriundas de esforços normais internos, que têm direções normais às seções transversais das barras. O valor da tensão normal em cada barra é a razão entre o esforço normal e a área da seção tranversal da barra, e pode ser calculada após a determinação dos deslocamentos ocorridos no nó 2 da estrutura.

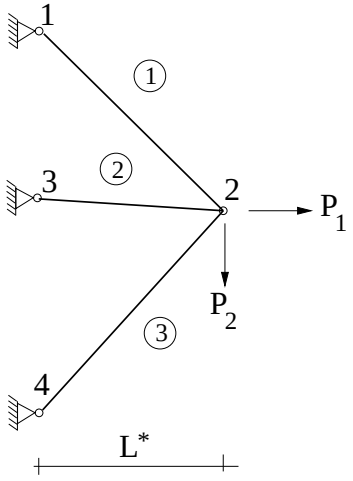

Figura 2.7: Possível configuração geométrica da treliça de três barras após aplicação de cargas.

Definindo-se $\sigma_i, i = 1, 2, 3$, como sendo as tensões normais nas barras e u_{2x} e u_{2y}

como sendo os deslocamentos do nó 2, nas direções x e y, respectivamente, escreve-se:

$$\sigma_i \leq \overline{\sigma}_i \qquad i = 1, 2, 3$$
$$u_{2x}, u_{2y} \leq \overline{u}_2$$

onde, $\overline{\sigma}_i$ e $\overline{u}_2$ são os limites máximos para as tensões normais nas barras e para os deslocamentos do nó 2.

O problema de otimização pode ser escrito como

$$\text{minimizar} \quad V(A_i) = \sum_{i=1}^{3} A_i L_i,$$

submetido a

$$\sigma_i \leq \overline{\sigma}_i \qquad i = 1, 2, 3$$
$$u_{2x}, u_{2y} \leq \overline{u}_2$$

Restrições adicionais são incorporadas no sentido de limitar o espaço de busca das variáveis de projeto e neste caso, tem-se:

$$A_i^{min} \leq A_i \leq A_i^{max} \qquad i = 1, 2, 3$$

Determinação dos deslocamentos e tensões

Para uma dada solução candidata, a partir do conjunto de áreas A_i fornecidas é necessário determinar os deslocamentos e tensões. Essa determinação passa pela resolução de um sistema de equações lineares envolvendo características elásticas e geométricas da estrutura além do carregamento aplicado sobre a mesma. Este sistema de equações representa a equação de equilíbrio da estrutura em termos dos deslocamentos sendo escrita como:

$$[K]\{u\} = \{F\}$$

onde $[K]$ é uma matriz quadrada de dimensão $n \times n$ chamada matriz de rigidez da estrutura, $\{F\}$ é o vetor de cargas com componentes calculadas em função do carregamento aplicado sobre a estrutura e $\{u\}$ é o vetor de deslocamentos a serem determinados após a resolução do sistema de equações. De posse dos deslocamentos é possível então a determinação das esforços e tensões normais em cada ponto de

interesse ao longo das barras da treliça. Esse sistema de equações para o exemplo ilustrado tem dimensão 2×2 sendo escrito como:

$$\begin{bmatrix} k_{11} & k_{12} \\ \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} u_{2x} \\ \\ u_{2y} \end{Bmatrix} = \begin{Bmatrix} F_{2x} \\ \\ F_{2y} \end{Bmatrix}$$

onde k_{ij} são os coeficientes de rigidez que dependem das características elásticas e geométricas das barras que compõem a estrutura. As componentes F_{2x} e F_{2y} são, para este exemplo, os valores dos carregamentos P_1 e P_2 aplicados diretamente sobre o nó 2 nas direções x e y, respectivamente. As incógnitas a serem determinadas são os deslocamentos u_{2x} e u_{2y}. Os coeficientes de rigidez para a treliça da Figura 2.6 são:

$$\begin{aligned} k_{11} &= \frac{E_1 A_1}{L_1}\text{cos}\alpha + \frac{E_2 A_2}{L_2} + \frac{E_3 A_3}{L_3}\text{cos}\beta \\ k_{12} = k_{21} &= -\frac{E_1 A_1}{L_1}\text{sen}\alpha\text{cos}\alpha + \frac{E_3 A_3}{L_3}\text{sen}\beta\text{cos}\beta \\ k_{11} &= \frac{E_1 A_1}{L_1}\text{sen}\alpha + \frac{E_2 A_2}{L_2} + \frac{E_3 A_3}{L_3}\text{sen}\beta \end{aligned}$$

onde, E_1, E_2 e E_3 são os módulos de elasticidade dos materiais de cada uma das três barras, L_1, L_2 e L_3 são os comprimentos de cada uma delas, α e β são os ângulos entre as barras (Figura 2.6). Os valores de $L_i, i = 1, 2, 3$, e $A_i, i = 1, 2, 3$ (variáveis de projeto), definem as características geométricas da treliça, enquanto que os valores de $E_i, i = 1, 2, 3$, as características elásticas. As componentes do vetor de cargas são dadas por:

$$\begin{aligned} F_{2x} &= P_1 \\ F_{2y} &= P_2 \end{aligned}$$

É possível notar, então, que para diferentes conjuntos de variáveis de projeto A_i obtém-se diferentes valores de deslocamentos $\{u\}$ sendo este vetor uma função implícita das variáveis A_i, não sendo possível escrever uma relação explícita entre essas grandezas. Para determinar $\{u\}$ precisa-se resolver um sistema de equações lineares, cuja matriz ($[K]$) depende das variáveis A_i.

De posse dos deslocamentos é possível determinar os esforços normais e conseqüentemente as tensões normais em cada uma das barras [21] e verificar se a solução

candidata em questão é factível ou não. A determinação dos esforços normais envolve operações matriciais que consideram características elásticas e geométricas de cada barra além dos deslocamentos dos nós de suas extremidades. Estas operações serão omitidas deste texto mas podem ser obtidas detalhadamente em [21].

A implementação computacional referente à análise estrutural envolvida na avaliação de soluções candidatas para a otimização se deu pelo acoplamento de um código em linguagem FORTRAN obtido em [22]. Este módulo em FORTRAN é um programa de elementos finitos com uma biblioteca para a análise de estruturas reticuladas do tipo treliça espacial que fornece os deslocamentos dos nós, os esforços e tensões normais nas barras das estruturas analisadas.

Capítulo 3

Meta-heurísticas Bio-inspiradas

Muitos problemas de otimização podem ser solucionados através de métodos de buscas determinísticas. Entretanto, esses métodos, na maioria das vezes, dependem de grande conhecimento do problema. As heurísticas também dependem de conhecimento do problema a ser resolvido. Contudo, elas podem modificar seus parâmetros internos durante a execução. Soluções possíveis podem ser combinadas a fim de obter um bom resultado. Diferente dos métodos determinísticos, as heurísticas não são formalmente provadas e nem sempre chegam ao melhor resultado, mas freqüentemente a uma boa solução. São métodos com base empírica, embora a exploração do espaço de busca seja feita de forma algorítmica. Essas técnicas, normalmente, são usadas em problemas nos quais a solução algorítmica disponível é de complexidade exponencial.

Heurística tem origem na palavra que vem da grega *heuristike* (heuristiké) que significa descobrir e que deu origem também ao termo eureca. A utilização da palavra *eureka* é atribuída a Arquimedes no conhecido episódio da descoberta de como medir o volume de um objeto irregular utilizando água. Arquimedes teria feito essa descoberta durante o banho e ficou tão excitado que saiu nú pelas ruas gritando: *Eureka! Eureka!* (Encontrei! Encontrei!).

As heurísticas podem ser classificadas e agrupadas como:

- Heurísticas de construção, como por exemplo o método guloso. São aquelas em que uma ou mais soluções são construídas elemento a elemento, seguindo

algum critério de otimização. O algoritmo é executado até que se tenha uma solução viável para o problema;

- Heurísticas de busca em vizinhança, como a busca local. Suas soluções partem de uma solução candidata inicial para uma solução candidata vizinha melhor. O processo continua até que algum critério de parada seja satisfeito;

- Heurísticas sistemáticas. É exemplo o *backtracking* controlado, onde o espaço de soluções é representado por uma árvore e é percorrido utilizando critérios de ramificação e corte;

- Heurísticas híbridas, de combinações entre heurísticas com estratégias diferentes.

Apesar de serem algoritmos mais gerais, algum conhecimento do problema ainda é necessário para que uma heurística explore corretamente o espaço de busca. Em contra-partida, as meta-heurísticas são métodos em que pouco conhecimento específico do problema é necessário, embora muitas vezes alguma característica seja utilizada para melhorar seu desempenho. Seu nome é a combinação da palavra heurística (vista anteriormente) com meta, que tem origem grega e exprime a idéia de nível superior, maior generalidade.

"A natureza é sábia e muito podemos aprender com ela" [23]. É improvável imaginar que alguém envolvido com ciências exatas diga uma frase como essa. Todavia, o campo da inteligência computacional tem aprendido muito observando o comportamento da natureza. Foi estudando essas atividades que surgiram as redes neurais artificiais, algoritmos evolutivos, sistemas imunológicos artificiais, colônias de formigas, entre outras meta-heurísticas inspiradas na natureza. Esse tipo de técnica tem tido utilização crescente na resolução de diversos problemas dentro das mais variadas áreas. Podem ser aplicados à engenharia, matemática, medicina, ciência da computação, economia, etc.

O presente capítulo irá estudar as meta-heurísticas bio-inspiradas. As seções que seguem apresentarão esses métodos, mas uma discussão maior será feita apenas daqueles importantes para este trabalho: algoritmos genéticos e sistemas imunológicos artificiais.

3.1 Rede Neural Artificial

Desde a criação das máquinas de calcular, o homem tem sonhado com a construção de um cérebro artificial. Apesar de sua grande complexidade, alguns métodos, através de grandes simplificações, tentam simular seu processamento. As redes neurais artificiais [24] são baseadas nos sistemas nervosos naturais e constituem-se de unidades interconectadas. Essas unidades são os neurônios. Segundo Forbes [25], as redes neurais artificiais são úteis em problemas onde não se pode achar uma solução algorítmica, mas é possível encontrar diversos exemplos de como procurá-la, ou onde pode-se identificar a estrutura das soluções de dados existentes. Ou seja, elas "aprendem" a encontrar os resultados através de exemplos.

As principais vantagens de se utilizar esse método são a adaptação eficiente a alterações e sua estrutura facilmente paralelizável. As redes neurais artificiais têm sido aplicadas com sucesso em diversos tipos de problemas. As principais classes de utilização são [25]: reconhecimento, inferência e clusterização.

3.2 Algoritmos Evolutivos

Os Algoritmos Evolutivos são métodos de busca estocásticos inspirados no processo de seleção natural: indivíduos mais aptos tem mais chances de sobrevivência do que os outros. Em cada iteração, ocorre um processo de seleção que elimina indivíduos menos aptos. Os indivíduos mais aptos são combinados afim de obter soluções melhores. Além disso, outro processo que também ocorre na natureza é a mutação. Esse processo evolutivo modifica os indivíduos com o objetivo de aumentar o poder exploratório do algoritmo. Isso acontece porque a mutação inibe a formação de padrões nos indivíduos gerados através das gerações (evitando a convergência prematura). Esse conjunto de operadores (seleção, cruzamento e mutação) conduz a população a evoluir suas soluções candidatas em busca de melhores resultados, ou seja, de indivíduos mais aptos.

Diversos são os algoritmos inspirados na teoria de sobrevivência do mais apto. Dentre eles se destacam os algoritmos genéticos (que serão mais detalhados na Seção 3.6), algoritmos meméticos, algoritmos de estimação de distribuição e as es-

tratégias evolutivas $(\mu + \lambda)$ e (μ, λ). Como as soluções candidatas tendem a evoluir através dos processos de combinação e mutação obtendo melhores indivíduos, os algoritmos evolutivos são largamente utilizados em problemas de otimização, sejam eles contínuos, discretos, mistos, uni- ou multi-modais, com ou sem restrições, etc.

3.3 Colônia de Formigas

Ant Colony Optimization [26, 27] (ACO) se inspira no comportamento de colônias de formigas para traçar rotas entre o formigueiro e o alimento. Ao caminhar, as formigas secretam uma substância chamada feromônio. As outras formigas se guiam pela densidade desse material no meio ambiente, ou seja, quanto maior for o número de formigas que passam por um certo caminho, maiores são as chances dessa rota ser escolhida pelas próximas formigas. A idéia básica do algoritmo é simular esse comportamento através de formigas artificiais que traçam caminhos em um grafo cujos vértices representam componentes da solução.

A quantidade de feromônio depositado nas arestas de uma rota é proporcional à qualidade da solução correspondente. Cada formiga artificial tende a escolher as arestas com mais feromônio. Com o tempo, as melhores escolhas/componentes são reforçadas e o processo converge para uma boa solução.

3.4 Otimização por Enxame de Partículas

A *Particle Swarm Optimization* (PSO) [28] tem sua execução espelhada no comportamento social dos pássaros em revoada ou peixes num cardume. Por esse motivo, é algumas vezes chamada de "bando de pássaros". Nessa meta-heurística, há um conjunto de partículas (ou pássaros) que percorrem o espaço de busca baseando-se em conceitos individuais e sociais. A ênfase dada à melhor solução encontrada por uma dada partícula, em sua movimentação, caracteriza seu comportamento individual. O grau de importância de um elemento em relação aos seus vizinhos reflete a socialidade. Cada partícula, associada a uma solução que evolui, é vizinha de um conjunto de outras partículas que nunca é modificado. Uma importante característica comportamental dessa meta-heurística é que a estrutura de vizinhança é feita com o

objetivo de evoluir todas as partículas de uma região caso progressos sejam obtidos por esta.

O PSO é utilizado na resolução de problemas de otimização, em especial, para espaços de busca contínuos.

3.5 Otimização Extrema

Esta meta-heurística baseia-se em ecossistemas onde a evolução é construída através da extinção de indivíduos mal-adaptados. Seu foco de melhoramento é na eliminação de elementos ruins. Apesar da clara semelhança com algoritmos evolutivos, a otimização extrema (*Extreme Optimization*) [29, 30] não é classificada como tal por não ser baseada numa população de soluções. Depois da solução inicial ser elaborada, um processo de eliminação do pior componente da solução, com base em uma função de aptidão, é repetidamente executado. Alterações são freqüentemente necessárias para que a solução continue consistente.

Como o próprio nome sugere, esta é uma meta-heurística a ser aplicada em problemas de otimização.

3.6 Algoritmos Genéticos

Como vimos, muitos dos métodos de otimização matemática requerem que os problemas a serem resolvidos atendam a uma série de requisitos para que possam ser aplicados. Entretanto, em grande parte dos problemas da vida real essas técnicas nem sempre podem ser aplicadas. A modelagem matemática pode levar a uma função objetivo não diferenciável, ou então, a restrições que não podem ser escritas de forma explícita.

Os Algoritmos Genéticos, ou AG's, são algoritmos probabilísticos largamente utilizados em processos de otimização. Como será visto aqui, não há necessidade de se conhecer explicitamente a função que se deseja otimizar. O cálculo da função objetivo pode ser feito como em uma "caixa preta". Dado um conjunto de variáveis, precisa-se apenas dos valores da função objetivo e das restrições. Essa metodolo-

gia também apresenta uma grande vantagem quando somente a função objetivo é conhecida: não há necessidade de conhecer as derivadas da função objetivo e das restrições.

Durante as próximas seções serão vistas as principais características dos Algoritmos Genéticos e diversas estratégias de como utilizá-los para resolver problemas de otimização.

3.6.1 Fundamentos

Os AG's são algoritmos probabilísticos para otimização de sistemas complexos baseados na teoria da evolução de Charles Darwin [31]. O objetivo desse algoritmo é simular matematicamente o processo de evolução biológica a fim de buscar melhores soluções candidatas para um dado problema.

Entre os anos de 1950 e 1960 vários cientistas da computação começaram a estudar os sistemas evolucionários de maneira independente. Estes sistemas se baseavam no princípio de que a teoria da evolução das espécies de Charles Darwin poderia ser utilizada como ferramenta de otimização para diversos problemas de engenharia [32].

Diversos problemas, principalmente no escopo da ciência da computação [33] e da engenharia aeronáutica [34], tiveram várias aplicações desenvolvidas durante a década de 1960 utilizando esse mesmo princípio.

Começou-se a pesquisar métodos para estudar o fenômeno da adaptação das espécies como a que acontece na natureza, e a se desenvolver estratégias para a utilização desses conceitos em sistemas computacionais. Foi em 1973 que John Holland publicou o primeiro trabalho sobre o assunto [5]. Logo depois, em 1975, veio o livro [6] que chegou, no início da década de 90, a sua segunda edição [7]. Ao contrário dos outros pesquisadores que aplicavam a teoria da evolução a problemas específicos, seu objetivo era usar a teoria para buscar uma metodologia mais geral para a resolução de problemas. Foram dessas pesquisas que surgiu o conceito dos algoritmos genéticos como uma abstração da evolução biológica e estabeleceu-se uma base teórica para a adaptação de soluções através da variação genética e da seleção natural [6].

Esse tipo de algoritmo tem recebido um grande impulso em diversas áreas de

aplicação científica há mais de 20 anos. Isso por poder ser utilizado nas mais diversas áreas e, principalmente, pelos excelentes resultados que vêm conseguindo. Sua utilização em problemas de modelagem computacional tem sido largamente aproveitada. A utilização de AG's se estende por diversas áreas: otimização estrutural [19, 35, 36], otimização de funções multimodais [37] e com restrições [38], processamento de imagem [39], controle de sistemas [40], entre muitos outros.

Estes algoritmos codificam uma solução para um problema específico em uma estrutura de dados, como a de um cromossomo, e aplicam operadores que re-combinam estas estruturas produzindo outras. Os AG's são fundamentados no processo de seleção natural de Charles Darwin e nos mecanismos da genética. Consistem basicamente em encontrar, dado um conjunto de elementos ou indivíduos, aquele que melhor atende a certas condições específicas [41].

Segundo a teoria da evolução, os indivíduos mais aptos apresentam maior probabilidade de reprodução, deixando mais descendentes e assim, perpetuando seus códigos genéticos. O fundamento dos algoritmos genéticos é buscar imitar estes princípios na construção de algoritmos computacionais a fim de buscar uma melhor solução a um dado problema através da evolução de uma população de indivíduos.

Cada indivíduo na população representa uma possível solução para um dado problema. O algoritmo genético se desenvolve pela criação de indivíduos cada vez mais aptos com relação ao objetivo do problema, procurando uma melhor solução para este [41].

O algoritmo genético foi estruturado de forma que as informações referentes a um determinado sistema pudessem ser codificadas de maneira análoga aos cromossomos biológicos.

Algumas definições relacionadas com os algoritmos genéticos e importantes para que seus passos possam ser melhor descritos, são apresentadas a seguir [41]:

- Cromossomo: cadeia de caracteres representando informações relativas às variáveis do problema;
- Indivíduo: solução representada por um cromossomo;
- Gene: é a unidade básica do cromossomo;

- População: conjunto de indivíduos ou soluções;
- Geração: número da iteração em que o algoritmo genético está sendo executado;
- Operações Genéticas: operações que o algoritmo realiza sobre os cromossomos;
- Aptidão ou *Fitness*: informação numérica do desempenho de cada cromossomo na população.

Para que um algoritmo genético evolua computacionalmente, pode-se seguir um esquema básico de iteração composto pelas fases do processo evolutivo que serão descritas nas seções seguintes. A Figura 3.1 mostra um fluxograma descrevendo um algoritmo genético padrão. Fica a cargo do desenvolvedor apenas qual a melhor implementação em cada uma dessas etapas. É importante destacar que não existe uma "combinação mágica" que funcione perfeitamente para todos os problemas.

3.6.2 Codificação e Decodificação do Problema

Geralmente existem dois componentes do algoritmo genético que são dependentes do problema: a codificação do mesmo e a função mérito. A representação de uma solução em um cromossomo é fundamental para um algoritmo genético, pois é a maneira de tratar computacionalmente as informações do problema [23].

O cromossomo biológico é composto de genes, que são responsáveis por determinadas características do indivíduo, como por exemplo, cor dos olhos, altura, cor da pele, etc. Através de uma analogia é possível construir um cromossomo artificial. Em problemas de otimização, pode-se representar uma solução candidata como uma seqüência binária (cromossomo), por exemplo, dos valores das variáveis.

A representação das possíveis soluções do espaço de busca de um problema define a estrutura do cromossomo a ser manipulado pelo algoritmo. Essa representação depende do tipo de problema e do que, essencialmente, se deseja manipular geneticamente.

A escolha da representação cromossomial é uma decisão importante tomada pelo analista e deve ser feita com base no problema. Algumas regras gerais devem ser seguidas:

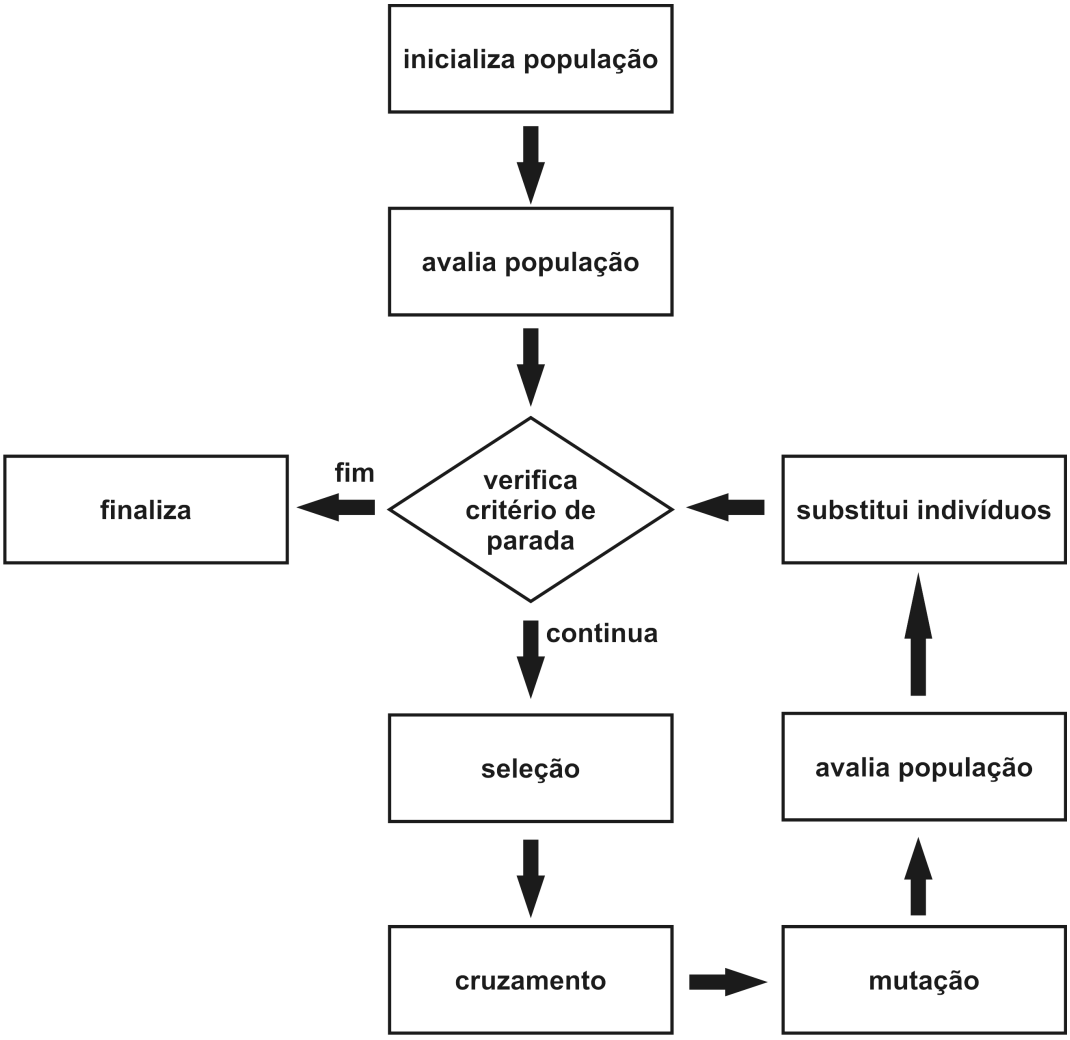

Figura 3.1: Diagrama com a seqüência de execução de um algoritmo genético

- A representação deve ser simples;
- Não deve haver representação para soluções proibidas ao problema;
- Todas as condições do problema devem estar implícitas na codificação.

Como será visto mais adiante, nos problemas com restrições, às vezes não é possível (ou adequado) utilizar uma codificação de acordo com os ítens listados anteriormente.

Existem várias formas de se representar uma possível solução para o problema a ser resolvido por um algoritmo genético. A representação binária é uma das mais utilizadas pela sua simplicidade e por poder ser facilmente utilizada em uma grande quantidade de diferentes problemas. Apesar desse tipo de representação ser útil em diversos casos, existem outros tipos de representação que podem ser mais adequados a certos problemas. Os principais tipos de representação são:

- Binário: pode ser utilizado para codificar soluções em diversos tipos de problemas. Os principais são números inteiros e contínuos;
- Real: a representação real é normalmente a mais adequada para problemas com variáveis contínuas;
- Permutação de símbolos: adequado para o caso de problemas onde alguma ordenação é importante;
- Árvore: estrutura de dados conveniente para representar uma expressão algébrica em problemas de regressão simbólica.

Usando a terminologia da biologia, a representação genotípica é feita pelo cromossomo. O fenótipo de uma solução é a decodificação desse cromossomo. A Tabela 3.1 exemplifica a codificação de um cromossomo binário (com seis bits) para um problema com números inteiros.

Tendo em mãos um indivíduo, que é candidato à solução do problema e possui um cromossomo que representa essa solução de alguma forma, precisa-se, agora, construí-la à partir dessa representação. Esse processo é chamado de decodificação do cromossomo.

Indivíduo	Cromossomo (Binário)	Fenótipo (Inteiro)
1	100011	35
2	001000	8
3	101011	43
4	111000	56

Tabela 3.1: Exemplos de representação binária

A vantagem da representação binária é sua fácil transformação para um número inteiro ou real. Na transformação para um real, considera-se um intervalo de valores (valores máximo e mínimo para cada variável) ou comprimento contínuo do domínio (c) de tal forma que:

$$x_r = \frac{x_b}{2^n - 1} c + x_{min} \tag{3.1}$$

Onde, $x_r \in [x_{min}, x_{max}]$, x_b é o inteiro correspondente ao binário, n é o número de bits de cada variável do qual o cromossomo é composto e c é o comprimento contínuo do domínio da variável dado por $c = x_{max} - x_{min}$.

A representação binária é considerada a maneira mais simples de traduzir as informações dos problemas de forma que possam ser tratados pelo computador. Por ser umas das formas mais genéricas, será a codificação utilizada no presente trabalho. As etapas que se seguem corresponderão ao caso binário.

3.6.3 Função de Avaliação

Além da questão da codificação (e conseqüentemente da decodificação), a função de avaliação é também parte da descrição do problema. A avaliação é o elo entre o algoritmo genético e o mundo externo.

A aptidão de um indivíduo é determinada pela decodificação de suas variáveis e pela avaliação da função mérito, chamada de função de avaliação, nas variáveis. A função de avaliação pode ser a própria função objetivo ou outra função constituída do conhecimento do problema específico.

Para exemplificar, a função matemática $f(x) = 3^x$ será utilizada como função

de avaliação para se obter a aptidão de três indivíduos. Segue a Tabela 3.2 onde pode-se visualizar os resultados.

Indivíduo	Cromossomo Binário	Decodificação	Aptidão
1	00001	1	3
2	00010	2	9
3	00100	4	81

Tabela 3.2: Exemplos de cálculos de aptidões

Pode-se observar, a partir dos cálculos das aptidões dos indivíduos da Tabela 3.2, que para um problema de otimização onde o objetivo é buscar o ponto de máximo, o indivíduo 3 é o mais apto dos três, por possuir maior aptidão. Caso o objetivo do problema fosse buscar o mínimo da função, o indivíduo 1 seria o mais apto dos apresentados.

3.6.4 Seleção

Segundo a teoria da seleção natural das espécies, os indivíduos mais áptos teriam maior chance de sobrevivência e de transmitir sua herança genética. A seleção dos indivíduos é baseada em sua aptidão, onde os mais aptos tem maior probabilidade de serem escolhidos para o processo de reprodução. Assim, se f_i é a avaliação do indivíduo i na população corrente de tamanho N, uma das formas de se obter a probabilidade p_i desse indivíduo ser selecionado é dada por:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{3.2}$$

onde assume-se que $f_i \geq 0$ para todo i.

Blickle e Thiele [42] analisaram os principais métodos de seleção usados em algoritmos genéticos. Os principais esquemas para este fim são [42]: torneio, truncada, *ranking* (este podendo ser linear ou exponencial) e roleta. Também em [42] são feitas análises referentes à aptidão média esperada e sua variância após o processo de seleção, taxa de reprodução (razão entre o número de indivíduos com uma certa

aptidão depois e antes da seleção), perda de diversidade (proporção dos indivíduos da população que não são selecionados), pressão (ou intensidade) e variância da seleção (variância normalizada após o processo de seleção). A pressão de seleção é dada pela divisão da diferença das médias da população antes e depois da seleção pela variância após esse processo (vide [42]).

Seleção por Roleta

Conhecida como seleção proporcional, ou por amostragem estocástica uniforme, neste método a probabilidade de um indivíduo ser selecionado é simplesmente proporcional ao valor de sua aptidão [42]. A chance de alguém ser escolhido é como na Equação 3.2 [42].

Um algoritmo para determinar os indivíduos selecionados utilizando a seleção proporcional pode ser visto na Figura 3.2. Este simples algoritmo, primeiramente, soma as aptidões dos indivíduos da população e posteriormente calcula as probabilidades de cada um deles ser selecionado. O restante do algoritmo refere-se à seleção dos indivíduos com base nessas probabilidades.

É fácil ver que esse mecanismo apenas funciona para problemas em que todas as aptidões são maiores ou iguais a zero. Uma alternativa a isto é deslocar as aptidões para área positiva. Entretanto, com esse mudança um dos indivíduos fica com probabilidade zero de ser selecionado.

Outro problema desse método é que, se um indivíduo possuir uma aptidão muito superior às demais ele terá uma probabilidade enorme de ser escolhido. Esse comportamento impede que outros indivíduos da população contribuam para sua evolução e pode causar uma convergência genética prematura. Esse tipo de indivíduo é conhecido como “super indivíduo”.

Seleção por Torneio

Na seleção por torneio, dois ou mais indivíduos são escolhidos ao acaso na população e são comparados. O melhor dos indivíduos vence o torneio. Este procedimento é realizado novamente diversas vezes até que o número de indivíduos necessário seja selecionado. Um algoritmo para esse processo pode ser visto na Figura 3.3.

1: **procedimento** SELECAORANKING(*populacao*, p_{min}, p_{max})
2: **para** $i = 1 : N$ **faça**
3: $soma \leftarrow f_i + soma$
4: **fim para**
5: **para** $i = 1 : N$ **faça**
6: $p_i \leftarrow \frac{f_i}{soma}$
7: **fim para**
8: **para** $i = 1 : numeroIndividuosSelecionar$ **faça**
9: $valor \leftarrow$ NUMEROAOACASO(0, 1)
10: $j \leftarrow 1$
11: **enquanto** $p_j < valor$ **faça**
12: $j \leftarrow j - 1$
13: **fim enquanto**
14: $selecionado \leftarrow$ PEGAR(*populacao*, j)
15: INSERIR($selecionado, populacaoSelecionados$)
16: **fim para**
17: **fim procedimento**

Figura 3.2: Pseudo-código para a seleção por roleta.

No algoritmo da Figura 3.3 pode ser visto que *numeroIndividuosSelecionar* é o número de indivíduos a serem selecionados, a população corrente é representada pela variável *populacao*, o número de indivíduos participantes do torneio por *numeroIndividuos* e a população de indivíduos selecionados é dada por *populacaoSelecionados*. Na linha 4 um indivíduo é selecionado ao acaso da população e em 5 este é adicionado à população de indivíduos selecionados ao acaso. Logo após, o melhor indivíduo dentre os obtidos ao acaso é inserido na população de indivíduos selecionados. O procedimento da linha 9 esvazia a população que é passada como parâmetro.

```
1: procedimento SELECAOPORTORNEIO(populacao, numeroIndividuos, N)
2:     para i = 1 : numeroIndividuosSelecionar faça
3:         para i = 1 : numeroIndividuos faça
4:             aux ← ESCOLHERAOACASO(populacao)
5:             INSERIR(aux, escolhidosAoAcaso)
6:         fim para
7:         selecionado ← PEGARMELHOR(escolhidosAoAcaso)
8:         INSERIR(selecionado, populacaoSelecionados)
9:         LIMPAR(escolhidosAoAcaso)
10:    fim para
11: fim procedimento
```

Figura 3.3: Pseudo-código para a seleção por torneio.

Seleção Truncada

No processo de seleção truncada, somente a fração $T \in [0, 1]$ dos melhores indivíduos pode ser selecionada e com igual probabilidade. Esse método é equivalente à seleção (μ, λ) usada em estratégias evolucionistas com $T = \frac{\mu}{\lambda}$ [42]. Um pseudo-código pode ser visto na Figura 3.4.

```
1: procedimento SELECAOTRUNCADA(populacao, T)
2:     ORDENAR(populacao)
3:     para i = 1 : numeroIndividuosSelecionar faça
4:         indice ← NUMEROAOACASO(1, T * N)
5:         selecionado ← PEGAR(populacao, indice)
6:         INSERIR(selecionado, populacaoSelecionados)
7:     fim para
8: fim procedimento
```

Figura 3.4: Pseudo-código para a seleção truncada.

No algoritmo da Figura 3.4 tem-se que N é o número de indivíduos da população,

o método da linha 2 ordena a população do melhor para o pior indivíduo, na 4 um valor é selecionado ao acaso tal que $indice \in [1, T * N]$ com *indice* inteiro e, em 5, o indivíduo na posição selecionada é obtido.

Seleção por *Ranking*

Aqui, os indivíduos são ordenados de acordo com os valores de suas aptidões. A probabilidade de seleção depende da posição dos indivíduos nesta lista ordenada. O ranking pode ser, por exemplo, linear ou exponencial. Segundo [23], a probabilidade linear de alguém ser selecionado de acordo com sua posição pode ser dada pela Equação 3.3.

$$p_i = p_{min} + \left[(p_{max} - p_{min}) \frac{i-1}{N-1} \right]; \quad i \in \{1, \ldots, N\} \tag{3.3}$$

onde, p_{max} é a probabilidade do melhor indivíduo ser selecionado e p_{min} é a probabilidade do pior deles ser selecionado. Como pode ser observado, a probabilidade de um indivíduo i da população de tamanho N ser selecionado varia linearmente de acordo com sua posição (*ranking*) na população ordenada. A Figura 3.5 descreve um algoritmo para essa forma de seleção.

