UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Desenvolvimento e Aplicação de Método de Otimização Via Multioperadores

Muriell de Rodrigues e Freire

Itajubá, abril de 2010

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Muriell de Rodrigues e Freire

Desenvolvimento e Aplicação de Método de Otimização Via Multioperadores

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Automação e Sistemas Elétricos Industriais.

Orientador:
Prof. Dr. Leonardo de Mello Honório

Abril de 2010

Itajubá - MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

F866d

Freire, Muriell de Rodrigues e
Desenvolvimento e aplicação de método de otimização via multioperadores / Muriell de Rodrigues e Freire. -- Itajubá, (MG) : [s.n.], 2010.
134 p. : il.

Orientador: Prof. Dr. Leonardo de Mello Honório.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Algoritmos evolutivos adaptativos. 2. Operadores evolutivos. 3. Otimização global. 4. Otimização com restrições. I. Honório, Leonardo de Mello, orient. II. Universidade Federal de Itajubá. III. Título.

Agradecimentos

Dedico meus sinceros agradecimentos para:

– a minha namorada Elisa, pelo carinho, compreensão, paciência e apoio;

– o meu pai Gilberto, pelo incentivo e confiança;

– o professor Leonardo de Mello, pela oportunidade, orientação e incentivo;

– os colegas do grupo CRTI, pelo companheirismo e ajuda nos momentos difíceis;

– todos os colegas do Mestrado em Engenharia Elétrica da UNIFEI;

– os professores do grupo CRTI, pelo apoio e amizade;

– os professores da UNIFEI pelos conhecimentos, auxílio e incentivo recebidos;

– todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho;

– o CNPq, pelo auxílio financeiro que possibilitou a realização deste trabalho.

“Tantas vezes pensamos ter chegado,
Tantas vezes é preciso ir além...”
Fernando Pessoa.

Resumo

Este trabalho apresenta uma proposta de uma nova metodologia baseada em algoritmos evolutivos adaptativos para resolver problemas de otimização com restrições. O algoritmo proposto trabalha com múltiplos operadores evolutivos e usa um mecanismo de adaptação realimentado para ajustar de forma automática os valores das probabilidades de seleção dos operadores evolutivos. A ideia principal é, a partir de um conjunto de operadores evolutivos, descobrir quais operadores são mais eficientes na resolução de um determinado problema, e em qual momento do processo cada operador deve ser utilizado. Desta forma pretende-se: melhorar o desempenho do algoritmo evolutivo proposto quando comparado com algoritmos evolutivos sem adaptação de probabilidades (com parâmetros fixos), aumentar a gama de problemas que um algoritmo evolutivo é capaz de resolver e diminuir o número de parâmetros a serem ajustados pelo usuário. Como forma de avaliar o desempenho do algoritmo proposto, um conjunto de 24 problemas testes foi utilizado e os resultados obtidos foram comparados com os resultados de outros algoritmos evolutivos sem adaptação de parâmetros.

Palavras-chave: Algoritmos Evolutivos Adaptativos, Operadores Evolutivos, Otimização Global, Otimização com Restrições.

Abstract

This work proposes a new methodology based on adaptive evolutionary algorithms to solve optimization problems with constraints. The proposed algorithm deal with multiple evolutionary operators and uses an adaptation mechanism with feedback to adjust automatically the values of the probabilities of selection of evolutionary operators. From a set of evolutionary operators, the main idea is to find out which operators are more efficient in solving a particular problem, and in what stage of the process each operator should be used. Thus it is intended: to improve the performance of evolutionary algorithm proposed when compared to algorithms without adjustment of probabilities (with fixed parameters), to increase the range of problems that an evolutionary algorithm is able to solve, and to reduce the number of parameters to be set by the user. In order to evaluate the performance of the proposed algorithm, a set of 24 benchmark problems was used and the results were compared to those obtained with other three evolutionary algorithms without adjustment of parameters.

Keywords: Adaptive Evolutionary Algorithms, Evolutionary Operators, Global Optimization, Optimization with Constraints.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

NFL	Teorema *No Free Lunch*,	17
EC	Computação Evolutiva,	33
EAs	Algoritmos Evolutivos,	35
ESs	Estratégias Evolutivas,	40
EP	Programação Evolutiva,	42
GAs	Algoritmos Genéticos,	43
LCS	Sistemas Classificadores com Aprendizagem,	44
GP	Programação Genética,	44
PSO	Otimização por Enxame de Partículas,	46
ACO	Otimização por Colônia de Formigas,	48
DE	Evolução Diferencial,	49
AIS	Sistemas Imunológicos Artificiais,	54
AEAs	Algoritmos Evolutivos Adaptativos,	57
PM	*Probability Matching*,	62
AP	*Adaptive Pursuit*,	64
MOES	*Multiple Operators Evolutionary System*,	82

1 Introdução

Um algoritmo de otimização busca encontrar o melhor resultado possível para um determinado problema, dado sua função objetivo e seu domínio. Não existe um algoritmo de otimização que resolva todos os tipos de problemas. Contudo, diversos algoritmos de otimização têm sido propostos para se resolver distintas classes de problemas (WHITACRE, 2007).

Nas diferentes áreas da engenharia, arquitetura, economia, física, química, etc., podemos encontrar problemas de otimização de alta complexidade (alta dimensionalidade, não linearidade, descontinuidade, multimodalidade, etc.), onde as técnicas clássicas de otimização não proporcionam grande eficiência. Sendo assim, a computação evolutiva surge como um método alternativo para se resolver problemas complexos de otimização (WEISE, 2008).

Na área da computação evolutiva, os algoritmos evolutivos têm se mostrado altamente efetivos na resolução de vários dos diferentes problemas de otimização do mundo real. Contudo, a utilização dos algoritmos evolutivos exige que o usuário tenha conhecimento prévio de como ajustar os parâmetros iniciais do algoritmo: normalmente, este ajuste é feito de maneira empírica (EIBEN et al., 2007). De fato, um bom conjunto de parâmetros iniciais aumenta a eficiência do algoritmo, e uma má escolha pode levar ao fracasso do método. Além disso, a qualidade da escolha dos parâmetros depende do problema que está sendo abordado. Uma técnica para encontrar automaticamente bons parâmetros seria muito útil.

Em um algoritmo evolutivo adaptativo, um mecanismo de adaptação deve controlar dinamicamente os parâmetros que regulam o processo de busca, como forma de otimizar o desempenho do algoritmo na resolução de um problema. O mecanismo de adaptação deve receber informações do problema através de um processo de realimentação.

A grande maioria dos problemas de otimização do mundo real está sujeita a restrições de igualdade e desigualdade. Na resolução destes problemas os algoritmos evolutivos têm sido amplamente utilizados, pois são técnicas de fácil implementação e são ferramentas de busca global. Contudo, os algoritmos evolutivos em sua versão original não contam com um mecanismo para fazer a manipulação das restrições, abrindo espaço para pesquisas de métodos que

adaptem os algoritmos evolutivos para resolver problemas de otimização com restrições.

1.1 Motivação

O teorema *No Free Lunch* (NFL) (WOLPERT; MACREADY, 1997) apresenta uma observação interessante quanto ao uso dos algoritmos evolutivos: na média, todos os algoritmos evolutivos possuem desempenho equivalente, quando aplicados aos infinitos problemas de otimização (ver Figura 1.1). Desta forma, não existe algoritmo para a resolução de todos problemas de otimização que seja genericamente (em média) superior que outro algoritmo competidor. Segundo o NFL, quando comparamos o desempenho de dois algoritmos somente podemos afirmar que um algoritmo evolutivo comporta-se melhor que outro com respeito à resolução de uma classe específica de problemas, e como consequência comporta-se inadequadamente para outras classes de problemas.

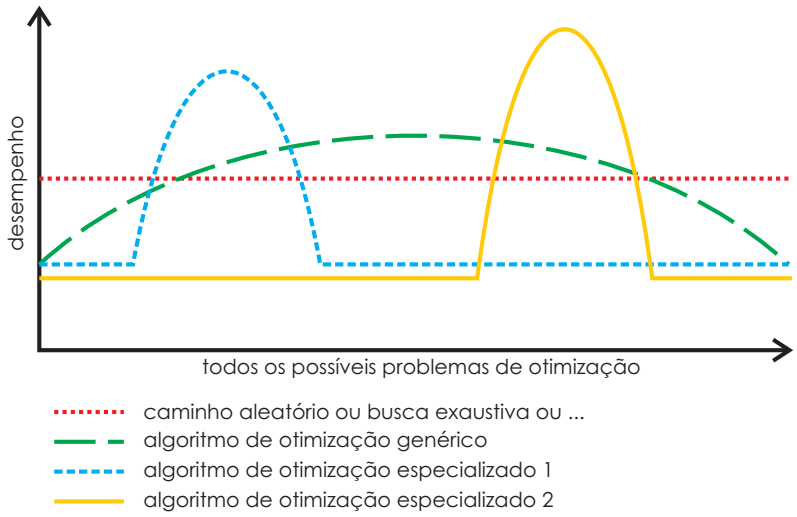


Figura 1.1: Uma visualização do teorema NFL.

De forma geral, podemos afirmar que se um dado algoritmo A é melhor que um algoritmo B em uma série de k problemas, então deve haver uma outra série de k problemas em que o algoritmo B tem um desempenho superior ao algoritmo A. Portanto, surge a ideia de usar diversos operadores de busca, cada um especializado em um determinado tipo de problema, em um único algoritmo evolutivo como forma de habilitar este sistema a resolver vários problemas diferentes, bastando para isso que o melhor operador para resolver aquele problema esteja entre os operadores implementados no algoritmo. Para tanto, este algoritmo evolutivo deve ser capaz de detectar qual o operador que maximiza o desempenho do algoritmo evolutivo como um todo e usá-lo com a maior frequência possível.

1.2 Objetivos

O objetivo principal deste trabalho de dissertação é propor um algoritmo evolutivo adaptativo com múltiplos operadores evolutivos que seja capaz de resolver uma grande gama de problemas de otimização restrita e que tenha poucos parâmetros a serem ajustados pelo usuário. Para tanto, um conjunto de nove operadores evolutivos será utilizado, onde as probabilidades de seleção do operadores serão ajustadas dinamicamente por um mecanismo de adaptação.

Um importante requisito para um algoritmo evolutivo adaptativo é ter um bom critério de avaliação da produtividade de seus operadores, visando ajustar de maneira adequada as probabilidades de seleção dos mesmos. Neste trabalho, temos o objetivo de propor e estudar três metodologias diferentes para calcular a produtividade de um operador quando aplicado na resolução de problemas com restrição.

Como forma de avaliar o desempenho do algoritmo proposto, um conjunto de 24 problemas testes será utilizado e os resultados obtidos serão comparados com os resultados de outros algoritmos evolutivos tradicionais (sem adaptação de parâmetros). Estes problemas são problemas de otimização restrita e foram selecionados de forma a se obter diversidade nas características que dificultam o processo de otimização global.

1.3 Organização do Trabalho

A organização deste documento foi feita da seguinte forma:

- **Capítulo 2 - Otimização de Sistemas**: Tem como objetivo fazer a fundamentação teórica sobre otimização de sistemas e sua aplicação no mundo real. A formulação geral para problemas de otimização é apresentada, juntamente com os conceitos básicos do tema: classificação e complexidade dos problemas de otimização, e os diferentes algoritmos de busca e otimização.

- **Capítulo 3 - Computação Evolutiva**: Faz uma introdução aos algoritmos evolutivos, apresentando os principais paradigmas da computação evolutiva. As vantagens e desvantagens da aplicação dos algoritmos evolutivos são discutidas.

- **Capítulo 4 - Algoritmos Evolutivos Adaptativos**: Faz a conceituação teórica das diversas técnicas para controle de parâmetros em algoritmos evolutivos e introduz os conceitos básicos sobre os algoritmos evolutivos adaptativos. Algumas técnicas e critérios encontrados na literatura atual são apresentados e discutidos.

- **Capítulo 5 - Algoritmo Proposto**: Descreve em detalhes o algoritmo desenvolvido: algoritmo evolutivo adaptativo para controle automático das probabilidades de seleção dos operadores evolutivos. A metodologia adotada para fazer a manipulação das restrições é apresentada e as estratégias usadas para avaliar a produtividade dos operadores é definida.

- **Capítulo 6 - Resultados e Discussões**: Os resultados obtidos da aplicação do algoritmo proposto aos 24 problemas testes são apresentados e comparados com outros três algoritmos tradicionais. Também temos a definição da metodologia experimental usada na obtenção dos resultados.

- **Capítulo 7 - Conclusões**: Apresenta as conclusões gerais obtidas da realização deste trabalho e sugestões para trabalhos futuros.

- **Anexo A - Problemas Testes**: Mostra a formulação completa dos 24 problemas de otimização restrita usados nos testes realizados.

2 *Otimização de Sistemas*

Este capítulo tem como objetivo fazer a fundamentação teórica sobre a otimização de problemas restritos, ou seja, problemas de otimização com restrições de desigualdade e igualdade. Para tanto, a formulação geral para problemas de otimização é apresentada, juntamente com uma classificação dos tipos de problemas que podem ser encontrados. Como o intuito de justificar a pesquisa na área de Computação Evolutiva, algumas propriedades comuns aos problemas de otimização, e que podem dificultar o processo de busca, serão apresentadas e comentadas. Por fim, temos uma breve introdução sobre os métodos de manipulação de restrições encontrados na literatura.

2.1 Introdução

Na Engenharia, uma das grandes linhas de pesquisa é a da Otimização de Sistemas, onde metodologias de otimização são desenvolvidas e aplicadas na busca por estados ótimos de um sistema ou por soluções ótimas para um determinado problema (HONORIO; SILVA; BARBOSA, 2007; HONORIO et al., 2007; VERMAAS et al., 2009). Normalmente, estes problemas são compostos de uma função objetivo e um conjunto de restrições, representadas por equações e inequações.

Problemas de otimização restrita, especialmente problemas de otimização não linear, onde as funções objetivo são minimizadas (ou maximizadas) sob um dado conjunto de restrições, são muito importantes e frequentemente aparecem no mundo real (TAKAHAMA; SAKAI, 2005a).

Nos problemas de otimização, o chamado ponto ótimo é um vetor com as variáveis de controle que satisfazem as restrições e geram um valor extremo na função objetivo: normalmente é denotado por $\vec{x}^*$. Sendo assim, o valor ótimo de um problema é o valor da função objetivo no ponto ótimo. Então, quando buscamos pela solução ótima de um problema, buscamos pelo par $(\vec{x}^*, f(\vec{x}^*))$, que pode ser local ou global.

A seguir serão descritos vários conceitos importantes para o entendimento dos métodos de

busca e otimização relacionados com esse trabalho.

2.2 Formulação de um Problema de Otimização

O objetivo da otimização global é encontrar o melhor elemento $\vec{x}^*$ possível de um conjunto $\mathbb{X}$ de acordo com um conjunto de critérios $C = \{f_1, f_2, \ldots, f_n\}$. Estes critérios são expressos através de funções matemáticas, as chamadas *funções objetivo* (WEISE, 2008). Sejam dados um conjunto $\mathbb{X} \subseteq \mathbb{R}^n$ e uma função objetivo $f : \mathbb{X} \mapsto Y$ com $Y \subseteq \mathbb{R}$. Um problema de otimização n-dimensional visa determinar um minimizador de f no conjunto $\mathbb{X}$, sendo escrito como:

$$Minimize\ f(\vec{x}),\ \ \vec{x} = [x_1, x_2, \ldots, x_n] \tag{2.1}$$

Semelhantemente ao problema acima, um problema de otimização n-dimensional que vise determinar um maximizador de f no conjunto $\mathbb{X}$, pode ser escrito da seguinte forma:

$$Maximize\ f(\vec{x}),\ \ \vec{x} = [x_1, x_2, \ldots, x_n] \tag{2.2}$$

Nestes problemas, o conjunto $\mathbb{X}$ é chamado de *conjunto viável* e define a região onde estão as possíveis soluções do problema. Também recebe o nome de *região viável*, *região factível*, *espaço de busca*, ou ainda *espaço de projeto*. Normalmente, para limitar uma certa região em $\mathbb{R}^n$, utiliza-se as *funções de restrição laterais*, definindo limites superiores e inferiores para $\mathbb{X}$. Os pontos de $\mathbb{X}$ são chamados de *pontos viáveis* ou *factíveis*.

Minimizar a função $f(\vec{x})$ é buscar por um vetor $\vec{x}^*$ que gere um valor mínimo extremo para a função. Assim, a função objetivo permite avaliar o grau de otimalidade de cada elemento do espaço de busca. A função objetivo, que também pode ser chamada de função custo ou ainda função de adaptação (função *fitness* - no contexto dos algoritmos evolutivos), pode ser unidimensional, ou seja, possui apenas uma variável de controle, ou multidimensional, isto é, possui mais de uma variável de controle.

O vetor $\vec{x} = [x_1, x_2, \ldots, x_n]$ possui as variáveis de controle ou de projeto que irão se modificar durante o processo de otimização, alterando o valor da função objetivo na busca por um valor extremo (mínimo ou máximo). Sendo assim, um ponto $\vec{x}^* \in \mathbb{X}$ é *minimizador global* de (2.1) se $f(\vec{x}^*) \leq f(\vec{x}),\ \forall \vec{x} \in \mathbb{X}$. Se existe uma vizinhança $\mathbb{U} \subseteq \mathbb{X}$ tal que $f(\vec{x}^*) \leq f(\vec{x})\ \ \forall \vec{x} \in \mathbb{U}$, então o ponto $\vec{x}^* \in \mathbb{U}$ é dito *minimizador local* de (2.1). No caso de estarmos resolvendo um problema de maximização de uma função objetivo, um ponto $\vec{x}^* \in \mathbb{X}$ é *maximizador global* de

(2.2) se $f(\vec{x}^*) \geq f(\vec{x})$, $\forall \vec{x} \in \mathbb{X}$. Também temos que, se existe uma vizinhança $\mathbb{U} \subseteq \mathbb{X}$ tal que $f(\vec{x}^*) \geq f(\vec{x}) \ \ \forall \vec{x} \in \mathbb{U}$, então o ponto $\vec{x}^* \in \mathbb{U}$ é dito *maximizador local* de (2.2). A figura 2.1 ilustra os conceitos de mínimo e máximo global/local para uma função objetivo bidimensional.

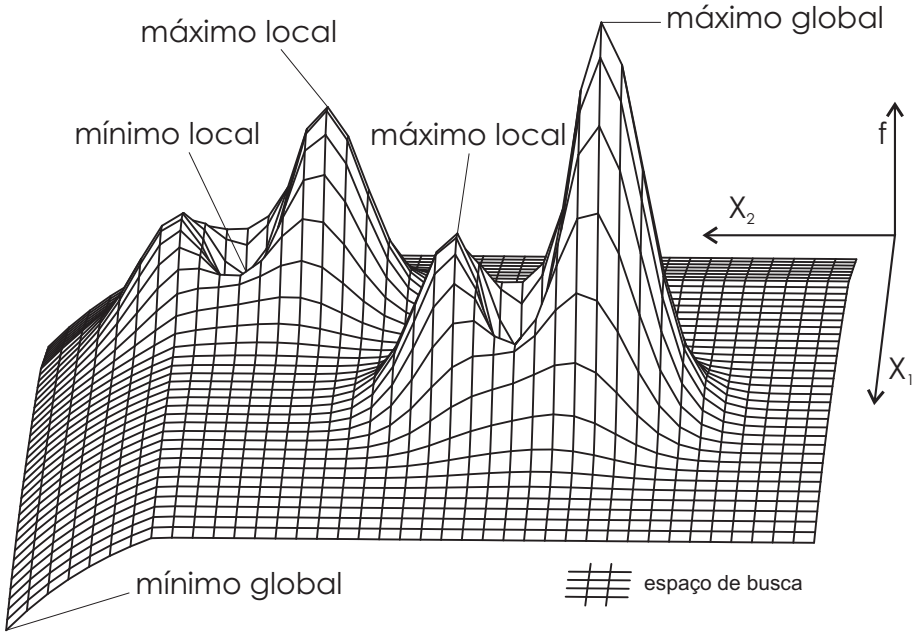


Figura 2.1: Ótimo global e local de uma função bidimensional.

Restrições são funções que delimitam o espaço de busca, através de igualdades ou desigualdades matemáticas, que estabelecem exigências para a solução ótima. As *restrições laterais* são funções que servem para limitar os valores dos parâmetros de controle do problema, sendo definidas como: limite inferior $\vec{x}^L = [x_1^L, x_2^L, \ldots, x_n^L]$ e limite superior $\vec{x}^U = [x_1^U, x_2^U, \ldots, x_n^U]$. Assim, se $x_i^L \leq x_i \leq x_i^U$ para $i = 1, 2, \ldots, n$, então $\vec{x} \in \mathbb{X}$. Nos problemas de otimização chamados de *Problemas de Otimização Restritos*, além das restrições laterais, podemos ter também *restrições de desigualdade* e *restrições de igualdade*.

Em geral, os problemas de otimização encontrados na engenharia são do tipo restritos e envolvem o ajuste de um vetor $\vec{x}$ (parâmetros de controle ou variáveis de entrada) de forma a otimizar (minimizar ou maximizar) uma dada função objetivo $f(\vec{x})$. Neste caso, a solução para o problema estará sujeita a satisfazer um conjunto de restrições, ou seja, para que $\vec{x}^*$ seja uma solução do problema deve satisfazer a todas as restrições de desigualdade, igualdade e limites superior e inferior para cada variável de controle. No caso da minimização, o problema geral de otimização restrita pode ser escrito conforme as expressões (2.3) e (2.4).

$$Minimize\ f(\vec{x}) \tag{2.3}$$

$$\text{Sujeito a} \begin{cases} g_j(\vec{x}) \leq 0, & j = 1,\ldots,q \\ h_j(\vec{x}) = 0, & j = q+1,\ldots,m \\ x_i^L \leq x_i \leq x_i^U, & i = 1,2,\ldots,n \end{cases} \tag{2.4}$$

onde $\vec{x} = [x_1, x_2, \ldots, x_n]$ é um vetor n-dimensional de variáveis de controle, $f(\vec{x})$ é uma função objetivo, $g_j(\vec{x}) \leq 0$ são q restrições de desigualdade, e $h_j(\vec{x}) = 0$ são $m-q$ restrições de igualdade. As funções f, g_j e h_j são funções reais lineares ou não lineares. Os valores x_i^L e x_i^U são os limites inferiores e superiores de x_i, respectivamente.

No problema de otimização definido pelas expressões (2.3) e (2.4), as restrições laterais vão definir as faixas (*ranges*) permitidas para os parâmetros de controle do sistema, ou seja, faixas de valores onde os parâmetros de controle podem variar: juntas formam o *espaço de busca* $\mathbb{X}$ do problema. Contudo, as restrições de desigualdade $g(\vec{x})$ e as restrições de igualdade $h(\vec{x})$ definem a *região viável* $\mathbb{F}$. Portanto, as soluções viáveis (*feasible solutions*) para um problema de otimização restrita pertencem ao conjunto $\mathbb{F} \subseteq \mathbb{X}$.

Restrições ativas são as restrições de desigualdade que são satisfeitas por $\vec{x}$ na igualdade, ou seja, para um dado ponto $\vec{x}$ existe uma restrição j tal que $g_j(\vec{x}) = 0$. Caso contrário, a restrição é dita inativa. É muito comum na otimização de sistemas encontrarmos soluções ótimas na borda da região viável, ou seja, soluções ótimas que geram a ativação de alguma restrição $g_j(\vec{x})$.

Um procedimento muito comum na otimização de problemas é transformar um problema de maximização em minimização, pois, uma vez implementado um algoritmo de minimização, este deverá ser usado para os dois tipos de problema. Para tanto, basta multiplicar a função objetivo do problema por -1, de forma que minimizar $-f(\vec{x})$ é maximizar $f(\vec{x})$: as soluções locais e globais de ambos os problemas serão as mesmas, com sinais opostos para os valores ótimos.

2.3 Características de um Problema de Otimização

Normalmente, os problemas de otimização estão divididos em algumas classes, com base em alguma característica predominante de sua função objetivo ou de suas restrições: Dimensionalidade, Multimodalidade, Convexidade da região factível, Objetivos Múltiplos, etc. Estas características vão impactar na eficiência de um determinado algoritmo, visto que os algoritmos

de busca e otimização são projetados para trabalhar com apenas algumas delas e não com todas. Assim, teremos algoritmos que resolvem bem uma determinada classe problemas e outras não. Contudo, é sabido que em problemas reais de otimização, podemos ter a combinação de várias destas características em apenas um problema (WHITACRE, 2007). Apesar de algumas delas não oferecerem grandes desafios quando ocorrem isoladas, sua combinação pode tornar o problema muito difícil de ser resolvido.

Whitacre (WHITACRE, 2007), em sua tese de doutorado, faz uma descrição bastante completa sobre as características que podem fazer de um problema de otimização um grande desafio:

- Dimensionalidade
- Multimodalidade
- Restrições Complexas
- Epistasia dos Parâmetros
- Problemas de Decepção
- Incertezas e/ou Ruídos
- Objetivos Múltiplos
- Objetivos Dinâmicos
- Custos Computacionais

Quando a função objetivo de um determinado problema de otimização possui muitos pontos ótimos (locais) o problema é chamado de *multimodal*. Uma função unimodal apresenta apenas um ponto de mínimo ou máximo. Já uma função multimodal possui várias inflexões de sua superfície de otimização, o que caracterizam múltiplos pontos de mínimo ou máximo. Para certos métodos de busca essa característica é indesejável, pois interrompe a busca por soluções melhores (ótimos globais), uma vez que o método fica preso em um ótimo local.

A característica de dimensionalidade pode comprometer a eficiência de alguns algoritmos de otimização quando os problemas são de grande dimensão ou de grande porte, uma vez que o espaço de busca pode se tornar muito grande. Contudo, se alguns parâmetros de controle do problema podem ser resolvidos separadamente a partir de outros parâmetros (conceito de epistasia), então o aumento da dimensão do problema não implicará em um grande aumento da sua complexidade.

O termo Epistasia é usado para indicar o grau de interação entre os parâmetros em uma função objetivo (ou em funções de restrição). Problemas de falta de interações epistáticas são completamente separáveis (isto é, problemas decomponíveis) onde cada um dos parâmetros pode ser resolvido isoladamente. Muitos problemas do mundo real têm pelo menos algum grau de epistasia.

A maioria dos problemas do mundo real não são definidos como tendo uma única função objetivo. Em muitas situações, várias funções objetivo precisam ser minimizadas ou maximizadas simultaneamente. Isto ocorre devido à complexidade dos problemas reais que, na maioria das vezes, apresentam objetivos conflitantes entre si, ou seja, a melhoria de um objetivo provoca a piora de outro. Um problema de otimização que possui mais de uma função objetivo é denominado de *problema multiobjetivo* ou *multicritério*. Exemplos de funções objetivo que podem aparecer juntas em um mesmo problema são: funções de custo, eficiência, risco, lucro, heurísticas baseadas na experiência de um especialista, etc.

Uma outra classe de problemas é a classe do problemas com função objetivo dinâmica, ou seja, problemas onde a função objetivo muda com o tempo $f(\vec{x},t)$. Desta forma, ótimo globais e locais podem variar com o tempo e o algoritmo de otimização deve estar apto a trabalhar com essas mudanças. Dentro da computação evolutiva, uma característica desejável para um algoritmo de otimização dinâmica é que ele consiga manter a diversidade de sua população: isso evitaria uma convergência prematura.

2.3.1 Classificação dos Problemas de Otimização

Os problemas de otimização podem ser classificados de acordo com as seguintes características:

1. Com base na existência de restrições
    - Problemas restritos (com restrições)
    - problemas irrestritos (sem restrições)
2. Com base no número de funções objetivo do problema
    - Problemas mono-objetivo
    - Problemas multiobjetivo
3. Com base na natureza dos parâmetros de controle

   - Problemas estáticos ou paramétricos
   - Problemas dinâmicos (quando os parâmetros são funções)

4. Com base na natureza das equações envolvidas

   - Problemas lineares
   - Problemas não lineares
   - Problemas de programação geométrica
   - Problemas de programação quadrática

5. Com base nos valores admissíveis para os pâmetros de controle

   - Problemas de programação com valores inteiros
   - Problemas de programação com valores reais
   - Problemas em codificação binária
   - Problemas de otimização combinatória

6. Com base na natureza determinística das variáveis

   - Problemas determinísticos
   - Problemas estocásticos

7. Com base na separabilidade das funções

   - Problemas separáveis
   - Problemas não separáveis

2.4 Métodos de Otimização

Atualmente, pode-se aplicar técnicas de otimização em várias áreas do conhecimento, sendo que sua aplicação em problemas de engenharia tem crescido consideravelmente (OLIVEIRA, 2006). Existem muitos métodos de otimização e cada um deles alcança melhor desempenho dependendo do tipo de problema considerado. Sendo assim, a escolha do método depende de uma série de características do problema a ser otimizado. É muito comum encontrarmos algoritmos especializados em resolver uma certa classe de problemas (WHITACRE, 2007).

Basicamente, os métodos de otimização estão divididos em dois grandes grupos: *determinísticos* e *probabilísticos* (também chamados de *estocásticos*). A figura 2.2 mostra uma taxonomia dos métodos de otimização global.

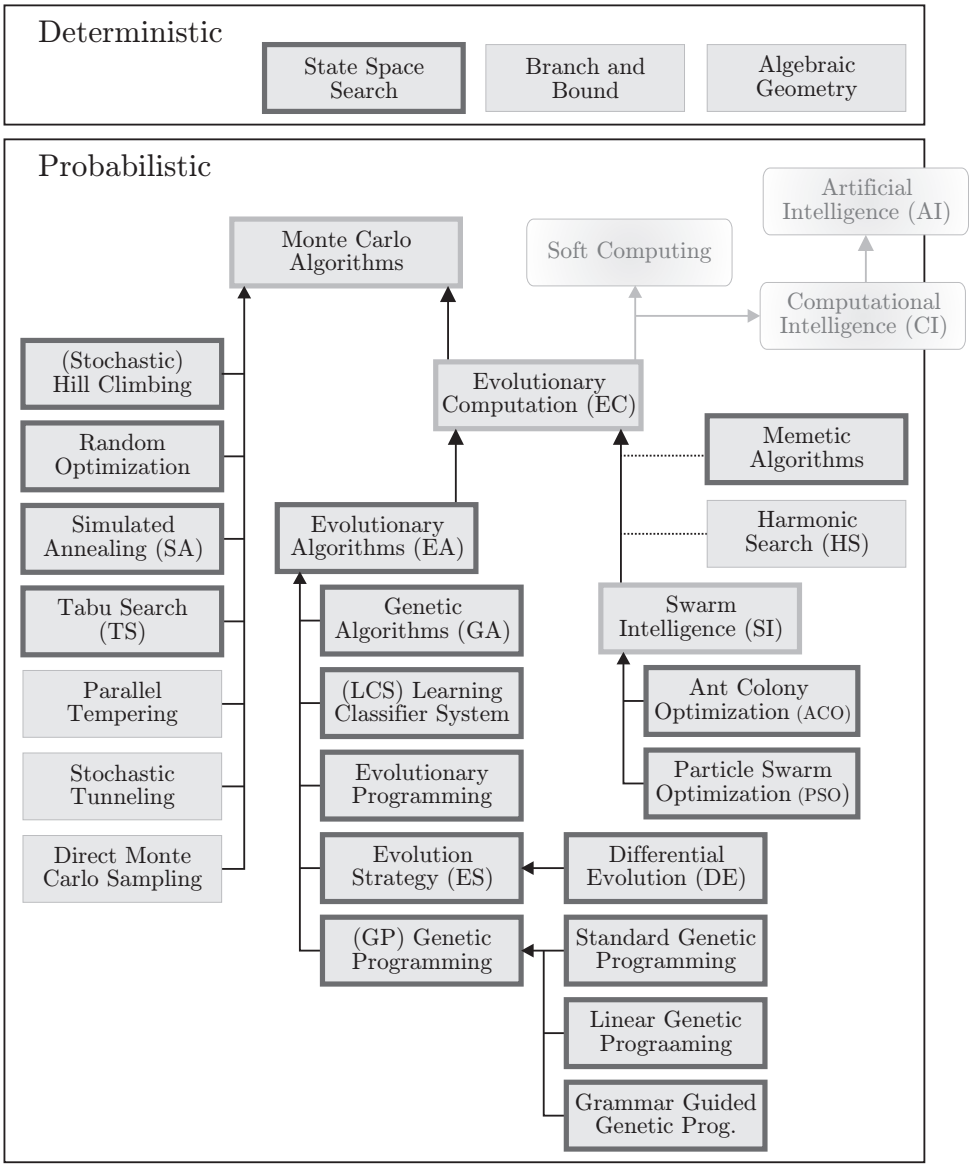


Figura 2.2: A taxonomia dos algoritmos de otimização global. Fonte: (WEISE, 2008)

2.4.1 Métodos Determinísticos

Os métodos de otimização baseados nos algoritmos determinísticos (maioria dos métodos clássicos) geram uma sequência determinística de possíveis soluções requerendo, na maioria das vezes, o uso de pelo menos a primeira derivada da função objetivo em relação às variáveis de projeto (HOLTZ, 2005). Assim, os métodos determinísticos usam apenas uma solução que vai sendo atualizada em cada iteração.

A maioria dos métodos determinísticos são baseados no cálculo de derivadas ou em aproximações destas, necessitando de informações do vetor gradiente, seja procurando o ponto onde ele se anula ou usando a direção para a qual aponta (OLIVEIRA, 2006). Estes métodos produzem bons resultados quando as funções são contínuas, convexas e unimodais. Já no caso dos

problemas multimodais, onde a função objetivo possui vários pontos de inflexão (a primeira derivada nesse ponto é zero - indica que é um ponto de máximo ou mínimo), os métodos determinísticos não são eficazes para se encontrar o ponto ótimo: eles costumam ficar presos em pontos de ótimo local. Esse problema é um dos grandes motivos para os desenvolvimento dos algoritmos estocásticos ou probabilísticos (WEISE, 2008).

Os métodos determinísticos apresentam teoremas que lhes garantem a convergência para uma solução ótima que não é necessariamente a solução ótima global. Como nesses métodos a solução encontrada é extremamente dependente do ponto de partida fornecido, pode-se convergir para um ótimo local, por isso não possuem bom desempenho em otimizar funções multimodais, isto é, funções que possuem vários ótimos locais.

Segundo (HOLTZ, 2005), os métodos de Programação Matemática ainda podem ser classificados em diferentes áreas de acordo com o comportamento da função objetivo e das restrições. Dentro da área de Pesquisa Operacional, podemos encontrar a seguinte classificação:

- **Programação Linear**: quando a função objetivo e as restrições são funções lineares das variáveis de projeto. O Método Simplex (HADLEY, 1982) é o método mais tradicional para solucionar este tipo de problema de otimização;
- **Programação Não Linear**: quando a função objetivo, ou pelo menos uma das restrições, é uma função não-linear das variáveis de controle. Nesta classe, os métodos que mais se destacam são: Método de Programação Linear Sequencial, Método de Programação Quadrática Sequencial, Método das Direções Viáveis e Método do Gradiente Reduzido, entre outros.

2.4.2 Métodos Probabilísticos

Os métodos de otimização baseados nos algoritmos probabilísticos são estruturados a partir de parâmetros *estocásticos*, ou seja, variáveis aleatórias. Nestes métodos, a função objetivo é usada apenas para avaliar as soluções encontradas e a busca por soluções ótimas é guiada por alguma *heurísticas*, que nada mais são que regras de como explorar os espaço de busca de maneira inteligente. Por não utilizarem a derivada da função objetivo, são considerados métodos de ordem zero.

Dentro do grupo de algoritmos probabilísticos, temos os algoritmos baseados na chamada *Computação Natural* ou *Evolutiva*, que adapta alguns conceitos do funcionamento da natureza para algoritmos de busca e otimização. Os mecanismos naturais que promovem a evolução

dos seres vivos podem ser considerados processos inteligentes. A maioria das pesquisas sobre evolução estuda o comportamento inteligente de seres vivos em coletividade buscando se auto-organizar para atingir um dado objetivo. Nestes termos, são incluídos na modelagem, mecanismos de auto-organização, adaptação, evolução, competição e cooperação, dentre outros.

Entre as técnicas mais conhecidas dos métodos naturais destacam-se os Algoritmos Genéticos (GAs), Recozimento Simulado (*Simulated Annealing*), Estratégias Evolutivas (ESs), Programação Evolutiva (EP), Programação Genética (GP), Evolução Diferencial (DE), Otimização por Enxame de Partículas (PSO), entre outras (OLIVEIRA, 2006). Estes algoritmos se baseiam em população de indivíduos, onde cada indivíduo representa um ponto de busca no espaço de soluções potenciais de um dado problema e imitam os princípios da natureza para criar procedimentos de otimização. Os Algoritmos Evolutivos (EAs) possuem alguns procedimentos de seleção baseados na aptidão dos indivíduos (avaliação da função objetivo) , e em operadores de cruzamento e mutação (também chamados de operadores evolutivos de busca e variação).

Os algoritmos evolutivos possuem características que os tornam aplicáveis a uma vasta quantidade de problemas de otimização. Algumas características desejáveis são:

- A função objetivo e as restrições não precisam necessariamente ter uma representação matemática;
- São métodos de otimização global, robustos e podem encontrar várias soluções;
- Não há restrição alguma quanto ao ponto de partida dentro do espaço de busca da solução;
- Podem otimizar um grande número de parâmetros discretos, contínuos ou combinações deles;
- Realizam buscas simultâneas em várias regiões do espaço de busca (paralelismo inerente);
- Utilizam informações de custo ou ganho e não necessitam obrigatoriamente de conhecimento matemático do problema (tais como derivadas);
- Podem ser eficientemente combinados com heurísticas e outras técnicas de busca local;
- São modulares e facilmente adaptáveis a qualquer tipo de problema.