Na linha 5 do algoritmo da Figura 3.5 o método *calculaProbabilidade* corresponde à Equação 3.3. O restante do procedimento se resume a selecionar os indivíduos baseado em suas probabilidades s_i. Observe que se $valor = 0$, então o pior indivíduo é selecionado e se $valor = s_N$, então o escolhido é o melhor deles.

Como já foi dito aqui, a seleção por *ranking* também pode ser exponencial. Esta é uma maneira de se aumentar a pressão seletiva e diminuir a diversidade da população. Porém, é uma maneira de resolver o problema do "super indivíduo", já que a probabilidade aqui depende exponencialmente da posição do indivíduo na população ordenada. Segundo [42], a Equação 3.4 pode ser utilizada para definir a probabilidade exponencial do indivíduo i ser selecionado.

$$p_i = \frac{c^{N-i}}{\sum_{j=0}^{N-1} c^j}; \quad i \in \{1, \ldots, N\} \tag{3.4}$$

onde c é um parâmetro definido pelo usuário tal que $0 < c < 1$.

```
1: procedimento SELECAORANKING(populacao, p_min, p_max)
2:     ORDENAR(populacao)
3:     s_0 ← 0
4:     para i = 1 : N faça
5:         s_i ← s_{i-1} + CALCULAPROBABILIDADE(i, p_max, p_min)
6:     fim para
7:     para i = 1 : numeroIndividuosSelecionar faça
8:         valor ← NUMEROAOACASO(0, s_N)
9:         j ← N
10:        enquanto s_j < valor faça
11:            j ← j − 1
12:        fim enquanto
13:        selecionado ← PEGAR(populacao, j)
14:        INSERIR(selecionado, populacaoSelecionados)
15:    fim para
16: fim procedimento
```

Figura 3.5: Pseudo-código para a seleção por ranking.

O algoritmo para a seleção por *ranking* exponencial é semelhante ao linear. A diferença está na linha 5 no algoritmo da Figura 3.5. Na seleção linear ela realiza o cálculo da Equação 3.3 enquanto que, no caso do *ranking* exponencial, a probabilidade vem da Equação 3.4.

3.6.5 Cruzamento ou *Crossover*

Nesta etapa do algoritmo é realizado o cruzamento entre o material genético dos indivíduos selecionados na etapa anterior. A partir dessa população, uma nova população é formada pelo cruzamento aleatório entre os cromossomos. Portanto, os filhos gerados recebem informações de seus progenitores, através do material genético proveniente deste cruzamento. Normalmente, a quantidade de filhos gerados é igual ao tamanho da população original. Uma probabilidade para a ocorrência

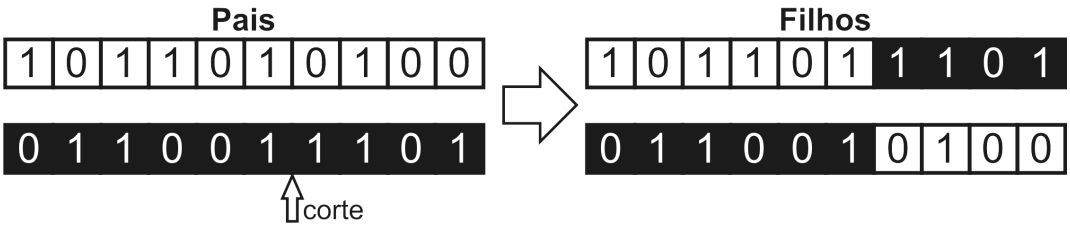

Figura 3.6: Representação para o *crossover* de um ponto

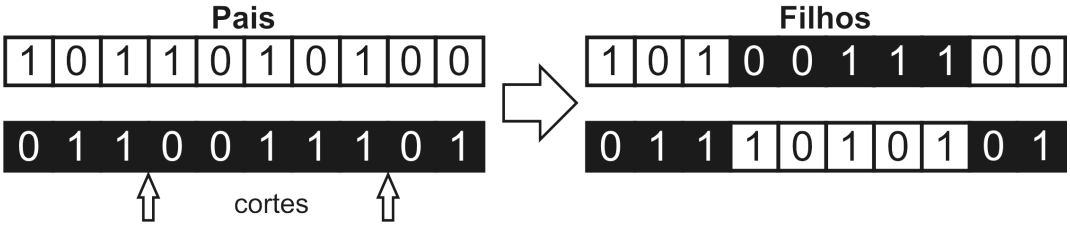

Figura 3.7: Representação para o *crossover* de dois pontos

do cruzamento deve ser definida, ou seja, normalmente, há chance de não ocorrer troca genética entre os indivíduos que seriam, então, simplesmente clonados. Esse parâmetro é conhecido como probabilidade de *crossover*.

Durante a permuta de material genético entre os indivíduos, haverá uma tendência da transmissão das características dominantes para as gerações futuras. Após algumas gerações pode-se observar que uma elevada taxa dos indivíduos possuem a presença de características dominantes.

O processo de cruzamento entre indivíduos com codificação binária pode ser feito através da escolha aleatória da posição onde ocorrerá o corte seguido pela troca de material genético. O número de rupturas geralmente varia entre 1 e 2. Essas duas formas de cruzamento são conhecidas como *crossover* de um e dois pontos. Quanto maior for o número de rupturas menor será a semelhança entre os pais e os filhos. As Figuras 3.6 e 3.7 mostram como as trocas genéticas ocorrem nos casos de *crossover* de um e dois pontos, respectivamente.

Outro conhecido método para se fazer a combinação genética entre os indivíduos selecionados é o *crossover* uniforme. Neste caso, os genes são trocados com base numa taxa escolhida. Para cada posição um número aleatório deve ser gerado e comparado com esse valor escolhido. O *crossover* uniforme tem uma grande capacidade de combinar padrões genéticos. A Figura 3.8 mostra um exemplo de *crossover*

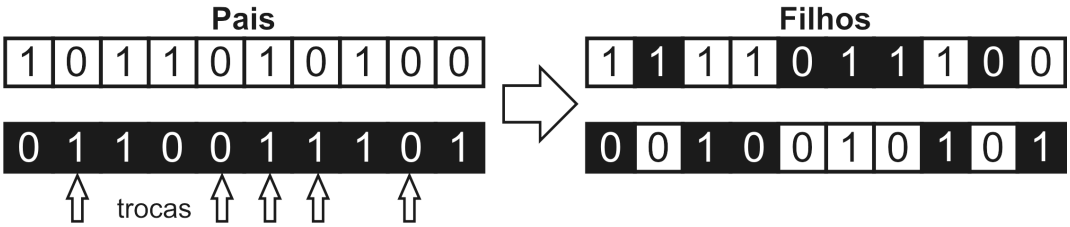

Figura 3.8: Representação para o *crossover* uniforme

Figura 3.9: Representação do operador de mutação

uniforme.

Diversas outras formas de cruzamento podem ser encontradas na literatura. Por exemplo, [23] tráz, além das apresentadas até aqui, o *crossover* baseado em maioria. Neste, n indivíduos são escolhidos entre os pais previamente selecionados. Para se gerar cada gene do filho verifica-se qual o valor na maioria desses pais.

3.6.6 Mutação

Após o cruzamento, os cromossomos dos indivíduos gerados pelo processo cruzamento passam por mais um operador genético: a mutação. Mutação é um operador exploratório que tem por objetivo aumentar a diversidade na população.

O procedimento mais comum para este operador é determinar uma taxa para a ocorrência da mutação. Para cada gene, um número entre 0 e 1 é sorteado ao acaso e, caso este seja menor que a taxa definida de mutação, então a mudança genética acontece. Essa taxa pode ser fixa ou variar ao longo da evolução.

São diversas as formas de se efetuar mutação. Por exemplo, pode-se trocar o valor do gene em questão por outro diferente. Trocar um valor 0 por um valor 1 ou ao contrário (no caso de cromossomo binário). A Figura 3.9 exemplifica o processo de mutação para um problema com codificação binária.

O processo descrito acima é o mais utilizado na literatura. Contudo, diversos outros métodos de mutação foram elaborados para criar diversidade na população e evitar sua convergência prematura. Um deles é o operador de mutação dirigida [23]. Neste método, depois de um dado número de gerações verifica-se, para cada posição no cromossomo, a ocorrência de um gene padrão nos melhores indivíduos. Caso um alelo comum seja verificado, o operador de mutação atua nesses genes e é aplicado a todos os indivíduos da população.

3.6.7 Otimização com Restrições

Os algoritmos genéticos são geralmente aplicados a problemas de otimização complexos que, freqüentemente, envolvem restrições. Os AGs, bem como outras meta-heurísticas inspiradas na natureza, requerem um mecanismo adicional para incorporar restrições de diversos tipos.

Um problema de otimização restrito padrão em $\Re^n$ é como visto em (2.1)-(2.4). As restrições de limitação, caso existam, não são incorporadas ao processo de codificação por serem triviais em algoritmos genéticos. Sua implementação é feita normalmente na codificação/decodificação do problema.

A padronização e a implementação de um algoritmo genético robusto para problemas com restrição não é uma tarefa fácil [43]. Atualmente, diversas técnicas estão sendo propostas para resolver problemas de otimização que tenham algum tipo de restrição. Muitas possibilidades tem sido exploradas e podem ser agrupadas como [43]:

- Técnicas de penalização;
- Métodos de reparo;
- Decodificadores especiais;
- Operadores especiais;
- Técnicas de seleção;
- Métodos híbridos.

Nos métodos de reparo, a técnica é mover os indivíduos não válidos para dentro de um domínio factível conhecido. Entretanto há situações onde essa operação é muito cara ou até impossível.

Nos sistemas que utilizam decodificadores especiais, o padrão é extrair um fenótipo factível de um dado genótipo. Porém, assim como nas técnicas de reparo, essa operação não é trivial e algumas vezes impossível.

Em algumas situações especiais, operadores genéticos podem ser utilizados para que sempre filhos factíveis sejam gerados do cruzamento de pais factíveis.

Os métodos híbridos combinam, freqüentemente, programação matemática com técnicas evolutivas. O presente trabalho utiliza um método onde duas meta-heurísticas, algoritmo genético e sistema imunológico artificial, são combinadas com o objetivo de melhorar os resultados. Esse algoritmo é discutido nas próximas seções.

As técnicas de seleção podem, por exemplo, privilegiar a escolha de indivíduos factíveis da população para a etapa do cruzamento. É comumente utilizada em conjunto com outros métodos. Na Seção 3.6.7, é apresentado um exemplo de técnica de seleção que tem obtido bons resultados em diversos testes encontrados na literatura: a ordenação estocástica. Alguns dos resultados obtidos por esse método são utilizados para comparações no Capítulo 5.

As técnicas de penalização são as mais comuns. Caracterizam-se por substituir as restrições do problema por algum tipo de penalização, transformando o problema de otimização restrita em um problema irrestrito. Por serem tão populares, elas são melhor detalhadas a partir da Seção 3.6.7.

Técnicas de Seleção

Na técnica da ordenação estocástica (stochastic ranking ou simplesmente SR) [44], o balanço entre a função objetivo e as restrições é feito por um processo de ordenação baseado num algoritmo de ordenação de bolha estocástico. Neste método é utilizada uma probabilidade p_f de se usar apenas a função objetivo como forma de comparar indivíduos na região infactível do espaço de busca. Ou seja, dados dois indivíduos a serem comparados, a probabilidade da comparação de acordo com a função objetivo é 1, se ambos são factíveis, ou p_f, caso contrário. Este procedimento

produz um bom equilíbrio no conjunto ordenado.

Por ser interessante que o algoritmo finalize com uma solução factível, a probabilidade p_f deve ser menor que 0.5, criando uma pressão contra indivíduos infactíveis. Essa pressão pode ser ajustada simplesmente ajustando o valor de p_f [45]. Quando $p_f = 0$, a ordenação se torna equivalente a uma super-penalização, onde todas as soluções factíveis são consideradas melhores que qualquer infactível. Por outro lado, se $p_f = 1$, as comparações entre os indivíduos será baseada somente na função objetivo, desconsiderando as restrições.

Técnicas de Penalização

Dentre a grande variedade de técnicas que podem ser utilizadas para se resolver um problema de otimização com restrições via algum algoritmo evolucionista, as técnicas de penalização são as mais populares [11].

Em geral, essa técnica se caracteriza por transformar um problema de otimização com diversas restrições em outro sem nenhuma restrição a ser considerada. Essa mudança tem como base a adição ou subtração de algum valor da aptidão dos indivíduos. O valor a ser modificado no cálculo da função objetivo é obtido através da violação das restrição do problema a ser otimizado. Assim, se um problema de otimização tem por objetivo maximizar uma certa função de avaliação e o cálculo é feito em um indivíduo infactível, ou seja, que viola alguma das restrições, então um valor com base nessa violação é subtraído (penalização) do resultado do cálculo dessa função.

Assim, segundo [11], conclui-se que a função objetivo de problemas utilizando penalização ficaria como

$$\phi(x) = f(x) \pm \left[\sum_{i=1}^{a} r_i G_i + \sum_{j=1}^{b} c_j H_j \right] \tag{3.5}$$

onde $f(x)$ é a função de avaliação do problema; G_i e H_j são as violações das inequações e equações de restrição, $g_i(x)$ e $h_j(x)$, respectivamente; r_i e c_j são valores positivos normalmente chamadas de “coeficientes de penalização”.

As formas mais comuns de G_i e H_j são dadas pelas equações 3.6 e 3.7.

$$G_i = \max[0, g_i(x)] \tag{3.6}$$

$$H_i = |h_i(x)| \tag{3.7}$$

É interessante observar que quando o indivíduo é factível, os valores das penalizações são nulos e a aptidão acaba sendo o valor da função objetivo inicial, ou seja, $\phi(x) = f(x)$ para os indivíduos factíveis.

Pode-se observar, na prática, que o bom desempenho do algoritmo utilizando a técnica de penalização depende fortemente dos coeficientes de penalização aplicados. Existem diversas formas de penalização que buscam adotar um valor mais apropriado para este coeficiente. Dentre estas, se destacam: "pena de morte", estática, dinâmica, adaptativa.

"Pena de Morte"

Nesse método, os indivíduos infactíveis são "mortos". A rejeição dos indivíduos infactíveis é eficiente computacionalmente, já que nenhum cálculo extra é necessário. Entretanto, essa solução apresenta problemas quando a população inicial não possui nenhum indivíduo factível. Nesse caso, o algoritmo acaba se tornando uma busca aleatória dentro do espaço de soluções (factíveis e infactíveis). Além disso, indivíduos infactíveis próximos à área factível ou que tivessem conteúdo/potencial genético são descartados. Por isso, este é um método que, apesar de simples implementação, pode apresentar péssimos resultados.

Penalização Estática

Nesta técnica utiliza-se uma penalização proporcional à violação das restrições. Quanto mais um indivíduo se afasta da área factível mais esse indivíduo é penalizado. Um coeficiente de penalização é utilizado nesse caso e o cálculo da aptidão é dado por

$$\phi(x) = f(x) \pm \sum_{i=1}^{m} \left(r_{k,i} \max[0, g_i(x)]^2 \right) \tag{3.8}$$

onde $r_{k,i}$ é o coeficiente de penalização usado, m é o número de restrições do problema e $k = 1, \ldots, l$, sendo l o número de níveis de violação das restrições definidas. O objetivo é dividir os indivíduos em diferentes grupos. Esses grupos são definidos pelos níveis de violação das restrições e diferentes conjuntos de fatores são aplicados a cada um desses grupos. Uma dificuldade é o número excessivo de coeficientes $r_{k,i}$ a serem definidos pelo usuário.

Penalização Dinâmica

Na técnica dinâmica pode-se, por exemplo, aumentar a penalização com o passar das gerações. Uma proposta para a função aptidão para um indivíduo numa geração t qualquer é

$$\phi(x) = f(x) \pm (ct)^a \Theta(b, x) \tag{3.9}$$

onde c, a e b são constantes dadas e a função Θ é definida como

$$\Theta(b, x) = \sum_{i=1}^{m} G_i^b(x) + \sum_{j=1}^{p} H_j(x) \tag{3.10}$$

com $G_i(x)$ e $H_j(x)$ sendo as violações das restrições de inequações e de equações, respectivamente, quando houver, e iguais a zero caso contrário.

Penalização Adaptativa

Bean e Hadj-Alouane desenvolveram uma técnica que utiliza uma função de penalização que tenta levar em conta como o processo evolutivo vem ocorrendo [46]. A aptidão de cada indivíduo é calculada através de coeficiente "evolutivo"(dependente de t) dependendo do caso. Esses casos são definidos pelos melhores indivíduos das últimas k gerações. Se esses melhores indivíduos forem factíveis, então utiliza-se o caso 1. Se os indivíduos forem infactíveis, então o caso 2 é utilizado. As constantes b_1 e b_2 são restritas como mostrados na Equação 3.13. A aptidão $\phi(x)$ pode, então, ser calculada como

$$\phi(x) = \quad f(x) \pm o(t) \left(\sum_{i=1}^{m} G_i^2(x) + \sum_{j=1}^{p} H_j(x) \right) \tag{3.11}$$

onde

$$o(t) = \begin{cases} \dfrac{o(t-1)}{b_1}, \text{ no caso 1} \\ o(t-1)b_2, \text{ no caso 2} \\ o(t-1), \text{ caso contrário} \end{cases} \tag{3.12}$$

$$b_1 > b_2 > 1 \tag{3.13}$$

Repare que o coeficiente de penalização $o(t)$ diminui se os melhores indivíduos das últimas k gerações forem factíveis, aumenta caso esses sejam infactíveis e permanece o mesmo nas demais situações. Note também, que a função $o(t)$ é atualizada a cada geração t.

Smith e Tate [47] definem outra solução de penalização adaptativa. Esta foi depois refinada em [48]. Nessa técnica, a magnitude da penalização é dinamicamente modificada de acordo com a aptidão do melhor indivíduo encontrado até aquele passo. Apenas restrições de inequações foram consideradas nesse caso e as aptidões são calculas como segue

$$\phi(x) = \quad f(x) \pm (b_{factivel} - b_{todos}) \left(\sum_{i=1}^{m} \frac{G_i^k(x)}{\Theta(t)} \right) \tag{3.14}$$

onde, $b_{factivel}$ é o valor da aptidão do melhor indivíduo da geração t, b_{todos} é o valor da aptidão do melhor dos indivíduos infactíveis da geração t, k é a constante que ajusta o quão severa será a penalização (o valor de $k = 2$ é sugerido na literatura) e a função $\Theta(t)$ é conhecida como NFT (*Near Feasibility Threshold*). Essa função define uma região em que soluções infactíveis são pouco penalizadas. Formulações para a função $\Theta(t)$ podem ser encontradas em [47–49].

Outro exemplo de penalização adaptativa pode ser encontrado em [36, 43]. O método conhecido como APM (*Adaptive Penalty Method*) é uma interessante e eficaz forma de se trabalhar com problemas de otimização com restrições por não necessitar de nenhum parâmetro adicional. Este método utiliza informação da própria população para produzir os dados necessários para a penalização. A função de aptidão proposta é dada por

$$F(x) = \begin{cases} f(x), & \text{se } x \text{ é factível,} \\ \overline{f}(x) + \displaystyle\sum_{j=1}^{m} k_j v_j(x) & \text{, caso contrário} \end{cases} \tag{3.15}$$

onde $m = a + b$ é o número total de restrições e

$$\overline{f}(x) = \begin{cases} f(x), & \text{se } f(x) > \langle f(x) \rangle \\ \langle f(x) \rangle & \text{, caso contrário} \end{cases} \tag{3.16}$$

com $\langle f(x) \rangle$ sendo igual à média dos valores das funções objetivas dos indivíduos da população corrente.

O parâmetro de penalização k_j é definido em cada geração como

$$k_j = |\langle f(x) \rangle| \frac{\langle v_j(x) \rangle}{\sum_{l=1}^{m} [\langle v_l(x) \rangle]^2} \tag{3.17}$$

onde $\langle v_l(x) \rangle$ é a média da l-ésima restrição na população corrente. Assumindo N como o número de indivíduos da população, então a Equação 3.17 pode ser expressa como

$$k_j = \frac{|\sum_{i=1}^{N} f(x^i)|}{\sum_{l=1}^{m} \left[\sum_{i=1}^{N} v_l(x^i) \right]^2} \sum_{i=1}^{N} v_j(x^i) \tag{3.18}$$

O objetivo do método é distribuir o coeficiente de penalização de forma que as restrições mais difíceis tenham um coeficiente de penalização relativamente maior.

3.7 Sistemas Imunológicos Artificiais

A ciência da computação vem a algum tempo buscando inspiração na natureza para aperfeiçoar sua forma de resolver problemas computacionais. Os sistemas biológicos têm constituído uma grande fonte de inspiração para a elaboração de soluções em sistemas computacionais dando origem, dentre outras meta-heurísticas, aos Sistemas Imunológicos Artificiais (SIA).

O sistema imunológico natural é um mecanismo biológico capaz de detectar e eliminar patologias, ou seja, responsável por garantir a integridade física de um indivíduo.

Algumas características importantes herdadas do sistema imunológico natural e que podem ser úteis na elaboração de algoritmos computacionais são [50]:

- Reconhecimento de padrões: as células e moléculas que não pertencem ao organismo são reconhecidas e eliminadas pelo sistema imunológico;
- Detecção de anomalia: o sistema imunológico pode detectar e reagir a patógenos a que o organismo nunca havia sido exposto anteriormente;
- Tolerância a ruídos: um reconhecimento perfeito não é necessário para que o sistema imunológico reaja contra um patógeno;
- Diversidade: existe uma quantidade limitada de células no sistema imunológico que são utilizadas para se obter o reconhecimento de infinitos elementos;
- Aprendizagem por reforço: a cada encontro com o mesmo patógeno, o sistema imunológico melhora a qualidade de sua resposta;
- Memória: os componentes do sistema imunológico bem sucedidos no reconhecimento e combate às patologias são armazenados para uma resposta futura;
- Robustez;
- Adaptabilidade.

Há muitas aplicações onde os sistemas imunológicos artificiais podem ser utilizados, podendo-se destacar [50]:

- Classificação, análise de dados e reconhecimento de padrões [51–57];
- Segurança computacional, detecção de falhas e anomalias [58–61];
- Aprendizagem de máquina [62, 63];
- Robótica, controle e eletrônica [64–67];
- Otimização e pesquisa [68–77];
- Arranjos seqüenciais (planejamento ou *scheduling*) [78–80].

O presente trabalho, dentre as diversas aplicabilidades dos sistemas imunológicos artificiais, será focado na resolução de problemas de otimização com restrições.

3.7.1 Sistema Imunológico

O sistema imunológico (SI) é responsável pela defesa do organismo. Ele é fundamental para garantir a integridade dos animais e deve agir de forma eficiente. Segundo Tizard, pode-se definir um SI como:

Definição 3.1 *O sistema que defende o animal contra o ataque constante de microorganismos é chamado de sistema imunológico [81].*

Quando um invasor entra no organismo, pode-se observar que em pouco tempo o mesmo será eliminado. Este processo é conhecido como resposta imunológica. Sua principal característica é a capacidade do organismo de reconhecer e destruir substâncias estranhas. O sistema imunológico é capaz de localizar invasores, células e moléculas que se apresentam diferentes das pertencentes ao indivíduo. Ou seja, através desse mecanismo o SI pode identificar bactérias, vírus e até mesmo células do próprio corpo com algum tipo de má formação (como tumores).

Existe uma grande variedade de mecanismos que compõem o sistema imunológico. Alguns são otimizados para a defesa contra um único invasor, enquanto outros atuam contra diversos invasores. Inclusive, existe uma considerável redundância, pois diversos desses mecanismos podem ser ativados contra um único agente infeccioso. Tudo isso torna o sistema imunológico a principal barreira do hospedeiro, além de realizar uma resposta rápida e efetiva contra os invasores.

O sistema imunológico é um conjunto de células, tecidos e órgãos destinados a defender o organismo contra as mais diversas ameaças. Esses ataques podem ser internos, como as células cancerígenas, ou externos, por meio de vírus, bactérias ou fungos. Parte desse sistema vem pronto desde o nascimento (imunidade inata) e constitui a primeira linha de defesa do organismo.

Porém, o meio permanece evoluindo constantemente. Os antígenos se aperfeiçoam tornando a imunidade nativa muitas vezes ineficaz. Portanto, um complemento à imunidade natural é requerida. A imunidade adquirida tem a capacidade de aprendizagem para reconhecer e se defender dos novos invasores. É com essa imunidade que as vacinas trabalham. As vacinas são substâncias capazes de ativar uma resposta imune adquirida fazendo com que o organismo aprenda a se defender de um certo antígeno.

No século XI, a varíola infestava a população mundial. Naquela época, foi possível observar que aqueles que conseguiam se recuperar de uma primeira contaminação tornavam-se resistentes a um posterior contato com a doença. Sabendo disso, crianças (que eram as principais vítimas do mal) eram infectadas com extratos de pústulas da varíola. O objetivo é que, se conseguissem sobreviver, essas crianças estariam imunizadas contra futuros contágios. Após algum tempo descobriu-se que a utilização de pústulas de variações mais brandas da varíola também tornava os indivíduos resistentes à doença. Esse método ficou conhecido como variolação.

No final do século XVIII, o médico inglês Edward Jenner observou que os trabalhadores que ordenhavam o gado e tinham pego varíola das vacas, uma variação branda, ficavam protegidas contra a variação humana da doença. Jenner passou a utilizar pústulas bovinas para a variolação. Esse substrato era conhecido como *vaccínia* ou *cowpox* e por isso o procedimento ficou conhecido por vacinação.

Depois de Jenner, no século XIX, Robert Koch provou que as doenças infecciosas eram causadas por diferentes microorganismos. Aproximadamente em 1900, Louis Pasteur, em seus estudos sobre a vacinação, concluiu que a exposição a uma cepa não virulenta de um agente causador de doenças pode futuramente protegê-lo contra uma infecção desse microorganismo ou de outro semelhante.

O volume de pesquisas envolvendo os sistemas imunológicos vem crescendo. Estudos recentes estão sendo feitos na descoberta de novas vacinas, análises dos componentes bioquímicos e genéticos e nas interações biológicas entre os organismos.

Imunidade Inata e Adquirida

Dentro os diversos mecanismos de defesa utilizados pelo SI existem aqueles que fazem parte do organismo de um indivíduo desde seu nascimento. A imunidade inata (natural ou nativa) é uma estrutura imunológica básica e geral. Essa estrutura é, normalmente, vinculada a uma espécie e não a cada indivíduo. O sistema inato é composto por mecanismos que defendem o organismo respondendo sempre da mesma forma, independente do invasor. É constituído pelas formas de defesa mais primárias, podendo ser encontrados até mesmo em plantas e fungos.

A imunidade natural compreende diversas formas de defesa. São elas:

- Barreiras físicas: pele, ácido gástrico, saliva, lágrimas, bactérias da flora normal do intestino e mucos;

- Fagócitos: também conhecidos como *Natural Killers* (NK ou matadores naturais), são células com capacidade de englobar partículas ou microorganismos estranhos e, posteriormente, eliminá-los. Importante contra infecções bacterianas, pois os vírus são muito pequenos e os parasitas demasiado grandes para serem fagocitados. A fagocitose também é importante na limpeza dos detritos celulares após infecções ou processos que levem à morte celular de tecidos. Entretanto, essas células morrem após algumas fagocitoses. Se o número de invasores e material estranho for grande, ocorre a geração de pús, um líquido pastoso e rico em proteínas estruturais;

- Sistema complemento: é um grupo de proteínas produzidas pelo fígado e que está presente no sangue. Inserem-se na membrana celular do invasor e levam o invasor à lise através da criação de poros pelos quais entra uma grande quantidade de água. A lise é o processo de ruptura ou dissolução da membrana plasmática ou da parede bacteriana, que leva à morte da célula;

- Resposta inflamatória: é uma reação inesperada ocasionada por células danificadas. Elas sensibilizam os receptores da dor e atraem fagócitos.

Outro tipo de imunidade é chamada de adquirida (ou específica). Ela confere ao SI a robustez de defender o organismo de material estranho. Sua principal capacidade é a de distinguir entre proteínas produzidas por células do próprio corpo (antígeno *self* ou próprio) e proteínas produzidas por invasores ou por células sob o controle de vírus (antígeno *non-self* ou não próprio). Esse reconhecimento é realizado através da reação entre os antígenos (invasores) e os linfócitos. Assim, para um antígeno reagir com esses receptores, um reconhecimento deve ocorrer. Considera-se reconhecimento, a ligação, tipo complementar, entre o padrão protéico dos antígenos e os reagentes.

Os linfócitos são células responsáveis pela imunidade específica. É possível destacar duas classes distintas de receptores [2]: os linfócitos T e B.

Os linfócitos conhecidos como células B possuem receptores de imunoglobulina, também chamados de receptores de células B, que são produzidos na forma solúvel

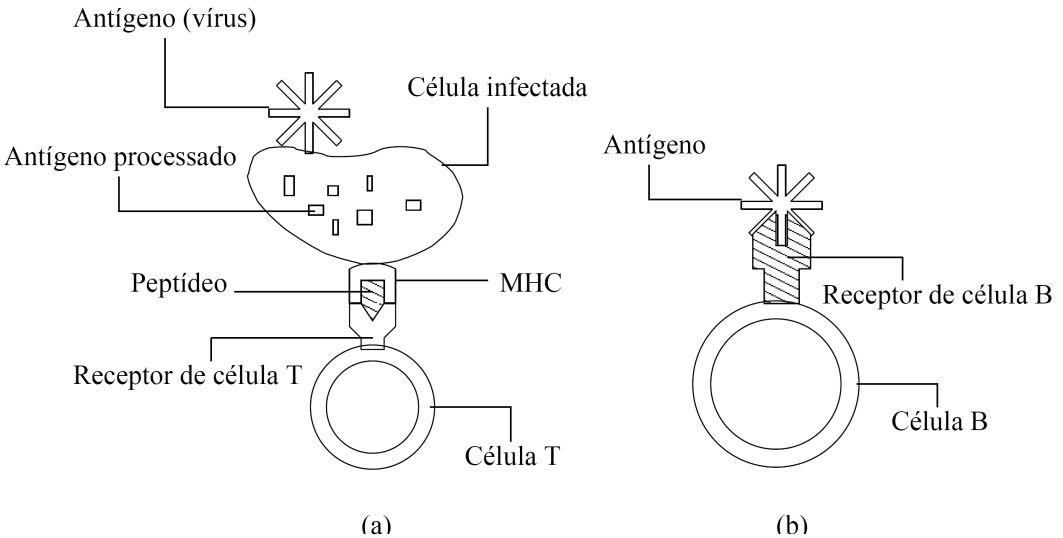

Figura 3.10: Reação das células do sistema imunológico com antígenos. Em (a) um linfócito T ligado a uma célula infectada por um vírus e em (b) um linfócito B associado diretamente a um antígeno [2].

como anticorpos, ou seja, cabe a eles a produção de imunoglobulinas (anticorpos). Cada clone (ou tipo) de célula B é responsável por um antígeno em específico.

Os outros linfócitos, chamados de células T, constituem a maior parte dos linfócitos do sangue. Esses podem ser divididos em diversas classes funcionais distintas [82]. As duas principais são as regulatórias (T4) e as citotóxicas (T8). Os linfócitos T4 são os que regulam as reações no caso do organismo ser invadido, ativando ou inibindo todas as outras células imunitárias. As células citotóxicas, assim como os linfócitos B, reconhecem os antígenos. Elas o fazem através de moléculas de proteína em sua superfície, conhecidas como receptores de células T. Ao contrário dos receptores de células B, esses não são decretados como anticorpos e não reagem com antígenos livres e intactos. Ao invés disso, eles reagem com moléculas do corpo que contenham MHC (*Major Histocompatibility Complex*, ou Complexo de Histocompatibilidade), que expressam fragmentos de antígenos (chamados de peptídeos) na superfície das células contaminadas pelo agente infeccioso. Caso haja um reconhecimento, enzimas são produzidas desencadeando na morte da célula ligada.

A Figura 3.10 mostra um exemplo de reação das células T (a), com uma célula infectada, e outro das células B (b), diretamente com um antígeno [2].

Para se ter uma resposta imunológica mais eficiente, os anticorpos precisam

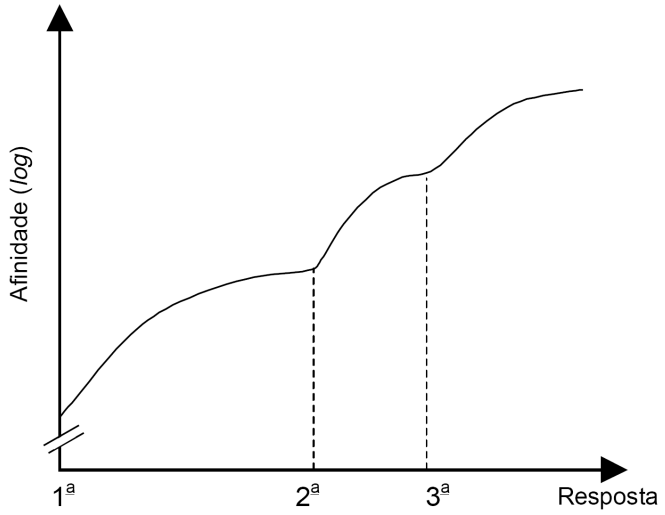

Figura 3.11: Gráfico de crescimento da afinidade depois de alguns contatos com um antígeno [3].

se aperfeiçoar. Quanto mais uma certa infecção nos afeta, mais o corpo se torna resistente a ele. Isso acontece, entre outros aspectos, por causa de uma técnica chamada de hipermutação ou seleção clonal. Na teoria da seleção clonal originalmente proposta por Burnet [3], a memória seria fornecida pela expansão do tamanho de um determinado clone específico ao estímulo antigênico, e a mutação aleatória seguida de seleção permitiria o aumento da afinidade desse clone. Com isso, somente os descendentes com alta afinidade antigênica serão selecionados.

A Figura 3.11 mostra o crescimento da afinidade durante diversas respostas imune adaptativas [3].

Diversificação dos Anticorpos

O repertório diversificado de anticorpos pode ser gerado de várias formas. Uma delas ocorre durante o desenvolvimento das células B por rearranjos do DNA que combinam diferentes segmentos gênicos. Um grande número de rearranjos podem ser utilizadas. Outro método, conhecido como diversidade juncional, é conseqüência da adição e substração de nucleotídeos nas junções entre os segmentos gênicos. Uma terceira possibilidade, um fator que potencializa a diversificação dos anticorpos, é a hipermutação somática que ocorre nos linfócitos B ativados. Esses processos são responsáveis por trazer diversidade à população de anticorpos e pelo refinamento da

afinidade com relação ao seu antígeno específico [83].