Naturalmente, também possuem algumas desvantagens:

- Como trabalham com população de soluções, podem ser mais lentos que outras alternativas;

- Possuem parâmetros que devem ser bem ajustados para obter eficácia;
- Possuem tempo de processamento maior que os métodos clássicos.

2.5 Métodos de Manipulação de Restrições

Normalmente, os algoritmo evolutivos de otimização são desenvolvidos para resolver problemas irrestritos (sem restrições). Portanto, para que estes algoritmo possam ser usados para resolver problemas de otimização restrita, temos que utilizar algum método de manipulação de restrições.

Em um problema de otimização restrita, o espaço de busca $\mathbb{X}$ consiste de dois subconjuntos disjuntos de subespaços: factíveis ($\mathbb{F}$ - atendem a todas as restrições simultaneamente) e infactíveis ($\mathbb{U}$ - violam uma ou mais restrições), sendo que no decorrer do processo de busca, a população de indivíduos candidatos à solução pode conter elementos que sejam tanto factíveis como infactíveis. A figura 2.3 ilustra candidatos à solução caracterizados como factíveis, infactíveis e uma solução ótima.

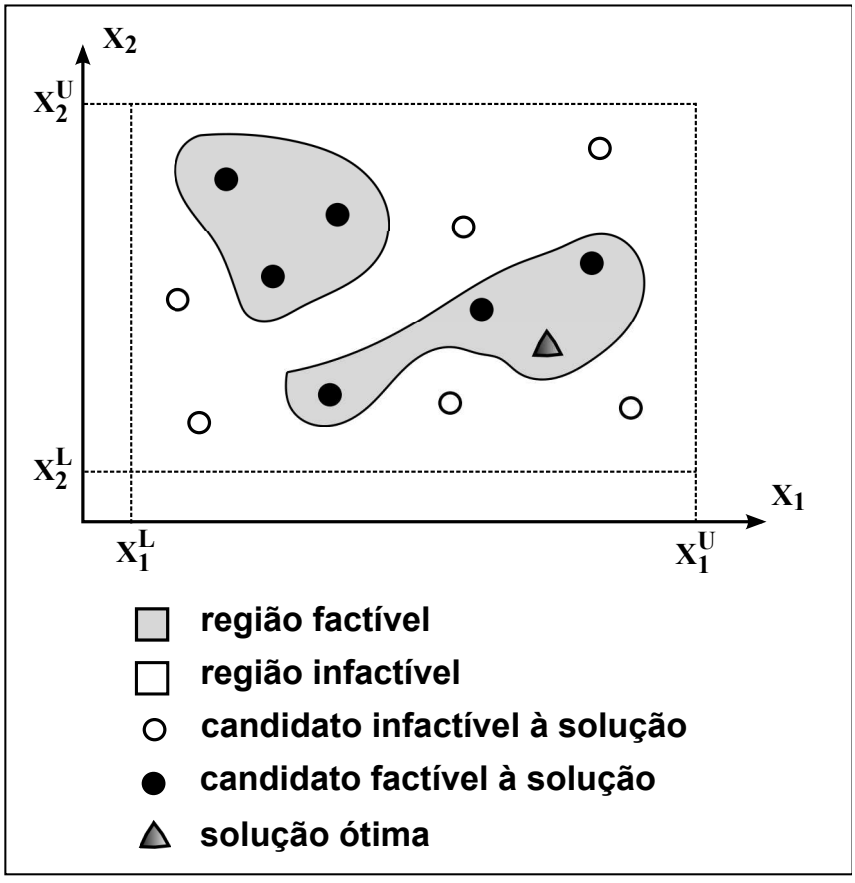


Figura 2.3: Representação pictórica do espaço de busca $\mathbb{X}$

Existem muitos estudos sobre a otimização de problemas restritos utilizando algoritmos evolutivos (TAKAHAMA; SAKAI, 2004; TAKAHAMA; SAKAI, 2005a; TAKAHAMA; SAKAI, 2005b; TAKAHAMA; SAKAI, 2006). Basicamente, os métodos de manipulação de restrições podem ser divididos em quatro categorias, de acordo com a forma com que as restrições são tradadas: 1) descarte ou reparação de soluções infactíveis, 2) penalidade para restrições não satisfeitas, 3) otimização multiobjetivo e 4) comparação lexográfica.

2.5.1 Descarte ou Reparação de Soluções Infactíveis

As restrições são usadas apenas para avaliar se um dado ponto de busca é factível ou não. Este método ou penaliza os pontos infactíveis com a morte, excluindo-o do conjunto de pontos de busca, ou repara o ponto de busca, recolocando-o de volta dentro da região factível. O processo de busca começa com um ou mais pontos factíveis e continua a busca por novos pontos dentro da região factível. Quando um novo ponto de busca é gerado (normalmente através de um processo estocástico) e não está sobre a região factível, o ponto é reparado ou descartado. No caso do descarte, um outro ponto é gerado em seu lugar. Contudo, a geração de pontos iniciais factíveis é muito difícil e computacionalmente dispendiosa, visto que na maioria dos problemas de otimização restrita a região factível é muito pequena.

2.5.2 Penalidade para Restrições Não Satisfeitas

A violação de restrições, que é a soma das violações de todas as funções de restrição, é combinada com a função objetivo do problema para gerar uma nova função objetivo, chamada de pseudo objetivo. Desta forma, o problema que era do tipo restrito se torna irrestrito, pois otimiza-se essa nova função objetivo que já incorpora as violações de restrição na forma de penalidades. Assim, a otimização da função objetivo e das violações de restrição é realizada pela otimização da função objetivo estendida.

Como exemplo, seja a função pseudo objetivo $f_p(\vec{x})$ dada pela expressão (2.5) e seja a função de penalidade $\phi(\vec{x})$ dada pela expressão (2.6):

$$f_p(\vec{x}) = f(\vec{x}) + k_p \phi(\vec{x}) \tag{2.5}$$

$$\phi(\vec{x}) = \left[\sum_{j=1}^{q} \|max\{0,\ g_j(\vec{x})\}\|^2 + \sum_{j=q+1}^{m} \|h_j(\vec{x})\|^2 \right] \tag{2.6}$$

onde $f(\vec{x})$ é a função objetivo original dada na equação (2.3); $g_j(\vec{x})$ e $h_j(\vec{x})$ são funções de restrições de desigualdade e igualdade, respectivamente, conforme (2.4); e o escalar k_p é um

multiplicador que quantifica a magnitude da penalidade. Para a eficiência do método, deve-se procurar um valor adequado para o fator de penalidade de forma a garantir que todas as restrições sejam obedecidas.

A principal dificuldade dos métodos baseados na função de penalidade é a difícil tarefa de ajustar valores apropriados para os coeficientes que ponderam a força da penalidade.

2.5.3 Otimização Multiobjetivo

Nesta categoria, os problemas de otimização restrita são resolvidos através de algum algoritmo de otimização de problemas multiobjetivo, de forma que a função objetivo e as funções de restrição são os objetivos a serem otimizados em conjunto. Assim, as restrições serão funções objetivo adicionais ao problema. No entanto, em muitos casos, resolver problemas de otimização multiobjetivo é mais difícil e computacionalmente mais dispendioso que resolver problemas de otimização com apenas uma função objetivo.

2.5.4 Comparação Lexográfica

Nesta categoria, a função de violação de restrições e a função objetivo são usadas separadamente: ambas a funções são otimizadas através de um método lexicográfico no qual a violação de restrições precede a função objetivo. Uma vez que a factibilidade dos pontos de busca é mais importante que a otimização (maximização ou minimização) da função objetivo, os algoritmos desta categoria buscam primeiro colocar os pontos de busca dentro da região factível para então otimizar a função objetivo do problema. Baseados nestas características, Takahama e Sakai propuseram os métodos α*-constrained* (TAKAHAMA; SAKAI, 2004; TAKAHAMA; SAKAI, 2005a; TAKAHAMA; SAKAI, 2005b) e ε*-constrained* (TAKAHAMA; SAKAI, 2006), que adota uma ordenação lexicográfica com o relaxamento das restrições. Estes métodos fazem o relaxamento das restrições de igualdade e desigualdade no estagio inicial do processo de busca e, gradualmente, vão diminuindo o relaxamento e convergindo os pontos infactíveis para a região factível. Para tanto, cria-se um novo mecanismo de comparação e ordenação dos pontos de busca, onde a comparação lexicográfica substitui a tradicional comparação ordinal (baseada na função de adaptação).

Devido a sua eficiência e robustez, o método ε*-constrained* foi o método escolhido neste trabalho para implementar o mecanismo de manipulação de restrições dos algoritmos implementados.

3 *Computação Evolutiva*

Este capítulo tem como objetivo a fundamentação teórica na área de Computação Evolutiva (EC - *Evolutionary Computation*), apresentando seus os principais conceitos e paradigmas. Alguns algoritmos evolutivos serão mostrados através de um pseudo-código como forma de exemplificar sua implementação. As principais vantagens e desvantagens da aplicação dos algoritmos evolutivos em problemas de busca e otimização serão apresentadas e discutidas.

3.1 Introdução

A Computação Evolutiva, ou também chamada de Computação Evolucionária, é um ramo da computação que engloba técnicas que simulam processos da natureza para gerar algoritmos computacionais. Abstraindo conceitos da Evolução Natural das Espécies e do Comportamento Coletivo de Animais ou Insetos, a EC possui uma abordagem alternativa para resolver problemas complexos de busca e otimização.

De acordo com (MICHALEWICZ, 1996), na EC a teoria da evolução das espécies é adaptada para modelos computacionais, visando a geração de algoritmos de busca e otimização de sistemas baseados nos mecanismos evolutivos encontrados na natureza. Esses mecanismos estão diretamente relacionados com a teoria da evolução de Darwin, onde ele afirma que a vida na Terra é o resultado de um processo de seleção, feito pelo meio ambiente, em que somente os mais aptos e adaptados possuirão chances de sobreviver e, consequentemente, reproduzir-se.

A ideia básica da EC é, dado um problema a ser resolvido, manter uma população de "estruturas de conhecimento", que vai evoluindo no decorrer do "tempo", através de um processo de competição e variação controlada. Cada estrutura na população representa uma solução candidata para o problema (FIALHO et al., 2007). Essa solução deve ser encontrada sobre um dado espaço de busca $\mathbb{X}$ (domínio do problema). Trata-se de uma metodologia de otimização, cujo processo produz soluções tão próximas da ótima quanto possível, dadas condições iniciais, restrições ambientais e parâmetros evolutivos corretos.

Os primeiros passos dados na área da EC foram de biólogos e geneticistas. Estes tinham interesse em simular os processos vitais de um ser humano em um computador. Na década de 60, um grupo de cientistas, em que o nome de Holland se destaca, iniciaram um estudo em que era implementada uma população de *n* indivíduos onde cada um possuía seu genótipo e estava sujeito a operações de seleção, recombinação e mutação. Tal estudo foi modelado e passou a ser conhecido como Algoritmo Genético (GA). Holland chegou a propor um quarto operador, a inversão. Entretanto ele não obteve destaque na comunidade científica, não vindo a ser muito utilizado. Uma das primeiras aplicações na área de EC foi relacionada a Algoritmos Genéticos (GAs). Bagley, em 1967, utilizou em uma parte de sua dissertação GA para desenvolver sistemas classificadores (MICHALEWICZ, 1996).

Segundo Fialho (FIALHO et al., 2007), várias abordagens para sistemas baseados na evolução foram propostas, tendo como principais diferenças os operadores utilizados e o tipo de codificação dos indivíduos. Dentre elas, podemos destacar:

- Estratégias Evolutivas (ESs): Rechenberg (1965);
- Programação Evolutiva (EP): Fogel, Owens e Walsh (1966);
- Algoritmos Genéticos (GAs): Holland (1975);
- Sistemas Classificadores com Aprendizagem (LCS): Holland (1975); e
- Programação Genética (GP): Koza (1992).

Com o avanço das pesquisas relacionadas com a EC, outras áreas surgiram e ganharam importância no meio científico. Novas metaheurísticas surgiram e todas baseadas nos conceitos da EC, por exemplo: Recozimento Simulado (SA, do inglês *Simulated Annealing*), Otimização por Colônia de Formigas (ACO, do inglês *Ant Colony Optimization*), Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) e Sistemas Imunológicos Artificiais (AIS, do inglês *Artificial Immune System*). Tais propostas também têm uma forte motivação física ou biológica tais como bandos de pássaros, cardumes, enxames ou também como a resposta imunológica dos animais.

Apesar de se tratar de técnicas criadas por autores diferentes, em momentos diferentes e com focos diferentes, todas têm uma estrutura básica em comum: realizam reprodução, impõem variações aleatórias, promovem competição e executam seleção de indivíduos de uma dada população. De acordo com Zuben (2000), sempre que estes quatro processos estiverem presentes, seja na natureza ou em simulação computacional, a evolução é o produto resultante.

Para um bom entendimento dos algoritmos relacionados com a EC, alguns conceitos devem ser definidos (FIALHO et al., 2007):

- **Indivíduo**: candidato à solução do problema em questão, a sua codificação depende da técnica evolutiva escolhida e do problema a ser resolvido.
- **Genes**: cada atributo do indivíduo. Por exemplo, quando o indivíduo é codificado na forma de um vetor de valores reais, cada posição do vetor é um gene; quando codificado na forma de árvore, cada nó da árvore é um gene.
- **População**: coleção de indivíduos que competem entre si pela sobrevivência.
- **Aptidão do Indivíduo (*fitness*)**: totalmente dependente do problema, define o quanto este candidato à solução está apto para solucionar o problema. Pode ser feita uma analogia com o papel do meio-ambiente na teoria da evolução das espécies: "só os indivíduos mais adequados/aptos/fortes sobrevivem".
- **Função de Avaliação**: terá o trabalho de julgar a aptidão de cada indivíduo da população. Deve julgar a qualidade/utilidade da solução que está sendo apresentada por um dado indivíduo.

3.2 Algoritmos Evolutivos

A expressão genérica Algoritmo Evolutivo (EA - *Evolutionary Algorithm*) é usada para classificar um amplo conjunto de técnicas heurísticas de resolução de problemas complexos que baseiam o seus funcionamento em um mecanismo semelhante ao do processo de evolução natural. Trabalhando com um conjunto de possíveis soluções para um dado problema, a metodologia utilizada por estas técnicas se fundamenta no uso de mecanismos de seleção das melhores soluções e geração de novas soluções candidatas através da recombinação de características das soluções selecionadas (WEISE, 2008).

Vários algoritmos têm sido propostos com a filosofia dos EAs (WEISE, 2008). A figura 3.1 ilustra o ciclo básico de funcionamento de um algoritmo evolutivo que opera sobre uma população P de indivíduos.

Um EA trabalha sobre um conjunto de indivíduos que representam possíveis soluções para o problema, codificados de acordo com algum mecanismo pré-determinado: a codificação relaciona-se com a modelagem do problema. Os indivíduos são avaliados de acordo com uma função de avaliação para o problema em questão. Normalmente, em problemas de otimização,

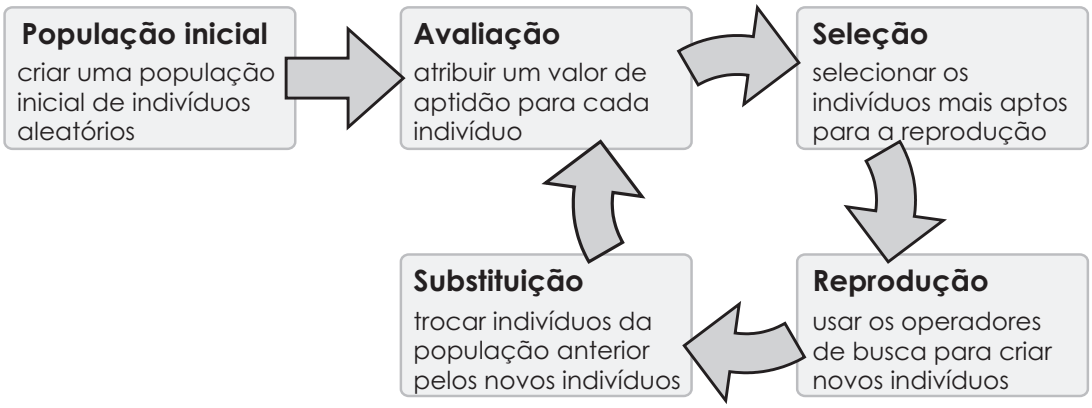


Figura 3.1: O ciclo básico dos algoritmos evolutivos.

o valor da função de avaliação (ou função de aptidão) para um indivíduo é igual ao valor da função de objetivo do problema. Contudo esta regra não é válida para todos os casos.

A operação do EA começa com a fase de inicialização dos indivíduos, que pode ser feita de forma completamente aleatória (seguindo uma distribuição uniforme de probabilidades) ou guiada por alguma heurística (quando temos algum conhecimento inicial do problema). O objetivo desta fase é espalhar os indivíduos sobre o espaço de busca para que os mesmos realizem o processo de buscar o ponto ótimo do problema. Para alguns EAs, garantir a diversidade da população inicial é um ponto importante para que seus mecanismos de busca funcionem eficientemente (WEISE, 2008).

As quatro fases seguintes são responsáveis pelo processo de evolução dos indivíduos. A evolução ocorre em um processo cíclico, onde a cada nova iteração novos indivíduos são gerados através da aplicação dos *operadores de busca* (também chamados de *operadores de reprodução* no contexto dos EAs) (WEISE, 2008). Este ciclo está dividido em quatro etapas:

- **Avaliação**: Etapa que consiste em atribuir um valor de aptidão (*fitness*) para cada indivíduo da população. Este valor avalia o quão bem cada indivíduo resolve o problema em questão e é utilizado para guiar o processo evolutivo.

- **Seleção**: Processo que determina quais são os indivíduos mais adequados para se aplicar os operadores de busca, com o objetivo de gerar os novos indivíduos que irão compor a nova população. De forma geral, escolhem-se os indivíduos com as melhores aptidões

- **Reprodução**: Etapa que gera um conjunto de descendentes a partir da aplicação dos operadores de busca (também chamados de operadores evolutivos ou estocásticos) sobre os indivíduos selecionados na etapa anterior.

- **Substituição**: Nesta etapa, um mecanismo que realiza o intercâmbio geracional será responsável por substituir os indivíduos da geração anterior pelos novos indivíduos (descendentes) criados na etapa anterior: uma nova população será gerada. Normalmente, a substituição se dá quando um filho é melhor que seu(s) pai(s).

Atualmente existem diversas políticas de seleção e substituição de indivíduos que permitem alterar as características de operação dos algoritmos evolutivos (WHITACRE, 2007). Por exemplo, se aplicarmos as políticas adequadas poderemos privilegiar os indivíduos mais adaptados em cada geração (estratégia chamada de elitista), aumentar a pressão seletiva sobre os indivíduos melhores adaptados, gerar um número reduzido de descendentes em cada geração, aumentar a diversidade da população, e outras muitas variantes. Em (WHITACRE, 2007) temos uma listagem bastante completa sobre alguns algoritmos de seleção e substituição.

A forma com a qual o algoritmo evolutivo vai explorar o espaço de busca do problema é definida pelos operadores de busca (também chamados de operadores evolutivos). Depois de mais de 40 anos de pesquisas na área da EC, existe uma grande diversidade de operadores de busca propostos na literatura (WHITACRE, 2007). Os diferentes tipos de operadores existentes, com suas particularidades, dão características peculiares às diferentes variantes de algoritmos evolutivos. Os operadores de recombinação (ou cruzamento), que permitem combinar características de dos ou mais indivíduos com a ideia de obter descendentes melhores adaptados, e os operadores de mutação, que introduzem diversidade (se refere a ter indivíduos distintos dentro da população) mediante a variações aleatórias, são os operadores de busca mais difundidos.

Como todo processo iterativo, um algoritmo evolutivo necessita de um critério de parada. Normalmente, o critério de parada de um algoritmo evolutivo leva em consideração a quantidade de gerações processadas, interrompendo o processo iterativo de evolução após um determinado número de gerações. Outras alternativas usam como critério de parada a variação dos valores de aptidão dos indivíduos, interrompendo o processo evolutivo quando o sistema convergir e nenhuma melhora considerável for observada, ou a estimativa do erro, interrompendo o processo evolutivo quando se atinge um erro mínimo em relação a um valor ideal para o problema. Uma outra alternativa para o critério de parada é realizar um número máximo de avaliações do problema (cálculo da função objetivo e restrições).

A grande vantagem de ser usar um EA é a sua facilidade de implementação e a sua eficiência em diversos problemas de otimização. Apesar de ser um método probabilístico, é bastante robusto e se bem implementado pode gerar respostas satisfatórias. Outra vantagem dos EAs, em comparação com outros métodos de otimização é a sua facilidade de trabalhar com sistemas do tipo “caixa preta”, fazendo apenas algumas hipóteses sobre as funções objetivo e encontrado

boas soluções, embora de forma probabilista, para o problema. Portanto, sabemos que os EAs conseguem ter um bom desempenho em muitas categorias de problemas diferentes (WEISE, 2008). Atualmente, os EAs possuem uma grande aplicação em problemas reais.

3.2.1 Exemplo de um Algoritmo Evolutivo Simples

Como exemplo, o Algoritmo 1 mostra uma típica implementação de EA. Dado que queremos resolver um problema P com n dimensões, o passo a passo do Algoritmo 1 e mostrado a seguir:

1. Na iteração $t = 0$, a função *criarPop* executa a tarefa de criar uma população inicial aleatória de $NPop$ indivíduos uniformemente distribuídos pelo espaço de busca $\mathbb{X}$ de P: $Pop = \{\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^{NPop}\}$ onde $\vec{x}^i = \left[x_1^i, x_2^i, \ldots, x_n^i\right]$.

2. O critério de parada para o "ciclo evolutivo" é um número máximo de iterações/gerações, de forma que o algoritmo realizará $Tmax$ iterações.

3. A função $avaliarAptidao$ avalia todos os $NPop$ indivíduos da população Pop, calculando o valor da função de aptidão para todos.

4. Um algoritmo de seleção é implementado na função *selecionar* de forma a selecionar os indivíduos que irão reproduzir e gerar descendentes. Uma função de comparação cmp_F é usada para a comparar a utilidade de dois indivíduos quaisquer.

5. Com a função *reproduzirPop* uma nova população Pop de $NPop$ indivíduos é gerada: os indivíduos que foram selecionados e armazenados em $Mate$ sofrerão a aplicação dos operadores de busca Ops (mutação, cruzamento, etc.).

6. A função *melhorIndividuo* retorna o melhor indivíduo da população Pop: esta será a melhor solução para P, dado o critério de comparação implementado em cmp_F.

De acordo com (FIALHO et al., 2007), o desempenho e a eficiência de um algoritmo evolutivo está fortemente relacionado à definição de alguns parâmetros a serem utilizados, cujos valores devem ser estudados de acordo com as necessidades do problema e dos recursos disponíveis. Na grande maioria dos casos, esses parâmetros são definidos empiricamente e com base na análise do problema. Neste contexto, podemos listar os seguintes parâmetros:

- **Tamanho da População**: afeta o desempenho global e a eficiência dos EAs. Se a população é pequena, consequentemente a cobertura do espaço de busca por geração

Algoritmo 1: Implementação de um EA simples.

Entrada: cmp_F: função de comparação que nos permite comparar dois candidatos
Entrada: Ops: conjunto de operadores de busca
Entrada: $NPop$: número de indivíduos da população
Entrada: $Tmax$: número máximo de iterações/gerações
Dados: t: contador de gerações
Dados: Pop: população com $NPop$ indivíduos
Dados: $Mate$: indivíduos selecionados para reprodução
Saída: $\vec{x}^*$: melhor solução encontrada para o problema

```
1  início
2      t ⟵ 0
3      Pop ⟵ criarPop(NPop)
4      enquanto t < Tmax faça
5          avaliarAptidao(Pop)
6          Mate ⟵ selecionar(Pop, cmp_F)
7          Pop ⟵ reproduzirPop(Mate, Ops)
8          t ⟵ t + 1
9      fim
10     retorna melhorIndividuo(Pop, cmp_F)
11 fim
```

também é pequena, o que influencia muito no desempenho do algoritmo. Se a população é grande, fornecendo uma cobertura representativa sobre o domínio do problema, a possibilidade de acontecer uma convergência prematura para uma solução local em vez de global se torna muito pequena. Entretanto, quanto maior a população, mais recursos computacionais e temporais serão necessários. Tipicamente, a população é muito menor do que o espaço de busca a ser explorado.

- **Probabilidades de Aplicação dos Operadores**: um EA é formado por um repertório de NOp operadores de busca responsáveis por explorar os espaço de busca localmente ou globalmente. A cada iteração t o operador de busca Op_i tem uma probabilidade p_i de ser escolhido e aplicado sobre um ou mais indivíduos. Saber ajustar esses valores de forma correta é um passo fundamental para o desempenho e a eficiência do algoritmo. Como exemplo, em um Algoritmo Genético (GA) básico temos apenas dois operadores: operador de cruzamento, aplicado com uma determinada *taxa de cruzamento*, e operador de mutação, aplicado com uma *taxa de mutação*. Sendo assim temos:

  1. **Taxa de Cruzamento**: Quanto maior a probabilidade de realização de cruzamentos, mais diversidade será inserida na população. Mas se essa taxa for muito alta, indivíduos com boas características para a solução do problema poderão ser perdidos; e se for muito baixa, a busca pode estagnar. Dependendo do estágio em que

se encontra a busca, ou seja, dependendo da maturidade da população, tal taxa pode sofrer variações.

2. **Taxa de Mutação**: Uma pequena probabilidade de realização de mutação já garante que a busca não se prenda em sub-regiões do espaço de busca, possibilitando que qualquer ponto seja atingido. Quanto maior a taxa de mutação, maior a aleatoriedade da busca. Assim como para o cruzamento, a taxa de mutação também pode variar de acordo com a maturidade da população.

3.3 Os Principais Paradigmas

Os primeiros algoritmos baseados nos conceitos da computação evolutiva surgiram por volta dos anos 60, abstraindo alguns mecanismos de evolução encontrados na natureza e os utilizando na resolução de problemas complexos de busca, otimização e aprendizado de máquina (WEISE, 2008).

Conforme ilustrado na figura 3.2, a família dos algoritmos evolutivos engloba cinco membros: Estratégias Evolutivas, Programação Evolutiva, Algoritmos Genéticos, Sistemas Classificadores com Aprendizagem e Programação Genética. A seguir teremos uma descrição bastante resumida dos algoritmos que são os principais paradigmas da computação evolutiva.

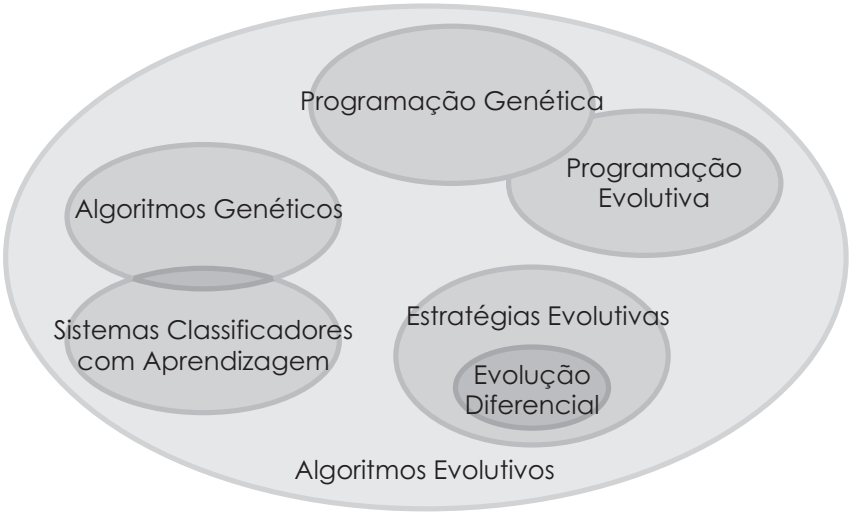


Figura 3.2: A família dos algoritmos evolutivos. Fonte: (WEISE, 2008)

3.3.1 Estratégias Evolutivas

As ESs (*Evolution Strategies*) foram desenvolvidas pelo pesquisador Rechenberg, em 1965 na Alemanha, na tentativa de resolver problemas de hidrodinâmica e de controle (FIALHO et

al., 2007), sendo a primeira técnica a explorar a teoria da evolução das espécies na solução de problemas. A primeira versão chamada de $(1+1)-ES$ (a sigla ES vem do inglês *Evolution Strategies*) ou de estratégia evolutiva de dois membros, utiliza apenas um pai e gera somente um filho por iteração. Este filho é mantido somente se for melhor (mais apto) que seu pai (WEISE, 2008).

Nas ESs, o principal operador é o operador de mutação. O operador de recombinação desempenha um papel secundário e pode ser omitido. Como exemplo, em um algoritmo $(1+1)-ES$, para gerar novos indivíduos a seguinte função pode ser usada:

$$x'_t = x_t + N(0, \sigma_t) \tag{3.1}$$

onde t se refere a geração atual, x'_t é o filho, x_t é o pai, e $N(0,\sigma_t)$ é um gerador de números aleatórios sobre um distribuição Gaussiana com média 0 e desvio padrão σ_t. Devemos notar que esta função trabalha com números reais.

De acordo com (FIALHO et al., 2007), as primeiras variantes de ESs não utilizavam o princípio da população, memorizando sempre apenas um ponto no espaço de busca (o "pai"), que poderia gerar um ou mais filhos, memorizando sempre apenas o melhor dos filhos para a próxima geração. Posteriormente, Rechenberg introduziu o conceito de população, tornando os algoritmos menos suscetíveis a convergência para mínimos locais, dado a maior diversidade.

Na estratégia $(\mu+1)-ES$ de Rechenberg, μ pais geram um único filho, que irá substituir o pior indivíduo da população. Schwefel propôs depois duas novas estratégia onde μ pais geram λ filhos: $(\mu+\lambda)-ES$ e $(\mu,\lambda)-ES$. Na primeira estratégia, o processo de seleção leva em conta tanto os filhos quanto os pais; na segunda estratégia, o processo de seleção utiliza apenas os filhos. O algoritmo 2 mostra um esquema genérico das ESs.

Algoritmo 2: Esquema genérico das ESs.

```
1  início
2      Gerar a população inicial
3      Calcular a aptidão de todos os indivíduos
4      enquanto (não "condição de parada") faça
5          Selecionar um conjunto de pais de maneira aleatória
6          Aplicar o operador de mutação nos pais selecionados
7          Calcular a aptidão de cada novo indivíduo
8          Realizar a seleção dos melhores indivíduos ("+" ou ",")
9      fim
10     Retornar o melhor indivíduo da população
11 fim
```

O processo de substituição das ESs é determinístico, de modo que somente os melhores indivíduos são mantidos para a próxima geração. Já o processo de seleção dos pais é aleatório: algum critério de seleção é usado (Roleta, *Ranking*, Torneio Estocástico, etc).

3.3.2 Programação Evolutiva

A Programação Evolutiva (EP - *Evolutionary Programming*) foi proposta por Lawrence J. Fogel em 1966. Neste paradigma, a inteligência é vista como um comportamento adaptativo. No inicio, seu objetivo era evoluir máquinas de estado finito para predição de símbolos (a aptidão de cada máquina era medida pelo número de símbolos corretamente previsto pela mesma), mas hoje em dia a EP é usada em uma vasta quantidade de problemas, não tendo uma estrutura de representação fixa (FIALHO et al., 2007). Um esquema genérico da EP é mostrado no algoritmo 3.

De acordo com Fogel (1994), a EP hoje pode ser considerada como virtualmente equivalente às Estratégias Evolutivas em problemas de otimização, apresentando apenas algumas pequenas diferenças no que se diz respeito aos procedimentos de codificação e seleção dos indivíduos.

O seu principal operador é a mutação. Os indivíduos de uma população são vistos como indivíduos de diferentes espécies, ao contrário das outras técnicas evolutivas. Assim, cada pai, através da mutação, gera apenas um filho.

Algoritmo 3: Esquema genérico de um algoritmo da EP.

```
1  início
2      Gerar a população inicial
3      Calcular a aptidão de todos os indivíduos
4      enquanto (não "condição de parada") faça
5          Aplicar o operador de mutação para gerar novos indivíduos
6          Calcular a aptidão dos novos indivíduo
7          Selecionar os indivíduos que ficarão para a próxima geração
8      fim
9      Retornar o melhor indivíduo da população
10 fim
```

Diferentemente das ESs, o processo de substituição dos algoritmos baseados no paradigma da EP é probabilístico, ou seja, feito através de torneios estocásticos (nesse método, N indivíduos são escolhidos aleatoriamente com a mesma probabilidade - dentre os escolhidos, o que tiver maior aptidão é selecionado para ficar para próxima geração). Já o processo de seleção dos pais é determinístico: todos os indivíduos da população geram descendentes a cada iteração.

3.3.3 Algoritmos Genéticos

Nos anos 60, o pesquisador John Holland estudou o projeto de sistemas artificiais baseados nos mecanismos dos sistemas naturais e idealizou os GAs (*Genetic Algorithms*). Seu objetivo inicial foi a formalização matemática e a explicação rigorosa dos processo de adaptação em sistemas naturais. Posteriormente, os GAs foram melhor desenvolvidos por seu grupo de pesquisa na Universidade de Michigan (WEISE, 2008).

O GA de Holland é um método para transformar uma população de indivíduos (soluções), representada pelo *cromossomos* (a representação é tradicionalmente feita através de uma codificação binária ou algum alfabeto finito codificado em cromossomos lineares) em uma nova população melhor adaptada ao ambiente (WEISE, 2008). A ideia é que as novas populações herdem as características dos melhores indivíduos da geração passada e através de algumas variações (busca local/global) se tornem cada vez mais adaptadas ao ambiente.

De acordo com (FIALHO et al., 2007), normalmente os pesquisadores e usuários dos GAs referem-se a "Algoritmos Genéticos" ou a "um Algoritmo Genético" e não "ao Algoritmo Genético". Isso se deve ao fato de que os GAs são uma classe de procedimentos com um conjunto de passos distintos e bem especificados, com cada um destes passos tendo inúmeras possíveis variações.

No Algoritmo 4 podemos ver um esquema genérico de um GA. Este algoritmo enfatiza a importância do operador de cruzamento (principal operador dos GAs) em relação ao operador de mutação e utiliza uma seleção de pais probabilística (por exemplo: Roleta ou Torneio estocástico) e uma substituição geracional (a nova geração é composta apenas pelos descendentes da geração antiga). Apesar de existir uma grande quantidade de variantes de GAs, o seu mecanismo fundamental consiste de três operações básicas: avaliação da aptidão de cada indivíduo, aplicação dos operadores genéticos, e formação de uma nova população.

Nos GAs, a codificação binária é a mais comum e, dada a sua universalidade, os operadores básicos são definidos com base em tal representação. Na terminologia adotada pela EC, a cadeia binária (*string binário*) que codifica um conjunto de soluções se chama *cromossomo*, cada segmento da cadeia que codifica uma variável é chamado de *gene* e o valor de cada posição cromossômica se chamada *alelo* (bit que pode ser 0 ou 1).

Em comparação com as ESs e a EP, o processo de seleção é baseado na aptidão dos indivíduos (indivíduos com maior aptidão tem uma probabilidade maior de se reproduzir e gerar descendentes) e o processo de substituição é de forma geracional, ou seja, toda a geração progenitora é substituída pelos novos indivíduos.

Algoritmo 4: Esquema de um GA genérico.

```
1  início
2      Gerar a população inicial
3      Calcular a aptidão de todos os indivíduos
4      enquanto (não "condição de parada") faça
5          Selecionar um conjunto de pais com base em suas aptidões
6          Criar dois filhos para cada par de pais usando o operador de cruzamento
7          Aplicar o operador de mutação nos novos indivíduos
8          Calcular a aptidão dos novos indivíduo
9          Todos os novos indivíduos entram na nova população e os pais morrem
10     fim
11     Retornar o melhor indivíduo da população
12 fim
```

3.3.4 Sistemas Classificadores com Aprendizagem

Outra ideia de John Holland, são os Sistemas Classificadores (CS) que classificam uma ou várias entradas e produzem uma ou várias saídas. Na sua forma mais simples, uma série de regras "*if this than that*" determinam a relação entre as entradas e as saídas. Em sistemas mais complexos, uma aplicação de um Sistemas Classificadores com Aprendizagem (LCS - *Learning Classifier System*) pode usar GA para adaptar as suas operações a um ambiente em constante alteração ou dependendo das suas saídas. Um LCS pode também evoluir regras de um conjunto de classificações iniciais aleatoriamente geradas.

Para a construção de tais sistemas é necessário: um ambiente, receptores que informam ao sistema sobre o que está ocorrendo, atuadores que permitem ao sistema manipular o seu ambiente e o sistema em si.

Por exemplo, um LCS para a área de investimentos em ações, "vigiaria" os valores de compra e venda do "papel" para produzir recomendações (Compra, Vende,...) baseado nas suas regras de classificação da informação. Um sistema evolutivo pode aprender através dos hábitos de compra e venda dos investidores, refinando as suas respostas para melhor se adaptar às necessidades do seu utilizador. Os primeiros Sistemas de Classificação apareceram nos anos 70, baseados nas ideias iniciais de John Holland.

3.3.5 Programação Genética

Segundo (FIALHO et al., 2007), a Programação Genética (GP - *Genetic Programming*) é a técnica mais recente sob a égide da EC, introduzida por Koza (1992). Trata-se de uma extensão dos GAs que tem por objetivo básico evoluir programas de computador ao invés de

soluções codificadas, usando os mesmos princípios da evolução natural. Ou seja, os objetos que constituem a população não são vetores que codificam soluções para o problema em questão. Tratam-se de programas que, quando executados, "são" as soluções-candidatas para o problema.

Na GP, os indivíduos (programas) são codificados em árvores de tamanho variável. Por exemplo, um programa simples que gera uma população aleatória de indivíduos é apresentado na figura 3.3: podemos ver um algoritmo que foi gerado por um GP como solução para um dado problema, a árvore que codifica o algoritmo gerado (indivíduo/solução) e sua implementação em uma linguagem de alto nível.