O principal objetivo da criação de diversidade entre os anticorpos é que as novas variações podem resultar em indivíduos de maior afinidade com o antigeno quando comparados ao original [83]. Os linfócitos capazes de produzir imunoglobulinas com poder de neutralização superior tendem a ser privilegiados no processo de seleção das células que irão se diferenciar. Esse processo é conhecido como maturação de afinidade.

Por conseqüência da alta taxa de mutação utilizada na hipermutação somática, a maioria das alterações produzidas gerariam imunoglobulinas de baixa afinidade, inválidas ou auto-reativas [84]. Ou seja, uma célula com afinidade antigênica alta, ao longo de sua reprodução, sob a mesma taxa de mutação, pode piorar essa afinidade pelo acúmulo de novas mutações ineficientes. Logo, acredita-se que esse processo seja composto por um curto pico de mutação seguido por um longo processo de seleção e expansão clonal [85]. Os procedimentos de seleção e clonagem devem preservar as células com alta afinidade. Esse processo é chamado de seleção clonal [86].

Segundo a teoria da seleção clonal [86], um antígeno de fora do corpo recrutaria um subconjunto de linfócitos capazes de reconhecê-lo. Aqueles com maior afinidade com o antígeno e menor afinidade com células do corpo serão selecionados para a clonagem em detrimento daqueles com menor poder de neutralização e maior reatividade com células próprias. Cada clone seria portador de um único tipo de receptor, ou seja, sua combinação não estaria presente em nenhuma outra célula. A Figura 3.12 exemplifica o processo de seleção clonal.

Na Figura 3.12, pode ser visto o funcionamento da seleção clonal maturando os linfócitos B e T. Primeiramente, as células-tronco hematopoiéticas (1), conhecidas por adquirir características de outras linhagens celulares dão origem aos linfócitos imaturos (2). Cada linfócito tem receptores diferentes dos demais. Aquelas que não reagem com as células do próprio corpo (3) amadurecem (4). As células maduras (ainda inativas) são expostas ao antígeno (5). Aquela com melhor afinidade é clonada [4]. Toda essa interação entre células, sejam elas do corpo ou não, é chamada de rede imunológica e teve sua teoria proposta por Jerne [87].

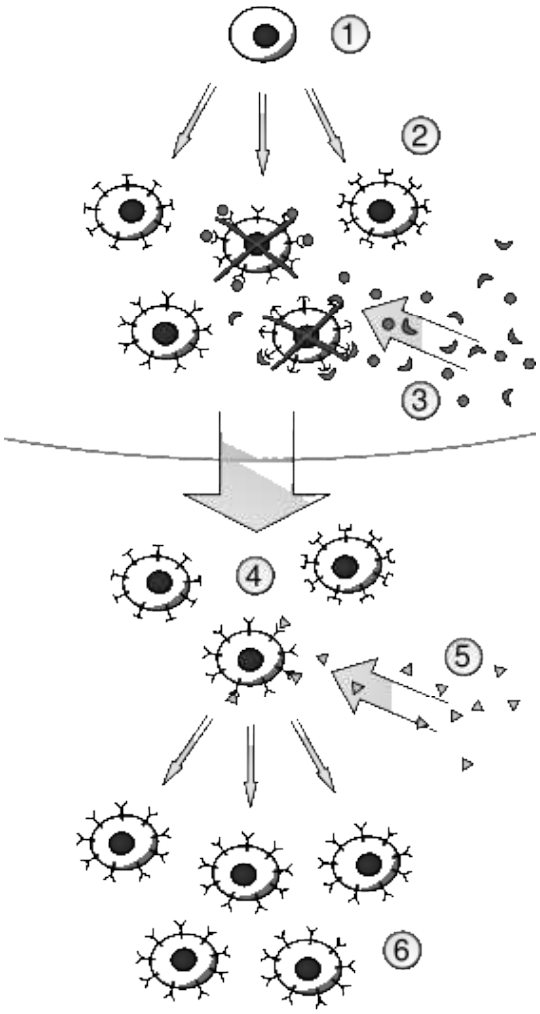

Figura 3.12: Diagrama que resume a seleção clonal dos linfócitos B e T [4].

3.7.2 Métodos Computacionais

Um novo ramo da teoria de sistemas inteligentes, denominado Sistemas Imunológicos Artificiais (SIA), tem surgido ao longo dos últimos anos e vem se consolidando através de workshops em diversas conferências desde o ano de 1996 [3]. Existem também diversas teorias e modelos computacionais com o objetivo de simular os diferentes componentes e mecanismos do sistema imunológico.

Em [88] mostra-se que os sistemas imunológicos artificiais são compostos por metodologias inteligentes, inspiradas no sistema imunológico biológico, para a solução de problemas do mundo real. Serão vistos aqui alguns processos computacionais inspirados na teoria do sistema imunológico natural. As características imune-biológicas mais importantes, neste caso, são: diversidade populacional, teoria da seleção clonal e da rede imunológica. Essas características são utilizadas na elaboração dos métodos SAND, CLONAG, ABNET e aiNET [3].

Dentre os algoritmos inspirados no SI, o método CLONALG [72, 89] será enfatisado, pois é um dos principais procedimentos imune-inspirados para resolução de problemas de otimização. Além disso, é parte importante nos algoritmos híbridos aqui apresentados.

SAND

O algoritmo SAND (*Simulated ANnealing approach for Diversity*) [90] é baseado na técnica de *Simulated Annealing* (Recozimento Simulado) [9]. O objetivo desse processo é obter uma população com uma ampla cobertura do espaço de busca sem a necessidade do conhecimento específico do problema. Este método pode ser utilizado em espaços de Hamming ou em espaços Euclidianos.

A característica imune simulada por este algoritmo é o crescimento da diversificação dos linfócitos. Seu objetivo é criar uma combinação de linfócitos não reagentes entre si e que consiga se ligar à maior quantidade de antígenos possível.

O *Simulated Annealing* se basea em como um metal se resfria e se arranja numa estrutura critalina de energia mínima (o processo real de *annealing* onde um material é aquecido e resfriado gradualmente) e a busca por um ótimo num sistema. Em cada iteração, o algoritmo procura na vizinhança da solução candidata corrente um

próximo ponto a ser investigado. A diferença entre os valores da função objetivo é utilizada como forma de seleção para um novo candidato. Aqui, a função objetivo ganha o nome de função de energia ou potencial. A meta-heurística objetiva a minimização da função de energia podendo, em algumas interações, aceitar vizinhos com um valor de potencial maior. A probabilidade de aceitar um candidato com valor da função potencial maior diminui com o tempo [9].

Pode-se utilizar o SAND em conjunto com outros métodos como redes neurais ou algoritmos evolutivos. Uma habilidade de destaque do sistema imunológico é a de gerar receptores antigênicos diversos, ou seja, os linfócitos são capazes de reconhecer uma grande quantidade de antigenos. O objetivo do SAND é diversificar uma população através da maximização de funções de energia baseadas na afinidade entre os membros da população de anticorpos. Pode ser utilizado na geração de condições iniciais para métodos de sistemas imunológicos artificiais, como conjunto de pesos iniciais para o treinamento de redes neurais artificiais, e para diversificar uma população num algoritmo evolutivo [3].

A cada iteração, como no *Simulated Annealing*, uma pequena perturbação é dada a uma solução candidata. O valor da variação entre as funções de energia é calculada (ΔE). Se $\Delta E < 0$, então a nova solução candidata substitui a anterior. Caso $\Delta E > 0$, um valor aleatório é comparado com a probabilidade de se aceitar essa nova opção. A probabilidade de um pior candidato ser aceito é calculada para T é a temperatura, como

$$P(\Delta E) = exp\left(\frac{-\Delta E}{T}\right) \tag{3.19}$$

No primeiro caso, o cálculo de energia é feito considerando o espaço de formas de Hamming. Para isso, define-se a distancia de Hamming como na Fórmula 3.7.2. A seguir, calcula-se F (distância de Hamming percentual entre todos os anticorpos) e D (percentual de indivíduos distintos), onde N é número de anticorpos de comprimento L. Dessa forma, a energia referente ao espaço de Hamming é calculada por

$$E = \frac{F + D}{2} \tag{3.20}$$

onde,

$$F = 100\frac{4}{LN^2}\sum_{i=1}^{N}\sum_{j=i+1}^{N} distanciaHamming(i, j)$$

e

$$D = 100\frac{1}{N}\sum_{i=1}^{N} s(i)$$

com

$$s(i) = \begin{cases} 1, & \text{se } anticorpo_i \neq anticorpo_j, \quad \forall j = 1, \ldots, N \\ 0, & \text{caso contrário} \end{cases}$$

$s(i)$ verifica se o anticorpo i é único e $distanciaHamming(i, j)$ calcula a distância de Hamming entre os indivíduos i e j.

O cálculo da distância de Hamming é dado por

$$distanciaHamming(i, j) = \sum_{k=1}^{L} d(i, j, k) \tag{3.21}$$

onde, $d(i, j, k) = 1$ se o alelo da posição k de um indivíduo i for diferente do alelo de mesma posição em j, $d(i, j, k) = 0$, em caso contrário.

Se um espaço de busca Euclidiano for utilizado, então o valor de energia a ser calculado considera a distância Euclidiana entre os indivíduos i e j.

$$E = \sum_{i=1}^{N}\sum_{j=i+1}^{N} distanciaEuclidiana(i, j) \tag{3.22}$$

onde a distância Euclidiana é dada por

$$distanciaEuclidiana(i, j) = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})_k^2} \tag{3.23}$$

onde, $x_{i,k}$ é o valor de cada variável x_k do indivíduo i e $x_{j,k}$ é o valor de cada variável x_k do indivíduo j.

A Figura 3.13 apresenta um pseudo-código para a execução do método SAND. Esse algoritmo baseia-se no apresentado em [3].

Para o algoritmo da Figura 3.13, primeiramente a energia da população corrente é calculada. Esse valor pode ser encontrado pelas Fórmulas (3.20) ou (3.22) dependendo do tipo de espaço utilizado. $energiaMax$ pode ser 100% se o espaço de formas utilizado for o de Hamming, porém, no caso Euclidiano, um outro critério de parada considerando a diversidade populacional deve ser usado. Algumas alternativas são apresentadas em [3].

```
1: procedimento SAND(nGeracoes, β, δ, energiaMax, populacao)
2:     GERAPOPULACAOINICIAL(populacao)
3:     energiaCorrente ← CALCULAENERGIA(populacao)
4:     iteracoesMesmaEnergia ← 0; α0 ← 1; T ← 0; geracao ← 0
5:     enquanto geracao < nGeracoes & energiaCorrente < energiaMax faça
6:         novaPopulacao ← HIPERMUTACAO(populacao, α)
7:         energiaNova ← CALCULAENERGIA(populacao)
8:         ΔE ← energiaCorrente − energiaNova
9:         se ΔE < 0 então
10:            populacao ← novaPopulacao; energiaCorrente ← energiaNova
11:            iteracoesMesmaEnergia ← 0
12:        senão
13:            se ΔE > 0 então
14:                p ← P(ΔE)
15:                numeroAleatorio ← NUMEROAOACASO(0, 1)
16:                se numeroAleatorio < p então
17:                    populacao ← novaPopulacao; energiaCorrente ← energiaNova
18:                    iteracoesMesmaEnergia ← 0
19:                fim se
20:            senão
21:                iteracoesMesmaEnergia ← iteracoesMesmaEnergia + 1
22:                se iteracoesMesmaEnergia ≥ δ = 0 então
23:                    T ← β × T; iteracoesMesmaEnergia ← 0; α ← α0
24:                fim se
25:                α ← β × α
26:            fim se
27:        fim se
28:        geracao ← geracao + 1
29:    fim enquanto
30: fim procedimento
```

Figura 3.13: Pseudo-código para o algoritmo SAND.

Posteriormente, uma estrutura de controle é iniciada. Se a variável *geracao* atingir o número máximo de gerações *nGeracoes* ou a energia corrente for menor que o valor limite para a energia, finaliza o processo. A população de anticorpos sobre hipermutação, ou seja, uma perturbação sob uma alta taxa de mutação e uma

nova população de anticorpos é gerada. A energia dessa nova população é calculada para que ΔE possa ser encontrado. Com o valor da variação de energia, então pode-se escolher entre as seguintes possibilidades:

- Se $\Delta E < 0$, a nova população é aceita;
- Se $\Delta E > 0$, então há uma probabilidade p da nova população ser aceita.
- Caso contrário, verifica-se qual o número de iterações em que o sistema está em estado estacionário e, caso seja maior que o limite definido em δ, a temperatura T é reduzida, a taxa de mutação α é tomada como seu valor inicial e a variável $iteracoesMesmaEnergia$ assume valor nulo.

CLONALG

Simulando o princípio de seleção clonal, o CLONALG (*CLOnal selection ALGorithm*, ou algoritmo de seleção clonal) originalmente proposto em [89] é um método utilizado em problemas de aprendizagem de máquina, reconhecimento de padrões e otimização [72]. O algoritmo trabalha com a idéia de que somente células com capacidade de reconhecer antígenos irão sobreviver e gerar descendentes. O processo é como a maturação do sistema imunológico natural. A diferença é que, ao contrário do SI, no presente procedimento todos os passos de evolução dos linfócitos são empregados apenas em anticorpos, pois aqui anticorpos e linfócitos não sofrem distinção. Logo, os anticorpos são selecionados, clonados e passam por um processo de mutação somática a altas taxas (hipermutação). Com esse processo espera-se gerar anticorpos com uma maior afinidade em relação ao antígeno em questão.

Para o problema de reconhecimento de padrões e aprendizado de máquina, seguindo a idéia de alguns imunologistas, há a necessidade de um conjunto de memória, composto pelos anticorpos do repertório que apresentem maior afinidade antigênica. A Equação 3.24 é um exemplo de como definir a quantidade de clones a serem gerados pelo i-ésimo anticorpo da população ordenada, segundo afinidade decrescente.

$$nClones = arredonda\left(\frac{\beta N}{i}\right) \tag{3.24}$$

onde, β é um fator multiplicativo, N é o número de anticorpos da população e *arredonda* é uma função que transforma seu argumento no inteiro mais próximo.

O pseudo-código na Figura 3.14 detalha o funcionamento do método. O algoritmo baseia-se no apresentado em [3]. Para começar, as variáveis são inicializadas. O algoritmo irá processar até o número de gerações máximas ser atingido ou a afinidade esperada ser obtida. Para cada antígeno a ser verificado, primeiro calcula-se as afinidades dos anticorpos da população em relação a ele. Com base nos valores obtidos, seleciona-se os *nSeleciona* melhores anticorpos para serem clonados. O número de clones para cada anticorpo selecionado é calculado pela Equação 3.24. Os clones gerados sofrem hipermutação e tem suas afinidades com o antígeno corrente calculadas. O anticorpo que melhor neutraliza o antígeno, ou seja, aquele com melhor afinidade antigênica é selecionado. Se esse anticorpo for melhor que aquele mais similar na população de anticorpos, então o indivíduo que estava na memória (população de anticorpos) é substituído. Concluindo, o *nSubstituidos* anticorpos são substituídos a fim de trazer diversidade ao sistema imunológico artificial. Essa troca pode ser feita por indivíduos da população de clones ou, mais comumente, gerados aleatoriamente.

O método apresentado pode ser facilmente modificado para resolução de problemas de otimização [3]. Ao invés de reconhecer uma população de antígenos, o objetivo é substituído por otimizar uma função f qualquer. A afinidade é calculada através dessa função. O sistema imunológico deve agir apenas contra esse antígeno. Outra diferença entre o processo apresentado previamente é que ao invés de selecionar apenas o melhor clone para compor a população de anticorpos, *nSeleciona* fazem esse papel.

O novo pseudo-código para execução do método CLONALG pode ser como apresentado na Figura 3.15. O processo, como pode ser observado, é muito semelhante ao da Figura 3.14.

Além do algoritmo da Figura 3.15, um diagrama descrevendo o fluxo de execução do algoritmo para resolver problemas de otimização pode ser visto na Figura 3.16.

O CLONALG é um dos métodos imune-inspirados mais utilizados para a resolução de problemas de otimização. Segundo Castro [3], o algoritmo pode ainda ser adaptado caso o objetivo seja encontrar multiplos ótimos da função. Neste caso, o número de indivíduos selecionados para a clonagem (*nSeleciona*) deveria ser igual ao tamanho da população, ou seja, a pressão de seleção seria ignorada para que

```
1: procedimento CLONALG(nGeracoes, β, nSeleciona, nSubstituidos, anticorpos, antigenos)
2:     geracao ← 0
3:     afinidadeGeral ← 0
4:     afinidadeMax ← CALCULAAFINIDADEMAXIMA(antigenos)
5:     numeroAntigenos ← TAMANHO(antigenos)
6:     enquanto geracao < nGeracoes&afinidadeGeral < afinidadeMax faça
7:         para i = 1 : numeroAntigenos faça
8:             afinidades ← CALCULAAFINIDADE(anticorpos, antigenos[i])
9:             selecionados ← SELECIONA(anticorpos, afinidades, nSeleciona)
10:            clones ← CLONA(selecionados, afinidades, β)
11:            clones ← HIPERMUTA(clones, afinidades)
12:            afinidades ← CALCULAAFINIDADE(clones, antigenos[i])
13:            melhorClone ← SELECIONA(clones, afinidades, 1)
14:            INSERE(melhorClone, anticorpos)
15:            SUBSTITUIRPOPULACAO(anticorpos, nSubstituidos)
16:        fim para
17:        afinidadeGeral ← SOMAAFINIDADES(anticorpos, antigenos)
18:        geracao ← geracao + 1
19:    fim enquanto
20: fim procedimento
```

Figura 3.14: Pseudo-código para o algoritmo CLONALG para problemas de reconhecimento de padrões ou aprendizado de máquina.

anticorpos com afinidades menores tivessem as mesmas chances de passarem pelo processo de maturação. Isso faria com que cada anticorpo fizesse uma busca local no espaço de busca [72]. Uma segunda alteração seria manter fixo o número de clones gerados por cada indivíduo da população de candidatos, não variando essa quantidade proporcionalmente à afinidade do anticorpo. Com isso, as chances evolutivas ficariam uniformes.

Como o algoritmo apresentado aqui tem o objetivo de encontrar um ponto ótimo

```
1: procedimento CLONALG(nGeracoes, β, nSeleciona, nSubstituidos, anticorpos)
2:     afinidades ← CALCULAAFINIDADE(anticorpos)
3:     para geracao = 0 : mGeracoes faça
4:         selecionados ← SELECIONA(anticorpos, afinidades, nSeleciona)
5:         clones ← CLONA(selecionados, afinidades, β)
6:         clones ← HIPERMUTA(clones, afinidades)
7:         afinidades ← CALCULAAFINIDADE(clones)
8:         novosAnticorpos ← SELECIONA(clones, afinidades, nSeleciona)
9:         INSERE(novosAnticorpos, anticorpos)
10:        SUBSTITUIRPOPULACAO(anticorpos, nSubstituidos)
11:    fim para
12:    afinidades ← CALCULAAFINIDADE(anticorpos)
13: fim procedimento
```

Figura 3.15: Pseudo-código para o algoritmo CLONALG para problemas de otimização.

de uma função e dada sua semelhança com os algoritmos evolutivos apresentados em seções anteriores, uma comparação entre esses métodos será apresentada. Enquanto os métodos evolutivos empregam um vocabulário inspirado na genética natural e são baseados na teoria da evolução neo-Darwiniana, o algoritmo de seleção clonal utiliza o espaço de formas e a terminologia imunológica para descrever as interações de antígenos e anticorpos, e utilizar a evolução celular no caso do processo de otimização [3].

Pode-se, com base nas similaridades discutidas, comparar os dois processos sob a perspectiva de cada um de seus passos.

No procedimento clonal, o processo de mutação é baseado numa probabilidade alta e inversamente proporcional à afinidade do anticorpo com os antígenos. Nos algoritmos evolutivos, a taxa de mutação é baixa e, normalmente, não varia de acordo com a aptidão dos indivíduos (apesar de essa estratégia ser permitida). Para esse último caso, a alteração genética tem como objetivo causar alterações nos indivíduos

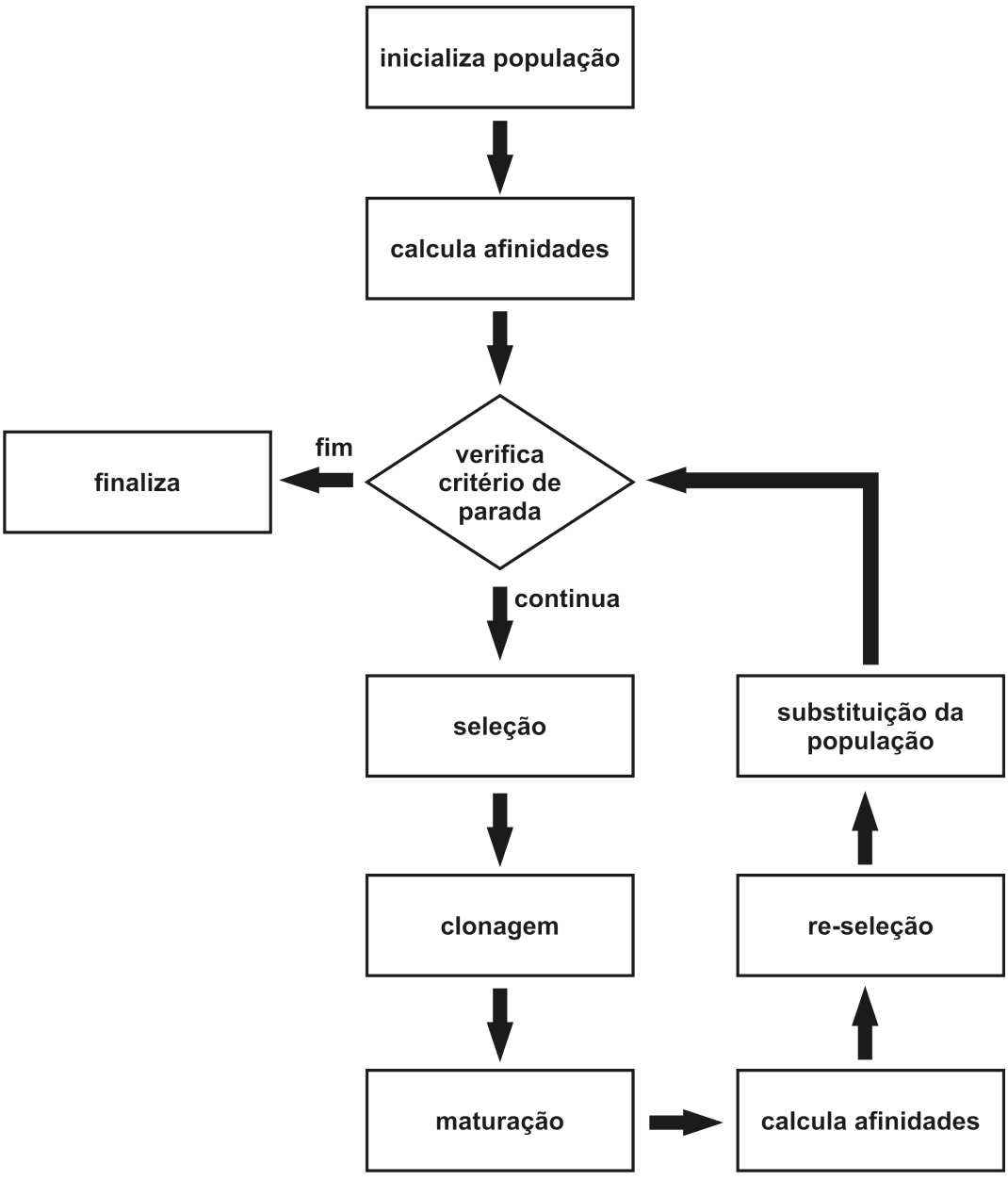

Figura 3.16: Diagrama de fluxo de execução para o algoritmo CLONALG.

à fim de, com isso, conseguir alguma pequena variação genética benéfica. No caso dos sistemas imunológicos artificiais, a idéia é evoluir um dado anticorpo à partir dessas alterações.

O processo de reprodução também é diferente. Nos algoritmos evolutivos, ela baseia-se no cruzamento entre indivíduos selecionados previamente. No caso do processo clonal, a reprodução é feita através da mitose (ou clonagem) dos indivíduos selecionados.

Comparando os algoritmos em relação à resolução de problemas de otimização tem-se, naturalmente mas não necessariamente, que os algoritmos evolutivos se propõem a resolver a otimização de uma função objetivo unimodal. Entretanto, os SIAs podem ser utilizados para problemas multimodais onde cada máximo local é um antígeno a ser relacionado com a população de anticorpos. Além disso, nos algoritmos evolutivos, apenas a melhor solução é encontrada. No processo clonal, vários máximos locais da função objetivo são apresentados como solução do problema.

Finalmente, a última e principal diferença entre os dois processos discutidos. Os algoritmos evolutivos foram originalmente propostas para solucionar problemas de otimização, enquanto o algoritmo clonal foi desenvolvido com o objetivo de implementar computacionalmente o princípio imunológico de seleção clonal para resolver problemas de reconhecimento de padrões e, só posteriormente, adaptado a problemas de otimização, focando no caso multimodal [3].

ABNET

A ABNET (*AntiBody NETwork*, ou rede de anticorpos) [91] é um algoritmo para treinamento de redes neurais booleanas e se fundamenta em vários princípios do sistema imunológico. Aprendizagem competitiva, geração automática da estrutura da rede e representação binária do conjunto de pesos são algumas de suas características que se destacam. O objetivo do método é explorar o espaço de formas de Hamming afim de determinar automaticamente uma arquitetura adequada para uma rede neural. Essa adequação do problema envolvia a experiência através de procedimentos de tentativa e erro. A arquitetura da rede baseia-se na idéia de que seus neurônios de saída e seus respectivos vetores de pesos representam os anticorpos. É a rede como um todo que faz o reconhecimento do antígeno. O princípio da seleção clonal

será utilizado como processo para evoluir a rede. Esse modelo é proposto para resolução de problemas de reconhecimento de padrões (especificamente binários) e clusterização.

O algoritmo trabalha com o princípio de que uma população de anticorpos deve reconhecer outra de antígenos, como o CLONALG visto na Seção 3.7.2. A população de anticorpos será modelada como uma rede neural booleana. Como os antígenos assim como os anticorpos são modelados em um espaço de formas de Hamming, então o cálculo da afinidade pode ser feito através da Equação 3.21. O anticorpo com maior afinidade antigênica é aquele que possuir a maior distância dada pela Equação 3.21 para um antígeno específico.

O processo é iniciado escolhendo-se um antígeno de forma aleatória, mas ponderando as probabilidades de seleção de acordo com a densidade de probabilidade na população de antígenos [3]. As afinidades dos anticorpos com este antígeno são calculadas. Aquele que melhor reconhecer o antígeno é selecionado e seu vetor de pesos é maturado. Posteriormente, o nível de concentração τ do anticorpo selecionado é incrementado. O anticorpo escolhido receconhece o antígeno selecionado, logo essa ligação é mantida num vetor $\overrightarrow{v}$. Para finalizar, o anticorpo mais estimulado é clonado e o pior deles, sob a mesma perspectiva, é candidato a ser eliminado.

O processo de maturação de afinidade é semelhante a uma mutação de múltiplos pontos com taxas de troca variáveis. As posições que não forem complementares são candidatas a serem alteradas pela hipermutação. A taxa de mutação (ou número de trocas) é definida pelo implementador.

Por conter características muito específicas de redes neurais artificiais, os processos de construção (clonagem) e poda (morte celular programada) da rede não serão discutidos aqui. Entretanto, o procedimento completo pode ser encontrado em [91].

Um diagrama com o processo do algoritmo pode ser visto na Figura 3.17.

aiNET

O aiNET (*Artificial Immune NETwork*, ou rede imunológica artificial) [71, 92] inspira-se na teoria da rede imunológica proposta por Jerne [87]. Suas principais características são: capacidade de descrever a estrutura interna dos antígenos (dados

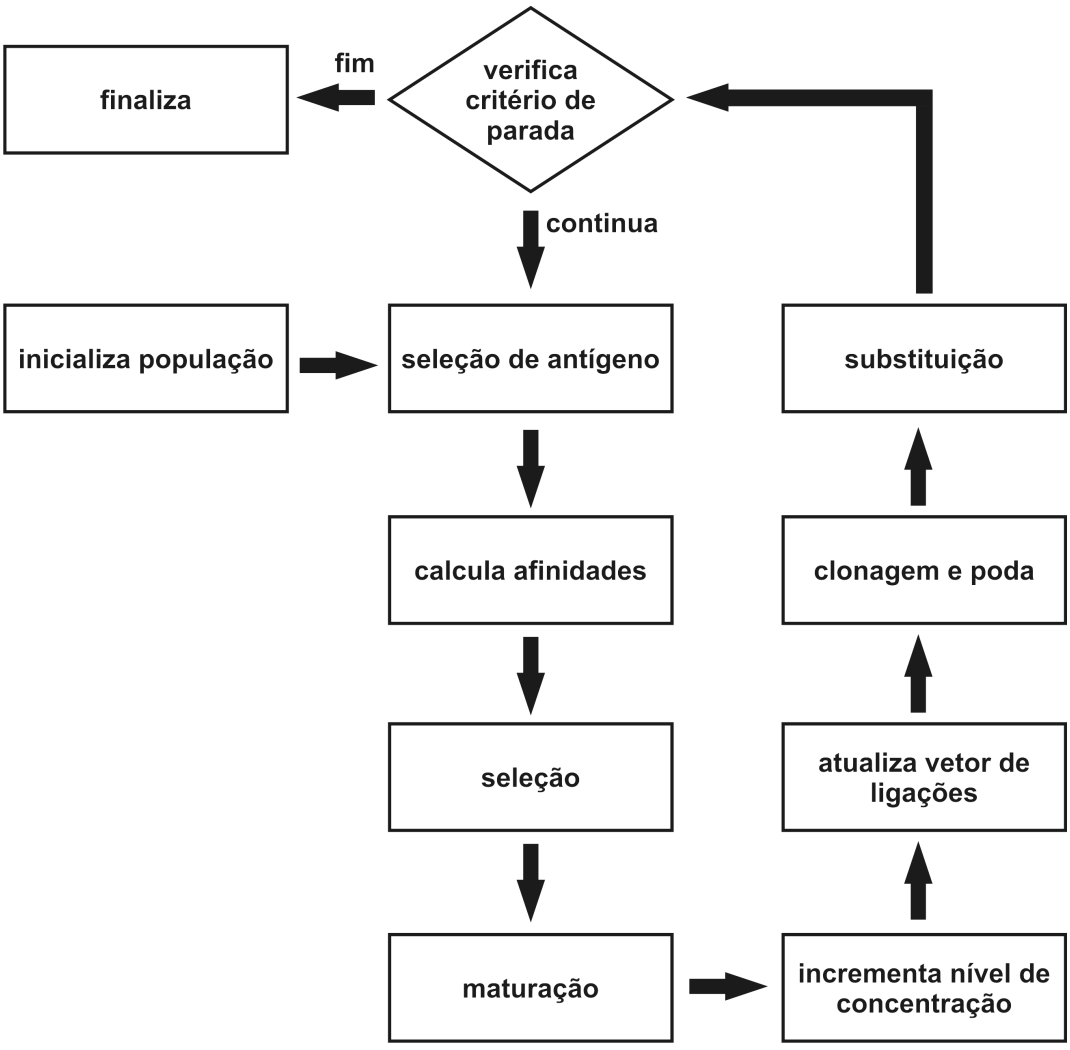

Figura 3.17: Diagrama de seqüência da execução do método ABNET.

de treinamento), seu perfil de distribuição de probabilidade e relações de vizinhança (clusters) [3]. O algoritmo CLONALG é utilizado como parte do processo de treinamento da rede. Ele irá controlar a quantidade e forma dos anticorpos que compõem a rede. Métodos de clusterização hierárquica e teoria dos grafos são utilizados para definir e interpretar a estrutura da rede. O aiNET pode ser aplicado a problemas de aprendizado de máquina, reconhecimento de padrões, compressão e clusterização [3].

Segundo Castro, este modelo computacional de rede imunológica pode ser descrito como

Definição 3.2 *A rede imunológica artificial, chamada aiNET, é um grafo com conexões ponderadas, não necessariamente totalmente interconectado, composto por um conjunto de nós, denominados anticorpos, e conjuntos de pares de nós chamados conexões, com um valor característico associado, chamado de peso da conexão ou simplesmente peso [3].*

O processo de execução do aiNET se divide em duas etapas: interação dos anticorpos com o antígeno e a relação entre os anticorpos da rede. Todas essas afinidades são calculadas através de medidas de similaridade. Segundo [3], o objetivo é utilizar métricas de distância para gerar uma população de anticorpos que melhor representem os antígenos a serem reconhecidos, sempre com um controle da cardinalidade do repertório. A afinidade entre anticorpo e antígeno é inversamente proporcional à distância entre eles. Quanto maior a afinidade anticorpo-antígeno, mais eficiente é a resposta imunológica. O aperfeçoamento do repertório é feito com base no algoritmo CLONALG 3.7.2. O reconhecimento mútuo entre a população da rede promove a supressão (eliminação) de indivíduos. Métodos estatísticos ou teoria dos grafos são utilizados para avaliar as conexões entre os componentes da rede.

A Figura 3.18 mostra um diagrama que descreve o algoritmo para o aiNET. Segundo essa imagem, etapas de execução devem ser feitas até que o critério de parada seja satisfeito. Em cada uma dessas interações e para cada antígeno algumas fases também devem ser cumpridas. Primeiramente, as afinidades entre os anticorpos e o antígeno são calculadas. Os *nSeleciona* anticorpos que melhor reconhecem o antígeno são selecionados. Esses são clonados gerando uma população de clones. Quanto maior a afinidade entre o anticorpo e o antígeno, maior é o número de clones

que este irá gerar. A população de clones passa pelo processo de hipermutação onde a taxa de mutação é inversamente proporcional à afinidade do anticorpo que o gerou. As afinidades entre os novos anticorpos (população de clones hipermutados) e o antígeno são avaliadas. Dessa população, os $\zeta\%$ dos melhores clones são re-selecionados e armazenados em uma matriz de memória de clones. A morte programada elimina todos os anticorpos dessa matriz cuja afinidade ao antígeno seja maior que um σ pré-definido. Esse limiar de supressão controla a especificidade dos anticorpos, a acurácia da clusterização e a plasticidade da rede [3]. Posteriormente, as afinidades entre os clones da matriz de memória são calculadas. Um de cada par que possuir afinidade menor que λ é eliminado (supressão clonal). Os clones que restarem do processo de maturação são incluídos na rede imunológica (anticorpos de memória).