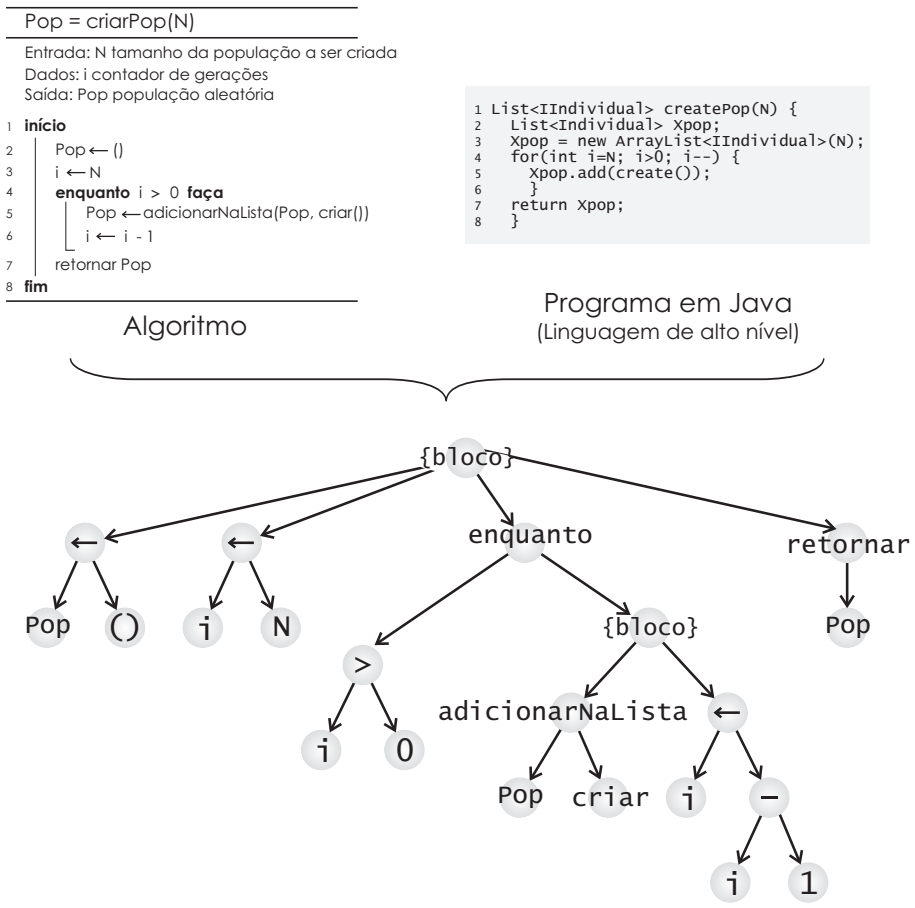


Figura 3.3: Algoritmo para exemplo de representação GP.

Com esse tipo de representação, têm-se como vantagem uma estrutura hierárquica, ao invés de vetores unidimensionais (GAs). Além disso, a estrutura é composta de simples funções, que podem ser facilmente codificadas usando uma linguagem de alto-nível (FIALHO et al., 2007).

3.4 Os Novos Paradigmas

Outras propostas mais recentes e que compartilham características com os paradigmas tradicionais da computação evolutiva serão apresentadas a seguir. Por fazerem parte do escopo deste trabalho, as metodologias Evolução Diferencial (DE - *Differential Evolution*) e Otimização por Exame de Partículas (PSO - *Particle Swarm Optimization*) terão uma descrição mais detalhada.

3.4.1 Otimização por Enxame de Partículas

A PSO (do inglês, *Particle Swarm Optimization*) foi proposta por Kennedy e Eberhart em 1995 para simular o comportamento de um bando de pássaros ou um cardume de peixes. É considerado um algoritmo de inteligência coletiva (*Swarm Intelligence Algorithm*) porque tenta imitar o comportamento dos organismos sociais (WEISE, 2008).

No caso dos pássaros, o bando escolhe um líder para guiar a busca por comida, de forma que todos os outros indivíduos serão atraídos pelo líder e passarão a segui-lo em sua busca. O líder do bando deve ser o indivíduo com melhor valor de aptidão: critério que garante que a busca seja global. Contudo, cada indivíduo do bando irá manter uma busca pessoal (busca local) por comida e, caso encontre alguma coisa, tentará informar aos outros indivíduos. Naturalmente, quando existir um indivíduo com melhores atributos (melhor aptidão) que o atual líder, este se tornará o novo líder do bando e atrairá os outros indivíduos para próximo de si.

A metodologia PSO trabalha com um exame de partículas (população) onde cada partícula representa uma solução em potencial para o problema. Cada partícula se move pelo espaço de busca $\mathbb{X}$ influenciada pela sua experiência pessoal (através da memorização da melhor posição de $\mathbb{X}$ que já esteve) e pela experiência coletiva do bando (o líder vai atrair as outras partículas para o melhor ponto de $\mathbb{X}$ que já visitou). Aqui temos o conceito de *influencia cognitiva*, quando uma partícula é influenciada pela melhor posição em que já esteve, e o conceito de *influencia social*, se relacionando com a influencia do líder sobre as outras partículas do bando.

Num algoritmo de PSO, cada indivíduo deve ser capaz de conhecer a região do espaço de busca em que se encontra de forma a evitar colisões com outros indivíduos (isso evita que dois indivíduos pesquisem uma mesma região do espaço de busca) e evitar perder de vista a direção do líder.

As partículas são representadas como vetores n-dimensionais, onde n é o número de variáveis de controle do problema a ser resolvido. Sendo assim, $\vec{x}_i(t)$ representa a partícula i no instante t, $\vec{v}_i(t)$ é a velocidade da partícula i no instante t, e t representa passos discretos de tempo (al-

goritmo iterativo). A posição da partícula i no instante $(t+1)$ é dada pela seguinte expressão:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{3.2}$$

onde $j = 1, 2, \ldots, n$.

De forma a guiar a busca de uma partícula para melhores posições no espaço de busca, o vetor velocidade integra os componentes de conhecimento social e pessoal de cada partícula. Sendo assim, a velocidade da partícula i no instante $(t+1)$ é calculada usando a seguinte expressão:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (x^*_{ij} - x_{ij}(t)) + c_2 r_2 (x^*_{Gj} - x_{ij}(t)) \tag{3.3}$$

onde:

- w (ponderação de inércia): Determina quanta influência a velocidade atual $\vec{v}_i(t)$ tem no cálculo de seu novo valor $\vec{v}_i(t+1)$. Esse parâmetro causa uma inercia no movimento das partículas, evitando mudanças bruscas da direção e movimento.
- $\vec{v}_i(t)$ (velocidade atual): É a atual velocidade da partícula i. Normalmente, se considera que no estado inicial do sistema as partículas possuem velocidade nula: $\vec{v}_i(0) = 0$.
- $\vec{x}_i(t)$ (posição atual): É a atual posição da partícula i.
- c_1 e c_1: Estes parâmetros são valores positivos e constantes que governam a influência dos fatores cognitivo e social, respectivamente.
- r_1 e r_2: São números aleatórios, sorteados no intervalo $[0,1]$ e com distribuição uniforme.
- $\vec{x}^*_i$: É o melhor ponto $pBest_i$ já visitado pela partícula i até o instante t. Em outras palavras, o ponto $pBest_i$ é um ótimo local para i:

$$pBest_i = \min_{t=1,\ldots,k} f(\vec{x}_i(t)),\ \vec{x}^*_i = arg \min_{t=1,\ldots,k} f(\vec{x}_i(t)) \tag{3.4}$$

- $\vec{x}^*_G$: É o melhor ponto $gBest$ já visitado pelo bando até o instante t: é um ponto de ótimo global para o sistema:

$$gBest = \min_i pBest_i,\ \vec{x}^*_G = arg \min_i f(\vec{x}^*_i) \tag{3.5}$$

A figura 3.4 mostra um exemplo gráfico do algoritmo de PSO em um problema bidimensional.

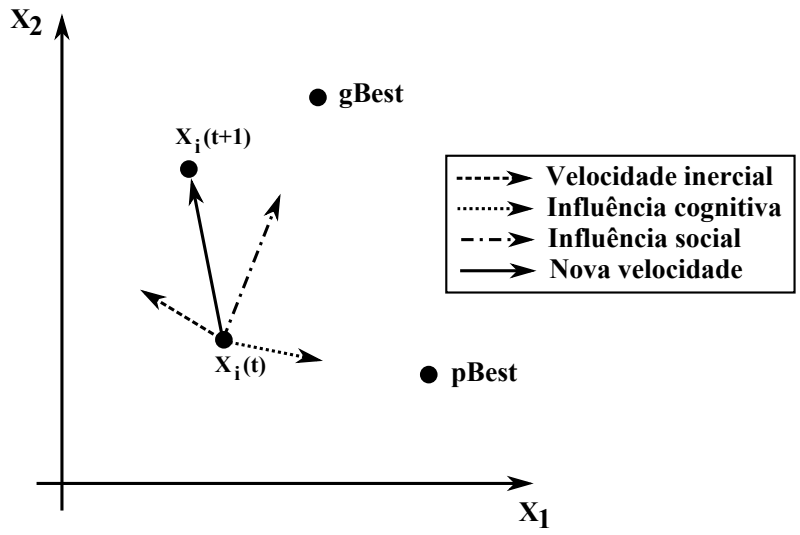


Figura 3.4: Exemplo gráfico do mecanismo de atualização da velocidade no algoritmo de PSO usando as equações (3.2) e (3.3)

Basicamente, existe duas versões da PSO que diferem quanto a estratégia usada para atualizar a velocidade das partículas. A primeira versão é chamada de "melhor global" (estratégia ***gBest***), onde existe apenas um líder no enxame: a cada iteração, testa-se todas as partículas para saber quem será o novo líder; esse novo líder carrega a informação do melhor pondo do espaço de busca já visitado pelo enxame. Nesta estratégia, as partículas além de fazerem uma busca local são atraídas pelo líder do enxame, fazendo com que os indivíduos, ao final do processo, convirjam para uma dada região do espaço. O Algoritmo 5 mostra um esquema genérico para a estratégia *gBest*, onde queremos minimizar um problema com função objetivo $f(\vec{x})$.

A segunda estratégia da PSO é conhecida como "melhor local" (***lBest***). Nesta variante, o espaço de busca é dividido em *NR* subregiões e cada subregião terá um grupo de *NP* partículas aleatoriamente espalhadas. As partículas têm como objetivo encontrar o melhor ponto de sua subregião, que será um ponto de ótimo local para o problema. Nesta estratégia, cada subregião possui o seu próprio líder e este influencia apenas as partículas de sua subregião. Desta forma, teremos um algoritmo que ao final do processo iterativo, retorna um conjunto de boas soluções (ótimos locais) para o problema (metodologia chamada de *niching*, onde o problema requer a identificação e manutenção de múltiplas soluções) (WEISE, 2008).

3.4.2 Otimização por Colônia de Formigas

Estes algoritmos são sistemas multiagentes nos quais o comportamento de cada agente (chamado de formiga, cada agente será uma solução para um determinado problema) se baseia na simulação do mecanismo natural de procura por comida das formigas. Os algoritmos ACO

Algoritmo 5: Esquema de uma algoritmo PSO baseado na estratégia *gBest*

Entrada: NP - número de partículas do enxame
Entrada: $Tmax$ - número máximo de iterações
Saída: $\vec{x}^*_G$ - melhor solução encontrada para o problema

```
início
    t ← 0
    Criar e inicializar aleatoriamente as NP partículas do enxame
    Calcular a aptidão de todas as partículas do enxame
    Inicializar x⃗*_i (pBest) para todas as partículas
    Inicializar v⃗_i(0) = 0 para todas as partículas
    enquanto t < Tmax faça
        t ← t + 1
        Encontrar o líder x⃗*_G do enxame (ponto gBest)
        para i ← 1 até NP faça
            para j ← 1 até n faça
                Calcular v_ij(t + 1) usando a expressão (3.3)
                Calcular x_ij(t + 1) usando a expressão (3.2)
            fim
            se f(x⃗_i) ≤ f(x⃗*_i) então
                x⃗*_i ← x⃗_i
            fim
        fim
    fim
    Retornar a melhor partícula do enxame (ponto gBest)
fim
```

(*Ant Colony Optimization*) são um dos exemplos mais bem sucedidos de sistemas baseados na simulação do comportamento natural de insetos/animais, e cuja aplicação tem sido feita nas mais diversas áreas do conhecimento (WEISE, 2008).

3.4.3 Evolução Diferencial

Desenvolvido por Storn e Price (STORN; PRICE, 1995), a Evolução Diferencial é um algoritmo evolutivo baseado em populações que pode ser classificado com um tipo de Estratégia Evolutiva (ver figura 3.2) Trata-se de uma método de busca direta estocástica onde a ideia principal é usar diferenças de vetores (daí o nome diferencial) para gerar perturbações nos indivíduos da população, evoluindo-os para a região do ótimo global do problema. Este método requer o ajuste de poucos parâmetros de controle, é de rápida convergência, é de fácil implementação, trabalha bem com problemas multimodais e é robusto quando bem implementado (WEISE, 2008).

No DE, a codificação dos indivíduos é feita usando vetores de números reais (codificação

real). Possui um mecanismo simples de mutação e um operador de cruzamento (operador diferencial) que realiza uma combinação linear de um certo número de indivíduos (normalmente três). Cada indivíduo gera apenas um filho por iteração; se o filho for melhor que o pai então assume o seu lugar na população. Desta forma, a etapa de seleção é determinística (WEISE, 2008).

Segundo (OLIVEIRA, 2006), a escolha do algoritmo DE para otimização numérica está baseada nas seguintes características:

- É um algoritmo de busca estocástica, originado dos mecanismos de seleção natural (algoritmo evolutivo);
- Possui capacidade para trabalhar com funções objetivo não lineares, não diferenciáveis e multimodais;
- É muito eficaz para resolver problemas de otimização com função objetivo descontínua, pois não requer informação sobre suas derivadas;
- Dificilmente fica preso em ótimos locais, pois busca a solução ótima global manipulando uma população de soluções, ou seja, buscando simultaneamente a solução em diferentes regiões do espaço de busca;
- Permite que as variáveis do problema sejam manipuladas como números ordinários reais (pontos flutuantes) sem processamento extra (caso da codificação binária), e, portanto, utiliza eficientemente os recursos do computador;
- Trabalha bem como otimizador local porque os diferenciais gerados por uma população convergente eventualmente tornam-se infinitesimais;
- É eficaz trabalhando com uma população relativamente pequena;
- É um algoritmo facilmente paralelizável;
- Na sua forma original, utiliza apenas três parâmetros de controle, além de um critério de parada.

Descrição do Algoritmo

Seja $\mathbb{X} \subseteq \mathbb{R}^n$ o espaço de busca n-dimensional de um dado problema P. Um algoritmo DE deve possuir uma população de NP vetores n-dimensionais, tal que $\vec{x}_i = [x_1, \ldots, x_n]$ para $i = 1, \ldots, NP$ é uma solução candidata para P, espalhados aleatoriamente pelo espaço de busca

$\mathbb{X}$ segundo uma distribuição de probabilidade uniforme. Através de um processo iterativo de evolução, os NP indivíduos desta população devem convergir para o ponto de ótimo global de P. A cada iteração t do algoritmo, os operadores de mutação e cruzamento de uma dada *estratégia DE* são aplicados aos indivíduos da população como forma de produzir um vetor filho $\vec{u}_i^{t+1}$ para cada vetor pai $\vec{x}_i^t$; se $\vec{u}_i^{t+1}$ possuir um valor de aptidão maior que $\vec{x}_i^t$, então devemos fazer $\vec{x}_i^{t+1} = \vec{u}_i^{t+1}$. Durante todo o processo, o número de indivíduos da população permanece fixo.

Os operadores da DE se baseiam no princípio da evolução natural cujos objetivos são manter a diversidade da população, evitar convergências prematuras e obter a melhor solução.

Operador de Mutação

A operação de mutação num algoritmo DE gera novos indivíduos, denotados por vetores modificados ou doadores, pela adição da diferença vetorial ponderada entre dois indivíduos aleatórios da população a um terceiro indivíduo. Esta é a ideia básica do DE, contudo existem várias estratégia para se realizar a operação de mutação, como será mostrado a seguir.

Existe uma nomenclatura especifica para os operadores de mutação DE, seguindo o seguinte formato: "**Técnica/VP/NV/TR**". Segue a nomenclatura:

- **Técnica**: Representa a heurística usada. A Evolução Diferencial será indicada como DE.
- **VP**: Vetor a ser perturbado, também chamado de vetor base.
  - **rand**: Indica que a seleção do vetor base é feita de forma aleatória.
  - **best**: Indica que o vetor base é o melhor indivíduo da população: $\vec{x}_{best}$.
  - **current-to-best**: Indica uma recombinação linear dos vetores pai e mutação.
- **NV**: Indica o número de pares de vetores que formam o vetor diferença.
- **TR**: Tipo do algoritmo usado para fazer o cruzamento (ou recombinação).
  - **bin**: O cruzamento é do tipo binomial.
  - **exp**: O cruzamento é do tipo exponencial.

Na iteração t do algoritmo DE, cada vetor pai $\vec{x}_i^t$ possui um vetor de mutação $\vec{v}_i^t = [v_{i1}^t, \ldots, x_{in}^t]$ gerado por algumas das estratégia de aprendizagem do DE. Em (OLIVEIRA, 2006) temos a descrição de 10 estratégias DE:

1. **DE/rand/1/bin**:

$$\vec{v}_i^t = \vec{x}_{r1}^t + F(\vec{x}_{r2}^t - \vec{x}_{r3}^t)$$

2. **DE/best/1/bin**:

$$\vec{v}_i^t = \vec{x}_{best}^t + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t)$$

3. **DE/rand/2/bin**:

$$\vec{v}_i^t = \vec{x}_{r1}^t + F(\vec{x}_{r2}^t - \vec{x}_{r3}^t) + F(\vec{x}_{r4}^t - \vec{x}_{r5}^t)$$

4. **DE/best/2/bin**:

$$\vec{v}_i^t = \vec{x}_{best}^t + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t) + F(\vec{x}_{r3}^t - \vec{x}_{r4}^t)$$

5. **DE/current-to-best/2/bin**:

$$\vec{v}_i^t = \vec{x}_i^t + K(\vec{x}_{best}^t - \vec{x}_i^t) + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t)$$

6. **DE/rand/1/exp**:

$$\vec{v}_i^t = \vec{x}_{r1}^t + F(\vec{x}_{r2}^t - \vec{x}_{r3}^t)$$

7. **DE/best/1/exp**:

$$\vec{v}_i^t = \vec{x}_{best}^t + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t)$$

8. **DE/rand/2/exp**:

$$\vec{v}_i^t = \vec{x}_{r1}^t + F(\vec{x}_{r2}^t - \vec{x}_{r3}^t) + F(\vec{x}_{r4}^t - \vec{x}_{r5}^t)$$

9. **DE/best/2/exp**:

$$\vec{v}_i^t = \vec{x}_{best}^t + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t) + F(\vec{x}_{r3}^t - \vec{x}_{r4}^t)$$

10. **DE/current-to-best/2/exp**:

$$\vec{v}_i^t = \vec{x}_i^t + K(\vec{x}_{best}^t - \vec{x}_i^t) + F(\vec{x}_{r1}^t - \vec{x}_{r2}^t)$$

Nas estratégias acima, os parâmetro F e K são os fatores de escalonamento do vetor diferença, cujos valores normalmente se encontram no intervalo $(0,1]$. O vetor $\vec{x}_{best}^t$ é o indivíduo com o melhor valor de aptidão no instante t. Os índices $r1$, $r2$, $r3$, $r4$ e $r5$ são valores inteiros sorteados aleatoriamente no intervalo $[1, NP]$, mutuamente exclusivos, e diferentes dos índices i e $best$.

A estratégia que devemos usar em um algoritmo DE vai depender do problema a ser resolvido: uma estratégia que funciona bem para um dado problema pode não funcionar bem quando aplicada a outro problema. Sendo assim, normalmente o usuário testa várias estratégias em um mesmo problema e escolhe a que for mais eficiente e/ou estável. O mesmo é valido para os

parâmetros F e K, cada problema possui um valor ou uma faixa de valor que torna o algoritmo eficiente e/ou estável.

Operador de Cruzamento

Após a fase de mutação (calculo do vetor $\vec{v}_i^t$), um processo de cruzamento é realizado onde, para cada par de vetores $\vec{x}_i^t$ e $\vec{v}_i^t$, um vetor filho $\vec{u}_i^{t+1}$ é gerado. Existem dois algoritmos para se fazer o cruzamento: binomial (**bin**) ou exponencial (**exp**).

- **Cruzamento Binomial**: É equivalente a decidir com certa probabilidade se uma posição j do vetor filho assumirá o valor do seus pai ou do vetor mutação. É definida como:

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^t & \text{se } \left(rand_j[0,1] \leq CR\right) \text{ ou } \left(j = j_{rand}\right) \\ x_{ij}^t & \text{caso contrário} \end{cases} \tag{3.6}$$

  onde CR é a taxa de cruzamento definida pelo usuário no intervalo $[0,1]$, j_{rand} é um número inteiro sorteado aleatoriamente no intervalo $[1, NP]$ para garantir que o vetor filho $\vec{u}_i^{t+1}$ tenha pelo menos uma variável diferente de seu pai $\vec{x}_i^t$, e $rand_j[0,1]$ é uma função que retorna números reais aleatórios dentro do intervalo $[0,1]$ e com distribuição uniforme.

- **Cruzamento Exponencial**: O cruzamento é executado nas variáveis enquanto o número aleatório $rand_j[0,1]$ for menor que a probabilidade de cruzamento CR.

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^t & \text{enquanto } \left(rand_j[0,1] \leq CR\right) \\ x_{ij}^t & \text{caso contrário} \end{cases} \tag{3.7}$$

  onde CR é a taxa de cruzamento definida pelo usuário no intervalo $[0,1]$, e $rand_j[0,1]$ é uma função que retorna números reais aleatórios dentro do intervalo $[0,1]$ e com distribuição uniforme.

No DE, o cruzamento é introduzido para aumentar a diversidade dos indivíduos que sofreram a mutação. A taxa de cruzamento CR representa a probabilidade do vetor filho herdar os valores das variáveis do vetor doador (mutação), devendo ser fornecida pelo usuário. Normalmente, para cada problema temos um valor ótimo para CR, ficando para o usuário descobrir qual é esse valor.

Operador de Seleção

Em uma dada iteração t, o operador de seleção compara o valor de aptidão de cada vetor filho $f(\vec{u}_i^{t+1})$ com o valor de seu vetor pai $f(\vec{x}_i^t)$: se o valor da aptidão do vetor filho for menor ou igual ao do vetor pai (para problemas de minimização), então o vetor filho substitui seu pai na próxima geração. O operador de seleção é representado da seguinte forma:

$$\vec{x}_i^{t+1} = \begin{cases} \vec{u}_i^{t+1} & \text{se } f(\vec{u}_i^{t+1}) \leq f(\vec{x}_i^t) \\ x_{ij}^t & \text{caso contrário} \end{cases} \tag{3.8}$$

Os operadores de mutação, cruzamento e seleção são aplicados iteração após iteração até que um dado critério de parada seja satisfeito. O Algoritmo 6 mostra um esquema de uma algoritmo DE usando a estratégia "DE/rand/1/bin".

No Algoritmo 6, a função $randint(min, max)$ retorna números inteiros aleatórios sorteados dentro do intervalo $[min, max]$ e função $rand_j[0,1]$ retorna números reais aleatórios sorteados dentro do intervalo $[0,1]$. Ambas estão baseadas em uma distribuição de probabilidade uniforme.

3.4.4 Sistemas Imunológicos Artificiais

O Sistema Imunológico Artificial (AIS - *Artificial Immune System*) é um mecanismo computacional composto por metodologias inteligentes inspiradas no sistema imunológico natural (SIN), usando heurística para solução de problemas do mundo real. No organismo humano, o SIN é o sistema que protege o corpo contra a invasão de substâncias estranhas (antígenos) ou organismos patogênicos através de respostas imunológicas.

Quando o corpo é exposto a estes elementos estranhos, o SIN identifica o anticorpo com mais alta afinidade iniciando o processo de proliferação, gerando clones. Alguns destes clones passam pelo processo de mutação em relação à célula original gerando um novo nível de afinidade para o antígeno. Os novos anticorpos com mais alta afinidade passam por um processo de maturação e podem tornar-se células Plasma (responsáveis pela produção de anticorpos e ataque aos antígenos) ou as células de Memória (possuem a habilidade de manter configurações das células com maior adaptabilidade permitindo dirigir repostas imunológicas mais rápidas a antígenos em específico) (CASTRO; ZUBEN et al., 2002).

As características de um sistema imunológico podem ser utilizadas para aprendizado e otimização. O AIS é uma metodologia de pesquisa que usa heurística somente para explorar

Algoritmo 6: Esquema de uma algoritmo DE usando a estratégia DE/rand/1/bin

Entrada: NP - número de indivíduos da população
Entrada: F - fator de escalonamento
Entrada: CR - taxa de cruzamento
Entrada: $Tmax$ - número máximo de iterações
Saída: $\vec{x}_{best}$ - melhor solução encontrada para o problema

```
início
    t ← 0
    Criar de maneira aleatória os NP indivíduos da população inicial
    Calcular a aptidão de todos os indivíduos da população
    enquanto t < Tmax faça
        para i ← 1 até NP faça
            Selecionar aleatoriamente r1 ≠ r2 ≠ r3 ≠ i
            j_rand ← randInt(1,n)
            para j ← 1 até n faça
                se rand_j[0,1] ≤ CR ou j = j_rand então
                    u_ij^{t+1} ← x_{r1j}^t + F(x_{r2j}^t − x_{r3j}^t)
                senão
                    u_ij^{t+1} ← x_ij^t
                fim
            fim
            se f(u_i^{t+1}) ≤ f(x_i^t) então
                x_i^{t+1} ← u_i^{t+1}
            senão
                x_i^{t+1} ← x_i^t
            fim
        fim
        t ← t + 1
    fim
    Retornar a melhor partícula da população (x_best)
fim
```

áreas de interesse no espaço de solução, fornecendo ferramentas para executar pesquisa local e global simultaneamente. Estas ferramentas são baseadas em dois conceitos: hipermutação e edição de receptores. Enquanto a hipermutação executa pequenos passos com objetivo de o anticorpo ter uma afinidade mais alta, determinando ótimos locais, a edição de receptores permite que o anticorpo execute grandes passos, caindo em posições onde a procura por anticorpos de mais alta afinidade possa ser mais promissora.

4 Algoritmos Evolutivos Adaptativos

Este capítulo tem como objetivo fazer a conceituação teórica das técnicas usadas para ajuste dos parâmetros de controle dos algoritmos evolutivos. Outro objetivo é introduzir os conceitos básicos sobre os algoritmos evolutivos adaptativos, especialmente os algoritmos com adaptação das probabilidades de seleção dos operadores evolutivos. Algumas técnicas e critérios encontrados na literatura atual são apresentadas e discutidas.

4.1 Ajuste de Parâmetros em Algoritmos Evolutivos

O desenvolvimento de Algoritmos Evolutivos (EAs) eficientes requer um certo número de atividades: definição de uma codificação apropriada para o problema, escolha da estratégia evolutiva a ser usada, desenvolvimento dos operadores de evolução, seleção e substituição apropriados, definição da função de aptidão e ajuste dos parâmetros que controlam o comportamento do EA. O ajuste dos parâmetros de controle é uma importante tarefa no desenvolvimento de um EA, uma vez que o ajuste ótimo de parâmetros vai variar de problema para problema e um conjunto de parâmetros ruim pode impactar significantemente na performance do algoritmo (WHITACRE, 2007).

Em um GA, por exemplo, devemos ajusta o tamanho da população, as probabilidades de aplicarmos os operadores de mutação e cruzamento, o tamanho do torneio de seleção, o número de gerações máximo (que determina o custo total do algoritmo, visto que avaliar a função objetivo frequentemente é a parte mais cara computacionalmente), e alguns parâmetros adicionais que vão depender dos operadores usados. Os valores destes parâmetros influenciam fortemente na busca por boas soluções. Infelizmente, o ajuste correto do valor de cada parâmetro é uma tarefa que normalmente consome muito tempo e esforço, podendo ser mais trabalhosa que a própria implementação do GA.

O processo de atribuir parâmetros de controle aos EAs é uma das etapas mais importantes para o correto balanceamento das capacidades de *exploração* e *explotação* dos algoritmos. Sa-

bendo que os valores dos parâmetros possuem impacto significativo na performance dos EAs, a atribuição de valores de má qualidade podem, por exemplo, causar convergência prematura do algoritmo ou torná-lo instável. Neste sentido, EAs adaptativos que realizem a tarefa de ajustar os parâmetros de forma automática e dinâmica podem ser muito vantajosos (WHITACRE, 2007). Desta forma, vamos chamar de Algoritmos Evolutivos Adaptativos (AEAs) os EAs que automaticamente ajustam um ou mais parâmetros durante o processo evolutivo.

4.1.1 Classificação das Técnicas

De maneira geral, podemos distinguir duas formas de atribuir valores aos parâmetros de um EA: a) calibração manual dos parâmetros e b) controle automático dos parâmetros. Na calibração manual, o usuário deve encontrar por tentativa e erro bons valores para os parâmetros que, depois de encontrados, permanecem fixos para um determinado problema ou aplicação do EA. Já o controle automático de parâmetros é uma alternativa na qual os parâmetros são ajustados durante a execução do problema, através dos conceitos de adaptação de parâmetros.

Segundo (EIBEN et al., 2007), os métodos para fazer o controle dos valores dos parâmetros de um EA podem ser classificados em uma das três categorias seguintes (ver figura 4.1):

1. **Métodos Determinísticos**: Quando os valores dos parâmetros são alterados por alguma regra determinística. Esta regra altera os valores dos parâmetros de forma predeterminada (especificação do usuário) e sem usar qualquer informação do processo de busca. Existe a possibilidade de termos um controle dinâmico, onde os parâmetros são atualizados segundo uma função determinística definida em função do tempo/iteração.

2. **Métodos Adaptativos**: Quando os valores dos parâmetros são obtidos por um mecanismo de realimentação que monitora o processo de evolução do algoritmo e recompensa ou castiga os valores dos parâmetros de acordo com seus desempenhos. O desempenho de um operador/parâmetro pode ser avaliado em função da melhora ou piora na função de aptidão do problema. Nesta categoria, os parâmetros são adaptados de forma global, ou seja, são adaptados em relação ao EA como um todo.

3. **Métodos Auto Adaptativos**: Os parâmetros a serem adaptados são codificados dentro da representação das soluções do problema, ou seja, cada indivíduo terá o seu próprio conjunto de parâmetros que são adaptados durante o processo evolutivo. A idéia é que cada indivíduo da população tenha o seu próprio conjunto ótimo de parâmetros para guiar a sua busca da forma mais eficiente possível. Nesta categoria, os parâmetros de um EA e a solução para o problema tratado vão coevoluir para um ponto de ótimo.

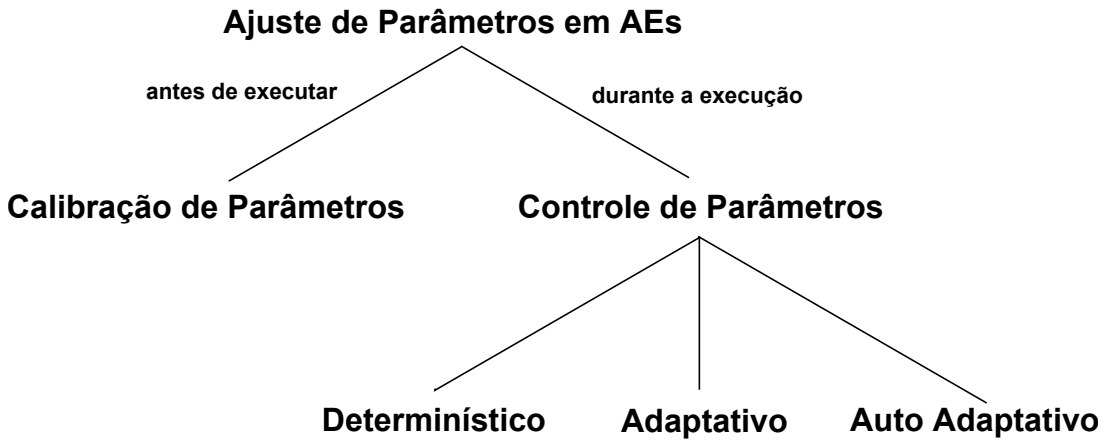


Figura 4.1: Taxonomia global do ajuste de parâmetros em EAs.

4.2 Operadores Evolutivos de Busca e Variação

Os operadores evolutivos realizam a tarefa de amostrar novos pontos do espaço de busca (novos indivíduos) usando como base as informações obtidas de um ou mais indivíduos da população. No início das pesquisas com EAs, os operadores de mutação e cruzamento eram os mais usados, mas atualmente existe um grande número de operadores evolutivos que foram desenvolvidos para resolver diversos tipos de problemas, sendo comum encontrarmos operadores especializados (desenvolvido para um problema específico) e operadores genéricos (desenvolvido para resolver uma grande gama de problemas) (WHITACRE, 2007).

Os operadores evolutivos mais populares são encontrados nos Algoritmos Genéticos (WRIGHT, 1991), na Evolução Diferencial (STORN; PRICE, 1995), na Adaptação da Matriz de Covariância (HANSEN; OSTERMEIER, 2001) e nos Algoritmos de Estimação de Distribuição (MUHLENBEIN; PAASS, 1996). Estes operadores geralmente empregam o uso de múltiplos pais e são extremamente bem sucedidos quando bem implementados e aplicados.

A grande maioria dos operadores evolutivos é classificada como sendo do tipo estocástico. Nestes operadores, sua execução envolve uma variável aleatória, cujo valor é gerado a partir de um sorteio guiado por alguma distribuição de probabilidades (uniforme, *Gaussiana*, etc.) predefinida. Esta característica permite que o operador exiba uma série de comportamentos diferentes e reduz as chances de uma mesma entrada (ou seja, os pais) gerar o mesmo resultado (isto é, a prole). Desta forma, o uso de variáveis estocásticas nos operadores evolutivos ajuda a melhorar a robustez do processo de pesquisa, visto que os operadores determinísticos sempre geram o mesmo resultado para uma mesma entrada (WHITACRE, 2007).

4.2.1 Algoritmos Evolutivos com Múltiplos Operadores

Uma opção para melhorar a robustez de um EA é o uso de vários operadores evolutivos, cada um contendo uma estratégia de busca (*linha de pesquisa*) e, consequentemente, sendo especialista em explorar um determinado tipo de problema. Pesquisas recentes têm indicado que o uso de múltiplos operadores pode ajudar a melhorar o desempenho geral do EA (MICHALEWICZ T. LOGAN; SWAMINATHAN, 1994; BARBOSA; Sá, 2000).

Por outro lado, a adição de múltiplos operadores (ou adição de muitas variáveis estocásticas em um operador de pesquisa) só é aconselhada se não for possível determinar qual é o tipo do problema e a estratégia de busca mais eficaz para resolver esse problema. Como exemplo, para otimizar um problema unimodal com uma superfície de aptidão "suave", a estratégia de busca obtida através de um operador evolutivo baseado no vetor gradiente deverá ser muito melhor do que qualquer outra estratégia de busca obtida com um operador estocástico (WHITACRE, 2007).

Contudo, para resolver problemas complexos e com restrições, é de se esperar que um EA com apenas uma estratégia de busca não seja eficiente em toda a superfície de aptidão (*fitness landscape*). Nesta condição, ter um EA com múltiplos operadores de busca pode gerar um processo de busca mais eficiente. Normalmente, as aplicações de EAs são feitas em problemas complexos, uma vez que os algoritmo tradicionais já resolvem bem e de forma mais rápida os problemas mais simples. Assim, é de se esperar de um EA robusto o uso de múltiplos operadores (WHITACRE, 2007).

4.2.2 Probabilidades dos operadores de busca

No desenvolvimento de um EA, além de termos que selecionar um conjunto de operadores adequado e eficaz, devemos também selecionar as probabilidades com que estes operadores serão selecionados. Assim, para um EA, as probabilidades ou taxas de aplicação dos operadores vão impactar no desempenho do algoritmo como um todo, uma vez que se aplicarmos muito um operador e não os outros podemos fazer o processo convergir rápido (perdendo diversidade na população) de mais ou ficar preso em um ponto de ótimo local.

De acordo com (Sá; BARBOSA, 1998), um bom conjunto de parâmetros iniciais aumenta a eficiência da busca de um EA. Por outro lado, uma má escolha pode levar ao fracasso do algoritmo. Além disso, a qualidade da escolha depende do problema que está sendo abordado.

Como foi dito anteriormente, o ajuste de parâmetros em um EA frequentemente é feito por

tentativa e erro. Contudo, em vez de rodar o EA varias vezes para se descobrir um bom conjunto de probabilidades para os operadores, seria interessante se o próprio algoritmo aprendesse esses parâmetros enquanto resolve o problema. Essa é a ideia principal do algoritmo proposto nesta dissertação e será melhor explicada na próxima seção.

4.3 Adaptação das Probabilidades dos Operadores

Segundo (Sá; BARBOSA, 1998), otimizar os parâmetros a serem utilizados por um EA pode ser um problema mais complexo que o próprio problema original a ser resolvido. Também não fica claro que a escolha inicial dos parâmetros é eficiente durante todo o decorrer do processo evolutivo. Daí surge a idéia de ajustá-los dinamicamente de acordo com o desempenho de cada operador na busca baseando-se em informações colhidas ao longo do processo evolutivo e usando regras de atualização (ou adaptação) para ajustar os parâmetros de controle.

Existem muitas maneiras de adaptar os diferentes parâmetros iniciais de um EA, uma idéia é adaptar as probabilidades dos operadores dando-lhes *crédito* de acordo com o seu *desempenho*. O desempenho, normalmente, é avaliado em função de uma melhorar na aptidão de um indivíduo da população após sofrer a aplicação de um dado operador.