Depois de executar o processo descrito para cada antígeno, as afinidades entre os anticorpos da rede devem ser avaliadas. A supressão da rede elimina os anticorpos de memória que possuírem afinidade inferior a λ. Um de cada par é removido. Para finalizar o ciclo e antes do critério de parada ser verificado, $nIncluidos$ novos anticorpos são incluídos (e não substituídos como nos algoritmos anteriores) na população de anticorpos.

Como vimos, o algoritmo aiNET é composto pelo CLONALG (para o problema de reconhecimento de padrões) mais a parte de atividade da rede imunológica. Além disso, dois passos supressivos acontecem: supressão clonal e da rede. Essa eliminação evita que anticorpos com alta afinidade entre si sobrevivam.

Segundo Castro e Von Zuben [92], vários critérios de convergência do algoritmo podem ser utilizados:

- Parar quando um número limite de iterações do algoritmo for executado;
- Interromper quando a dimensão da rede imunológica atingir um limite;
- Avaliar o erro médio entre os antígenos e as células de memória e limitar quando esse chegar a um valor pré-definido;
- Interromper quando a variação do erro médio calculado é nula durante um certo número de interações.

Análises mais específicas como extração de conhecimento da rede e comparações

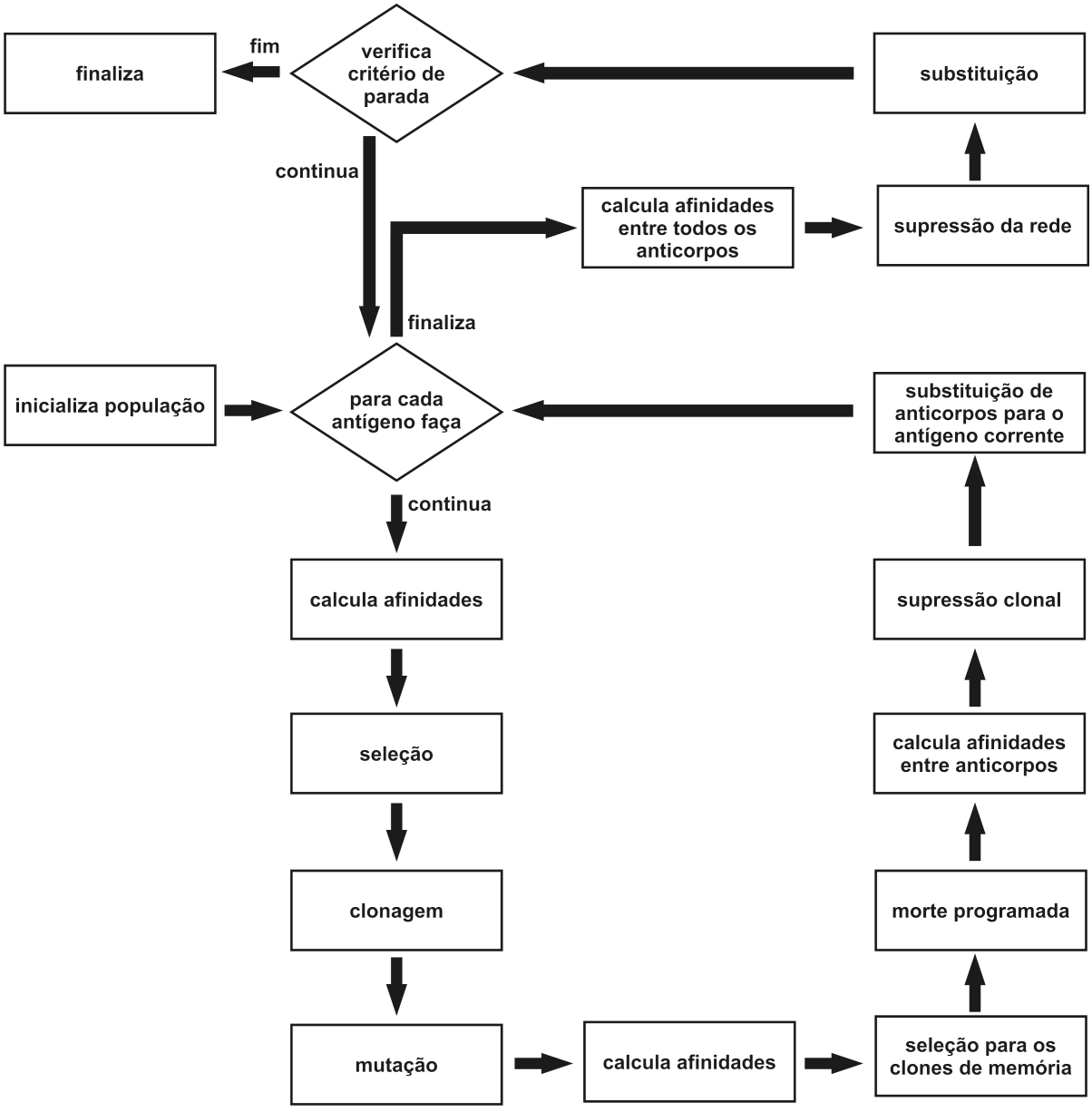

Figura 3.18: Diagrama de seqüência da execução do algoritmo aiNET.

com redes neurais artificiais não serão discutidas aqui por envolver conhecimento em clusterização, o que não é foco do presente trabalho. Entretanto, em [3] há uma análise envolvendo técnicas de clusterização hierárquica e gradual. Além disso, [92] apresenta similaridades, diferenças e comparações de resultados experimentais entre o aiNET e redes neurais.

Outros

Diversos métodos computacionais imune-inspirados foram elaborados. Porém, esses outros modelos são, normalmente, baseados em algum dos modelos apresentados aqui. Exemplos são os algoritmos opt-aiNET [70], copt-aiNET [78], RABNET[93] e omni-aiNET [69].

O primeiro e o último métodos são uma adaptação do algoritmo aiNET para resolução de problemas de otimização. O opt-aiNET trabalha internamente com a versão do CLONALG para otimização ao contrário do aiNET original, que usa a versão para reconhecimento de padrões e aprendizagem de máquina. A principal característica do opt-aiNET é ter uma população de anticorpos com tamanho variável. Isso é interessante para problemas de otimização multimodais já que variando o tamanho da rede automaticamente, o algoritmo consegue procurar uma maior variedade de mínimos locais, independente de sua quantidade. O número de mínimos locais (e globais) varia para cada problema.

Baseado no opt-aiNET, o omni-aiNET também é um método utilizado para problemas de otimização. A diferença entre os métodos é a adição de um mecanismo conhecido como *Gene Duplication* (ou duplicação gênica) [94–96].

O copt-aiNET é um método de otimização combinatorial, enquanto que o RABNET é um algoritmo baseado no ABNET que utiliza variáveis contínuas para treinamento de redes neurais.

Capítulo 4

Algoritmos Híbridos

Apesar da eficácia das meta-heurísticas, em alguns casos, os problemas podem não ser bem resolvidos apenas com a utilização do método e parâmetros corretos. Existem situações em que combinar características de vários algoritmos resulta em métodos mais eficientes. Os algoritmos híbridos consistem na combinação de procedimentos a fim de obter melhores resultados para os problemas propostos. Não existem regras para gerar as combinações. Elas podem ser feitas usando características de heurísticas, meta-heurísticas, outros métodos de resolução disponíveis.

Os algoritmos híbridos têm sido largamente utilizados na literatura [10–15, 35, 97–99] com o objetivo de melhorar o desempenho dos métodos previamente utilizados. Por exemplo, o presente trabalho apresenta alguns métodos em que combina-se um algoritmo genético com um sistema imunológico artificial a fim de resolver problemas de otimização com restrições.

Embora diversas combinações possam ser elaboradas, e vários experimentos já foram realizados, os estudos foram focados em métodos híbridos que combinam meta-heurísticas bio-inspiradas.

4.1 Classificação

Yen et al. [100] descreve quatro formas para classificar as formas híbridas de algoritmos populacionais inspirados na natureza (APiN):

- Híbridos *pipelining*. Nessa classe de algoritmo, uma meta-heurística é sequencialmente executado com outra técnica. Algumas vezes um APiN é processado primeiro e, posteriormente, uma busca local é executada na população final gerada. Diversas combinações seqüenciais podem ser elaboradas;

- Híbridos assíncronos. Esses métodos mantém uma população compartilhada entre um algoritmo e outra técnica de busca. Ambos procedimentos que compõem o algoritmo híbrido trabalham cooperativamente, modificando as soluções da população em comum;

- Híbridos hierárquicos. Um APiN e outro procedimento atuam em diferentes níveis no problema. Por exemplo, para uma rede neural um APiN pode ser utilizado para especificar a topologia da rede que é passada para outro algoritmo de otimização (back-propagation) que computa os pesos [101];

- Algoritmos com uso de operadores adicionais. Podem ser construídos pela inclusão de qualquer procedimento de busca como uma operador de movimento adicional que melhora um ou mais indivíduos pela sua aplicação. A população gerada pelo processo é inserida juntamente com o conjunto de filhos gerados pelos operados de movimento padrões do algoritmo.

O algoritmo proposto aqui integra um AG com um SIA. Como a execução do processo com ambos algorítmos é seqüêncial, então este pode ser classificado como um híbrido pipelining.

4.2 Inspiração

Entre 1995 e 1999, Hajela, juntamente com outros co-autores, trabalharam com a idéia de usar um algoritmo genético interno a outro AG com o objeto de aumentar a afinidade (ou reduzir a distância) entre indivíduos infactíveis (anticorpos, como será visto nas próximas seções) e factíveis (antígenos) [12–15]. O AG interno utiliza uma distância genotípica (Hamming) como aptidão com o objetivo de mover indivíduos infactíveis para a região factível. Ou seja, para o fim proposto, não há necessidade de cálculos adicionais da função objetivo do problema, que normalmente, gastam

muito mais tempo de processamento. As chamadas à função objetivo são feitas apenas pelo algorítmo genético externo.

Posteriormente, Coello and Cruz-Cortés[10, 11] publicaram trabalhos baseados na idéia dos algoritmos de Hajela. Apesar de se basear na teoria imunológica, o processo interno de aproximação dos indivíduos infactíveis da região factível ainda era executado via um AG. Foram feitos experimentos em diversos problemas e uma nova variação, utilizando computação paralela, foi apresentada.

Nas seções seguintes descrevem-se os híbridos AG-SIA desenvolvidos ao logo da tese.

4.3 AG-SIA - Primeira versão

O algoritmo híbrido proposto em [11] utiliza um algoritmo genético para otimizar a solução de problemas e, internamente, outro AG para manter e tornar factíveis essas soluções candidatas. A primeira versão do AG-SIA [98] é semelhante às inspirações citadas anteriormente. Uma das diferenças é a utilização do algorítmo CLONALG (imune-inspirado), ao invés do AG, para movimentar os indivíduos infactíveis.

Esta proposta combina algoritmos genéticos com sistemas imunológicos artificiais. O AG trabalha na função objetivo e o SIA procura, caso necessário, trazer os indivíduos infactíveis para a região factível do problema. Essa etapa do algoritmo também é desenvolvida utilizando um algoritmo genético, porém inspirado no sistema imunológico.

Como na maioria dos problemas evolutivos, uma população inicial é aleatoriamente gerada. Provavelmente, mas não obrigatoriamente, esta população será uma mistura de indivíduos factíveis e infactíveis.

Se existirem indivíduos factíveis e infactíveis na população, então dividi-se esta em dois grupos separados: um de indivíduos factíveis (antígenos) e outro com infactíveis (anticorpos). O objetivo com isso é fazer os anticorpos se aproximarem dos antígenos, ou melhor, aproximar os indivíduos infactíveis dos melhores factíveis até que eles entrem na região factível do problema.

Caso existam apenas indivíduos infactíveis na população, então o indivíduo "menos infactível", ou seja, aquele que menos violar as restrições do problema, funcionará como antígeno para o algoritmo. Outra estratégia possível é gerar indivíduos até algum factível ser encontrado. Porém esse método foi desconsiderado pela dificuldade (na maioria das vezes) em se encontrar um indivíduo factível aleatoriamente.

Com a população dividida entre antígenos e anticorpos, o sistema imunológico artificial começa a operar. Seu objetivo é aproximar os indivíduos infactíveis dos factíveis.

A aptidão de todos os anticorpos (indivíduos infactíveis) é inicializada com zero. Essa ação não trará nenhum problema ao algorítmo pois, como veremos, a aptidão não é considerada em indivíduos infactíveis.

Os indivíduos são clonados e hipermutados, operações evolutivas padrões do método CLONALG. Como sugerido em [72], o número de clones é mantido fixo com o objetivo de obter resultados mais diversificados, já que este método é comumente utilizado para problemas de otimização multimodal. Depois disso, os anticorpos gerados têm suas afinidades calculadas. Para isso, compara-se cada um dos elementos do conjunto de anticorpos com os antígenos. O resultado de cada uma dessas comparações é representado pela variável z. Essa relação representa a distância entre os dois indivíduos (anticorpo e antígeno) e é calculada com base na distância de Hamming segundo

$$z = \sum_{i=1}^{L} w_i, \tag{4.1}$$

onde L é o tamanho do cromossomo, $w_i = 1$ se o alelo da posição i de um dado anticorpo for diferente do alelo de mesma posição do antígeno e $w_i = 0$, caso contrário.

Os anticorpos com maior afinidade antigênica são selecionados para compor a população na próxima geração. As afinidades são calculadas pela soma dos valores de z dos anticorpos para cada antígeno. O algoritmo volta ao passo onde as aptidões de todos os anticorpos são inicializadas com zero. Essas operações são realizadas por um número determinado de iterações.

Finalizando o sistema imunológico artificial, os antígenos e anticorpos voltam à população de indivíduos do algoritmo genético externo. O algoritmo agora passa a operar como um algoritmo genético normal, passando pelas fases de seleção, cruza-

mento e mutação. Esta proposta usa seleção por torneio binário, ou seja, dois indivíduos são aleatoriamente escolhidos da população de anticorpos e, aquele com maior aptidão, será utilizado como um dos pais para a operação de crossover que resultará em dois indivíduos. Porém, o presente algoritmo não utiliza uma seleção por torneio comum. Ao invés disso, uma alternativa é sugerida. Ao invés de dois indivíduos serem selecionados aleatoriamente, apenas um é escolhido dessa forma. O outro é obtido seguindo a seqüência de cada um dos indivíduos da população. Ou seja, o torneio começa com o primeiro indivíduo da população disputando com um indivíduo obtido aleatoriamente. Depois disso, o segundo irá competir com mais um selecionado ao acaso, e assim por diante. Com isso, garante-se que apesar da seleção ser por torneio, todos os indivíduos da população tem chance de serem escolhidos, já que todos os indivíduos participarão de pelo menos uma das disputas.

Os filhos gerados sofrem mutação com probabilidade de um alelo alterado a cada variável. Essa taxa $p_m = \frac{1}{k}$, onde k é o tamanho (em bits) de cada variável dos indivíduos da população, segue que, em média, há a mudança de um bit para cada variável.

Os filhos resultantes passam a compor a nova população.

Um ponto importante a ser considerado é a seleção por torneio binário do algoritmo genético, que segue a seguinte regra para dois indivíduos sorteados da população:

- Se um dos indivíduos for factível e o outro infactível, vence o indivíduo factível;
- Se ambos os indivíduos forem factíveis, vence o indivíduo com melhor aptidão;
- Se ambos os indivíduos forem infactíveis, ganha o indivíduo menos infactível, ou seja, aquele que violar menos as restrições propostas.

Os indivíduos resultantes dos processos de seleção, cruzamento e mutação, constituirão a nova população. O algoritmo volta ao passo em que se divide a população entre indivíduos factíveis e infactíveis ou antígenos e anticorpos.

O algoritmo itera até encontrar uma solução ótima para o problema proposto ou ao atingir um número limite de gerações.

4.3.1 Extensão para o Caso de Números Reais

O algoritmo apresentado em [98] utiliza a distância de Hamming para calcular a aptidão dos anticorpos do sistema imunológico artificial. Esse valor, obtido através de comparações entre os genótipos dos indivíduos, é interessante pois pode ser utilizado em qualquer problema codificado em binário.

Entretanto, outros cálculos de distância podem ser utilizados. Para o caso de variáveis reais, por exemplo, pode-se usar a distância entre os pontos. Uma norma viável para problemas com variáveis reais seria a distância euclidiana, também apresentada na referência [98]. Assim, o valor de z é calculado como

$$z = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.2}$$

onde x e y são os fenótipos de dois indivíduos e n a dimensão do problema.

4.4 AG-SIA - Segunda versão

Uma segunda versão do algoritmo híbrido foi proposta em [99]. O método é muito semelhante ao apresentado na Seção 4.3. A diferença entre os algoritmos está no processo de troca da população de anticorpos. No procedimento apresentado, os anticorpos com maior afinidade são selecionados para compor a nova população. Na referência [99], a renovação da população de anticorpos seleciona o melhor anticorpo entre cada anticorpo e seus clones hipermutados. Esse processo aumenta a diversidade dos anticorpos gerados.

Além disso, como visto, o SIA pode utilizar diversas formas de cálculos para a afinidade entre os anticorpos (indivíduos infactíveis) e os antígenos (indivíduos factíveis). Um estudo comparando diversas formas de cálculos de afinidade pode ser encontrados em [99]. Nesse trabalho são apresentados resultados utilizando distância euclidiana, de Hamming e variações dessa.

Por causa do baixo custo computacional, por obter bons resultados e por ser uma norma mais geral, podendo ser utilizada sem nenhuma adaptação em diversos problemas, os experimentos foram feitos utilizando a distância de Hamming. Este algoritmo será representado no Capítulo 5 por AG-SIA^E.

O pseudo-código deste algoritmo pode ser visto nas Figuras 4.1 e 4.2 [35]. O

```
1: procedimento AG-SIA(nGeracoesAG,nIteracoesSIA)
2:     computaAptidaoViolacao(populacao)
3:     para i = 1 : nGeracoesAG faça
4:         separa(populacao, anticorpos, antigenos)
5:         para j = 1 : nIteracoesSIA faça
6:             clona(anticorpos, temp)
7:             mutacao(temp)
8:             calculaDistancias(antigenos, temp)
9:             selecionaMelhores(temp, anticorpos)
10:        fim para
11:        uniao(anticorpos, antigenos, populacao)
12:        selecaoPorTorneio(populacao, temp)
13:        crossover(temp)
14:        mutacao(temp)
15:        computaAptidaoViolacao(temp)
16:        substituiPopulacao(populacao, temp)
17:    fim para
18: fim procedimento
```

Figura 4.1: Pseudo-código para o AG-SIA.

diagrama da Figura 4.3 mostra o algoritmo de forma geral para um melhor entendimento.

4.5 AG-SIA - Terceira versão

Em [35], foi apresentada uma terceira versão do algoritmo híbrido. O novo método utiliza um procedimento chamado *Clearing* que foi proposto por Petrowski em [102]. O procedimento de *clearing*, limpeza ou remoção, originalmente utilizado para se resolver problemas de otimização multimodal, é um método de nicho inspirado pelo princípio do compartilhamento de recursos limitados entre indivíduos

```
1: função CLONA(anticorpos, temp)
2:     temp ← anticorpos
3:     para i = 1 : numClones faça
4:         para j = 1 : anticorpos.size faça
5:             INSERE(temp, anticorpos[j])
6:         fim para
7:     fim para
8: fim função
9: função SELECIONAMELHORES(temp, anticorpos)
10:    REMOVETODOS(anticorpos)
11:    para i = 1 : temp.size faça
12:        INSERE(tmp, temp[j])
13:        se i mod numClones = 0 então
14:            PEGAMELHOR(tmp, anticorpo)
15:            INSERE(anticorpos, anticorpo)
16:            REMOVETODOS(tmp)
17:        fim se
18:    fim para
19: fim função
20: função SELECAOPORTORNEIO(populacao, temp)
21:    REMOVETODOS(temp)
22:    para i = 1 : populacao.size faça
23:        ALEATORIO(r)
24:        PEGAMELHOR(populacao[i], populacao[r], melhor)
25:        INSERE(temp, melhor)
26:    fim para
27: fim função
```

Figura 4.2: Funções auxiliares do AG-SIA.

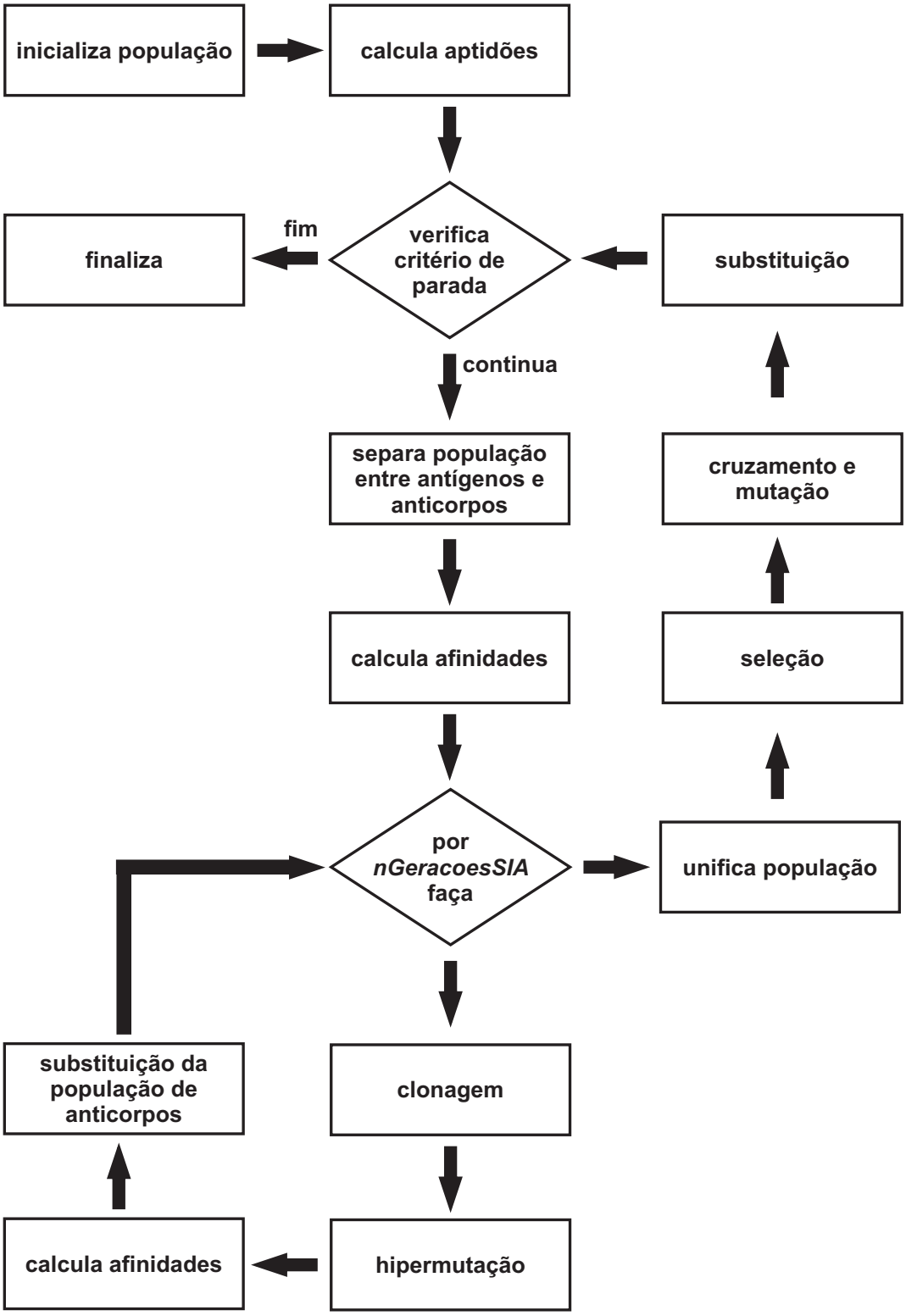

Figura 4.3: Diagrama do fluxo de execução do método híbrido.

de sub-populações caracterizadas por alguma similaridade [37]. O processo entrega todos os rescursos de cada sub-população para seu melhor indivíduo. Segundo [37], o procedimento é normalmente aplicado depois dos cálculos de aptidão (e neste caso, da violação) e antes que os operadores de seleção sejam aplicados. Os indivíduos são ordenados do melhor para o pior e todos aqueles dentro de uma distância crítica de cada pivô têm suas aptidões zeradas. O pivô é o melhor indivíduo não removido na seqüência ordenada. O processo continua até que todas as soluções sejam consideradas, ou seja, ou o indivíduo foi considerado como pivô em alguma iteração ou foi removido.

Diferentemente do que foi proposto em [37], o processo de *clearing* utilizado aqui é aplicado na substituição da população corrente pela população de filhos. Um novo conjunto de indivíduos é gerado unindo ambas populações (corrente e de filhos). O método é executado nesse novo conjunto. Além disso, os valores das aptidões não são zerados, como na referência original. Ao invés disso, os indivíduos que seriam removidos são marcados. A nova população (aquela que passará para a próxima geração) é composta pelos indivíduos não marcados. Caso seja necessário, para manter o tamanho da população constante, essa nova população é completada com os melhores indivíduos marcados gerados pelos processos de cruzamento e mutação.

Segundo [37], este procedimento quando aplicado sozinho não produz bons resultados. Com o objetivo de manter os nichos o operador de *crossover* é aplicado a indivíduos similares, ou seja, que tem grande possibilidade de estarem no mesmo nicho.

O restante do algoritmo permanece inalterado. A Figura 4.4 apresenta o pseudo-código para o método de *clearing* utilizado. Esse algoritmo é apresentado nas tabelas de resultados do Capítulo 5 como AG-SIA^C.

4.6 AG-SIA - Quarta versão

A quarta versão do algoritmo híbrido [103] é a que mais difere das demais. Apesar dos bons resultados obtidos pelos métodos apresentados em [35, 98, 99], a perda de diversidade é visívelmente excessiva. Com o objetivo de manter (ou até mesmo aumentar) a diversidade da população de soluções, um novo algoritmo AG-SIA é

```
1: função SUBSTITUIPOPULACAO^C(populacao, temp)
2:     UNIAO(populacao, temp, tmp); ORDENA(tmp)
3:     para i = 1 : tmp.size faça
4:         para j = i + 1 : tmp.size faça
5:             se not EMARCADO(tmp[j]) então
6:                 CALCULADISTANCIA(tmp[i], tmp[j], d)
7:                 se d < criticalDistance então
8:                     MARCA(tmp[j])
9:                 fim se
10:            fim se
11:        fim para
12:    fim para
13:    REMOVETODOS(populacao)
14:    para i = 1 : tmp.size faça
15:        se not EMARCADO(tmp[i]) então
16:            se temp.size! = populacao.size então
17:                INSERE(populacao, tmp[i])
18:            fim se
19:        fim se
20:    fim para
21:    ORDENA(temp); i ← 1
22:    enquanto temp.size! = populacao.size faça
23:        se EMARCADO(tmp[i]) então
24:            INSERE(populacao, temp[i])
25:            i ← i + 1
26:        fim se
27:    fim enquanto
28: fim função
```

Figura 4.4: Função de mudança de população utilizando *clearing*.

proposto. A nova técnica divide a população entre indivíduos factíveis e infactíveis, como nas versões anteriores. Depois disso, um algoritmo genético é utilizado para otimizar os indivíduos factíveis considerando apenas os valores da função objetivo (o que é trivial, já que essas soluções não violam nenhuma das restrições). Enquanto isso, os indivíduos infactíveis são operados por um sistema imunológico artificial com o objetivo de minimizar os valores de suas restrições. O SIA utilizado aqui é o CLONALG, apresentado na Seção 3.7.2.

Posteriormente, a população gerada pelo AG é verificada. Caso sejam localizadas soluções que violem alguma restrição, essas são movidas para a população do SIA. De forma similar, os indivíduos factíveis encontrados pelo CLONALG são transferidos para a população do AG, finalizando a iteração. Em gerações onde o número de indivíduos factíveis é ímpar, a pior dessas soluções é movida da população do AG para a do SIA. Isso é feito apenas com o objetivo de facilitar a aplicação dos operados evolutivos do algoritmo genético.

O número total de indivíduos (população do AG mais população do SIA) é mantido fixo. A quantidade de soluções em cada uma das populações varia de acordo com quantos indivíduos factíveis/infactíveis são criados a cada geração.

Um pseudo-código para o algoritmo híbrido pode ser visto na Figura 4.5.

Na Figura 4.5, a função SEPARA move os indivíduos para as populações de acordo com sua factibilidade, a função PEGAPIOR retorna o pior indivíduo da população, a função INSERE inclui um conjunto de indivíduos de uma população em outra e a função SUBSTITUI troca todos os indivíduos de uma população pelo de outra. As funções AG (que opera nos indivíduos factíveis) e SIA (que trabalha as soluções infactíveis) são melhor detalhadas nas Figuras 4.6 e 4.7, respectivamente.

Um AG simples é utilizado para otimizar os indivíduos factíveis. Os indivíduos são selecionados utilizado um torneio binário. O torneio é o mesmo apresentado na Seção 4.3. Ele garante que todos os indivíduos participarão de pelo menos um torneio, reduzindo o ruído no processo de seleção. Depois disso, são aplicados os operados de recombinação genética e mutação a fim de se gerar uma nova população. O operador de cruzamento utilizado aqui é o *crossover* uniforme.

A Figura 4.6 tráz um pseudo-código para o AG descrito aqui (utilizado na Linha 9 no algoritmo da Figura 4.5).

```
1: procedimento AG-SIA(nGeracoes)
2:     computaAptidaoViolacao(pop)
3:     separa(pop, popFactiveis, popInfactiveis)
4:     para i = 1 : nGeracoes faça
5:         se modulo(popFactiveis, 2)!=0 então
6:             piorIndividuoFactivel ← pegaPior(popFactiveis)
7:             insere(piorIndividuoFactivel, popInfactiveis)
8:         fim se
9:         AG(popFactiveis, novaPopAG)
10:        SIA(popInfactiveis, novaPopSIA)
11:        separa(novaPopAG, popFactiveis, novaPopInfactiveis)
12:        separa(novaPopSIA, novaPopFactiveis, popInfactiveis)
13:        substitui(novaPopFactiveis, popFactiveis)
14:        substitui(novaPopInfactiveis, popInfactiveis)
15:    fim para
16: fim procedimento
```

Figura 4.5: O algoritmo para a quarta versão do hibrido AG-SIA.

```
1: procedimento AG(popFactivel, novaPopAG)
2:     selecaoPorTorneio(popFactivel, temp)
3:     cruzamento(temp)
4:     mutacao(temp)
5:     computaAptidaoViolacao(temp)
6:     substituiPop(popFactivel, temp)
7:     insere(temp, novaPopAG)
8: fim procedimento
```

Figura 4.6: AG utilizado quarta versão do algoritmo híbrido.

Para a Figura 4.6, tem-se que a função selecaoPorTorneio (vista na Seção 4.3) seleciona os indivíduos para os operados evolutivos, que são considerados nas funções cruzamento e mutacao.

O SIA adotado aqui é baseado no CLONALG proposto por Castro [70, 72]. A população é ordenada e depois clonada. Como já foi visto na Seção 3.7.2 e segundo Castro, para um SIA multimodal o número de clones deve ser o mesmo para todos os anticorpos. Essa escolha ajuda a aumentar a diversidade da população. Logo após a clonagem a hipermutação é aplicada. A taxa de mutação é diretamente proporcional à posição do anticorpo na população ordenada. O menor valor para essa taxa mutação é o valor utilizado no AG e cresce até uma taxa máxima definida. Os novos anticorpos são avaliados (cálculos de função objetivo e violações) e os melhores entre cada anticorpo original e seus clones formarão a nova população. Esse processo de sobrevivência consiste em um torneio onde o vencedor é o indivíduo que menos violar as restrições. Todavia, há casos em que indivíduos factíveis e infactíveis serão comparados: um número ímpar de indivíduos nas populações faz com que o pior indivíduo factível entre no conjunto de soluções do SIA ou uma hipermutação em um dos clones de um anticorpo pode gerar uma solução factível. Nesses casos, a seleção do melhor indivíduo segue a regra do torneio apresentada na Seção 4.3.

Um pseudo-código para o SIA utilizado aqui é apresentado na Figura 4.7 (referenciado na Figura 4.5). A Figura 4.8 mostra um fluxograma de execução para a quarta versão do algoritmo híbrido. A função HIPERMUTA aplica o operador de mutação a uma taxa diretamente proporcional à posição do anticorpo na população ordenada.

1: **procedimento** SIA($popInfactiveis, novaPopSIA$)
2: ORDENA($popInfactiveis$)
3: CLONA($popInfactiveis, temp$)
4: HIPERMUTA($temp$)
5: COMPUTAAPTIDAOVIOLACAO($temp$)
6: $novaPopSIA \leftarrow$ PEGAMELHORES($popInfactiveis, temp$)
7: **fim procedimento**

Figura 4.7: SIA utilizado no algoritmo híbrido.

As principais diferenças entre os AG-SIA apresentados (Seções 4.3, 4.4 e 4.5) e este são: a forma como as afinidades dos anticorpos são calculadas e a complexidade computacional do algoritmo. Nos métodos anteriores, o valor da afinidade se baseava

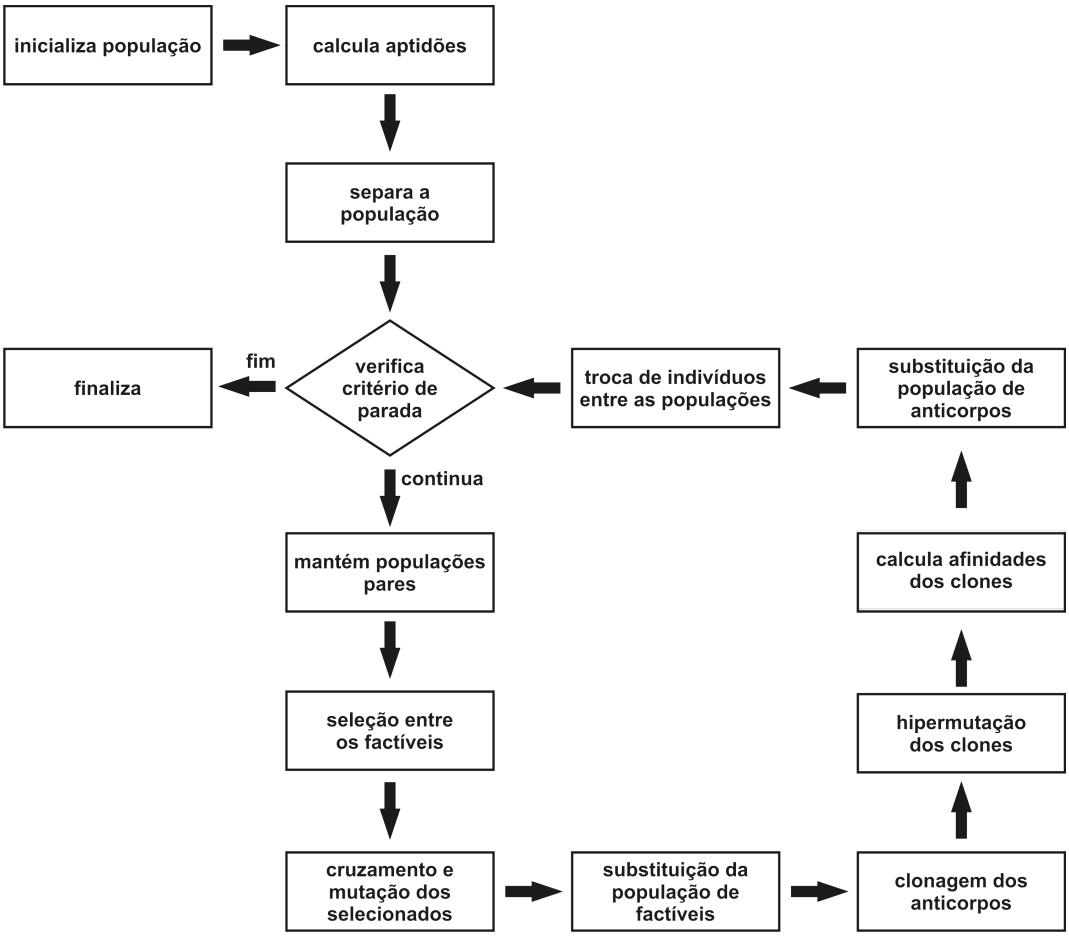

Figura 4.8: Diagrama do fluxo de execução da quarta versão do AG-SIA.

na similaridade entre os indivíduos factíveis (antígenos) e infactíveis (anticorpos). Aqui, a afinidade é definida como a soma das violações das restrições. Antes, para cada geração do algoritmo genético, $nIteracoesSIA$ iterações internas do SIA eram percorridas. No AG-SIA desta seção ambas meta-heurísticas trabalham seqüencialmente: uma iteração do AG e outra do SIA para cada geração global. Além disso, o método de *clearing* aumenta ainda mais a complexidade. Um algoritmo com este método requer o cálculo das distâncias entre os indivíduos a cada geração. Mais ainda, o operador de cruzamento foi aplicado apenas entre soluções similares. Esse processo melhora os resultados encontrados pelo algoritmo com o método de *clearing*, entretanto aumenta a complexidade computacional.

Uma outra vantagem indireta do algoritmo pode ser observada: sua fácil adaptação ao paralelismo. O AG e o SIA podem operar independente um do outro, apenas se sincronizando ao final de cada geração (para a troca dos indivíduos). A Figura 4.9 mostra um fluxograma de execução para a quarta versão do algoritmo híbrido proposto. A programação distribuída é de fundamental importância para resolução de grandes problemas. Entretanto, esse tipo de implementação não será abordada no presente trabalho mas comentada como uma possibilidade de trabalho futuro.

O algoritmo híbrido apresentado nessa seção é referenciado por AG-SIA nas tabelas de resultados do Capítulo 5.

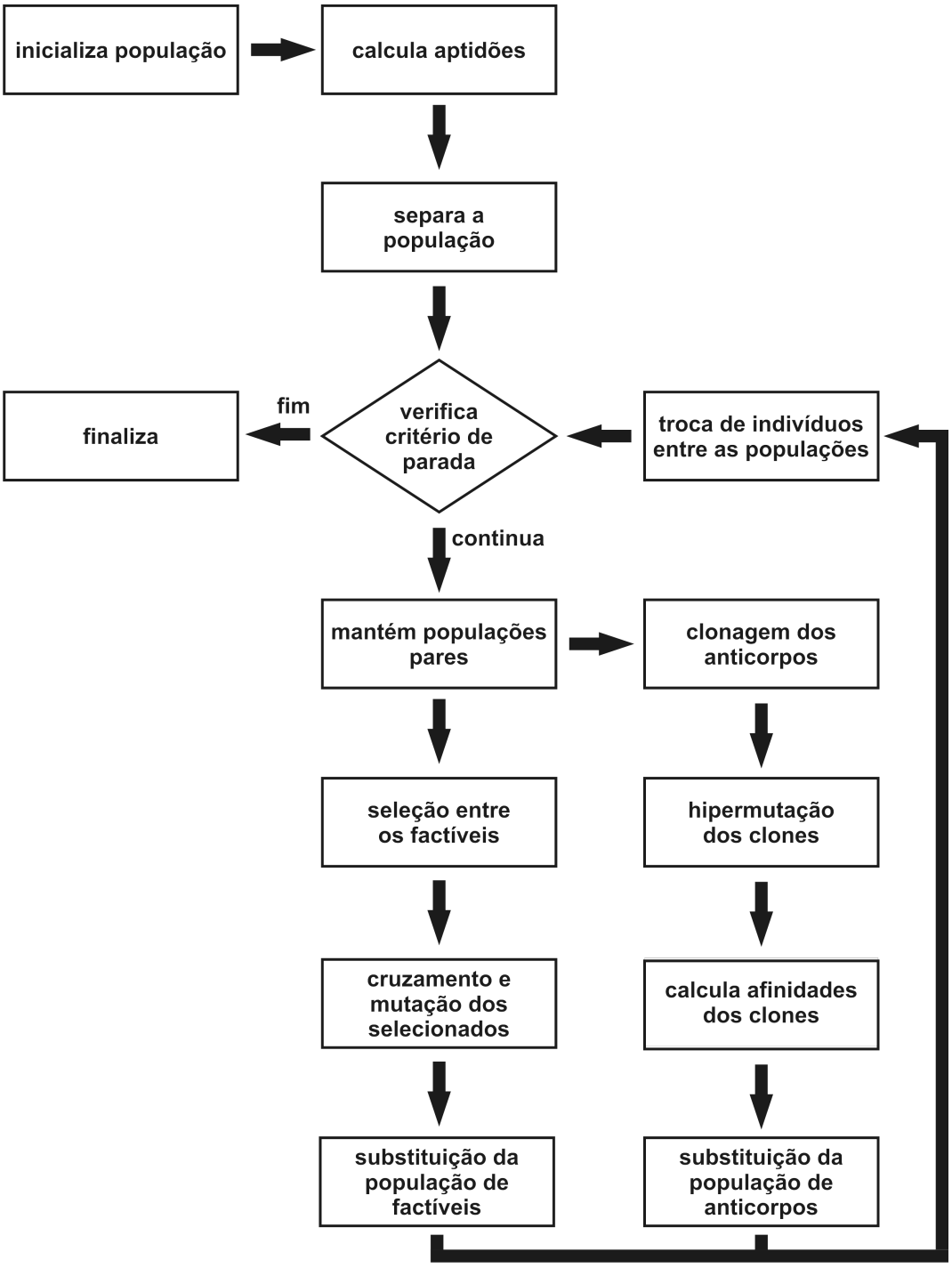

Figura 4.9: Diagrama do fluxo de execução da quarta versão do AG-SIA adaptado à execução paralela.

Capítulo 5

Experimentos Computacionais

Diversos experimentos foram realizados com o objetivo de testar as várias versões de algoritmos propostos (Capítulo 4) e compará-los com outros métodos de resolução de problemas com restrições existentes na literatura. Os testes abrangeram problemas de otimização em matemática, química, engenharia mecanica e estrutural e problemas de confiabilidade.

Para todos os problemas tratados serão apresentadas tabelas comparando os desempenhos dos algoritmos propostos com resultados disponíveis na literatura. Os melhores resultados serão marcados em negrito.

5.1 Implementação

Os algoritmos foram implementados utilizando a linguagem Java. Apesar de não ser muito difundida entre aplicações científicas, a linguagem Java tem grande destaque em sistemas comerciais. Sua principal característica, a de ser multiplataforma, permite que seus programas possam ser executados em qualquer dispositivo que tenha uma JVM (*Java Virtual Machine*) instalada. Esta facilidade de desenvolvimento e muitas outras características importantes foram o que levaram a implementação desse trabalho em Java.

Como para a maioria dos problemas práticos envolvendo otimização apresenta um custo computacional muito grande no cálculo da função objetivo e/ou das res-

trições, as comparações feitas na literatura envolvem, na maioria das vezes, medidas relacionadas ao número necessário dessas avaliações. Assim, nenhuma comparação referente ao tempo de processamento dos experimentos foi realizada. Além disso, a linguagem Java é conhecida por ter seu tempo de execução mais alto que outras como C, C++ e FORTRAN. Como a velocidade de processamento não era um ponto a ser medido, não havia problemas com a liguagem de programação adotada.

Todos os testes realizados utilizaram o mesmo conjunto de parâmetros. Isso é interessante pois mostra a eficiência dos algoritmos nos mais diversos problemas, com características variadas. O mesmo programa foi utilizado para resolver problemas de otimização com variáveis discretas, contínuas e mistas. Os parâmetros utilizados pelos AG-SIA^E (Seção 4.4) e AG-SIA^C (Seção 4.5) foram:

- Tamanho da população: 20 indivíduos;
- Taxa de mutação (AG e SIA): $0,04$;
- Número de clones de cada anticorpo: 3;
- Número de iterações do SIA: 20;
- Codificação: binária utilizando Gray code com 25 bits, por variável;
- Crossover: uniforme com taxa de 50%;
- Probabilidade de ocorrência do *crossover*: 100%.

Além desses parâmetros, o AG-SIA^C ainda requer a definição do raio do *clearing*. Foi utilizado um valor de 10% do comprimento do cromossomo.

Para o AG-SIA os parâmetros utilizados são:

- Tamanho da população (AG + SIA): 50 indivíduos;
- Taxa de mutação do AG: $0,01$;
- Taxa mínima de mutação do SIA: $0,01$ (mesma que a do AG);
- Taxa máxima de mutação do SIA: $0,03$;
- Número de clones de cada anticorpo: 1;

- Codificação: binária utilizando Gray code com 25 bits, por variável;
- *Crossover*: uniforme com taxa de 50%;
- Probabilidade de ocorrência do *crossover*: 90%.

5.1.1 *Java Native Interface*

Como citado anteriormente, para a avaliação de uma solução candidata durante o processo de evolução, particularmente nos problemas de otimização estrutural, foi utilizado um módulo de Elementos Finitos [22] capaz de fornecer os valores das grandezas a serem verificadas nas restrições do problema. A utilização desse módulo, conhecido como programa STAP (an in-core solution STatic Analysis Program) foi possível em razão das facilidades de uma biblioteca integrada da linguagem Java: a *Java Native Interface* (JNI). Essa adaptabilidade pode ser extendida a outros linguagens que geram código nativo (C, C++, Delphi, COBOL, etc).

Como já foi visto, um programa desenvolvido em Java pode ser executado em qualquer plataforma que tenha uma JVM instalada. Com o uso dessa biblioteca, entretanto, essa vantagem é retirada já que o código a ser utilizado é nativo de um sistema. Mais informações sobre a *Java Native Interface* podem ser obtidas em [104].

5.2 Experimento 1 - Suite de Funções

Um conjunto de problemas de teste com variáveis contínuas iniciado em [105] e expandido, recentimente, em [38] é freqüentemente utilizado para avaliar a eficiência de técnicas evolucionárias de otimização com restrições. Os resultados apresentados aqui são avaliados em três níveis, considerando o número de cálculos da função objetivo. As tabelas de 5.1 à 5.21 apresentam os valores a serem comparados dispondo de três possibilidades para o limite máximo de 5000, 50000 e 500000 avaliações da função objetivo (na). Em cada uma das tabelas, o primeiro bloco composto pelas três primeiras linhas mostram os resultados para 5000 cálculos da função objetivo. Os segundo e terceiro blocos correspondem aos valores obtidos com 50000 e 500000 avaliações da fução objetivo.

As tabelas apresentam os valores da encontrados pelos algoritmos para o melhor, mediana e pior caso. Além disso, são mostrados também a média, o desvio padrão (dp) e o número de execuções em que ao menos um resultado factível foi encontrado (erf). É importante observar que apenas valores factíveis foram considerados.

Os resultados para as funções g_{20}, g_{21} e g_{22} não são mostrados, pois nenhuma solução factível foi encontrada por nenhuma das técnicas propostas ou usadas para comparação. As formulações para os problemas dessa seção podem ser encontrados em [38]. Os testes foram feitos num total de 25 execuções independentes (utilizando-se sementes distintas para o gerador de números pseudo-aleatórios). O APM [36, 43], método de penalização adaptativa, e o SR [44], técnica de ordenação estocástica, são utilizados para comparar dos resultados encontrados pelos algoritmos híbridos propostos.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	−12.54560	−9.66313	−9.63801	1.83	−3.54334	23	5000
SR	−12.53710	−8.89068	−8.50374	2.15	−5.51379	9	5000
AG-SIA^E	−12.42924	−9.97209	−9.70580	1.69	−3.11102	**25**	5000
AG-SIA^C	**−13.58816**	**−12.07793**	**−11.80723**	**1.31**	**−8.12925**	**25**	5000
AG-SIA	−6.44972	−3.97147	−4.13181	1.74	−2.05377	8	5000
APM	−14.99420	−14.98342	−14.97430	$3.26E-2$	−14.83784	**25**	50000
SR	**−14.99857**	**−14.99300**	**−14.99313**	**3.81E − 3**	**−14.98008**	24	50000
AG-SIA^E	−14.78591	−14.65819	−14.60320	$2.16E-1$	−13.25278	**25**	50000
AG-SIA^C	−14.86625	−14.76883	−14.76332	$4.81E-2$	−14.64413	**25**	50000
AG-SIA	−14.99379	−14.97697	−14.96744	$5.38E-2$	−14.90857	**25**	50000
APM	−14.99996	−14.99954	−14.96378	$9.79E-2$	−14.57225	**25**	500000
SR	−14.99977	−14.99951	−14.99931	$5.42E-4$	−14.99757	**25**	500000
AG-SIA^E	−14.98813	−14.98056	−14.97966	$5.66E-3$	−14.96613	**25**	500000
AG-SIA^C	−14.98887	−14.97595	−14.97514	$8.06E-3$	−14.93905	**25**	500000
AG-SIA	**−14.99999**	**−14.99999**	**−14.99999**	**3.73E − 6**	**−14.99998**	**25**	500000

Tabela 5.1: Resultados do problema g_1. O melhor valor é −15.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	-0.5944454	-0.4985342	-0.4854191	$5.91E-2$	-0.3512636	**25**	5000
SR	-0.5864694	-0.5283935	-0.5312732	$3.37E-2$	-0.4541653	**25**	5000
AG-SIA^E	-0.6110063	-0.5275892	-0.5284940	$3.61E-2$	-0.4487732	**25**	5000
AG-SIA^C	-0.5732360	-0.5236345	-0.5213102	$2.90E-2$	-0.4498942	**25**	5000
AG-SIA	$\mathbf{-0.6713061}$	$\mathbf{-0.5923598}$	$\mathbf{-0.5851114}$	**1.78E − 3**	$\mathbf{-0.5231159}$	**25**	5000
APM	-0.7745439	-0.6517182	-0.6663961	$6.24E-2$	-0.5706031	**25**	50000
SR	-0.7486565	-0.6644478	-0.6653669	$7.01E-2$	-0.5154426	**25**	50000
AG-SIA^E	-0.7332407	-0.6444320	-0.6526506	$4.90E-2$	-0.5351180	**25**	50000
AG-SIA^C	-0.7482104	-0.6469565	-0.6455348	$4.70E-2$	-0.5435568	**25**	50000
AG-SIA	$\mathbf{-0.7840179}$	$\mathbf{-0.7097032}$	$\mathbf{-0.7149940}$	**1.88E − 3**	$\mathbf{-0.6319040}$	**25**	50000
APM	-0.8015561	$\mathbf{-0.7787246}$	-0.7724801	$2.26E-2$	-0.7261164	**25**	500000
SR	-0.7854180	-0.7405743	-0.7256381	$5.57E-2$	-0.6079553	**25**	500000
AG-SIA^E	-0.7934579	-0.7428760	-0.7383075	$3.73E-2$	-0.6111616	**25**	500000
AG-SIA^C	-0.7834507	-0.7437401	-0.7351923	$3.51E-2$	-0.6567412	**25**	500000
AG-SIA	$\mathbf{-0.8018923}$	-0.7775314	$\mathbf{-0.7769726}$	**2.21E − 3**	$\mathbf{-0.7441443}$	**25**	500000

Tabela 5.2: Resultados do problema g_2. O melhor valor é -0.8036191.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**−0.4770853**	**−0.1529288**	**−0.2313726**	$9.65E-2$	**−0.1104597**	24	5000
SR	–	–	–	–	–	0	5000
AG-SIA^E	−0.2367439	−0.0035223	−0.0253767	$5.03E-2$	−0.0000028	20	5000
AG-SIA^C	−0.1749013	−0.0029394	−0.0166668	$3.39E-1$	−0.0000003	22	5000
AG-SIA	−0.3657447	−0.0695542	−0.0960841	**8.41E − 3**	−0.0035011	**25**	5000
APM	−0.9173612	−0.5289950	−0.5508323	$1.44E-1$	−0.3099690	**25**	50000
SR	–	–	–	–	–	0	50000
AG-SIA^E	−0.2853399	−0.0199313	−0.0437244	$6.24E-2$	−0.0001201	**25**	50000
AG-SIA^C	−0.4007668	−0.0172259	−0.0517864	$8.53E-2$	−0.0000029	**25**	50000
AG-SIA	**−0.9899670**	**−0.8853369**	**−0.8512225**	**1.15E − 2**	**−0.5988370**	**25**	50000
APM	−1.0004896	−1.0004466	−1.0004036	$1.11E-4$	−0.9999571	**25**	500000
SR	–	–	–	–	–	0	500000
AG-SIA^E	−0.7615700	−0.2659286	−0.2976153	$1.93E-1$	−0.0267321	**25**	500000
AG-SIA^C	−0.8879961	−0.4592956	−0.4908986	$2.12E-2$	−0.0406379	**25**	500000
AG-SIA	**−1.0004921**	**−1.0004551**	**−1.0004487**	**1.54E − 5**	**−1.0002798**	**25**	500000

Tabela 5.3: Resultados do problema g_3. O melhor valor é −1.0005001.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	−30566.3455	−30383.8927	−30375.9930	$1.24E+2$	−30103.4912	**25**	5000
SR	−30574.5977	−30396.2109	−30374.8434	$1.22E+3$	−30170.9648	**25**	5000
AG-SIA^E	**−30642.9014**	**−30520.1760**	−30491.8207	$9.42E+1$	−30225.0856	**25**	5000
AG-SIA^C	−30641.3563	−30515.4392	**−30495.9788**	**8.06E+1**	**−30281.1385**	**25**	5000
AG-SIA	−30552.3862	−30392.3863	−30390.9200	$1.74E+4$	−30110.2253	**25**	5000
APM	**−30664.6630**	−30564.6626	−30541.6923	$8.76E+1$	−30367.2095	**25**	50000
SR	−30660.2910	−30508.0762	−30491.9935	$1.47E+3$	−29985.3633	**25**	50000
AG-SIA^E	−30661.1218	−30630.4060	−30622.2918	$3.87E+1$	−30416.7505	**25**	50000
AG-SIA^C	−30664.4368	−30636.4277	−30632.8975	**1.99E+1**	**−30581.5809**	**25**	50000
AG-SIA	−30663.4742	**−30648.2437**	**−30641.5461**	$6.16E+2$	−30557.5550	**25**	50000
APM	−30665.5238	−30665.4709	−30665.0947	$8.55E-1$	−30661.7610	**25**	500000
SR	−30664.7051	−30651.1426	−30633.9923	$4.79E+1$	−30453.1055	23	500000
AG-SIA^E	−30665.4397	−30663.4553	−30662.6091	3.21	−30651.3820	**25**	500000
AG-SIA^C	−30665.3889	−30663.7472	−30662.9301	2.28	−30655.5270	**25**	500000
AG-SIA	**−30665.5379**	**−30665.5126**	**−30665.4726**	**1.34E−2**	**−30665.0819**	**25**	500000

Tabela 5.4: Resultados do problema g_4. O melhor valor é −30665.5386717.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	–	–	–	–	–	0	5000
SR	**5141.8721**	**5193.0991**	**5221.2872**	**8.76E + 1**	**5421.4668**	**9**	5000
AG-SIA^E	–	–	–	–	–	0	5000
AG-SIA^C	–	–	–	–	–	0	5000
AG-SIA	–	–	–	–	–	0	5000
APM	5137.3998	5262.4268	5420.4905	$3.54E+2$	5993.0124	9	50000
SR	**5126.5215**	**5149.3882**	**5157.0944**	**3.507E + 1**	**5258.4805**	**16**	50000
AG-SIA^E	–	–	–	–	–	0	50000
AG-SIA^C	–	–	–	–	–	0	50000
AG-SIA	–	–	–	–	–	0	50000
APM	5127.3606	5244.5322	5312.6175	$2.45E+2$	5993.0113	**24**	500000
SR	5126.5195	**5137.3833**	**5155.8345**	$3.87E+1$	5258.4736	13	500000
AG-SIA^E	5166.0885	5189.4730	5204.1544	**3.85E + 1**	**5256.9017**	3	500000
AG-SIA^C	–	–	–	–	–	0	500000
AG-SIA	**5126.5027**	5163.2389	5253.1241	$3.09E+2$	5721.1774	19	500000

Tabela 5.5: Resultados do problema g_5. O melhor valor é 5126.4967140.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	-6834.5344	-3836.6674	-5120.8467	$1.58E+3$	-1276.4607	**24**	5000
SR	$\mathbf{-6922.1831}$	-5787.4535	$\mathbf{-5605.5055}$	$\mathbf{1.09E+3}$	$\mathbf{-2215.5444}$	18	5000
AG-SIA^E	-6650.0106	-5626.4849	-5318.8276	$1.22E+3$	-1372.5897	23	5000
AG-SIA^C	-6584.6677	$\mathbf{-5847.3395}$	-5553.2601	$1.22E+3$	-1394.2647	20	5000
AG-SIA	-4539.0174	-3541.6212	-3433.1394	$4.85E+5$	-2058.0008	16	5000
APM	-6939.3001	-3984.0864	-6413.3729	$1.11E+3$	-1319.0707	24	50000
SR	-6956.6471	-6682.5586	-6612.7806	$2.96E+2$	-5772.6938	15	50000
AG-SIA^E	-6960.8910	-6904.2614	-6901.5555	$\mathbf{2.97E+1}$	-6786.0714	**25**	50000
AG-SIA^C	-6955.0718	-6907.1951	-6896.5274	$6.85E+1$	-6464.7069	**25**	50000
AG-SIA	$\mathbf{-6961.6620}$	$\mathbf{-6959.3884}$	$\mathbf{-6956.0150}$	$1.37E+2$	$\mathbf{-6901.6117}$	**25**	50000
APM	$\mathbf{-6961.7961}$	-6961.7710	-6961.7742	$1.42E-2$	-6961.7592	24	500000
SR	$\mathbf{-6961.7961}$	-6961.7827	-6961.7863	$5.19E-3$	-6961.7827	13	500000
AG-SIA^E	-6961.7894	-6961.7659	-6961.7682	$7.87E-3$	-6961.7491	**25**	500000
AG-SIA^C	$\mathbf{-6961.7961}$	-6961.7558	-6961.7574	$1.23E-2$	-6961.7424	**25**	500000
AG-SIA	$\mathbf{-6961.7961}$	$\mathbf{-6961.7961}$	$\mathbf{-6961.7961}$	**0.00**	$\mathbf{-6961.7961}$	**25**	500000

Tabela 5.6: Resultados do problema g_6. O melhor valor é -6961.8138755.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	43.2072211	86.2003283	93.1306621	$3.28E+1$	152.181957	**25**	5000
SR	f31.9050388	50.1892052	**51.5728580**	**1.33E + 1**	**74.3306732**	**25**	5000
AG-SIA^E	34.7181969	72.7931753	110.996260	$1.25E+2$	776.563144	**25**	5000
AG-SIA^C	**28.6809702**	**45.0236259**	55.4336500	$2.90E+1$	211.927963	**25**	5000
AG-SIA	178.9944807	645.9088755	1044.7180285	$8.42E+5$	3298.940758	12	5000
APM	**24.5673615**	28.3454008	29.2488815	3.67	38.5389701	**25**	50000
SR	25.0301437	28.3579140	30.0073784	5.61	46.1200676	24	50000
AG-SIA^E	25.4217483	31.6841466	33.9099435	$1.02E+1$	95.1135317	**25**	50000
AG-SIA^C	24.9646997	**26.0109090**	**26.1673499**	**9.77E − 1**	**28.9968472**	**25**	50000
AG-SIA	24.7210266	28.6900899	30.7579049	4.99	58.6746432	**25**	50000
APM	24.4860035	26.8199501	27.0714049	2.27	34.4138410	**25**	500000
SR	**24.4150887**	28.1761951	28.4540093	4.00	41.6882782	21	500000
AG-SIA^E	24.5600669	28.4441414	29.8455249	4.35	41.0650770	**25**	500000
AG-SIA^C	24.4482393	**24.7079128**	**24.8274137**	**3.20E − 1**	**25.9281175**	**25**	500000
AG-SIA	24.4857599	26.4192052	27.3073734	6.89	33.3677490	**25**	500000

Tabela 5.7: Resultados do problema g_7. O melhor valor é 24.3062090.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**−0.0958250**	−0.0958248	−0.9582019	$1.17E-5$	−0.0957811	**25**	5000
SR	**−0.0958250**	**−0.0958250**	−0.0874181	$2.25E-2$	−0.0291438	24	5000
AG-SIA^E	**−0.0958250**	**−0.0958250**	−0.0944914	$9.34E-3$	−0.0291438	**25**	5000
AG-SIA^C	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	5000
AG-SIA	**−0.0958250**	−0.0958032	−0.0923795	$1.74E-4$	−0.0290356	**25**	5000
APM	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	50000
SR	**−0.0958250**	**−0.0958250**	−0.0851560	$2.49E-2$	−0.0291438	**25**	50000
AG-SIA^E	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	50000
AG-SIA^C	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	50000
AG-SIA	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	50000
APM	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	500000
SR	−0.0958251	−0.0958251	−0.0878233	$2.21E-2$	−0.0291438	**25**	500000
AG-SIA^E	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	500000
AG-SIA^C	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	500000
AG-SIA	**−0.0958250**	**−0.0958250**	**−0.0958250**	**0.00**	**−0.0958250**	**25**	500000

Tabela 5.8: Resultados do problema g_8. O melhor valor é -0.0958250.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	684.6761547	703.3695986	714.3836304	$2.96E+1$	825.0185225	**25**	5000
SR	684.8146362	696.2620239	702.4067383	$1.38E+1$	734.6710205	**25**	5000
AG-SIA^E	682.9336332	690.1082142	693.8352805	9.39	726.2259899	**25**	5000
AG-SIA^C	**681.8268793**	**686.3363980**	**686.9471871**	**3.20**	**694.0967521**	**25**	5000
AG-SIA	689.4433368	743.6043655	752.9479625	$2.33E+2$	882.3997388	**25**	5000
APM	**680.7658016**	681.6898093	682.1195215	1.36	687.0250951	**25**	50000
SR	680.8317261	683.3109131	685.3920581	6.91	715.9301758	**25**	50000
AG-SIA^E	680.8432734	683.2076576	683.8952635	2.42	690.3016947	**25**	50000
AG-SIA^C	680.7949849	**681.4916867**	**681.5876354**	**4.48E − 1**	**682.6530750**	**25**	50000
AG-SIA	680.9040567	681.911573	682.6416455	7.23	694.7290743	**25**	50000
APM	**680.6474288**	**680.7799403**	**680.7979077**	**1.72E − 2**	681.4545510	**25**	500000
SR	680.7388916	681.9906006	682.1157532	1.31	687.0321655	22	500000
AG-SIA^E	680.6545927	681.3924261	681.4811963	$5.75E-1$	683.0550587	**25**	500000
AG-SIA^C	680.6801414	680.8877626	680.9005951	$1.30E-1$	**681.2122556**	**25**	500000
AG-SIA	680.6583415	680.8019185	680.9023721	$7.62E-2$	681.6543442	**25**	500000

Tabela 5.9: Resultados do problema g_9. O melhor valor é 680.6300573.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	–	–	–	–	–	0	5000
SR	–	–	–	–	–	0	5000
AG-SIA^E	8821.3618	11666.3139	12543.9797	$3.70E+3$	24155.3658	**18**	5000
AG-SIA^C	**7769.5966**	**10143.4935**	**11684.9899**	$3.95E+3$	24097.0972	15	5000
AG-SIA	16111.1235	16294.1137	16329.6423	**3.79E+2**	**16583.6895**	3	5000
APM	**7080.4052**	**7581.6931**	**7683.2236**	**6.59E+2**	**8387.5723**	3	50000
SR	–	–	–	–	–	0	50000
AG-SIA^E	7157.2501	8297.4039	8630.7599	$1.42E+3$	14944.8744	**25**	50000
AG-SIA^C	7142.5267	7675.6942	7873.9504	$7.17E+2$	10682.2014	23	50000
AG-SIA	7260.0426	8033.9286	8133.2079	$5.24E+3$	9488.6850	**25**	50000
APM	7068.6338	8181.6974	8154.6199	$7.75E+2$	9769.1018	9	500000
SR	7538.5068	7538.5068	7538.5068	–	**7538.5068**	1	500000
AG-SIA^E	7054.8350	8123.7119	8417.9070	$1.27E+3$	13205.017	**25**	500000
AG-SIA^C	**7053.5055**	**7314.0347**	**7361.5796**	**2.69E+2**	8701.1528	**25**	500000
AG-SIA	7065.6712	7453.0779	7736.6282	$3.74E+3$	8913.8646	**25**	500000

Tabela 5.10: Resultados do problema g_{10}. O melhor valor é 7049.2480205.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	0.7499568	0.8204312	0.8380892	$7.94E-2$	0.9972218	**25**	5000
SR	**0.7499117**	**0.7596881**	**0.7707811**	$3.51E-2$	**0.8982297**	18	5000
AG-SIA^E	0.7501970	0.8125605	0.8489786	$8.62E-2$	0.9968189	24	5000
AG-SIA^C	0.7500200	0.8214822	0.8463420	$8.83E-2$	0.9995183	22	5000
AG-SIA	0.7499638	0.7606574	0.7861055	**3.45E − 2**	0.9853051	24	5000
APM	0.7499568	0.8204122	0.8377019	$7.92E-2$	0.9970093	**25**	50000
SR	0.7499093	0.7531331	0.7697072	$3.07E-2$	0.8891737	24	50000
AG-SIA^E	0.7499000	0.7549079	0.7824960	$5.07E-2$	0.9782222	**25**	50000
AG-SIA^C	0.7499013	0.7659117	0.7999625	$6.49E-2$	0.9904011	**25**	50000
AG-SIA	**0.7499000**	**0.7499015**	**0.7499033**	**2.09E − 5**	**0.7499186**	**25**	50000
APM	0.7499540	0.8118332	0.8334051	$7.65E-2$	0.9936831	**25**	500000
SR	0.7499090	0.7583824	0.7708550	$2.60E-2$	0.8309575	23	500000
AG-SIA^E	**0.7499000**	0.7499317	0.7499695	$1.62E-4$	0.7510275	**25**	500000
AG-SIA^C	0.7499001	0.7499380	0.7508844	$3.90E-3$	0.7709872	**25**	500000
AG-SIA	**0.7499000**	**0.7499000**	**0.7499001**	**1.33E − 7**	**0.7499005**	**25**	500000

Tabela 5.11: Resultados do problema g_{11}. O melhor valor é 0.7499.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	–	–	–	–	–	0	5000
SR	**−1**	**−1**	−0.9985502	$2.66E-3$	−0.9924536	**25**	5000
AG-SIA^E	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	5000
AG-SIA^C	**−1**	**−1**	−0.9999999	$2.04E-7$	−0.999999	**25**	5000
AG-SIA	**−1**	−0.9999831	−0.9965932	$2.56E-5$	−0.9862545	**25**	5000
APM	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	50000
SR	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	50000
AG-SIA^E	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	50000
AG-SIA^C	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	50000
AG-SIA	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	50000
APM	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	500000
SR	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	500000
AG-SIA^E	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	500000
AG-SIA^C	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	500000
AG-SIA	**−1**	**−1**	**−1**	**0.00**	**−1**	**25**	500000

Tabela 5.12: Resultados do problema g_{12}. O melhor valor é -1.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**0.0556113**	**0.0966101**	0.8095851	$2.93E-1$	1.3202477	**25**	50000
SR	0.0670723	0.3804254	0.5788253	$6.24E-1$	2.4329371	13	50000
AG-SIA^E	0.2970864	0.9747390	1.0930967	$8.92E-1$	4.6175239	9	50000
AG-SIA^C	0.5150024	0.9535153	1.3761583	1.25	4.8781921	5	50000
AG-SIA	0.0591664	0.2923595	**0.3060395**	**4.00E − 2**	**0.7116657**	**25**	50000
APM	0.0556111	0.9288099	0.8829213	$3.83E-1$	1.8784189	**25**	500000
SR	0.1003231	0.5086704	0.7698459	$7.59E-1$	2.4842145	12	500000
AG-SIA^E	0.2970188	0.9754438	1.1859940	1.11	7.2200664	24	500000
AG-SIA^C	0.5148793	0.9138921	1.2220168	1.14	4.8769912	6	500000
AG-SIA	**0.0539418**	**0.0539761**	**0.0554841**	**5.14E − 3**	**0.0905853**	**25**	500000

Tabela 5.13: Resultados do problema g_{13}. O melhor valor é 0.0539415.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**−46.890438**	−43.194835	−43.027409	2.41	−38.807925	**25**	50000
SR	–	–	–	–	–	0	50000
AG-SIA^E	−42.961579	−42.961579	−42.961579	–	**−42.961579**	1	50000
AG-SIA^C	−44.439144	−42.838209	−42.838209	**1.60**	−41.237273	2	50000
AG-SIA	−46.0645522	**−43.54487**	**−43.7705545**	2.08	−41.0973521	18	50000
APM	−46.890438	−43.203376	−43.044209	2.40	−38.807998	**25**	500000
SR	–	–	–	–	–	0	500000
AG-SIA^E	−45.689376	−42.445971	−42.315373	1.77	−37.332921	22	500000
AG-SIA^C	−46.373549	−42.848299	−42.496643	1.83	−38.419681	22	500000
AG-SIA	**−47.6832053**	**−47.531358**	**−47.4342606**	**6.04E − 1**	**−46.8616135**	**25**	500000