De acordo com (Sá; BARBOSA, 1998), as técnicas propostas na literatura diferem nas particularidades de como essa idéia é implementada. Os pontos principais em questão são:

- Quando e quanto dar de recompensa ao operador?
- Só o próprio operador merece a recompensa?
- Como avaliar a qualidade de um operador?
- Devemos levar em conta o histórico de produtividade?
- Devemos estabelecer probabilidades mínimas de aplicação dos operadores?

A adaptação das probabilidades de aplicação dos operadores evolutivos é um dos aspectos de desenvolvimento dos EAs que tem sido mais exaustivamente estudado na área de EC, visto o difícil e tedioso trabalho de ajustar estes parâmetros, particularmente quando o algoritmo possui mais de dois operadores (WHITACRE, 2007). A decisão de quais operadores usar e com qual frequência é um desafio no desenvolvimento dos EAs, envolvendo o processo de tentativa e erro.

4.3.1 Qualidade do Operador

Um algoritmo adaptativo de alocação de operadores é um algoritmo que iterativamente escolhe um de seus operadores para aplicá-lo a um ambiente externo. O ambiente retorna uma recompensa para o operador usado (que pode ser zero se nenhuma melhora ocorreu) e o algoritmo de alocação usa essa recompensa e o seu estado interno para adaptar as probabilidades com que os operadores são escolhidos. É importante ter em mente que o ambiente considerado aqui é não estacionário, significando que a distribuição de probabilidade, segundo a qual a recompensa é gerada para um determinado operador, muda durante a execução do algoritmo de alocação (THIERENS, 2005).

Dado um EA com um conjunto de K operadores $\mathbf{O} = \{Op_1, \ldots, Op_K\}$, e um vetor de probabilidades $\vec{P}(t) = [P_1(t), \ldots, P_K(t)]$, temos que:

$$\forall t : \ 0 \ \leq \ P_i(t) \ \leq \ 1 \tag{4.1}$$

$$\sum_{i=1}^{K} P_i(t) \ = \ 1 \tag{4.2}$$

onde t é o passo de tempo discreto. O algoritmo de alocação de operadores seleciona um operador para ser executado na proporção dos valores de probabilidade especificados em $\vec{P}(t)$. Quando um operador Op_i é executado no tempo t, uma recompensa $R_i(t)$ é retornada. Cada operador tem uma recompensa associada, que é uma variável aleatória não estacionária. Todas as recompensas são armazenadas no vetor de recompensas $\vec{R}(t) = [R_1(t), \ldots, R_K(t)]$, de forma que, quando um operador Op_i recebe mais de uma recompensa em um mesmo passo de tempo t, estas recompensas são acumuladas em $R_i(t)$.

Além dos vetores $\vec{P}(t)$ e $\vec{R}(t)$, o algoritmo de alocação de operadores possui o vetor de qualidade $\vec{Q}(t) = [Q_1(t), \ldots, Q_K(t)]$ que especifica uma estimativa da qualidade dos operadores em função das recompensas recebidas por eles no passado. O valor inicial para $\vec{Q}$ é $\vec{Q}(0) = 1$. A estimativa de qualidade para um operador indica o quão bom ele tem sido para o estágio atual da busca. Sempre que um operador Opi é executado, a sua estimativa de qualidade $Q_i(t)$ é adaptada através da seguinte expressão:

$$Q_i(t+1) = Q_i(t) + \alpha[R_i(t) - Q_i(t)] \tag{4.3}$$

onde α é a taxa de adaptação: $0 \leq \alpha \leq 1$. O parâmetro α faz o controle da memória do algoritmo de alocação onde, para $\alpha \to 1$ resulta num sistema sem memória e com máxima adaptação, e para $\alpha \to 0$ resulta num sistema com máxima memória mas sem nenhuma adaptação. A

inserção de memória num sistema adaptativo minimiza o efeito de variações bruscas oriundas de um bom desempenho local.

Quando o algoritmo de alocação de operadores é executado por um período de T passos de tempo, o objetivo é maximizar o valor esperado para o total de recompensas recebido pelo algoritmo: *Maximize* $\mathscr{E}[R]$, onde $\mathscr{E}[R] = \sum_{t=1}^{T}\sum_{i=1}^{K} R_i(t)$. Para um EA, isso implica em resolver um dado problema da melhor forma possível, ou seja, aplicando os operadores da forma mais eficiente possível.

A seguir serão mostrados dois métodos para transformar o vetor de qualidade $\vec{Q}(t)$ um um conjunto de probabilidades $\vec{P}(t)$ para alocação de operadores.

4.3.2 Estratégia *Probability Matching*

A ideia básica do algoritmo de alocação de operadores usando a estratégia *Probability Matching* (PM), é calcular a probabilidade de seleção de um operador $P_i(t)$ como sendo proporcional a sua qualidade $Q_i(t)$, de forma que:

$$P_i(t) = \frac{Q_i(t)}{\sum_{j=1}^{K} Q_j(t)} \tag{4.4}$$

Contudo, se usarmos a expressão (4.4) para calcular as probabilidades de aplicação dos operadores poderemos perder alguns operadores durante o processo, ou seja, uma vez que $P_i(t)$ seja igual a 0, o operador não será selecionado mais e consequentemente não receberá recompensas, o que faz a sua qualidade permanecer nula até o final do processo. Essa é uma propriedade indesejável para um EA, visto que operadores ruins em um determinado estágio podem vir a se tornarem bons em estágios futuros do processo de busca. Assim, todos os operadores devem estar ativos, de forma que suas estimativas de qualidade estejam sempre sendo atualizadas (BARBOSA; Sá, 2000).

Para garantir que todos os operadores estejam ativos durante todo o processo de busca, vamos definir um valor mínimo P_{min} para as probabilidades de seleção dos operadores $P_i(t)$, tal que: $0 < P_{min} \leq P_i(t) < 1$. Como resultado, o valor máximo P_{max} que qualquer probabilidade $P_i(t)$ pode assumir é dado por: $P_{max} = 1 - (K-1)P_{min}$, onde K é o número de operadores. Desta forma, a expressão (4.4) é substituída pela seguinte:

$$P_i(t) = P_{min} + (1 - K \cdot P_{min})\frac{Q_i(t)}{\sum_{j=1}^{K} Q_j(t)} \tag{4.5}$$

Podemos ver nas expressões (4.3) e (4.5) que quando um operador não recebe nenhuma

recompensa por muito tempo sua qualidade $Q_i(t)$ converge para 0 e sua probabilidade $P_i(t)$ converge para P_{min}. Podemos ver também que quando apenas um dos operadores receber recompensas por um longo período de tempo sua probabilidade $P_i(t)$ irá convergir para P_{max}.

Finalmente, o algoritmo de alocação de operadores usando a *Probability Matching* pode ser especificado como mostra o Algoritmo 7.

Algoritmo 7: Algoritmo de Alocação de Operadores baseado na estratégia *Probability Matching*

Entrada: $\vec{P}$: vetor de probabilidades
Entrada: $\vec{Q}$: vetor de qualidade
Entrada: K: número de operadores
Entrada: P_{min}: valor mínimo para as probabilidades
Entrada: α: taxa de adaptação
Dados: s: índice do operador selecionado

início
  $t \leftarrow 0$
  para $i \leftarrow 1$ *até* K **faça**
    $P_i(t) \leftarrow \frac{1}{K}$
    $Q_i(t) \leftarrow 1$
  fim
  enquanto *(não "condição de parada")* **faça**
    $s \leftarrow SelecionarOperador(\vec{P})$
    $R_s(t) \leftarrow AplicarOperador(Op_s)$
    $Q_s(t+1) \leftarrow Q_s(t) + \alpha[R_s(t) - Q_s(t)]$
    para $i \leftarrow 1$ *até* K **faça**
      $P_i(t+1) \leftarrow P_{min} + (1 - K \cdot P_{min}) \frac{Q_i(t+1)}{\sum_{j=1}^{K} Q_j(t+1)}$
    fim
    $t \leftarrow t+1$
  fim
fim

No Algoritmo 7, a função *SelecionarOperador* retorna o índice do operador selecionado (sorteio onde a probabilidade de um operador *i* ser sorteado é $P_i(t)$), e a função $AplicarOperador$ retorna uma recompensa para o operador selecionado.

A estratégia PM é uma boa estratégia para adaptar as probabilidades do algoritmo de alocação de operadores, contudo, perde muito em termos da maximização do valor esperado para o total de recompensas recebidas pelo algoritmo. Suponha que nos temos um algoritmo com apenas dois operadores Op_1 e Op_2 com valores constantes de recompensas R_1 e R_2. Da expressão (4.5) podemos encontrar a seguinte relação:

$$\frac{P_1(t) - P_{min}}{P_2(t) - P_{min}} = \frac{R_1}{R_2}$$

Assuma que $R_1 > R_2$. Em uma estratégia ideal, o algoritmo deveria maximizar o uso do operador Op_1 como forma de obter o melhor desempenho possível. Contudo, quanto mais próximo for os valores de R_1 e R_2 pior será o desempenho do algoritmo. Como exemplo, se $R_1 = 10, R_2 = 9$ e $P_{min} = 0,1$, então teremos $P_1 = 0,52$ e $P_2 = 0,48$, o que está longe do desejável que seria $P_1 = 0,9$ e $P_2 = 0,1$.

4.3.3 Estratégia *Adaptive Pursuit*

Os algoritmos *Pursuit* são uma classe de algoritmos de rápida convergência para fazer o aprendizado de autômatos, propostos por Thathachar e Sastry (THATHACHAR; SASTRY, 1985). A estratégia *Adaptive Pursuit* (AP) aplicada ao algoritmo de alocação de operadores busca adaptar o vetor de probabilidades $\vec{P}(t)$ de forma a maximizar o uso do operador Op_i^* que possui a maior qualidade $Q_i(t)$.

Como na estratégia PM, o algoritmo de alocação de operadores com a estratégia AP seleciona um operador i para ser executado de acordo com as probabilidades do vetor $\vec{P}(t)$, e atualiza a correspondente qualidade $Q_i(t)$ do operador com a recompensa recebida. O que difere as duas estratégias é como as probabilidades são adaptadas. Na estratégia AP, após a atualização do vetor qualidade, o operador Op_i^* com maior qualidade $Q_i(t)$ é considerado como sendo o melhor operador para o estágio atual do processo e sua probabilidade $P_i(t)$ é incrementada de acordo com a seguinte expressão:

$$P_i(t+1) = (1-\beta)P_i(t) + \beta \tag{4.6}$$

onde $Op_i^* = argmax_i[Q_i(t+1)]$ e β é a taxa de adaptação da probabilidades. Os outros operadores (não ótimos) terão a suas probabilidades decrementadas da seguinte forma:

$$\forall\, Op_j \neq Op_i^* :\ P_j(t+1) = (1-\beta)P_j(t) \tag{4.7}$$

Podemos ver na expressão (4.6) que se um operador i é o melhor operador durante um certo período de tempo a sua probabilidade de seleção irá convergir para 1, enquanto a outras irão convergir para 0. Como dito na seção anterior, essa característica não é desejada para um ambiente não estacionário.

Para adaptar a estratégia AP para ambientes não estacionários, vamos mudar o esquema de atualização das probabilidades por adicionar um valor mínimo P_{min} permitido. Sendo assim, vamos forçar os valores das probabilidades a ficarem dentro do intervalo $[P_{min}, P_{max}]$, onde $0 < P_{min} < P_{max} < 1$. Chamando o melhor operador atual de $Op_i^* = argmax_i[Q_i(t+1)]$, vamos

adaptar as probabilidades de $\vec{P}(t+1)$ da seguinte forma:

$$P_i(t+1) = P_i(t) + \beta[P_{max} - P_i(t)] \tag{4.8}$$

e

$$\forall\, Op_j \neq Op_i^* :\ P_j(t+1) = P_j(t) + \beta[P_{min} - P_j(t)] \tag{4.9}$$

onde temos a seguinte restrição:

$$P_{max} = 1 - (K-1)P_{min} \tag{4.10}$$

Na restrição (4.10), note que para a relação $P_{max} > P_{min}$ ser verdadeira (desejável para que haja adaptação) temos que respeitar a seguinte restrição: $P_{min} < \frac{1}{K}$. Um valor interessante para a probabilidade mínima é $P_{min} = \frac{1}{2K}$ que resulta no seguinte valor para a probabilidade máxima: $P_{max} = \frac{1}{2} + \frac{1}{2K}$. Uma forma de imaginar a operação do algoritmo AP com esses valores é pensar que o algoritmo deve escolher o melhor operador em metade do tempo, enquanto na outra metade todos os operadores tem igual probabilidade de serem escolhidos.

Finalmente, o algoritmo de alocação de operadores usando a estratégia *Adaptive Pursuit* pode ser especificado como mostra o Algoritmo 8.

No Algoritmo 8, a função *SelecionarOperador* retorna o índice do operador selecionado (sorteio onde a probabilidade de um operador i ser sorteado é $P_i(t)$), a função $AplicarOperador$ retorna uma recompensa para o operador selecionado, e a função $argmax_i$ retorna o índice do operador com o maior valor de qualidade $Q_i(t)$.

Considerando novamente o caso em que temos dois operadores Op_1 e Op_2 sendo aplicados a um ambiente estacionário que retorna sempre as mesmas recompensas, $R_1 = 10$ e $R_2 = 9$, e sendo $P_{min} = 0,1$, podemos concluir que após algumas iterações (o que vai depender do parâmetro β) o algoritmo AP vai convergir para $P_1 = 0,9$ e $P_2 = 0,1$, ou seja, teremos o melhor operador sendo executado em quase 90% do tempo. Também vemos que o operador mais fraco é executado com a probabilidade mínima, resultado que nos interessa para ambientes não estacionários. Contudo, o vetor de qualidade continua a ser adaptado segundo a expressão (4.3), o que permite que qualquer mudança na distribuição de recompensas o sistema vai ser capaz de reagir e modificar suas probabilidades para um outro cenário.

4.3.4 Comparação entre as estratégias

Vamos fazer uma comparação entre as estratégias PM, AP e Fixa (estratégia onde as probabilidades iniciais dos operadores são iguais e permanecem fixas durante todo o processo). Para

Algoritmo 8: Algoritmo de Alocação de Operadores baseado na estratégia *Adaptive Pursuit*

Entrada: $\vec{P}$: vetor de probabilidades
Entrada: $\vec{Q}$: vetor de qualidade
Entrada: K: número de operadores
Entrada: P_{min}: valor mínimo para as probabilidades
Entrada: α: taxa de adaptação das qualidades
Entrada: β: taxa de adaptação das probabilidades
Dados: s: índice do operador selecionado
Dados: m: índice do melhor operador
Dados: P_{max}: valor máximo para as probabilidades

início
  $t \leftarrow 0$
  $P_{max} \leftarrow 1-(K-1)P_{min}$
  para $i \leftarrow 1$ *até* K **faça**
    $P_i(t) \leftarrow \frac{1}{K}$
    $Q_i(t) \leftarrow 1$
  fim
  enquanto *(não "condição de parada")* **faça**
    $s \leftarrow SelecionarOperador(\vec{P})$
    $R_s(t) \leftarrow AplicarOperador(Op_s)$
    $Q_s(t+1) \leftarrow Q_s(t) + \alpha[R_s(t) - Q_s(t)]$
    $m \leftarrow argmax_i[Q_i(t+1)]$
    $P_m(t+1) = P_m(t) + \beta[P_{max} - P_m(t)]$
    para $i \leftarrow 1$ *até* K **faça**
      se $i \neq m$ **então**
        $P_i(t+1) = P_i(t) + \beta[P_{min} - P_i(t)]$
      fim
    fim
    $t \leftarrow t+1$
  fim
fim

tanto, considere um ambiente não estacionário com cinco operadores $Op_i : i = 1,...,5$. Cada operador Op_i recebe uma recompensa R_i uniformemente distribuída dentro de seu respectivo intervalo: $R_1 = \mathscr{U}[0,2]$, $R_2 = \mathscr{U}[1,3]$, $R_3 = \mathscr{U}[2,4]$, $R_4 = \mathscr{U}[3,5]$, e $R_5 = \mathscr{U}[4,6]$ (ver figura 4.2). Após um intervalo fixo de tempo ΔT, estas distribuições são aleatoriamente trocadas entre os operadores, de forma que o atual melhor operador Op_i^* (inicialmente é o Op_5) deve mudar para outro operador. Neste esquema, se pudéssemos sempre usar o melhor operador teríamos um valor esperado para as recompensas de $\mathscr{E}[R^*] = 5$. Contudo, isso não será possível, pois temos que pagar o preço por explorar os efeitos de outros operadores. O objetivo aqui será então adaptar as probabilidades de forma a maximizar o uso do melhor operador de acordo com a dinâmica do ambiente e reagir da forma mais rápida possível quando houver uma mudança

nas distribuições de probabilidade.

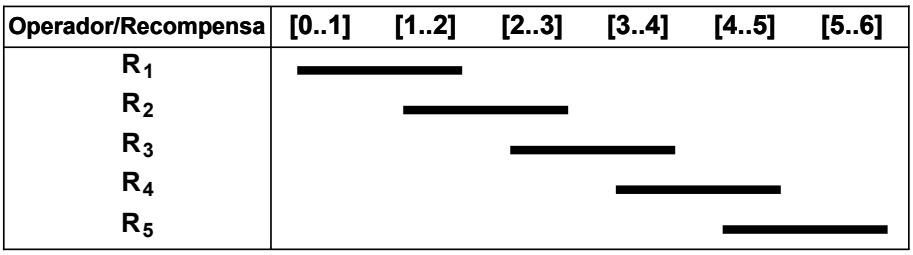


Figura 4.2: Intervalos com distribuição uniforme para cada operador.