Tabela 5.14: Resultados do problema g_{14}. O melhor valor é −47.7648885.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**968.965583**	**969.924408**	**969.924408**	**1.35**	**970.883266**	**2**	5000
SR	–	–	–	–	–	0	5000
AG-SIA^E	–	–	–	–	–	0	5000
AG-SIA^C	–	–	–	–	–	0	5000
AG-SIA	–	–	–	–	–	0	5000
APM	**962.640839**	965.298033	966.220198	3.06	971.286895	**25**	50000
SR	962.973328	**963.785889**	**963.965942**	**1.02**	**965.377930**	5	50000
AG-SIA^E	963.051821	966.700903	967.167665	3.39	972.217032	3	50000
AG-SIA^C	–	–	–	–	–	0	50000
AG-SIA	962.8614989	967.0526586	967.3117621	7.18	971.0313437	11	50000
APM	962.640483	965.297403	966.218809	3.06	971.285135	**25**	500000
SR	962.482971	962.698608	963.219477	$8.15E-1$	964.853943	21	500000
AG-SIA^E	961.767880	964.695332	965.472313	3.16	972.216474	12	500000
AG-SIA^C	–	–	–	–	–	0	500000
AG-SIA	**961.7150233**	**961.7150499**	**961.7157591**	**3.91E − 3**	**961.7230629**	**25**	500000

Tabela 5.15: Resultados do problema g_{15}. O melhor valor é 961.7150222.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	−1.8598855	−1.4981970	−1.4960260	$2.60E-1$	−1.0315077	17	5000
SR	−1.7976446	−1.5929062	−1.5427779	$1.81E-1$	−1.1406803	22	5000
AG-SIA^E	−1.8691629	−1.6957917	−1.6466281	$1.66E-1$	−1.1723105	**25**	5000
AG-SIA^C	**−1.8847265**	**−1.7729120**	**−1.7525989**	$8.77E-2$	**−1.5163932**	**25**	5000
AG-SIA	−1.7594457	−1.3946147	−1.3994435	**1.36E − 2**	−1.2353074	20	5000
APM	−1.8968595	−1.8343355	−1.8293013	$4.45E-2$	−1.7492531	**25**	50000
SR	−1.8618839	−1.7203965	−1.6604525	$1.89E-1$	−0.9743471	24	50000
AG-SIA^E	−1.9020257	−1.8343978	−1.8201434	$6.43E-2$	−1.5358576	**25**	50000
AG-SIA^C	**−1.9046090**	**−1.8963019**	**−1.8938212**	$8.09E-3$	**−1.8700456**	**25**	50000
AG-SIA	−1.8972113	−1.8481757	−1.8307088	**3.21E − 3**	−1.6374893	**25**	50000
APM	−1.9051516	**−1.9050654**	−1.9032724	$5.45E-3$	−1.8818044	**25**	500000
SR	−1.8696414	−1.7357190	−1.7042652	$1.78E-1$	−1.0286503	24	500000
AG-SIA^E	−1.9049337	−1.8935411	−1.8863406	$1.93E-2$	−1.8168770	**25**	500000
AG-SIA^C	−1.9051032	−1.9044968	**−1.9042373**	**6.93E − 4**	**−1.9025437**	**25**	500000
AG-SIA	**−1.9051537**	−1.9047733	−1.9039405	$4.98E-3$	−1.8963863	**25**	500000

Tabela 5.16: Resultados do problema g_{16}. O melhor valor é −1.9051553.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	**8940.168230**	**8940.168230**	**8940.168230**	–	**8940.168230**	**1**	50000
SR	–	–	–	–	–	0	50000
AG-SIA^E	–	–	–	–	–	0	50000
AG-SIA^C	–	–	–	–	–	0	50000
AG-SIA	–	–	–	–	–	0	50000
APM	8875.515487	8948.681152	8980.116781	$\mathbf{1.10E+2}$	9276.896157	**24**	500000
SR	–	–	–	–	–	0	500000
AG-SIA^E	–	–	–	–	–	0	500000
AG-SIA^C	–	–	–	–	–	0	500000
AG-SIA	**8863.988067**	**8931.139105**	**8960.752075**	$9.50E+2$	**9240.696876**	17	500000

Tabela 5.17: Resultados do problema g_{17}. O melhor valor é 8853.5396748.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	–	–	–	–	–	0	5000
SR	–	–	–	–	–	0	5000
AG-SIA^E	–	–	–	–	–	0	5000
AG-SIA^C	**−0.5348792**	**−0.2637329**	**−0.3002225**	**1.22E − 1**	**−0.1420471**	**4**	5000
AG-SIA^E	–	–	–	–	–	0	5000
APM	−0.7869344	−0.7869344	**−0.7869344**	–	**−0.7869344**	1	50000
SR	−0.8621180	**−0.8423864**	−0.7611128	$1.35E-1$	−0.4947852	15	50000
AG-SIA^E	−0.8546679	−0.5379352	−0.5779785	$1.26E-1$	−0.3574985	18	50000
AG-SIA^C	−0.8586420	−0.6608961	−0.7214536	$1.09E-1$	−0.5130703	9	50000
AG-SIA	**−0.8626188**	−0.7575913	−0.7279657	**1.24E − 2**	−0.4898580	**25**	50000
APM	−0.8641388	**−0.8640182**	**−0.8619589**	**2.34E − 3**	**−0.8575919**	8	500000
SR	−0.8646700	−0.8491147	−0.7718814	$1.46E-1$	−0.4999466	19	500000
AG-SIA^E	−0.8641044	−0.6547800	−0.6715419	$1.29E-1$	−0.4882556	23	500000
AG-SIA^C	−0.8652776	−0.6698892	−0.7442860	$1.04E-1$	−0.5615551	9	500000
AG-SIA	**−0.8656750**	−0.8529690	−0.8286874	$3.50E-2$	−0.6078096	**25**	500000

Tabela 5.18: Resultados do problema g_{18}. O melhor valor é −0.8660254.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	255.724401	437.289256	490.432396	$2.04E+2$	1115.49838	**25**	5000
SR	184.100540	307.581970	341.076149	$1.38E+2$	780.532165	**25**	5000
AG-SIA^E	121.955285	212.209075	215.377929	$5.96E+1$	437.843290	**25**	5000
AG-SIA^C	**108.631591**	198.263474	200.606560	**4.41E+1**	**301.788178**	**25**	5000
AG-SIA	109.828792	**180.4855963**	**197.4698653**	$3.28E+3$	339.8308239	**25**	5000
APM	55.5094357	90.2558150	100.584946	$2.83E+1$	156.224591	**25**	50000
SR	64.2224197	97.0017242	110.557444	$3.61E+1$	210.558655	24	50000
AG-SIA^E	65.9034885	117.350395	118.912289	$3.32E+1$	208.347350	**25**	50000
AG-SIA^C	63.5711752	**83.4115213**	89.0866357	**2.11E+1**	155.190918	**25**	50000
AG-SIA	**54.1241333**	83.4956654	**82.8110546**	$2.83E+2$	**124.2977299**	**25**	50000
APM	47.5302335	67.7416098	67.5369704	$1.34E+1$	101.205100	**25**	500000
SR	49.4397850	78.6502228	79.2372183	$2.10E+1$	126.887253	**25**	500000
AG-SIA^E	45.2934737	84.4632468	90.4197446	$2.50E+1$	152.756103	**25**	500000
AG-SIA^C	40.6118492	65.5883385	66.0103369	**1.19E+1**	94.8934192	**25**	500000
AG-SIA	**39.8681536**	**58.6522001**	**62.8015911**	$1.85E+2$	**86.8022057**	**25**	500000

Tabela 5.19: Resultados do problema g_{19}. O melhor valor é 32.6555929.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	16.670943	41.018660	41.018660	**3.44E + 1**	65.366380	**2**	50000
SR	–	–	–	–	–	0	50000
AG-SIA^E	–	–	–	–	–	0	50000
AG-SIA^C	–	–	–	–	–	0	50000
AG-SIA	**−107.453797**	**−107.453797**	**−107.453797**	–	**−107.453797**	1	50000
APM	9.7424748	37.553526	37.553526	**3.93E + 1**	65.364580	2	500000
SR	–	–	–	–	–	0	500000
AG-SIA^E	286.83437	286.83437	286.83437	–	286.83437	1	500000
AG-SIA^C	4.2437108	89.368312	118.06984	$1.11E + 2$	289.29904	4	500000
AG-SIA	**−260.271062**	**−124.731854**	**−115.481847**	$5.70E + 3$	**23.714543**	**25**	500000

Tabela 5.20: Resultados do problema g_{23}. O melhor valor é -400.0551000.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM	−5.5076836	−5.4848828	−5.4474105	$7.47E-2$	−5.2005319	**25**	5000
SR	−5.5050306	−5.3985375	−5.3958694	$9.90E-2$	−5.0622700	**25**	5000
AG-SIA^E	−5.5079887	**−5.5068253**	**−5.5055916**	**3.11E − 3**	−5.4901054	**25**	5000
AG-SIA^C	**−5.5079953**	−5.5064843	−5.5052758	$3.20E-3$	**−5.4943009**	**25**	5000
AG-SIA	−5.5078730	−5.4750420	−5.4529664	$4.73E-3$	−5.2397219	**25**	5000
APM	**−5.5080131**	−5.5079922	−5.5051182	$8.38E-3$	−5.4703893	**25**	50000
SR	−5.5080125	−5.4986755	−5.4598721	$7.11E-2$	−5.2227953	23	50000
AG-SIA^E	**−5.5080131**	−5.5080119	−5.5080083	$8.89E-6$	−5.5079661	**25**	50000
AG-SIA^C	**−5.5080131**	−5.5080106	−5.5080055	$1.33E-5$	−5.5079409	**25**	50000
AG-SIA	**−5.5080131**	**−5.5080131**	**−5.5080125**	**2.71E − 6**	**−5.5080048**	**25**	50000
APM	**−5.5080131**	**−5.5080131**	**−5.5080131**	**0.00**	**−5.5080131**	**25**	500000
SR	**−5.5080131**	**−5.5080131**	**−5.5080131**	**0.00**	**−5.5080131**	**25**	500000
AG-SIA^E	**−5.5080131**	**−5.5080131**	**−5.5080131**	**0.00**	**−5.5080131**	**25**	500000
AG-SIA^C	**−5.5080131**	**−5.5080131**	**−5.5080131**	**0.00**	**−5.5080131**	**25**	500000
AG-SIA	**−5.5080131**	**−5.5080131**	**−5.5080131**	**0.00**	**−5.5080131**	**25**	500000

Tabela 5.21: Resultados do problema g_{24}. O melhor valor é −5.5080133.

Os resultados com 5000 avaliações da função objetivo não são apresentados nas Tabelas 5.13, 5.14, 5.17 e 5.20, pois nenhuma das técnicas propostas ou utilizadas para comparação obteve soluções factíveis.

As Tabelas 5.22, 5.23 e 5.24 apresentam as porcentagens de melhores resultados obtidos pelas técnicas para cada fução (melhor, média, mediana, dp, pior e erf). As tabelas comparativas foram separadas considerando o número de avaliações das funções objetivo, que foram: 5000, 50000 e 500000 , respectivamente.

Como pode ser visto na Tabela 5.22, com 5000 cálculos das funções objetivas, o algoritmo híbrido com *clearing* produziu melhores resultados em todos os casos. Ao mesmo tempo, o AG-SIA^E chegou ao mesmo valor considerando o número de execuções com resultado factível (erf).

Considerando 50000 avaliações das funções objetivas (Tabela 5.23), o AG-SIA produziu melhores resultados em todos os casos com exceção do melhor caso e erf, onde o APM obteve (pouca) vantagem.

Quando se observa a Tabela 5.24, que compara os resultados obtidos utilizando 500000 avaliações das funções objetivo, é fácil perceber que o AG-SIA obteve resultados significativamente melhores em todos os indicadores. Sua vantagem chega a ultrapassar os 90% no caso de erf.

	Melhor	Mediana	Média	Pior	erf
APM	$14,29\%$	$9,52\%$	$9,52\%$	$9,52\%$	$47,62\%$
SR	$23,81\%$	$19,05\%$	$19,05\%$	$19,05\%$	$38,10\%$
AG-SIA^E	$14,29\%$	$19,05\%$	$9,52\%$	$4,67\%$	$\mathbf{52,38}\%$
AG-SIA^C	$\mathbf{47,62}\%$	$\mathbf{42,86}\%$	$\mathbf{33,33}\%$	$\mathbf{38,10}\%$	$\mathbf{52,38}\%$
AG-SIA	$14,29\%$	$9,52\%$	$9,52\%$	$9,52\%$	$33,33\%$

Tabela 5.22: Técnicas com os melhores resultados para os problemas utilizando 5000 avaliações da função objetivo.

	Melhor	Mediana	Média	Pior	erf
APM	**52**, **38**%	23, 81%	23, 81%	23, 81%	**80.95**%
SR	19, 05%	28, 57%	19, 05%	19, 05%	28, 57%
AG-SIA^E	14, 29%	9, 52%	9, 52%	14, 29%	66, 67%
AG-SIA^C	19, 05%	28, 57%	23, 81%	28, 57%	61, 90%
AG-SIA	47, 62%	**47**, **62**%	**57**, **14**%	**47**, **62**%	76.19%

Tabela 5.23: Técnicas com os melhores resultados para os problemas utilizando 50000 avaliações da função objetivo.

	Melhor	Mediana	Média	Pior	erf
APM	23, 81%	33, 33%	23, 81%	19, 05%	80.95%
SR	19, 05%	14, 29%	14, 29%	14, 29%	23, 81%
AG-SIA^E	19, 05%	14, 29%	14, 29%	19, 05%	66, 67%
AG-SIA^C	23, 81%	23, 81%	28, 57%	28, 57%	66, 67%
AG-SIA	**85**, **71**%	**66**, **67**%	**71**, **43**%	**71**, **43**%	**90**, **48**%

Tabela 5.24: Técnicas com os melhores resultados para os problemas utilizando 500000 avaliações da função objetivo.

5.3 Experimento 2 - Problema de Programação Não-Linear com Variáveis Mistas

Esse problema, citado em [106, 107], corresponde à maximização de uma função não-linear com variáveis mistas. Ele é composto por três variáveis contínuas e duas inteiras. Além disso, o problema contém três inequações de restrição. Ele é o *Problem 6* em [107] e o *Test-problem 14* em [106]. O problema de otimização pode ser escrito como:

$$f(x,y) = -5.357854x_1^2 - 0.835689y_1x_3 - 37.29329y_1 + 40792.141$$

a ser maximizado sujeito às restrições

$$g_1(x,y) = 85.3344070 + 0.0056858y_2x_3 + 0.0006262y_1x_2 - 0.0022053x_1x_3 - 92 \leq 0$$
$$g_2(x,y) = 80.5124900 + 0.0071317y_2x_3 + 0.0029955y_1y_2 - 0.0021813x_1^2 - 110 \leq 0$$
$$g_3(x,y) = 9.3009610 + 0.0047026x_1x_3 + 0.0012547y_1x_1 - 0.0019085x_1x_2 - 25 \leq 0$$

O espaço de busca é limitado por $27 \leq x_1, x_2, x_3 \leq 45$, com $x_i \in \Re$, $78 \leq y_1 \leq 102$ e $33 \leq y_2 \leq 45$, com y_j inteiros. A solução ótima é $(27, x_2, 27, 78, y_2)$ para quaisquer valores de x_2, y_2 dentro dos intervalos dados. O valor obtido com essa solução é $f(x,y) = 32217,43$.

Os resultados obtidos pelos algoritmos APM [106] e *Stochastic Ranking* foram utilizados nas comparações. Os resultados estatísticos são apresentados na Tabela 5.25. Todos os AG-SIA e o SR utilizaram 5000 cálculos da função objetivo contra 30000 dos considerados pelo APM [106]. Os resultados encontrados são similares aos obtidos pelo APM [106]. Entretanto, o número de cálculos da função objetivo foi muito menor que o usado pela referência (5000 contra 30000). É importante destacar o desempenho dos AG-SIA^E e AG-SIA, pois além de achar o melhor resultado, como nas outras técnicas, também foram mais eficientes nos valores da mediana e média.

Método	Melhor	Mediana	Média	dp	Pior	erf	na
APM[106]	**32217.43**	–	**32217.43**	–	**32217.43**	–	30000
SR	**32217.43**	32217.35	32217.05	1.20	32209.27	**50**	**5000**
AG-SIA^E	**32217.43**	**32217.43**	**32217.43**	$5.89E-3$	32217.39	**50**	**5000**
AG-SIA^C	**32217.43**	**32217.43**	32217.42	$7.80E-3$	32217.37	**50**	**5000**
AG-SIA	**32217.43**	**32217.43**	**32217.43**	**5.03E − 6**	32217.41	**50**	**5000**

Tabela 5.25: Resultados encontrados no Experimento 2. O melhor valor é -32217.42778.

5.4 Experimento 3 - Problema de Programação Não-Linear com Variáveis Inteiras

Esse é um problema de programação não-linear com cinco variáveis inteiras e oito inequações de restrição. O Problema 2 é o *Problem 8* em [108] e pode ser escrito como:

$$f(x) = x_1^2 + x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 - 8x_1 - 2x_2 - 3x_3 - x_4 - 2x_5$$

a ser minimizado sujeito às restrições

$$\begin{aligned}
g_1(x) &= x_1 + x_2 + x_3 + x_4 + x_5 - 400 \leq 0 \\
g_2(x) &= x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 - 800 \leq 0 \\
g_3(x) &= 2x_1 + x_2 + 6x_3 - 200 \leq 0 \\
g_4(x) &= x_3 + x_4 + 5x_5 - 200 \leq 0 \\
g_5(x) &= 55 - (x_1 + x_2 + x_3 + x_4 + x_5) \leq 0 \\
g_6(x) &= 48 - (x_1 + x_2 + x_3 + x_4) \leq 0 \\
g_7(x) &= 34 - (x_2 + x_4 + x_5) \leq 0 \\
g_8(x) &= 104 - (6x_1 + 7x_5) \leq 0
\end{aligned}$$

onde, $0 \leq x_i \leq 99$ para $i = 1, 2, 3, 4, 5$. A solução ótima é $f(x) = 807$ com $x = (16, 22, 5, 5, 7)$.

Uma comparação entre os algorítmos híbridos e o SR pode ser encontrada na Tabela 5.26, que mostra os valores estatísticos obtidos neste problema com 20000

cálculos da função objetivo em cinquenta execuções. Todos os métodos obtém a melhor solução do problema. A melhor média foi conseguida pelo AG-SIA^E e a pior solução pelo AG-SIA^C

Método	Melhor	Mediana	Média	dp	Pior	erf	na
SR	**807**	860	872.76	61.65	1036	**50**	**20000**
AG-SIA^E	**807**	**807**	**808.96**	**5.80**	842	**50**	**20000**
AG-SIA^C	**807**	**807**	812.12	7.84	**833**	**50**	**20000**
AG-SIA	**807**	808	821.58	24.03	921	**50**	**20000**

Tabela 5.26: Resultados encontrados no Experimento 3. O melhor valor é 807.

5.5 Experimento 4 - Problema de Programação Não-Linear com Variáveis Inteiras

Esse é o *Problem 16* em [108]. O problema contém treze variáveis inteiras e nove inequações de restrição. O problema de minimização pode ser descrito como:

$$f(x,y) = C^T x - 0.5 x^T Q x + D^T y$$

sujeito às restrições

$$\begin{aligned}
g_1(x,y) &= 2x_1 + 2x_2 + y_6 + y_7 - 10 \leq 0 \\
g_2(x,y) &= 2x_1 + 2x_3 + y_6 + y_8 - 10 \leq 0 \\
g_3(x,y) &= 2x_2 + 2x_3 + y_7 + y_8 - 10 \leq 0 \\
g_4(x,y) &= y_6 - 8x_1 \leq 0 \\
g_5(x,y) &= y_7 - 8x_2 \leq 0 \\
g_6(x,y) &= y_8 - 8x_3 \leq 0 \\
g_7(x,y) &= y_6 - 2x_4 - y_1 \leq 0 \\
g_8(x,y) &= y_7 - 2y_2 - y_3 \leq 0 \\
g_9(x,y) &= y_8 - 2y_4 - y_5 \leq 0
\end{aligned}$$

onde,

$$\begin{aligned} 0 \leq x_i \leq 1 \qquad & i = 1,2,3,4 \\ 0 \leq y_i \leq 1 \qquad & i = 1,2,3,4,5,9 \\ 0 \leq y_i \leq 3 \qquad & i = 6,7,8 \\ C^T = (5,5,5,5) \qquad & Q = 10 \times I \\ D^T = (-1,-1,-1,-1,-1,-1,-1,-1,-1) & \end{aligned}$$

Na Tabela 5.27 uma comparação de desempenho dos algoritmos híbridos com o SR é apresentada utilizando 1500 cálculos da função objetivo onde o AG-SIA^E e o AG-SIA^C se destacaram por encontrar os melhores valores.

Método	Ótimo	Mediana	Média	dp	Pior	erf	na
SR	**−15**	**−15**	−14.86	$4.05E-1$	−13	**50**	**1500**
AG-SIA	**−15**	**−15**	**−15**	**0.00**	**−15**	**50**	**1500**
AG-SIA^C	**−15**	**−15**	**−15**	**0.00**	**−15**	**50**	**1500**
AG-SIA	**−15**	**−15**	−14.14	1.04	−12	**50**	**1500**

Tabela 5.27: Resultados obtidos para o Experimento 4. O melhor valor é −15.

5.6 Experimento 5 - Mola Sob Tração/Compressão

Este é um problema de otimização de um projeto de um componente mecânio que é uma mola submetida à tração ou compressão, mostrada na Figura 5.1. O objetivo é minimizar o volume V de uma mola sob tração/compressão constantes. As variáveis de projeto são o número de espirais ativas da mola ($N = x_1 \in [2, 15]$), o diâmetro de cada volta ($D = x_2 \in [0.25, 1.3]$) e o diâmetro do arame ($d = x_3 \in [0.05, 2]$). O

volume e as restrições mecânicas são dados por:

$$
\begin{aligned}
V(x) &= (x_1 + 2)x_2x_3^2 \\
g_1(x) &= 1 - \frac{x_2^3 x_1}{71785x_3^4} \leq 0 \\
g_2(x) &= \frac{4x_2^2 - x_3x_2}{12566(x_2x_3^3 - x_3^4)} + \frac{1}{5108x_3^2} - 1 \leq 0 \\
g_3(x) &= 1 - \frac{140.45x_3}{x_2^2x_1} \leq 0 \\
g_4(x) &= \frac{x_2 + x_3}{1.5} - 1 \leq 0
\end{aligned}
$$

onde

$$2 \leq x_1 \leq 15 \qquad 0.25 \leq x_2 \leq 1.3 \qquad 0.05 \leq x_3 \leq 2$$

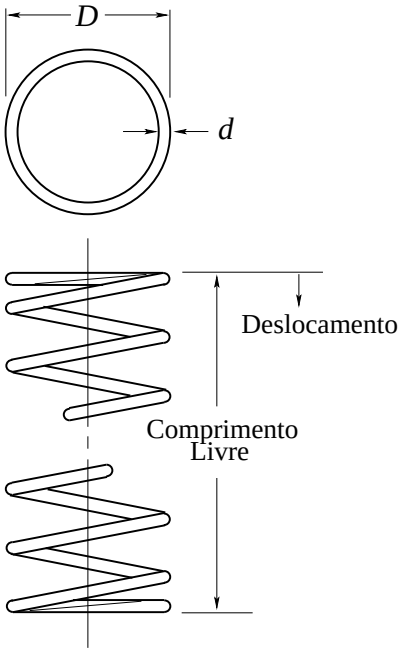

Figura 5.1: Mola utilizada no problema

O número de avaliações da função objetivo é adotado como 36000. Uma comparação de resultados pode ser vista na Tabela 5.28, onde o melhor resultado foi encontrado pelos AG-SIAs, com um volume final igual a 0.012520. É importante notar que o AG-SIA^C encontra os melhores valores em todos os casos estatísticos

considerados. A Tabela 5.29 mostra os valores encontrados para as variáveis de projeto e restrições para os melhores resultados de cada método. A referência [109] não apresenta esses valores.

	Melhor	Mediana	Média	dp	Pior	erf
ES [109]	0.012688	–	0.013014	–	0.017037	–
AG-SIA^E	**0.012666**	0.0128082	0.0129453	$3.31E-4$	0.0140502	**50**
AG-SIA^C	**0.012666**	**0.012704**	**0.012740**	**9.21E − 5**	**0.013121**	**50**
AG-SIA	**0.012666**	0.012892	0.013131	$6.28E-4$	0.015318	**50**
APM	0.012684	0.013575	0.014022	$1.47E-3$	0.017794	**50**
SR	0.012679	0.013716	0.014089	$1.30E-3$	0.017794	49

Tabela 5.28: Valores encontrados para a Mola sob Tração/Compressão. O sobrescrito C denota que o algoritmo utiliza o método de *clearing*.

	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
x_1	11.162923	11.187366	11.6611924	12.070748	11.375795
x_2	0.358887	0.358465	0.3505298	0.344304	0.355485
x_3	0.051779	0.051762	0.0514305	0.051168	0.051638
V	**0.012666**	**0.012666**	**0.012666**	0.0126838	0.012679

Tabela 5.29: Variáveis de projeto encontradas nos melhores resultados para o problema de tração/compressão da mola.

5.7 Experimento 6 - Redutor de Velocidade

O objetivo deste problema é minimizar o peso W de um componente mecânico que é um redutor de velocidade, mostrado na Figura 5.2. As variáveis de projeto são a largura da face ($b = x_1 \in [2.6, 3.6]$), o módulo dos dentes ($m = x_2 \in [0.7, 0.8]$), número de dentes ($n = x_3 \in [17, 28]$), tamanho da haste 1 entre os suportes ($l_1 = x_4 \in [7.3, 8.3]$), tamanho da haste 2 entre os suportes ($d_2 = x_7$). A varíavel x_3 é

inteira e as demais são contínuas. As restrições incluem limitações da tensão de flexão e de superficie da engrenagem de dentes, deslocamento transversal das hastes 1 e 2 gerado pela força transmitida e as tensões nas hastes 1 e 2.

O peso e as restrições mecânicas podem ser dados por

$$
\begin{aligned}
&W = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right)\\
&-1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right)\\
&+0.7854\left(x_4x_6^2 + x_5x_7^2\right)\\
&g_1(x) = 27x_1^{-1}x_2^{-2}x_3^{-1} \leq 1\\
&g_2(x) = 397.5x_1^{-1}x_2^{-2}x_3^{-2} \leq 1\\
&g_3(x) = 1.93x_2^{-1}x_3^{-1}x_4^3x_6^{-4} \leq 1\\
&g_4(x) = 1.93x_2^{-1}x_3^{-1}x_5^3x_7^{-4} \leq 1\\
&g_5(x) = \frac{1}{0.1x_6^3}\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + \{16.9\}10^6\right]^{0.5} \leq 1100\\
&g_6(x) = \frac{1}{0.1x_7^3}\left[\left(\frac{745x_5}{x_2x_3}\right)^2 + (157.5)\,10^6\right]^{0.5} \leq 850\\
&g_7(x) = x_2x_3 \leq 40 \qquad g_8(x) = x_1/x_2 \geq 5\\
&g_9(x) = x_1/x_2 \leq 12\\
&g_{10}(x) = (1.5x_6 + 1.9)\,x_4^{-1} \leq 1\\
&g_{11}(x) = (1.1x_7 + 1.9)\,x_5^{-1} \leq 1\\
&2.6 \leq x_1 \leq 3.6 \quad 0.7 \leq x_2 \leq 0.8 \quad 17 \leq x_3 \leq 28\\
&7.3 \leq x_4 \leq 8.3 \quad 7.8 \leq x_5 \leq 8.3 \quad 2.9 \leq x_6 \leq 3.9
\end{aligned}
$$

A Tabela 5.30 apresenta uma comparação dos resultados encontrados pelo algoritmo proposto e outros da literatura. Nesse caso, pode-se observar que todas as técnicas encontraram, aproximadamente, o mesmo valor para a melhor solução, exceto a estratégia evolutiva (ES, [109]). O melhor valor foi obtido pelo algoritmo APM (2996.3482024). Entretanto, o método proposto encontrou 2996.3482734, o que representa uma diferença de 0.00000237%. A Tabela 5.31 apresenta os valores das variáveis de projeto e as restrições para os melhores resultados. Foram utilizados 36000 cálculos da função objetivo neste problema.

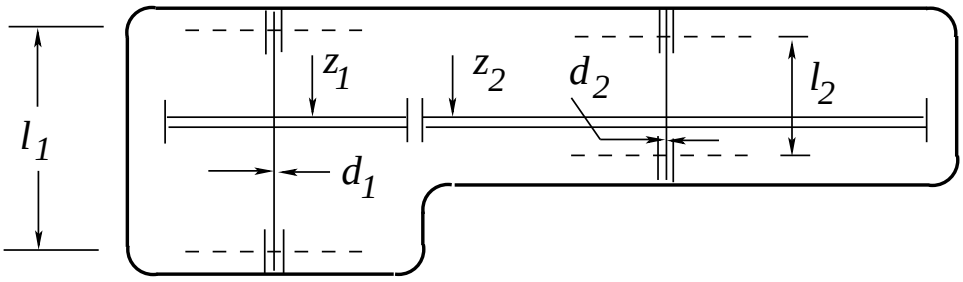

Figura 5.2: Redutor de velocidade.

	Melhor	Mediana	Média	dp	Pior	erf
ES [109]	3025.0051	–	3088.7778	–	3078.5918	–
AG-SIA^E	2996.3543	2996.3814	2996.3943	$3.31E-2$	2996.5135	**50**
AG-SIA^C	2996.3580	2996.3840	2996.3925	$2.61E-2$	2996.4678	**50**
AG-SIA	2996.3483	2996.3495	2996.3501	$7.45E-3$	2996.3599	**50**
APM	**2996.3482**	**2996.3482**	3033.8807	$1.10E+2$	3459.0948	19
SR	2996.3483	2996.3488	**2996.3491**	**1.01E − 3**	**2996.3535**	**50**

Tabela 5.30: Valores encontrados para o problema do redutor de velocidade.

	ES [109]	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
x_1	3.506163	3.500007	3.500002	3.500001	3.500000	3.500000
x_2	0.700831	0.700000	0.700000	0.700000	0.700000	0.700000
x_3	17	17	17	17	17	17
x_4	7.460181	7.300217	7.300178	7.300008	7.300000	7.300001
x_5	7.962143	7.800002	7.800001	7.800001	7.800000	7.800001
x_6	3.362900	3.350219	3.350224	3.350215	3.350215	3.350215
x_7	5.308949	5.286683	5.286689	5.286683	5.286683	5.286683
W	3025.0051	2996.3543	2996.3580	2996.3483	**2996.3482**	2996.3483

Tabela 5.31: Valores encontrados para as variáveis de projeto nos melhores resultados para o problema do redutor de velocidade.

5.8 Experimento 7 - Viga Soldada

O objetivo é minimizar o custo $C(h, l, t, b)$ de uma viga soldada, como mostrada na Figura 5.3, onde $h \in [0.125, 10]$ e $0.1 \leq l, t, b \leq 10$. A função objetivo e as restrições são como segue:

$$C(h, l, t, b) = 1.10471h^2l + 0.04811tb(14.0 + l)$$
$$g_1(\tau) = 13,600 - \tau \geq 0 \qquad g_2(\sigma) = 30,000 - \sigma \geq 0$$
$$g_3(b, h) = b - h \geq 0 \qquad g_4(P_c) = P_c - 6,000 \geq 0$$
$$g_5(\delta) = 0.25 - \delta \geq 0$$

onde as expressões para τ, σ, P_c e δ são dados por

$$\tau = \sqrt{(\tau')^2 + (\tau'')^2 + l\tau'\tau''/\alpha} \qquad \tau' = \frac{6000}{\sqrt{2}hl}$$
$$\alpha = \sqrt{0.25(l^2 + (h + t)^2)} \qquad \sigma = \frac{504000}{t^2b}$$
$$P_c = 64746.022(1 - 0.0282346t)tb^3 \quad \delta = \frac{2.1952}{t^3b}$$
$$\tau'' = \frac{6000(14 + 0.5l)\alpha}{2(0.707hl(l^2/12 + 0.25(h + t)^2))}$$

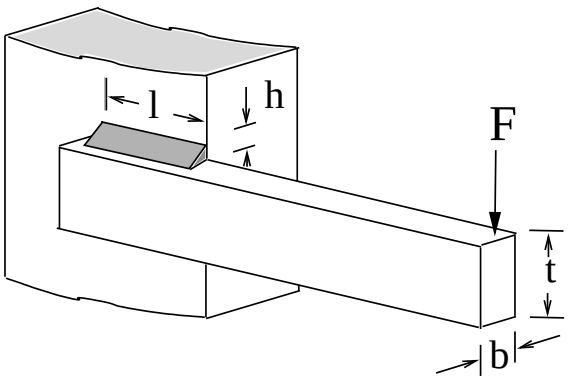

Figura 5.3: Viga soldada.

A Tabela 5.32 mostra uma comparação dos resultados. O melhor custo (2.38144) foi encontrado pelo APM. Todavia, o AG-SIA^C apresenta os melhores resultados em todos os outros casos. A Tabela 5.33 mostra as variáveis de projeto correspondentes

às melhores soluções encontradas por cada técnica. O número de cálculos da função objetivo utilizadas foram 320000.

	Melhor	Mediana	Média	dp	Pior	erf
AG-SIA^E	2.38542	2.47793	2.53184	$1.64E-1$	3.17973	**50**
AG-SIA^C	2.38164	**2.42056**	**2.43588**	**6.42E − 2**	**2.71210**	**50**
AG-SIA	2.38335	2.92121	2.99298	$2.02E-1$	4.05600	**50**
APM	**2.38144**	3.27244	3.49560	$9.09E-1$	5.94803	**50**
SR	2.59610	4.21812	4.33259	1.29	10.1833	**50**

Tabela 5.32: Valores encontrados para o custo da viga soldada.

	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
h	0.2453628	0.2444623	0.2434673	0.2442419	0.2758192
l	6.1996494	6.2154949	6.2507296	6.2231189	5.0052613
t	8.2744407	8.2912650	8.2914724	8.2914718	8.6261101
b	0.2453759	0.2444625	0.2443690	0.2443690	0.2758194
Cost	2.38542	2.38164	2.38335	**2.38144**	2.59610

Tabela 5.33: Variáveis de projeto dos melhores valores encontrados para o problema da viga soldada.

5.9 Experimento 8 - Vaso de Pressão

Esse problema [110–113] corresponde à minimização do peso de um vaso de pressão cilíndrico com duas tampas esféricas. O vaso de presão é mostrado na Figura 5.4. São quatro variáveis de projeto (medidas em polegadas): a espessura do vaso de pressão (T_s), a espessura da tampa (T_h), o raio interno do vaso (R) e a altura do componente cilíndrico (L). Dentre essas, duas são variáveis discretas (T_s e T_h) e duas são contínuas (R e L). Este é um problema com restrições, não-linear e com variáveis mistas (discretas e contínuas). Os limites das variáveis são

$0.0625 \leq T_s, T_h \leq 5$ (em passos constantes de 0.0625) e $10 \leq R, L \leq 200$. O peso a ser minimizado e as restrições são dadas por

$$W(T_s, T_h, R, L) = 0,6224 T_s T_h R + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_s^2 R$$
$$g_1(T_s, R) = T_s - 0.0193R \geq 0$$
$$g_2(T_h, R) = T_h - 0.00954R \geq 0$$
$$g_3(R, L) = \pi R^2 L + 4/3\pi R^3 - 1,296,000 \geq 0$$
$$g_4(L) = -L + 240 \geq 0$$

As primeiras duas restrições referem-se aos limites inferiores das razões T_s/R e T_h/R. A terceira restrição corresponde ao limite inferior para o volume do vaso e o último, ao limite superior de altura para o componente cilíndrico.

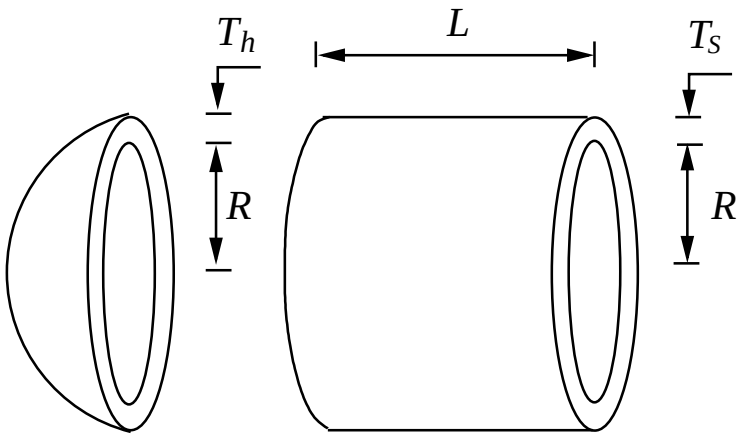

Figura 5.4: Vaso de Pressão.

A Tabela 5.34 apresenta uma comparação dos resultados obtidos pelos algoritmos. Todas as técnicas utilizaram 80000 chamadas à função objetivo, com exceção do SIA em [11], onde foi utilizado 150000. A melhor solução foi encontrada pelo AG-SIA proposto e corresponde a um peso final igual a 6059.8546470. O AG-SIA^C encontrou os melhores resultados estatísticos nos outros casos, com exceção do desvio padrão, onde o melhor valor também foi obtido pelo AG-SIA. A Tabela 5.35 mostra os detalhes das melhores soluções.

	Melhor	Mediana	Média	dp	Pior	erf	na
SIA [11]	6061.123	–	6734.085	–	7368.060	–	150000
AG-SIA^E	6060.608	6776.900	6670.493	$4.41E+2$	7340.052	**50**	**80000**
AG-SIA^C	6072.325	**6123.263**	**6213.910**	$1.35E+2$	**6443.125**	**50**	**80000**
AG-SIA	**6059.855**	6426.710	6545.126	$\mathbf{1.24E+2}$	7388.160	**50**	**80000**
APM	6065.822	6434.435	6632.376	$5.15E+2$	8248.003	**50**	**80000**
SR	6832.584	7073.107	7187.314	$2.67E+2$	8012.651	**50**	**80000**

Tabela 5.34: Valores dos pesos encontrados para o problema do vaso de pressão.

	SIA [11]	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
T_s	0.8125	0.8125	0.8125	0.8125	0.8125	1.1250
T_h	0.4375	0.4375	0.4375	0.4375	0.4375	0.5625
R	42.0870	42.0911	41.9952	42.0973	42.0492	58.1267
L	176.7791	176.7276	177.9198	176.6509	177.2522	44.5941
W	6061.1229	6060.6084	6072.3247	**6059.8546**	6065.8217	6832.5836

Tabela 5.35: Variáveis de projeto dos melhores resultados para o problema do vaso de pressão.

5.10 Experimento 9 - Viga Engastada e Livre

Esse problema [114] corresponde à minimização do volume de uma viga engastada e livre, mostrada na Figura 5.5 e sujeita a uma carga de $P = 50000$N. São dez as variáveis de projeto que correspondem à altura (H_i) e a largura (B_i) da seção transversal retangular de cada uma das cinco partes que compõem a viga. As variáveis B_1 e H_1 são inteiras. B_2, B_3, H_2 and H_3 assumem valores discretos, tal que $B_2, B_3 \in \{2.4, 2.6, 2.8, 3.1\}$ e $H_2, H_3 \in \{45.0, 50.0, 55.0, 60.0\}$. Finalmente, B_4, H_4, B_5 e H_5 são contínuas. As variáveis são definidas em centímetros e o módulo de Young do material é igual a 200 GPa. O volume da viga e as restrições do problema podem ser calculados como:

$$
\begin{aligned}
&V(H_i, B_i) = 100 \sum_{i=1}^{5} H_i B_i \\
&g_i(H_i, B_i) = \sigma_i \leq 14000\text{N/cm}^2 \quad i = 1, \ldots, 5 \\
&g_{i+5}(H_i, B_i) = H_i / B_i \leq 20 \quad i = 1, \ldots, 5 \\
&g_{11}(H_i, B_i) = \delta \leq 2.7\text{cm}
\end{aligned}
$$

onde, δ é o deslocamento da ponta da viga na direção vertical.

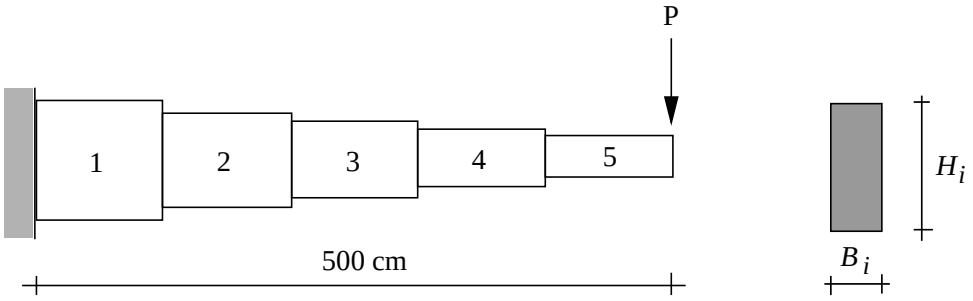

Figura 5.5: Viga engastada e livre

A Tabela 5.36 apresenta os resultados encontrados pelas diferentes técnicas. O número de cálculos da função objetivo utilizado foi de 35000 para todos os casos, com exceção da referência [114], que dispôs de 10000. O método SR produziu o melhor resultado com um volume final igual a 64599.6509715. Os melhores resultados em todos os outros casos considerados foram obtidos pelo AG-SIA^C.

	Melhor	Mediana	Média	dp	Pior	erf	na
Ref. [114]	64815	–	–	–	–	–	10000
AG-SIA^E	64834.38	70262.99	71053.83	$4.18E+3$	85677.13	**50**	35000
AG-SIA^C	65206.08	**68895.15**	**68677.78**	$\mathbf{2.10E+3}$	**73124.68**	**50**	35000
AG-SIA	64834.70	74987.16	76004.24	$6.93E+3$	102981.06	**50**	35000
APM	66030.05	79466.10	83524.21	$1.44E+4$	151458.17	**50**	35000
SR	**64599.65**	70508.33	71240.03	$3.90E+3$	83968.45	47	35000

Tabela 5.36: Volume final encontrado para a viga engastada e livre.

	Ref. [114]	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
B_1	3	3	3	3	3	3
B_2	3.1	3.1	3.1	3.1	3.1	3.1
B_3	2.6	2.6	2.6	2.6	2.6	2.6
B_4	2.3000	2.3077	2.4060	2.2947	2.2094	2.2837
B_5	1.8000	1.8140	1.8096	1.8250	2.0944	1.7532
H_1	60	60	60	60	60	60
H_2	55	55	55	55	60	55
H_3	50	50	50	50	50	50
H_4	45.5000	45.4312	44.9155	45.2153	44.0428	45.5507
H_5	35.0000	34.7314	35.0885	35.1191	31.9867	35.0631
V	64815	64834.38	65206.08	64834.70	66030.05	**64599.65**
ne	**10000**	3,000	35000	35000	35000	35000

Tabela 5.37: Variáveis de projeto dos melhores resultados no problema da viga engastada e livre.

5.11 Experimento 10 - Treliça de 10 barras

Esse é um problema muito conhecido que corresponde à minimização do peso de uma treliça de 10 barras esquematizada na Figura 5.6. A função objetivo é extremamente simples (linear em relação às variáveis de projeto) e as restrições são funções não-lineares implícitas das variáveis de projeto. As restrições envolvem as tensões em cada membro e os deslocamentos de cada nó.

A minimização é sujeita às restrições de tensão (normalizadas) descritas na Equação 5.1.

$$\frac{\sigma_i}{\overline{\sigma}} - 1 \leq 0, \quad i = 1, 2, \ldots, m \tag{5.1}$$

onde m é o número de barras.

A otimização do problema também está sujeita às restrições de deslocamento (normalizadas) apresentadas na Equação 5.2.

$$\frac{u_j}{\overline{u}} - 1 \leq 0, \quad j = 1, 2, \ldots, n \tag{5.2}$$

onde n é o número de graus de liberdade do modelo discreto. Uma análise deve ser feita com o objetivo de verificar se todas as restrições estão sendo satisfeitas. Nas demais treliças desse trabalho são utilizadas as mesmas expressões para as restrições.

As variáveis de projeto são as áreas da secção transversal de cada barra (A_i, $i = 1, 10$). As tensões são limitadas a $\pm$ 25 ksi e os deslocamentos a 2 in (polegadas), nas direções x e y. A massa específica do material é 0.1 lb/in^3, o módulo de Young é E = 10^4 ksi e a carga aplicada nos nós 2 e 4, no sentido vertical, é de 100 kips.

Dois casos são analisados: um com variáveis discretas ou com variáveis contínuas. Para o problema com variáveis discretas as áreas das secções transversais são escolhidas dentro do conjunto $\mathcal{S}$ que contem as 32 possibilidades que se seguem (in^2): 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.93, 3.13, 3.38, 3.47, 3.55, 3.63, 3.88, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.97, 11.50, 13.50, 14.20, 15.50, 16.90, 18.80, 19.90, 22.00, 26.50, 30.00, 33.50. No caso de utilizar variáveis contínuas, essas áreas variam de 0.1 in^2 até 33.50 in^2.

Os números de cálculos da função objetivo considerados são de 90000 e 280000 para os casos discreto e contínuo, respectivamente.

A Tabela 5.38 apresenta os valores encontrados para o peso final no caso discreto.

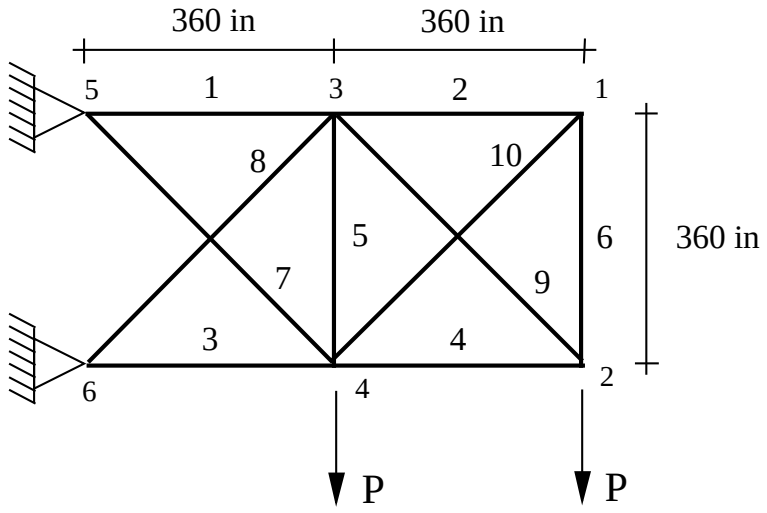

Figura 5.6: Treliça de 10 barras

Todos os algoritmos AG-SIA, bem como o APM, produziram o melhor resultado (5490.738).

A Tabela 5.40 mostra os valores obtidos no caso contínuo, onde o algoritmo híbrido AG-SIA encontrou a melhor solução (5061.1600404). Este mesmo algoritmo encontrou os melhores resultados em todos os casos considerados.

Nas tabelas 5.39 e 5.41 podem ser vistas as variáveis de projeto dos melhores resultados para todas técnicas nos casos discreto e contínuo, respectivamente.

	Melhor	Mediana	Média	dp	Pior	erf
AG-SIA^E	**5490.74**	5522.74	5536.76	$5.14E+1$	5741.98	**50**
AG-SIA^C	**5490.74**	5505.14	**5509.04**	$\mathbf{1.82E+1}$	**5574.54**	**50**
AG-SIA	**5490.74**	**5504.54**	5513.90	$2.56E+1$	5575.28	**50**
APM	**5490.74**	5558.74	5585.98	$1.48E+2$	6443.23	**50**
SR	5491.72	5648.46	5664.21	$9.64E+1$	6020.77	**50**

Tabela 5.38: Pesos finais encontrados para o problema da treliça de 10 barras – caso discreto.

	AG-SIA^E [35]	AG-SIA^C [35]	AG-SIA	APM	SR
1	33.50	33.50	33.50	33.50	33.50
2	1.62	1.62	1.62	1.62	1.62
3	22.90	22.90	22.90	22.90	22.90
4	14.20	14.20	14.20	14.20	15.50
5	1.62	1.62	1.62	1.62	1.62
6	1.62	1.62	1.62	1.62	1.62
7	7.97	7.97	7.97	7.97	7.97
8	22.90	22.90	22.90	22.90	22.00
9	22.00	22.00	22.00	22.00	22.00
10	1.62	1.62	1.62	1.62	1.62
W	**5490.74**	**5490.74**	**5490.74**	**5490.74**	5491.72

Tabela 5.39: Variáveis de projeto encontradas nas melhores soluções para o problema da treliça de 10 barras – caso discreto.

	Melhor	Mediana	Média	dp	Pior	erf
AG-SIA^E	5065.28	5077.27	5079.50	$1.09E+1$	5110.02	**50**
AG-SIA^C	5062.64	5080.65	5081.73	$1.23E+1$	5117.12	**50**
AG-SIA	**5061.16**	**5064.36**	**5068.85**	**7.78**	**5084.56**	**50**
APM	5062.12	5070.54	5133.22	$2.48E+2$	6430.55	**50**
SR	5061.71	5079.53	5077.67	$1.01E+1$	5101.17	**50**

Tabela 5.40: Valores encontrados para os pesos finais no problema da treliça de 10 barras – caso contínuo.

	AG-SIA^E	AG-SIA^C	AG-SIA	APM	SR
1	30.20108	30.17176	30.52684	30.95080	30.01400
2	0.10015	0.10138	0.10000	0.10000	0.10000
3	22.68561	22.83014	22.91574	22.92083	26.14460
4	16.14570	15.35501	15.48294	15.55024	15.29260
5	0.10017	0.10011	0.10000	0.10000	0.10000
6	0.57732	0.52462	0.54620	0.60959	0.55610
7	7.57211	7.55375	7.47594	7.46973	7.43980
8	21.30937	21.60682	21.01566	20.83562	21.00560
9	21.14685	21.32977	21.55362	21.35644	21.93900
10	0.10010	0.10021	0.10000	0.10000	0.10000
W	5065.28	5062.64	**5061.16**	5062.12	5061.71

Tabela 5.41: Variáveis de projeto para os melhores pesos finais encontrados para o problema da treliça de 10 barras – caso contínuo.

5.12 Experimento 11 - Treliça de 25 barras

Esse exemplo é um problema de otimização tradicional na engenharia estrutural. Seu objetivo é minimizar o peso de uma treliça de 25 barras mostrada na Figura 5.7. Esse estrutura é composta por barras de comprimento L_k, as áreas das secções transversais a_k são as variáveis de projeto, k é o índice das barras da estrutura e ρ é a massa específica do material. Assim, o peso da estrutura pode ser simplesmente escrito como

Para esta treliça, as barras são agrupadas, a fim de manter a simetria do projeto. Para cada grupo, uma única área A é utilizada para as secções transversais. Logo, o problema é escrito como: encontrar o conjunto de áreas $\mathbf{a} = \{A_1, A_2, \ldots, A_l\}$ que minimize o peso da estrutura.

A tensão em cada barra deve ter seu valor no intervalo $[-40, 40]$ ksi e os deslocamentos máximos dos nós 1 e 2 limitados a $0,35$ in, nas direções x e y. As áreas das secções transversais, que são variáveis de projeto, devem ter seus valores escolhidos no conjunto (em polegadas quadradas): 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,

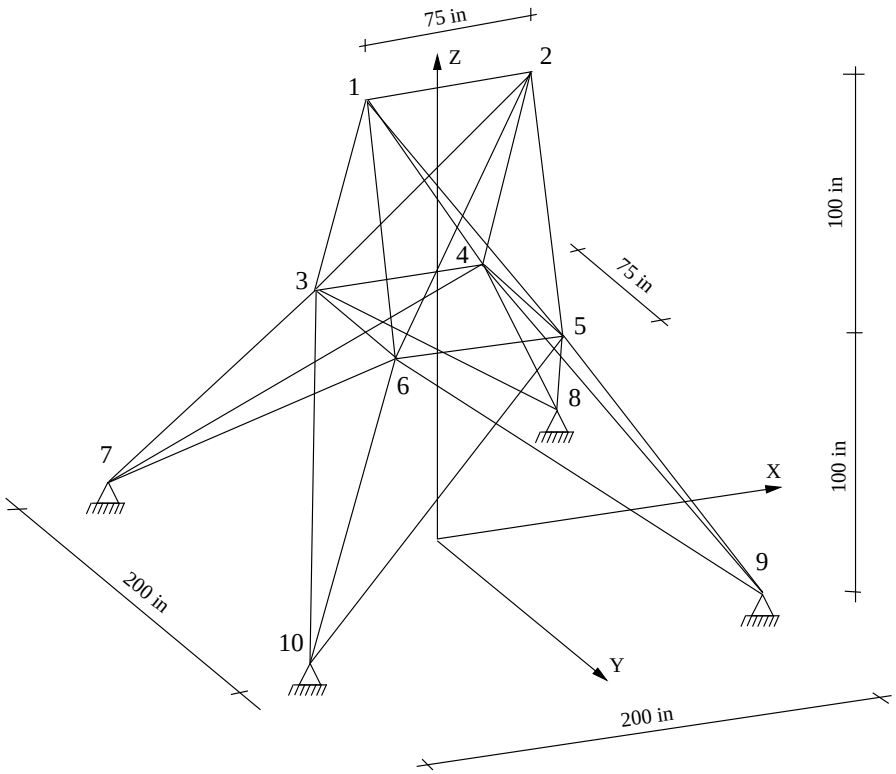

Figura 5.7: Treliça de 25 barras.

1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.1, 3.2. As variáveis de projeto são agrupadas conforme mostraa Tabela 5.43. O material do qual as barras são compostas tem massa específica igual a 0.1 lb/in^3 e o módulo Young é igual a 10^4 ksi. O carregamento aplicado sobre a estrutura é apresentada na Tabela 5.42. As Tabelas 5.44 e 5.45 apresentam comparações entre as melhores

Nó	F_x	F_y	F_z
1	1	−10.0	−10.0
2	0	−10.0	−10.0
3	0.5	0	0
6	0.6	0	0

Tabela 5.42: Carga para a treliça de 25 barras (em kips).

Grupo	Conectividades
A_1	1-2
A_2	1-4, 2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	3-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Tabela 5.43: Agrupamento de barras para a treliça de 25 barras.

soluções encontradas na literatura e com os algoritmos híbridos. Os resultados obtidos na Tabela 5.45 utilizaram apenas 800 cálculos da função objetivo. Os valores encontrados por [115] foram obtidos por um AG com codificação binária. Todos os algorítmos híbridos e o APM [36] encontraram um peso final igual a 484.854 lb. Esse resultado é o melhor entre os apresentados. Para obter esse resultado, os algoritmos fizeram 20000 cálculos da função objetivo. Utilizando 800 avaliações, o algoritmo AG-SIA^E encontrou 487.329 lbs. Este foi um pouco pior que o melhor

peso encontrado, porém muito melhores que os das referências [115] e [116] com o mesmo número de cálculos da função objetivo.

A referência [114] usa 30000 avaliações da função objetivo. Em [117] um algoritmo genético com codificação binária *steady-state* (não geracional) é proposto utilizando uma função de penalização constante e 40000 cálculos da função objetivo. O algoritmo "*Templeman* ampliado" foi utilizado na referência [116].

É importante destacar que: todas as soluções são factíveis, correspondem a soluções distintas (com exceção do melhor resultado) e os AG-SIAs, juntamente com o APM, obtiveram o menor peso para a estrutura.

As estatísticas para os resultados obtidos podem ser encontrados na Tabela 5.46.

Variables	Ref.[114]	Ref.[117]	APM [36]	AG-SIA	AG-SIA^C	AG-SIA
A_1	0.1	0.1	0.1	0.1	0.1	0.1
A_2	1.2	0.5	0.3	0.3	0.3	0.3
A_3	3.2	3.4	3.4	3.4	3.4	3.4
A_4	0.1	0.1	0.1	0.1	0.1	0.1
A_5	1.1	1.5	2.1	2.1	2.1	2.1
A_6	0.9	0.9	1.0	1.0	1.0	1.0
A_7	0.4	0.6	0.5	0.5	0.5	0.5
A_8	3.4	3.4	3.4	3.4	3.4	3.4
W	493.80	486.29	**484.854**	**484.854**	**484.854**	**484.854**
nfe	30,000	40,000	20,000	20,000	20,000	20,000

Tabela 5.44: Comparação dos resultados para o problema da treliça de 25 barras, onde W é o peso final da estrutura em libras.

Variables	Ref.[115]	Ref.[116]	AG-SIA^E	AG-SIA^C	AG-SIA
A_1	0.1	0.1	0.1	0.1	1.0
A_2	1.8	1.9	0.4	0.3	0.5
A_3	2.3	2.6	3.4	3.4	3.0
A_4	0.2	0.1	0.1	0.1	0.1
A_5	0.1	0.1	2.4	2.0	1.5
A_6	0.8	0.8	0.9	1.0	1.1
A_7	1.8	2.1	0.5	0.2	0.9
A_8	3.0	2.6	3.4	3.4	3.4
W	546.01	562.93	487.329	487.718	512.185
nfe	800	800	800	800	800

Tabela 5.45: Comparação dos resultados, utilizando 800 cálculos da função objetivo, para o problema da treliça de 25 barras, onde W é o peso final da estrutura em libras.

5.13 Experimento 12 - Treliça Anelar de 60 barras

Este exemplo corresponde à minimização do peso de uma treliça anelar de 60 barras discutida em [106, 118], mostrada na Figura 5.8. O raio externo do anel tem 100 in e o raio interno 90 in. A treliça é submetida a três casos de carregamento detalhados na Tabela 5.47. O material tem módulo de Young igual a 10^4 ksi e massa específica de 0.1 lb/in^3. O problema apresenta 198 restrições onde 180 se referem às tensões ($\sigma_i = 60$ ksi, $i = 1$ to 60) e 18 aos deslocamentos. Os limites dos deslocamentos são: 1.75 no nó 4, 2.25 no nó 13 e 2.75 no nó 19. As áreas das secções transversais das barras são variam continuamente de 0.5 in^2 até 5 in^2 e são agrupadas em 25 grupos como na Tabela 5.48.

Os valores estatísticos obtidos pelos algoritmos para o problema da treliça de 60 barras podem ser vistos na Tabela 5.49. O menor peso para a estrutura foi encontrado pelo AG-SIA e corresponde a 310.1365960lb. O AG-SIA^C obteve os melhores resultados em todos os outros casos considerados. A Tabela 5.50 mostra as variáveis de projeto para os melhores pesos obtidos pelos métodos aqui comparados.

	Melhor	Mediana	Média	dp	Pior	erf	na
Ref.[115]	546.01	–	–	–	–	–	**800**
Ref.[116]	562.93	–	–	–	–	–	**800**
AG-SIA^E	487.329	511.554	512.506	15.73	562.222	**50**	**800**
AG-SIA^C	487.718	514.850	515.520	11.31	548.501	**50**	**800**
AG-SIA	512.185	565.890	571.286	$145E+3$	677.668	**50**	**800**
Ref.[117]	486.29	–	–	–	–	–	40000
Ref.[114]	493.80	–	–	–	–	–	30000
APM [36]	**484.854**	–	**485.967**	–	**490.742**	–	20000
AG-SIA^E	**484.854**	**486.023**	486.196	**1.49**	492.554	**50**	20000
AG-SIA^C	**484.854**	**486.023**	486.485	2.10	496.783	**50**	20000
AG-SIA	**484.854**	488.515	491.007	7.35	519.160	**50**	20000

Tabela 5.46: Valores estatísticos encontrados para o peso final no problema da treliça de 25 barras.

Carga	nó	F_x	F_y
1	1	−10.0	0
	7	9.0	0
2	15	−8.0	3.0
	18	−8.0	3.0
3	22	−20.0	10.0

Tabela 5.47: Cargas para a estrutura treliça anelar de 60 barras (kips).

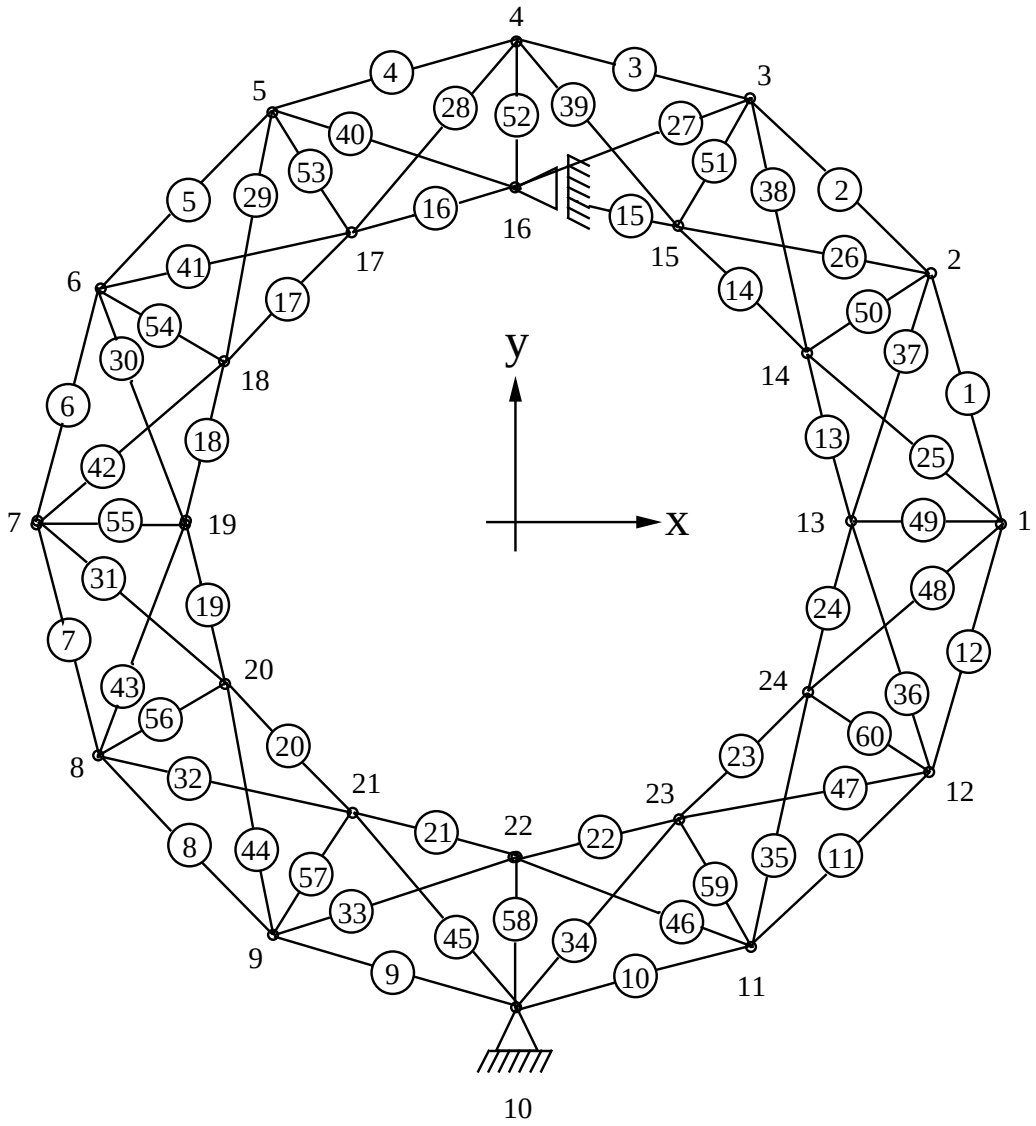

Figura 5.8: Treliça anelar de 60 barras.

5.14 Experimento 13 - Treliça de 72 Barras

Este problema é a minimização do peso de uma treliça de 72 barras, mostrada na Figura 5.9. As variáveis de projeto são as áreas das secções transversais das barras.

Grupo	Barras	Grupo	Barras
A_1	49 to 60	A_{14}	25, 37
A_2	1, 13	A_{15}	26, 38
A_3	2, 14	A_{16}	27, 39
A_4	3, 15	A_{17}	28, 40
A_5	4, 16	A_{18}	29, 41
A_6	5, 17	A_{19}	30, 42
A_7	6, 18	A_{20}	31, 43
A_8	7, 19	A_{21}	32, 44
A_9	8, 20	A_{22}	33, 45
A_{10}	9, 21	A_{23}	34, 46
A_{11}	10, 22	A_{24}	35, 47
A_{12}	11, 23	A_{25}	36, 48
A_{13}	12, 24		

Tabela 5.48: Agrupamento das variáveis de projeto para o problema da treliça anelar de 60 barras.

Essas áreas variam de 0.1 in^2 à 5 in^2. As 72 áreas são agrupadas em 16 conjuntos (variáveis de projeto) organizados como na Tabela 5.51. As restrições correspondem aos deslocamentos máximos de 0.25 in nos nós de 1 até 16 nas direções x e y e a tensão em cada barra é limitada ao intervalo de $[-25, 25]$ ksi. A massa específica é 0.1 lb/in^3 e o módulo de Young é igual a 10^4 ksi. Duas cargas, definidas na Tabela 5.52, são considerados para essa estrutura.

As Tabelas 5.53 e 5.54 apresentam uma comparação dos resultados obtidos pelos algoritmos. Ela contém as áreas das seções transversais de cada barra e os pesos finais para o problema da treliça de 72 barras. A melhor solução foi encontrada pelo AG-SIA (381.293 lb).

As referências [119, 120] usaram técnicas de otimização tradicionais. Conceitos de aproximação para otimização estrutural foram utilizados em [121]. Para este problema foram utilizados 35000 avaliações da função objetivo, que é a mesma usada pelo APM [36]. Os resultados de [114] foram obtidos com um AG.

	Melhor	Mediana	Média	dp	Pior	erf
CPM	315.4792	–	337.3386	–	382.9452	–
APM [36]	311.8757	–	333.0190	–	384.1990	–
AG-SIA^E	312.5987	315.7730	317.0082	4.49	331.9970	**20**
AG-SIA^C	310.8810	**313.8001**	**313.5535**	**1.19**	**315.2008**	**20**
AG-SIA	**310.1366**	314.0261	318.7236	8.84	340.0199	**20**

Tabela 5.49: Valores encontrados para o peso final da estrutura em anel.

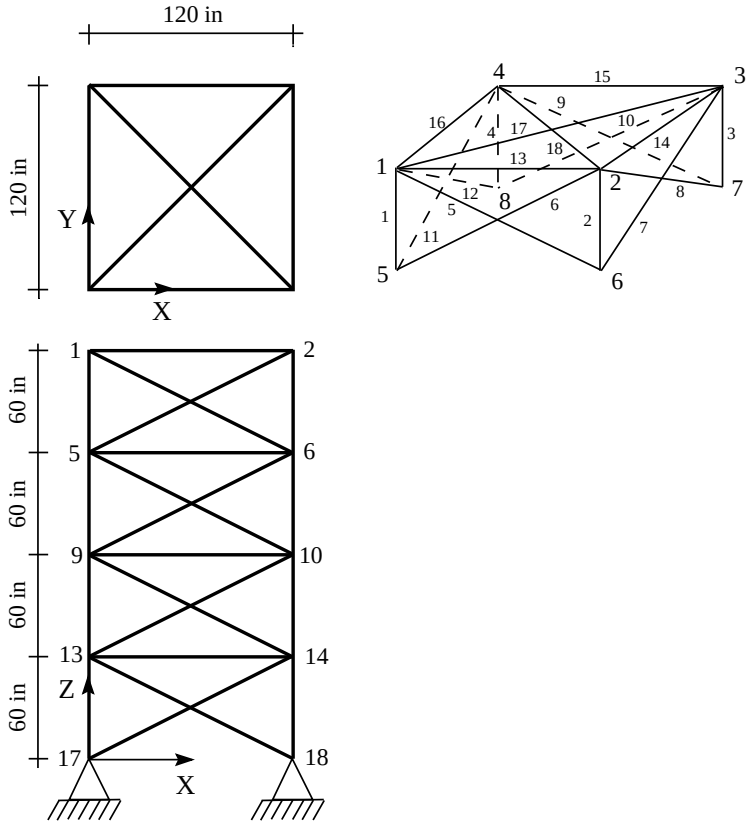

Figura 5.9: Treliça de 72 barras.

Os resultados estatísticos podem ser vistos na Tabela 5.55. As melhores soluções em todos os casos considerados foram obtidas pelo AG-SIA.

5.15 Experimento 14 - Problema de Confiabilidade

Nessa seção, um problema de um sistema de confiabilidade é resolvido e os resultados obtidos são comparados com os da literatura. Os objetivos do problema de otimização são três e devem ser resolvidos separadamente: maximização da confiança do sistema (R_s), minimização de seu custo total (C) e minimização de seu peso (W).

A formulação para este problema pode ser encontrada em [122, 123]. Em seu trabalho, Dhingra [122] apresenta as melhores soluções encontradas. O problema de otimização da confiabilidade corresponde à maximizar:

$$R_s(m,r) = \prod_{j=1}^{t} [1 - (1 - r_i)^{m_i}]$$

sujeito às restrições

$$g_1(m) = \sum_{i=1}^{4} (v_i m_i^2) - V_{lim} \leq 0$$

$$g_2(m,r) = \sum_{i=1}^{4} C(r_i) \left[m_i + exp\left(\frac{m_i}{4}\right) \right] - C_{lim} \leq 0$$

$$g_3(m) = \sum_{i=1}^{4} w_i m_i exp\left(\frac{m_i}{4}\right) - W_{lim} \leq 0$$

$$C(r_i) = \alpha_i \left(\frac{-T}{ln(r_i)} \right)^{\beta_i}$$

$$1 \leq m_i \leq 10 \qquad 0.5 \leq r_i \leq 1 - 10^{-6}$$

onde v_i é o produto do peso e volume por elemento no estágio i, w_i é o peso de cada componente no estágio i e $C(r_i)$ é o custo de cada componente com confiança r_j no estágio i with $i = 1, 2, \ldots, t$.

A otimização do custo corresponde à minimar:

$$C(m,r) = \sum_{i=1}^{4} C(r_i) \left[m_i + exp\left(\frac{m_i}{4}\right) \right]$$

sujeito às restrições

$$g_1(m) = \sum_{i=1}^{4}(v_i m_i^2) - V_{lim} \leq 0$$

$$g_2(m,r) = \prod_{j=1}^{t}[1-(1-r_i)^{m_i}] - R_{lim} \leq 0$$

$$g_3(m) = \sum_{i=1}^{4} w_i m_i exp\left(\frac{m_i}{4}\right) - W_{lim} \leq 0$$

Finalmente, a otimização do peso pode ser escrita como a minimização

$$W(m,r) = \sum_{i=1}^{4} w_i m_i exp\left(\frac{m_i}{4}\right)$$

sujeito às restrições

$$g_1(m) = \sum_{i=1}^{4}(v_i m_i^2) - V_{lim} \leq 0$$

$$g_2(m,r) = \sum_{i=1}^{4} C(r_i)\left[m_i + exp\left(\frac{m_i}{4}\right)\right] - C_{lim} \leq 0$$

$$g_3(m) = \prod_{j=1}^{t}[1-(1-r_i)^{m_i}] - R_{lim} \leq 0$$

Dhingra [122] utilizou um método de programação matemática não-linear e Li e Gen [123] usaram um AG. Foram utilizadas 50 execuções independentes contra 10 de [123]. São apresentados nas Tabelas 5.56, 5.57 e 5.58 os resultados obtidos com 15000 cálculos da função objetivo para os três problemas de otimização. Os melhores resultados, em geral, foram obtidos pelo AG-SIA^C. Quando se considera o problema de minimização do custo, o AG-SIA^E encontra as melhores soluções. Todos os algoritmos produziram os mesmos resultados quando o objetivo foi minimizar o peso.

Variables	APM [36]	CPM	AG-SIA^E	AG-SIA^C	AG-SIA
A_1	1.1202346	1.1906158	1.2503042	1.1726822	1.1473206
A_2	2.0219941	2.2771261	2.2284460	2.0621954	2.0305310
A_3	0.5087977	0.6055718	0.5622851	0.5100718	0.5000035
A_4	1.7272727	1.5733138	1.4938727	1.7654699	2.0943461
A_5	1.5205279	1.3753666	1.5821010	1.6658467	1.5536466
A_6	0.5263930	0.5087977	0.5561846	0.5720687	0.5755588
A_7	1.9032258	1.9340176	1.9705521	1.9062078	1.9149985
A_8	2.1275660	2.0527859	1.9176961	1.9484566	1.9744345
A_9	0.9882698	1.2390029	1.0018366	1.0510688	1.0299557
A_10	2.0527859	1.8196481	1.8867111	1.7466194	1.7932299
A_11	2.0527859	1.6392962	1.9104339	1.6457268	1.6610834
A_12	0.7243402	0.5263930	0.5052771	0.5153524	0.5022349
A_13	1.9604106	2.1979472	2.0828714	2.0797600	2.0271904
A_14	1.2302053	1.2346041	1.2470049	1.2616581	1.2514830
A_15	0.9970674	1.0498534	1.2055308	1.1398907	1.0486630
A_16	0.6055718	0.7595308	0.6175502	0.6888714	0.6256583
A_17	0.7287390	0.6143695	0.7475708	0.7888196	0.7737988
A_18	1.0938416	1.1202346	1.1097740	1.0401628	1.0545479
A_19	1.1158358	1.1158358	1.1202651	1.1605230	1.1312696
A_20	1.1686217	1.1554252	1.1848861	1.1499384	1.1418152
A_21	1.0674487	1.1862170	1.0072308	1.0097773	1.1089411
A_22	1.0630499	1.0718475	1.0644257	1.0680071	1.0525820
A_23	0.5879765	0.7903226	0.6158278	0.8199369	0.6823657
A_24	1.0674487	1.2653959	1.1064241	1.0693682	1.1227735
A_25	1.2697947	1.2697947	1.2981291	1.2743438	1.2486593
W	311.8757625	315.4792513	312.5986895	310.8809775	**310.1365960**

Tabela 5.50: Variáveis de projeto das melhores soluções para a treliça anelar.

Grupo	Barra
A_1	1, 2, 3, 4
A_2	5, 6, 7, 8, 9, 10, 11, 12
A_3	13, 14, 15, 16
A_4	17,18
A_5	19, 20, 21, 22
A_6	23, 24, 25, 26, 27, 28, 29, 30
A_7	31, 32, 33, 34
A_8	35,36
A_9	37, 38, 39, 40
A_{10}	41, 42, 43, 44, 45, 46, 47, 48
A_{11}	49, 50, 51, 52
A_{12}	53,54
A_{13}	55, 56, 57, 58
A_{14}	59, 60, 61, 62, 63, 64, 65, 66
A_{15}	67, 68, 69, 70
A_{16}	71,72

Tabela 5.51: Agrupamento das barras para a treliça de 72 barras.

Carga	Nó	F_x	F_y	F_z
1	1	5	5	−5
2	1	0	0	−5
	2	0	0	−5
	3	0	0	−5
	4	0	0	−5

Tabela 5.52: Carga para a treliça de 72 barras (kips).

Var.	Ref.[120]	Ref.[119]	Ref.[121]	Ref.[114]	APM [36]
A_1	0.161	0.1492	0.1585	0.161	0.15500
A_2	0.557	0.7733	0.5936	0.544	0.54534
A_3	0.377	0.4534	0.3414	0.379	0.27496
A_4	0.506	0.3417	0.6076	0.521	0.51853
A_5	0.611	0.5521	0.2643	0.535	0.60365
A_6	0.532	0.6084	0.5480	0.535	0.66607
A_7	0.100	0.1000	0.1000	0.103	0.10159
A_8	0.100	0.1000	0.1509	0.111	0.13008
A_9	1.246	1.0235	1.1067	1.310	1.19954
A_{10}	0.524	0.5421	0.5793	0.498	0.47368
A_{11}	0.100	0.1000	0.1000	0.110	0.10059
A_{12}	0.100	0.1000	0.1000	0.103	0.10945
A_{13}	1.818	1.4636	2.0784	1.910	1.95307
A_{14}	0.524	0.5207	0.5034	0.525	0.51653
A_{15}	0.100	0.1000	0.1000	0.122	0.10000
A_{16}	0.100	0.1000	0.1000	0.103	0.10105
W	381.2	395.97	388.63	383.12	387.036

Tabela 5.53: Comparação dos resultados para a treliça de 72 barras. W é o peso final em lb.

Var.	AG-SIA^E	AG-SIA^C	AG-SIA
A_1	0.17662	0.17765	0.15745
A_2	0.50777	0.56793	0.54583
A_3	0.47313	0.48696	0.35137
A_4	0.57060	0.53717	0.58946
A_5	0.41377	0.62115	0.54317
A_6	0.46883	0.52298	0.53262
A_7	0.14094	0.11038	0.10192
A_8	0.11203	0.20989	0.10358
A_9	1.36150	1.06770	1.32186
A_{10}	0.48918	0.43852	0.51468
A_{11}	0.10950	0.11557	0.10250
A_{12}	0.11671	0.22485	0.11831
A_{13}	2.10058	2.15986	1.92280
A_{14}	0.60663	0.54524	0.49524
A_{15}	0.14099	0.11610	0.10014
A_{16}	0.12831	0.11145	0.10120
W	392.679	395.874	**381.293**

Tabela 5.54: Comparação dos resultados para a treliça de 72 barras. W é o peso final em lb.

	Melhor	Mediana	Média	dp	Pior	erf
APM [36]	387.036	–	402.588	–	432.953	–
AG-SIA^E	392.6791785	402.0249273	403.8586862	9.53	434.4808421	**20**
AG-SIA^C	395.8735023	402.2874897	405.1349864	8.60	424.3371334	**20**
AG-SIA	**381.292964**	**384.802909**	**385.160978**	**2.61**	**391.491002**	**20**

Tabela 5.55: Pesos finais encontrados para a treliça de 72 barras.

	Melhor	Mediana	Média	dp	Pior	erf
NMP [122]	0.99961	-	-	-	-	-
AG ($\gamma = 1$) [123]	**0.999955**	-	-	-	-	-
AG ($\gamma = 2$) [123]	0.999954	-	-	-	-	-
AG ($\gamma = 4$) [123]	0.999954	-	-	-	-	-
AG-SIA	0.999954	0.999945	0.999939	$1.98E-5$	0.999880	**50**
AG-SIA^C	**0.999955**	**0.999946**	**0.999946**	**6.60E − 6**	**0.999917**	**50**
AG-SIA	**0.999955**	0.999918	0.999885	$2.65E-5$	0.998785	**50**

Tabela 5.56: Resultados encontrados para a maximização da confiabilidade R_s.

	Melhor	Mediana	Média	dp	Pior	erf
NMP [122]	20.7252	-	-	-	-	-
AG-SIA	20.14028	**20.37048**	**20.35425**	**1.82E − 1**	**20.80367**	**50**
AG-SIA^C	20.14380	20.40202	20.38474	$1.87E-1$	20.84020	**50**
AG-SIA	**20.13973**	21.23375	21.65518	1.95	26.67229	**50**

Tabela 5.57: Resultados encontrados para a minimização do custo C.

	Melhor	Mediana	Média	dp	Pior	erf
NMP [122]	34.6687	-	-	-	-	-
AG-SIA	**34.668686**	**34.668686**	**34.668686**	**0.00**	**34.668686**	**50**
AG-SIA^C	**34.668686**	**34.668686**	**34.668686**	**0.00**	**34.668686**	**50**
AG-SIA	**34.668686**	**34.668686**	**34.668686**	**0.00**	**34.668686**	**50**

Tabela 5.58: Resultados encontrados para a minimização do peso W.

Capítulo 6

Conclusões

Neste trabalho foram propostas, implementadas e avaliadas computacionalmente várias possibilidades de hibridização de algoritmos genéticos com sistemas imunológicos artificiais para a resolução de problemas de otimização com restrições. Os resultados obtidos foram utilizados para verificar a eficiência dos algoritmos aqui apresentados através de comparações com valores obtidos na literatura por outros métodos.

Problemas testes das mais diversas áreas foram utilizados nas comparações. Primeiramente, testes comuns da literatura foram executados. Depois, cinco problemas de engenharia mecância. Por fim, foram comparados resultados de problemas de otimização estrutural e de sistemas de confiabilidade. Os algoritmos híbridos AG-SIA mostraram, através dos resultados apresentados e comparados, ser algoritmos eficientes e robustos. Boas soluções são encontradas mesmo sendo os problemas de naturezas muito diferentes. Os testes são compostos de problemas com variáveis contínuas, discretas/inteiras ou mistas, além de serem, normalmente, não-lineares.

Apesar das diversas aplicações e variações do algoritmo híbrido apresentadas no presente trabalho, muitas são as propostas para estudos futuros. Por ser um algoritmo com uma estrutura mais desacoplada com relação às meta-heurísticas que o compõem, o AG-SIA poderia ser implementado e testado utilizando uma arquitetura de computadores distribuída. Uma outra opção que poderia ser utilizada em conjunto com essa, mas não necessariamente dessa forma, seria manter as populações do AG e do SIA fixas por algumas gerações, ou seja, sem que os indivíduos de

cada população sejam enviados para a outra. Com isso, espera-se que os algoritmos específicos tenham mais oportunidade de trabalhar com seus indivíduos antes de trocá-los. Além disso, no caso de paralelismo, os pontos necessários de sincronização poderiam diminuir. Por fim, diversos outros problemas estruturais de maior porte poderiam ser solucionados pelos algoritmos aqui apresentados.

Uma outra proposta a ser considerada é a utilização de outros métodos de SIA para a resolução tanto do problema de aproximar os indivíduos da região factível do problema (AG-SIA^E e AG-SIA^C), quanto para minimizar as restrições dos elementos infactíveis. Uma idéia seria utilizar o opt-aiNET ou o omni-aiNET como alternativas ao CLONALG (algoritmo utilizado neste trabalho).

Referências Bibliográficas

[1] FRIENLANDER, A. *Elementos de Programação Não-linear*. [S.l.]: Editora Unicamp, 1994.

[2] REIS, M. A. dos. *Forense Computacional e sua Aplicação em Segurança Imunológica*. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas - SP, 1 2003.

[3] CASTRO, L. N. de. *Engenharia Imunológica: Desenvolvimento e Aplicações de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais*. Tese (Doutorado) — Universidade de Campinas - UNICAMP, 5 2001.

[4] KARONEN, I. Drawn in inkscape. 2006. Disponível em: <http://en.wikipedia.org/wiki/Clonal_selection>.

[5] HOLLAND, J. H. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, v. 2, n. 2, p. 88–105, 1973.

[6] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[7] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1992.

[8] FARMER, J. D.; PACKARD, N. H.; PERELSON, A. S. The immune system, adaptation, and machine learning. *Physica D*, v. 22, p. 187–204, 1986.

[9] KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, v. 220, 4598, p. 671–680, 1983.

[10] COELLO, C. A. C.; CORTÉS, N. C. A parallel implementation of an artificial immune system to handle constraints in genetic algorithms: Preliminary results. In: *Congress on Evolutionary Computation 2002 (CEC'2002)*. [S.l.]: IEEE Service Center, 2002. p. 819–824.

[11] COELLO, C. A. C.; CORTÉS, N. C. Hybridizing a genetic algorithm with an artificial immune system for global optimization. *Engineering Optimization*, v. 36, n. 5, p. 607–634, October 2004.

[12] HAJELA, P.; LEE, J. Constrained genetic search via schema adaptation. an immune network solution. In: *1st World Congress of Stuctural and Multidisciplinary Optimization*. Goslar, Germany: Pergamon Press, 1995. p. 915–920.

[13] HAJELA, P.; LEE, J. Constrained genetic search via schema adaptation. An immune network solution. *Structural Optimization*, v. 12, p. 11–15, 1996.

[14] HAJELA, P.; YOO, J. S. Immune network modelling in design optimization. In: CORNE, D.; DORIGO, M.; GLOVER, F. (Ed.). *New Ideas in Optimization*. [S.l.]: McGraw-Hill, 1999. p. 167–183.

[15] YOO, J. S.; HAJELA, P. Immune network simulations in multicriterion design. *Structural Optimization*, v. 18, p. 85–94, 1999.

[16] BORTOLOSSI, H. J. *Cálculo Diferencial a Várias Variáveis - Uma Introdução à Teoria de Otimização*. [S.l.]: Editora PUC Rio, 2003.

[17] BENNATON, J. F. Fermat e o início da história dos problemas de otimização. 2001. Disponível em: <http://www.lps.usp.br/neo/jocelyn/historia_jocelyn.htm>.

[18] FREIRE, W. P. *Um Algoritmo de Direções Viáveis para Otimização Convexa Não Diferenciável*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro - COPPE, 2005.

[19] LEMONGE, A. C. C. *Aplicação de Algoritmos Genéticos em Otimização Estrutural*. Tese (Doutorado) — Prog. de Engenharia Civil, COPPE/UFRJ, 1999.

[20] J.S.ARORA. *Introduction to Optimum Design*. [S.l.]: McGraw-Hill, New Jersey, 1989.

[21] JR, J. M. G. e W. W. *Análise de Estruturas Reticuladas*. [S.l.]: Editora Guanabara Dois, 1981.

[22] BATHE, K.-J. *Finite Element Procedures in Engineering Analysis*. [S.l.]: Prentice Hall, Inc., Eaglewood Cliffs, New Jersey, 1982.

[23] LINDEN, R. *Algoritmos Genéticos: Uma Importante Ferramenta da Inteligência Computacional*. [S.l.]: Brasport, 2006.

[24] PITTS, W.; MCCULLOUGH, W. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Cambridge, MA, USA: MIT Press, 1943.

[25] FORBES, N. *Imitation of Life: How Biology Is Inspiring Computing*. Cambridge, MA, USA: MIT Press, 2004.

[26] COLORNI, A.; DORIGO, M.; MANIEZZO, V. Distributed optimization by ant colonies. In: *Proceedings of the European Conference on Artificial Life*. Paris, France: Elsevier, 1991. p. 134–142.

[27] DORIGO, M.; MANIEZZO, V.; COLORNI, A. *The Ant System: An Autocatalytic Optimizing Process*. Milano, Italy, 1991.

[28] EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, p. 39–43, Oct 1995.

[29] BOETTCHER, S. Extremal optimization of graph partitioning at the percolation threshold. *MATH.GEN.*, v. 32, p. 5201, 1999.

[30] BOETTCHER, S.; PERCUS, A. G. Extremal optimization: Methods derived from co-evolution. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco: Morgan Kaufmann, 1999.

[31] DARWIN, C. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. [S.l.: s.n.], 1859.

[32] MITCHELL, M. *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT Press, 1996. ISBN 0-262-13316-4.

[33] FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley, 1966.

[34] RECHENBERG, I. *Cybernetic Solution Path of an Experimental Problem*. Farnborough, Hants, U.K., 1965.

[35] BERNARDINO, H. S.; BARBOSA, H. J. C.; LEMONGE, A. C. C. A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*. Singapore: IEEE Press, 2007.

[36] LEMONGE, A. C. C.; BARBOSA, H. J. C. An adaptive penalty scheme for genetic algorithms in structural optimization. *Int. Journal for Numerical Methods in Engineering*, v. 59, n. 5, p. 703–736, 2004.

[37] SINGH, G.; DEB, K. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: *Genetic And Evolutionary Computation Conference, GECCO-2006*. Seattle, WA, USA: [s.n.], 2006.

[38] LIANG, J. J. et al. *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*. [S.l.], 2006. 1-24 p. Disponível em: <http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC06/cec2006.zip>.

[39] QIAN, Y. Image interpretation with fuzzy-graph based genetic algorithm. *International Conference on Image Processing*, v. 1, p. 545–549, 1999.

[40] NASSIF, N.; KAJL, S.; SABOURIN, R. Optimization of HVAC control system strategy using two-objective genetic algorithm. *International Journal of HVAC&R Research*, v. 11, 2005.

[41] BORGES, F. P. e S. *Otimização via Algoritmo Genético do Processo Construtivo de Estruturas de Concreto Submetidos á Retração Restringida Tendo em Vista a Fissuração nas Primeiras Idades*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Abril 2002.

[42] BLICKLE, T.; THIELE, L. *A Comparison of Selection Schemes Used in Genetic Algorithms*. Gloriastrasse 35, 8092 Zurich, Switzerland, 1995.

[43] BARBOSA, H. J. C.; LEMONGE, A. C. C. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In: *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. New York: Morgan Kaufmann Publishers, 2002. p. 287–294. ISBN 1-55860-878-8.

[44] RUNARSSON, T. P.; YAO, X. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, v. 4, n. 3, p. 284–294, September 2000.

[45] RUNARSSON, T. Approximate evolution strategy using stochastic ranking. In: YEN, G. G. et al. (Ed.). *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*. Vancouver, BC, Canada: IEEE Press, 2006. p. 745–752. ISBN 0-7803-9487-9.

[46] HADJ-ALOUANE, A. B.; BEAN, J. C. A genetic algorithm for multiple-choice integer program. *Operational Research*, v. 45, p. 92–101, 1997.

[47] SMITH, A. E.; TATE, D. M. Genetic optimization using a penalty function. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*. [S.l.]: Ed. Los Altos, 1993. p. 499–505.

[48] COIT, D.; SMITH, A.; TATE, D. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *Journal on Computing*, p. 173–182, 1996.

[49] KULTUREL-KONAK, S. et al. Exploiting tabu search memory in constrained problems. *INFORMS J. on Computing*, INFORMS, v. 16, n. 3, p. 241–254, 2004.

[50] CASTRO, L. N. de; TIMMIS, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. 1. ed. [S.l.]: Springer, 2002.

[51] ALEXANDRINO, J. L.; FILHO, E. C. de B. C. Investigation of a new artificial immune system model applied to pattern recognition. In: *HIS '06: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*. Washington, DC, USA: IEEE Computer Society, 2006. p. 16. ISBN 0-7695-2662-4.

[52] CASTRO, L. N. de; TIMMIS, J. Artificial immune systems: A novel approach to pattern recognition. In: CORCHADO, L. A. J.; FYFE, C. (Ed.). *Artificial Neural Networks in Pattern Recognition*. [S.l.]: University of Paisley, 2002. p. 67–84. ISBN 84-95721-22-8.

[53] DENG, J.; JIANG, Y.; MAO, Z. An artificial immune network approach for pattern recognition. *International Conference on Natural Computation*, IEEE Computer Society, Los Alamitos, CA, USA, v. 3, p. 635–640, 2007.

[54] FIGUEREDO, G. P.; EBECKEN, N. F. F.; BARBOSA, H. J. C. The SUPRAIC algorithm: A suppression immune based mechanism to find a representative training set in data classification tasks. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2007. p. 59–70.

[55] LEUNG, K.; CHEONG, F.; CHEONG, C. Generating compact classifier systems using a simple artificial immune system. *IEEE Trans Syst Man Cybern B Cybern*, v. 37, n. 5, p. 1344–56, 2007.

[56] SECKER, A.; FREITAS, A. A.; TIMMIS, J. AISEC: An Artificial Immune System for E-mail Classification. In: SARKER, R. et al. (Ed.). *Proceedings of the Congress on Evolutionary Computation*. Canberra. Australia, 2003. p. 131–139.

[57] TANG, N.; VEMURI, V. R. An artificial immune system approach to document clustering. In: *SAC'05: Proceedings of the 2005 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2005. p. 918–922. ISBN 1-58113-964-0.

[58] HAAG, C. R. et al. An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions. In: *GECCO '07: Proc. of the 2007 Genetic and Evolutionary Computation Conference*. New York, NY, USA: ACM, 2007. p. 2717–2724. ISBN 978-1-59593-698-1.

[59] KIM, J.; ONG, A.; OVERILL, R. Design of an artificial immune system as a novel anomaly detector for combating financial fraud in retail sector. In: *Congress on Evolutionary Computation*. [S.l.: s.n.], 2003.

[60] XIE, H.; HUI, Z. An intrusion detection architecture for ad hoc network based on artificial immune system. In: *PDCAT '06: Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies*. Washington, DC, USA: IEEE Computer Society, 2006. p. 1–4. ISBN 0-7695-2736-1.

[61] ZHI-TANG, L.; YAO, L.; LI, W. A novel fuzzy anomaly detection algorithm based on artificial immune system. In: *HPCASIA '05: Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*. Washington, DC, USA: IEEE Computer Society, 2005. p. 479. ISBN 0-7695-2486-9.

[62] GLICKMAN, M.; BALTHROP, J.; FORREST, S. A machine learning evaluation of an artificial immune system. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 13, n. 2, p. 179–212, 2005. ISSN 1063-6560.

[63] TIMMIS, J. *On Parameter Adjustment of the Immune Inspired Machine Learning Algorithm AINE*. Canterbury, Kent. CT2 7NF., November 2000. 18 p.

[64] LAU, H. Y. K.; KO, A. An immuno robotic system for humanitarian search and rescue. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2007. p. 191–203.

[65] LEE, J. et al. Clonal selection algorithms for 6-DOF PID control of autonomous underwater vehicles. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2007. p. 182–190.

[66] OATES, R. et al. The application of a dendritic cell algorithm to a robotic classifier. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2007. p. 204–215.

[67] OWENS, N. D. et al. On immune inspired homeostasis for electronic systems. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2007. p. 216–227.

[68] ACAN, A. Clonal selection algorithm with operator multiplicity. In: *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 2004. p. 1909–1915.

[69] COELHO, G. P.; ZUBEN, F. J. V. omni-aiNet: An immune-inspired approach for omni optimization. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2006. p. 294–308.

[70] CASTRO, L. N. de; TIMMIS, J. An artificial immune network for multimodal function optimization. In: *Proc. of the 2002 IEEE World Congress on Computational Intelligence*. Honolulu, Hawaii, USA: [s.n.], 2002. I, p. 669–674.

[71] CASTRO, L. N. de; ZUBEN, F. J. V. ainet: An artificial immune network for data analysis in data mining: A heuristic approach. In: _____. Idea Group Publishing, 2002. cap. 12, p. 231–259. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/lnunes/DMHA.pdf>.

[72] CASTRO, L. N. de; ZUBEN, F. J. V. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 3, p. 239–251, 2002.

[73] FRESCHI, F.; REPETTO, M. Multiobjective optimization by a modified artificial immune system algorithm. In: *Proceedings of the International Conference on Artificial Immune Systems*. [S.l.: s.n.], 2005. p. 248–261.

[74] GASPAR, A.; COLLARD, P. Two models of immunization for time dependent optimization. In: *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.: s.n.], 2000. v. 1, p. 113–118.

[75] J.Y.WU. Artificial immune system for solving constrained global optimization problems. In: *Artificial Life 2007, ALIFE '07*. Honolulu, HI: [s.n.], 2007. p. 92–99.

[76] PANIGRAHI, B. et al. A clonal algorithm to solve economic load dispatch. *Electric Power Systems Research*, v. 77, p. 1381–1389, August 2006.

[77] TAN, K. C.; MAMUN, C. K. G. A. A.; EI, E. Z. An evolutionary artificial immune system for multi-objective optimization. *European Journal of Operational Research*, April 2007.

[78] GOMES, L. de C. T. et al. Copt-aiNet and the gene ordering problem. In: LIFSCHITZ, S. et al. (Ed.). *Workshop Brasileiro de Bioinformática*. [S.l.: s.n.], 2003. p. 28–37.

[79] KEKO, H.; SKOK, M.; SKRLEC, D. Artificial immune systems in solving routing problems. In: *The International Conference on Computer as a Tool - EUROCON*. [S.l.: s.n.], 2003. p. 62 – 66.

[80] TSAI, J.-T. et al. Improved immune algorithm for global numerical optimization and job-shop scheduling problems. *Applied Mathematics and Computation*, v. 194, p. 406–424, 2007.

[81] TIZARD, I. R. *Immunology: Introduction*. [S.l.]: Saunders College Publishing, 1995.

[82] CORMACK, D. H. *Essential Histology*. [S.l.]: Lippincott Williams & Wilkins, 2001.

[83] JR., C. A. J. et al. *Imunobiologia: O Sistema Imune na Saúde e na Doença*. [S.l.]: Artmed, 2006.

[84] OPREA, M.; PERELSON, A. S. Somatic mutation leads to efficient affinity maturation when centrocytes recycle back to centroblasts. Santa Fe Institute, 1997.

[85] FORSDYKE, D. R. The origins of the clonal selection theory of immunity as a case study for evaluation in science. *The FASEB Journal*, v. 9, p. 164–166, 1995.

[86] BURNET, F. M. *The Clonal Selection Theory of Acquired Immunity*. [S.l.]: Cambridge University Press, 1959.

[87] JERNE, N. K. Towards a network theory of the immune system. *Ann Immunol (Paris)*, v. 125C, n. 1-2, p. 373–89, 1974.

[88] DASGUPTA, D. *Artificial Immune Systems and Their Applications*. 1. ed. [S.l.]: Springer, 1998.

[89] CASTRO, L. N. de. The clonal selection algorithm with engineering applications. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'2000)*. [S.l.: s.n.], 2000. p. 36–37.

[90] CASTRO, L. N. de; ZUBEN, F. J. V. *Artificial Immune Systems: Basic Theory and Applications*. Campinas - SP, 1999. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/tr_dca/trdca0199.pdf>.

[91] CASTRO, L. N. de; ZUBEN, F. J. V.; JR., G. A. de D. The construction of a boolean competitive neural network using ideas from immunology. *Neurocomputing*, v. 50, p. 51–85, 2003.

[92] CASTRO, L. N. de; ZUBEN, F. J. V. An evolutionary immune network for data clustering. In: *Proceedings of IEEE SBRN*. [S.l.: s.n.], 2000. p. 84–89.

[93] KNIDEL, H.; CASTRO, L. N. de; ZUBEN, F. J. V. Rabnet: a real-valued antibody network for data clustering. In: *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005. p. 371–372.

[94] FRANÇA, F. O. de; ZUBEN, F. J. V.; CASTRO, L. N. de. An artificial immune network for multimodal function optimization on dynamic environments. In: *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005. p. 289–296.

[95] HOLLAND, P. W. H. et al. Gene duplications and the origins of vertebrate development. *Dev Suppl*, 1994.

[96] OHNO, S. *Evolution by gene duplication*. [S.l.]: Springer-Verlag, 1970. Unknown Binding.

[97] BARBOSA, H. J. C.; LAVOR, C. C.; RAUPP, F. M. P. A GA-simplex hybrid algorithm for global minimization of molecular potential energy functions. *Annals of Operations Research*, p. 189–202, 2005.

[98] BERNARDINO, H. S.; BARBOSA, H. J. C.; LEMONGE, A. C. C. Constraint handling in genetic algorithms via artificial immune systems. In: GRAHL, J. (Ed.). *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2006)*. Seattle, WA, USA: [s.n.], 2006. Disponível em: <http://www.cs.bham.ac.uk/~wbl/biblio/gecco2006etc/papers/lbp134.pdf>.

[99] BERNARDINO, H. S.; BARBOSA, H. J. C.; LEMONGE, A. C. C. Um algoritmo genético híbrido para problemas de otimização com restrições. In: *Proceedings of the XXVII Iberian Latin American Congress on Computational Methods in Engineering - CILAMCE'2006*. Belém, Pará, Brazil: [s.n.], 2006.

[100] YEN, J. et al. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, v. 28, n. 2, p. 173–191, April 1998.

[101] HARP, S.; SAMAD, T.; GUHA, A. Towards the genetic synthesis of neural networks. In: SCHAFFER, J. (Ed.). *Proc. of the Third Int. Conf. on Genetic Algorithms and their Applications*. [S.l.]: Morgan Kaufmann, San Mateo, CA, 1989. p. 360–369.

[102] PETROWSKI., A. A clearing procedure as a niching method for genetic algorithms. In: *Proc. Third IEEE International Conference on Evolutionary Computation*. [S.l.: s.n.], 1996. p. pages 798–803.

[103] BERNARDINO, H. S. et al. A new hybrid AIS-GA for constrained optimization problems in mechanical engineering (por aparecer). In: *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*. Hong Kong: IEEE Press, 2008.

[104] LIANG, S. *The Java Native Interface - Programmer's Guide and Specification*. [S.l.]: Addison Wesley Longman, 1999.

[105] KOZIEL, S.; MICHALEWICZ, Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, v. 7, n. 1, p. 19–44, 1999.

[106] BARBOSA, H. J. C.; LEMONGE, A. C. C. A new adaptive penalty scheme for genetic algorithms. *Information Sciences*, v. 156, p. 215–251, 2003.

[107] COSTA, L.; OLIVEIRA, P. Evolutionary algorithm approach to the solution of mixed integer non-linear programming problems. *Computers and Chemical Engineering*, v. 25, p. 257–266, 2001.

[108] MOHAN, C.; NGUYEN, H. T. A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems. *Comput. Optim. Appl.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 14, n. 1, p. 103–132, 1999. ISSN 0926-6003.

[109] EFRÉN, M.; COELLO, C. A. C.; RICARDO, L. Engineering optimzation using a simple evolutionary algorithm. In: *15th Intl. Conf. on Tools with Art. Intelligence – ICTAI'2003*. CA, USA: [s.n.], 2003. p. 149–156.

[110] COELLO, C. A. C. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Computers in Industry*, v. 41, n. 2, p. 113–127, January 2000.

[111] DEB, K. GeneAS: A robust optimal design technique for mechanical component design. In: *Evolutionary Algorithms in Engineering Applications*. [S.l.]: Springer-Verlarg, Berlin, 1997. p. 497–514.

[112] KANNAN, B.; KRAMER, S. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, n. 116, p. 405–411, 1995.

[113] SANDGREN, E. Nonlinear integer and discrete programming in mechanical design. In: *Proc. of the ASME Design Technology Conference*. Kissimee, FL: [s.n.], 1988. p. 95–105.

[114] ERBATUR, F. et al. Optimal design of planar and space structures with genetic algorithms. *Computers & Structures*, v. 75, p. 209–224, 2000.

[115] KRISHNAMOORTY, C.; RAJEEV, S. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, v. 118, n. 5, 1992.

[116] ZHU, D. An improved Templeman's algorithm for optimum design of trusses with discrete member sizes. *Engineering Optimization*, v. 9, p. 303–312, 1986.

[117] WU, S.; CHOW, P. Steady-state genetic algorithms for discrete optimization of trusses. *Computers & Structures*, v. 56, n. 6, p. 979–991, 1995.

[118] PATNAIK, S.; HOPKINS, D.; CORONEOS, R. Structural optimization with aproximate sensitivities. *Computer & Structures*, v. 58, n. 2, p. 407–418, 1996.

[119] GELLATLY, R.; BERKE, L. *Optimal Structural Design*. [S.l.], 1971.

[120] VENKAYYA, V. B. Design of optimum structures. *Journal of Computers & Structures*, v. 1, n. 1-2, p. 265–309, 1971.

[121] SCHIMIT, L.; FARSHI, B. Some approximation concepts in structural synthesis. *AIAA Journal*, v. 12, p. 692–699, 1974.

[122] DHINGRA, A. Optimal apportionment of reliability and redundancy in series systems under multiple objectives. *IEEE Transactions on Reliability*, v. 41, n. 4, p. 576–582, 1992.

[123] YIN-XIU, L.; GEN, M. Nonlinear mixed integer programming problems using genetic algorithm and penalty function. *IEEE International Conference on Systems, Man, and Cybernetics*, v. 4, p. 2677–2682, 1996.