Considerando $P_{min} = \frac{1}{2K} = 0,1$, a seguir temos os cálculos para cada estratégia das probabilidades de escolha do operador ótimo e os valores esperados para o total de recompensas recebidas pelo algoritmo:

1. **Algoritmo não Adaptativo**: Estando as probabilidades fixas durante todo o processo de busca, esta estratégia simplesmente seleciona cada um dos operadores com igual probabilidade. Sendo assim, a probabilidade de escolha do operador ótimo Op_i^* é:

$$Prob[Op_s = Op_i^*] \; = \; \frac{1}{K} \; = \; 0,2$$

   O valor esperado para o total de recompensas é dado por:

$$\begin{aligned} \mathscr{E}[R^{fixo}] &= \sum_{i=1}^{K} \mathscr{E}[R_i] Prob[Op_s = Op_i] \\ &= \frac{\sum_{i=1}^{K} \mathscr{E}[R_i]}{K} \\ &= 3,0 \end{aligned}$$

2. **Algoritmo *Probability Matching***: Nesta estratégia, a probabilidade de escolha do operador ótimo Op_i^* é:

$$Prob[Op_s = Op_i^*] \; = \; P_{min} + (1 - K \cdot P_{min}) \frac{\mathscr{E}[R_i^*]}{\sum_{j=1}^{K} \mathscr{E}[R_j]} \; = \; 0,2666\ldots$$

   O valor esperado para o total de recompensas é dado por:

$$\begin{aligned} \mathscr{E}[R^{PM}] &= \sum_{i=1}^{K} \mathscr{E}[R_i] Prob[Op_s = Op_i] \\ &= \sum_{i=1}^{K} i[P_{min} + (1 - K \cdot P_{min}) \frac{\mathscr{E}[R_i]}{\sum_{j=1}^{K} \mathscr{E}[R_j]}] \\ &= 3,333\ldots \end{aligned}$$

3. **Algoritmo *Adaptive Pursuit***:Nesta estratégia, a probabilidade de escolha do operador ótimo Op_i^* é:

$$Prob[Op_s = Op_i^*] \;=\; 1-(K-1)\cdot P_{min} \;=\; 0,6$$

O valor esperado para o total de recompensas, sendo Op_i^* o operador ótimo, é dado por:

$$\begin{aligned}\mathscr{E}[R^{AP}] &= \sum_{j=1}^{K} \mathscr{E}[R_j] Prob[Op_s = Op_j] \\ &= P_{max}\,\mathscr{E}[R_i^*] + P_{min} \sum_{j=1, j\neq i}^{K} \mathscr{E}[R_j] \\ &= 4,0\end{aligned}$$

Podemos concluir dos valores mostrados acima que a estratégia AP é a melhor. Se desconsiderarmos o tempo de adaptação das probabilidades, a estratégia AP vai aplicar o melhor operador a cada cenário do ambiente em quase 60% do tempo, contra 20% para a estratégia sem adaptação e quase 27% para a estratégia PM. Podemos ver também que o valor esperado para a recompensa da estratégia AP é 20% maior que a estratégia PM e 33% maior que a estratégia das probabilidades fixa.

4.3.5 Avaliação do Desempenho de um Operador

Uma vez definida a estratégia de adaptação das probabilidades, devemos definir como vamos avaliar o desempenho de um operador evolutivo: cada operador, após a sua execução, deve receber uma recompensa proporcional ao seu desempenho. Uma recompensa R_i é o resultado da interação do operador Op_i com o ambiente, que no caso do um EA significa uma medida da performance ou produtividade do operador. Para um EA com adaptação de probabilidades, devemos usar as recompensas obtidas em cada estágio do processo evolutivo para incrementar a probabilidade dos operadores que são mais produtivos em detrimento dos que não estão tendo uma boa performance (Sá; BARBOSA, 1998).

Em (WHITACRE, 2007), antes do processo de adaptação das probabilidades, cada operador Op_i é executado (aplicação do operador i em um indivíduo da população para gerar um novo indivíduo) M_i vezes, de forma que cada execução gera uma recompensa $r_i(j)$ onde $j = 1,\ldots,M_i$. A cada τ iterações do ciclo evolutivo, o operador Op_i tem a sua probabilidade de seleção atualizada (adaptada) em função de sua recompensa média $R_i(t)$ recebida no período, tal que:

$$R_i(t) = \frac{1}{M_i} \sum_{j=1}^{M_i} r_i(j) \tag{4.11}$$

onde cada incremento de tempo t é relativo a um ciclo de adaptação, que por sua vez representa τ ciclos evolutivos (número de gerações do EA entre uma adaptação e outra).

Na expressão (4.11), a recompensa $R_i(t)$ representa múltiplas interações entre o sistema adaptativo e seus ambiente. Essa definição é interessante pois aumenta a possibilidade de um operador com baixa probabilidade ser usado e normaliza as recompensas para critério de comparação de desempenho.

Considerando que cada operador gera apenas um filho (novo indivíduo), as recompensas $r_i(j)$ são definidas em função da variação Δf do valor da função de aptidão (função *fitness*) de um indivíduo após a aplicação do operador Op_i. Whitacre (WHITACRE, 2007) apresenta algumas possibilidades para o calculo de $r_i(j)$, o que pode ser feito seguindo algum critério de interpretação da variação Δf gerada pelo operador. Sendo F_{filho} o valor da função de aptidão do novo indivíduo gerado pelo operador Op_i, e considerando um problema de maximização, a recompensa $r_i(j)$ gerada na j-ésima aplicação de Op_i pode ser calcula das seguintes formas:

- **Critério I1**: A recompensa é calculada com referência local (em relação ao valor de aptidão do indivíduo pai F_{pai}) e o crédito é fixo (independe do ganho produzido):

$$r_i(j) = \begin{cases} 1 & \text{se } F_{filho} > F_{pai} \\ 0 & \text{caso contrário} \end{cases} \tag{4.12}$$

- **Critério I2**: A recompensa é calculada com referência local (em relação a F_{pai}) e proporcional ao ganho/perda de aptidão observado:

$$r_i(j) = F_{filho} - F_{pai} \tag{4.13}$$

- **Critério I3**: A recompensa é calculada com referência local (em relação a F_{pai}) e proporcional ao ganho de aptidão observado:

$$r_i(j) = Max(0,\, F_{filho} - F_{pai}) \tag{4.14}$$

  Diferentemente do critério I2, este critério não penaliza o operador (crédito zero) quando o indivíduo filho possuir uma aptidão inferior ao seu pai.

- **Critério I4**: A recompensa é calculada com referência global (em relação a média dos valores de aptidão dos indivíduos da população $F_{\text{média}}$) e o crédito é fixo (independe do ganho produzido):

$$r_i(j) = \begin{cases} 1 & \text{se } F_{filho} > F_{\text{média}} \\ 0 & \text{caso contrário} \end{cases} \tag{4.15}$$

- **Critério I5**: A recompensa é calculada com referência global e o crédito é proporcional ao ganho/perda relativo observado:

$$r_i(j) = \frac{(F_{filho} - F_{melhor})}{(F_{melhor} - F_{\text{média}})} \tag{4.16}$$

onde $F_{\text{média}}$ é a média dos valores de aptidão dos indivíduos da população; e F_{melhor} é o valor de aptidão do melhor indivíduo da população. Neste critério penalizamos um operador (crédito negativo) quando ela não gera um filho melhor que o melhor indivíduo da população.

- **Critério I6**: A recompensa é calculada com referência global (em relação a média dos valores de aptidão dos indivíduos da população $F_{\text{média}}$)) e o crédito é proporcional ao ganho observado:

$$r_i(j) = Max(0,\ F_{filho} - F_{melhor}) \tag{4.17}$$

- **Critério I7**: A recompensa é calculada com referência global (em relação ao indivíduo cuja aptidão é maior que 90% das aptidões de todos os indivíduos da população $F_{90\%}$) e o crédito é proporcional ao ganho observado:

$$r_i(j) = Max(0,\ F_{filho} - F_{90\%}) \tag{4.18}$$

- **Critério I8**: A recompensa é calculada através de um processo de classificação do indivíduo filho gerado em relação a todos os indivíduos da população:

$$r_i(j) = \sum_{p=1}^{NP} \phi(filho, p) \tag{4.19}$$

$$\phi(filho, p) = \begin{cases} 1 & \text{se } F_{filho} > F_j \\ 0 & \text{senão} \end{cases}$$

onde NP é o número de indivíduos da população, e F_j é o valor de aptidão do p-ésimo indivíduo da população.

Em todos os critérios mostrados acima podemos considerar o custo do operador para gerar o filho já que alguns operadores requerem mais de uma avaliação da função de aptidão para gerá-lo: geralmente, a avaliação da função de aptidão é o processamento mais caro de um EA. Sendo assim, não importa como a produtividade é medida, parece natural dividi-la pelo custo da produção.

Quando um operador gera mais de um filho, consideramos que esses filhos foram gerados em eventos distintos para um mesmo operador, de forma que para cada filho teremos um crédito

associado.

Em (BARBOSA; Sá, 2000), temos o conceito de *Ancestralidade*: para cada recompensa $R_i(t)$ podemos distribuí-la pelos operadores que geraram os pais do indivíduo filho (e possivelmente também os avós e bisavós) considerando que a capacidade de gerar pais que geram bons filhos também merece ganho de produtividade. Esse recurso implica em uma estrutura de dados adicional para armazenar a ancestralidade (árvore com os operadores) além da inserção de dois novos parâmetros: *levels*, o número de ancestrais que será levado em conta e *decay* que corresponde a porcentagem de crédito que será atribuída a cada nível de ancestralidade.

5 *Algoritmo Proposto*

Este capítulo descreve o algoritmo evolutivo proposto que se baseia em técnicas adaptativas para ajustar de forma automática os valores das probabilidades de seleção dos operadores evolutivos. A ideia inicial é, a partir de um conjunto de nove operadores, descobrir quais operadores são mais eficientes na resolução de um determinado problema de otimização restrita, e em qual momento cada operador deve ser utilizado. O objetivo principal desta proposta é melhorar o desempenho do algoritmo quando comparado com algoritmos evolutivos sem adaptação de probabilidades (com parâmetros fixos). A metodologia adotada para fazer a manipulação das restrições é apresentada, bem como os operadores evolutivos e as ferramentas de adaptação e interpretação da produtividade dos operadores.

5.1 Manipulação das Restrições: Método ε-*constrained*

O Método de manipulação de restrições ε-*constrained* foi inicialmente proposto por Takahama e Sakai (TAKAHAMA; SAKAI, 2006) e sua eficiência foi comprovada através da aplicação em vários problemas teste (*benchmarks*) conhecidos.

Como foi dito na seção 2.5.4, este método trabalha com o relaxamento das restrições e otimiza o problema adotando uma estratégia de comparação lexográfica onde a satisfação das restrições é mais importante que a otimização da função objetivo. Esta abordagem permite transformar uma algoritmo de otimização de problemas irrestritos em um algoritmo de otimização de problemas restritos.

5.1.1 Violação de Restrição e Comparação ε-*Level*

Considerando as formulações e definições da seção 2.2, das expressões (2.3) e (2.4) temos que um problema (P) de minimização é composto de uma função objetivo $f(\vec{x})$ e m restrições: q restrições de desigualdade $g_j(\vec{x}) \leq 0$ e $m-q$ restrições de igualdade $h_j(\vec{x}) = 0$. O vetor $\vec{x} = [x_1, \ldots, x_n]$ é um vetor n-dimensional com as variáveis de controle de (P) sujeitas as restrições

de limite inferior x_i^L e limite superior x_i^U.

No método ε-*constrained*, $\phi(\vec{x})$ é definido com sendo a função de violação de restrição (também chamada de função de penalidade), cujo valor é calculado através da soma de todas as restrições:

$$\phi(\vec{x}) = \sum_{j=1}^{q} \left\| max\{0,\ g_j(\vec{x})\} \right\|^p + \sum_{j=q+1}^{m} \left\| h_j(\vec{x}) \right\|^p \tag{5.1}$$

onde p é um número real positivo. Neste trabalho, usaremos $p = 1$ em todas as simulações.

Dado o conjunto de valores $(f(\vec{x}),\ \phi(\vec{x}))$ para um determinado ponto de busca $\vec{x}$, o método de comparação lexográfica chamado de ε-*level comparison* é definido de forma que $\phi(\vec{x})$ precede $f(\vec{x})$, pois considera-se a factibilidade do ponto $\vec{x}$ mais importante que a minimização da função objetivo $f(\vec{x})$. Na expressão (5.1), podemos ver que quando a violação de restrição de um ponto for maior do que 0 significa que este ponto é infactível e não tem valor como solução de (P).

Considere f_i e ϕ_i como o valor da função objetivo e a violação de restrição para um ponto $\vec{x}_i$ $(i = 1, 2)$, respectivamente. Então, para qualquer ε satisfazendo $\varepsilon \geq 0$, os comparadores ε-*level* $<_\varepsilon$ e $\leq_\varepsilon$ entre $(f_1,\ \phi_1)$ e $(f_2,\ \phi_2)$ são definidos da seguinte forma:

$$(f_1,\ \phi_1) <_\varepsilon (f_2,\ \phi_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{se } \phi_1, \phi_2 \leq \varepsilon \\ f_1 < f_2, & \text{se } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{caso contrário} \end{cases} \tag{5.2}$$

$$(f_1,\ \phi_1) \leq_\varepsilon (f_2,\ \phi_2) \Leftrightarrow \begin{cases} f_1 \leq f_2, & \text{se } \phi_1, \phi_2 \leq \varepsilon \\ f_1 \leq f_2, & \text{se } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{caso contrário} \end{cases} \tag{5.3}$$

onde o parâmetro ε é um valor real positivo e define o nível do relaxamento das restrições. O objetivo deste parâmetro é relaxar as restrições num estágio inicial do processo e conduzir os ponto de busca em direção a região de factibilidade $\mathbb{F}$.

No caso de $\varepsilon = \infty$, os comparadores $<_\infty$ e $\leq_\infty$ são equivalentes aos comparadores ordinais $<$ e $\leq$ entre os valores das funções objetivo. Também, no caso de $\varepsilon = 0$, os comparadores $<_0$ e $\leq_0$ são equivalentes a uma ordem lexográfica na qual o valor da violação de restrição $\phi(\vec{x})$ precede o valor da função objetivo $f(\vec{x})$.

5.1.2 As Propriedades do Método *ε-constrained*

O método *ε-constrained* converte um problema de otimização restrita em um problema irrestrito através do uso dos comparadores *ε-level* nos métodos de busca. Um problema de otimização resolvido por este método, isto é, um problema no qual os comparadores ordinais são substituídos pelos *ε-level comparisons*, $(P_{\leq_\varepsilon})$ é definido da seguinte forma:

$$(P_{\leq_\varepsilon}) \ \ minimize_{\leq_\varepsilon} \ \ f(\vec{x}) \tag{5.4}$$

onde $minimize_{\leq_\varepsilon}$ significa a minimização baseada no *ε-level comparison* $\leq_\varepsilon$. Também, um problema (P^ε) é definido de forma que as restrições de (P), isto é, $\phi(\vec{x}) = 0$, são relaxadas e substituídas com $\phi(\vec{x}) \leq \varepsilon$:

$$\begin{aligned} (P^\varepsilon) \ \ & minimize_{\leq_\varepsilon} \ \ f(\vec{x}) \\ & \text{sujeito a} \ \ \phi(\vec{x}) \leq \varepsilon \end{aligned} \tag{5.5}$$

É interessante notar que (P^0) é equivalente a (P).

Para os três tipos de problemas, (P^ε), $(P_{\leq_\varepsilon})$ e (P), os seguintes teoremas são dados (ver demonstrações em (TAKAHAMA; SAKAI, 2009):

- **Teorema 1**: Se existe uma solução ótima para (P^0), então qualquer solução de $(P_{\leq_\varepsilon})$ é uma solução ótima de (P^ε)
- **Teorema 2**: Se existe uma solução ótima para (P), então qualquer solução de $(P_{\leq_0})$ é uma solução ótima de (P)
- **Teorema 3**: Deixe $\{\varepsilon_n\}$ ser um sequência estritamente decrescente de valores reais positivos convergindo para 0. Deixe $f(\vec{x})$ e $\phi(\vec{x})$ serem funções contínuas de $\vec{x}$. Assuma que uma solução ótima $\vec{x}^*$ de (P^0) existe e uma solução ótima $\vec{x}_n^*$ de $(P_{\leq_{\varepsilon_n}})$ existe para qualquer ε_n. Então, qualquer ponto de acumulação para a sequência $\{\vec{x}_n^*\}$ é uma solução ótima para (P^0).

Os teoremas 1 e 2 mostram que um problema de otimização restrita pode ser transformado em um problema de otimização irrestrita equivalente pelo uso do comparador *ε-level comparison*, de forma que se este método for incorporado em algum método de otimização irrestrita problemas de otimização restrita poderão ser resolvidos. Assim, podemos dizer que o método *ε-constrained* é um *método de transformação de algoritmos* porque pode converter um algoritmo de otimização irrestrita em um algoritmo de otimização restrita. O teorema 3 mostra

que, no método ε-*constrained*, uma solução ótima de (P) pode ser obtida convergindo ε para 0, estratégia semelhante a aumentar o coeficiente de penalidade para o infinito nos métodos da função de penalização (ver item 2.5.2).

5.1.3 Controle do ε-*Level*

Usualmente, o ε-*level* não precisa ser controlado, pois muitos problemas de otimização restritos podem ser resolvidos usando-se apenas a ordem lexicográfica dos pontos de busca, o que significa manter o parâmetro ε constante e igual a 0. Contudo, para problemas com restrições de igualdade, o ε-*level* deve ser controlado convenientemente para obtermos soluções de alta qualidade.

De acordo com a metodologia apresentada em (TAKAHAMA; SAKAI, 2009), uma forma simples e eficiente de controlar o parâmetro ε é usando uma função exponencial decrescente $\varepsilon(t)$, como mostra a equação 5.6. O valor inicial $\varepsilon(0)$ é o valor da função de violação de restrição do θ-ésimo indivíduo da população inicial de em um EA. O parâmetro ε é atualizado até o número de iterações t do algoritmo ser maior ou igual ao parâmetro T_c. Após o número de iterações exceder T_c, o parâmetro ε é ajustado para 0 como forma de obter soluções com valores mínimos de violação de restrição.

$$\begin{aligned} \varepsilon(t) &= \begin{cases} \varepsilon(0)(1-\frac{t}{T_c})^{cp}, & 0 < t < T_c \\ 0, & t \geq T_c \end{cases} \\ \varepsilon(0) &= \phi(\vec{x}_\theta) \end{aligned} \tag{5.6}$$

onde $\vec{x}_\theta$ é o θ-ésimo indivíduo de uma população com NP indivíduos (ordenada do melhor para o pior), onde $\theta = 0, 2NP$; T_c é o parâmetro que controla o período de convergência de ε; e cp é o parâmetros que controla a velocidade de redução do relaxamento das restrições.

5.2 Operadores Evolutivos

Nesta seção temos a descrição dos nove operadores evolutivos (ou operadores de busca) em codificação real usados no desenvolvimento do algoritmo proposto. Os operadores foram escolhidos de forma empírica, objetivando-se formar um grupo onde cada operador fosse especializado em resolver um determinado tipo problema.

Normalmente os operadores evolutivos são classificados em dois grupos distintos: *operadores de mutação* (criam um novo indivíduo partindo de um único pai) e *operadores de*

recombinação (geram novos indivíduos através da recombinação de características de dois ou mais indivíduos pais). Seguindo essa classificação, foram escolhidos nove operadores, sendo seis de recombinação (BLX, WHX, ELX, UNX, DER e DEB) e três de mutação (NUM, UNM e BDM). Os operadores de recombinação foram modificados de forma a gerar um único filho.

Um décimo operador é apresentado, o operador CUT (Operador de Reparação), usado para recolocar pontos de busca dentro do espaço de busca, caso tenham saído. Este operador não é um operador evolutivo, apenas uma ferramenta auxiliar: executado sempre ao final da execução de qualquer operador de recombinação.

Nos operadores evolutivos apresentados a seguir, cada operador pode receber um ou mais indivíduos pais e gerar apenas um indivíduo filho de forma que o j-ésimo pai é definido como $\vec{x}_j^{\,p} = [x_{j,1}^p, \ldots, x_{j,n}^p]$ e o filho é definido como $\vec{x}^{\,f} = [x_1^f, \ldots, x_n^f]$. Para todos os operadores evolutivos, o indivíduo pai $\vec{x}_1^{\,p}$ será sempre o melhor de todos os pais selecionados.

Números aleatórios sorteados sobre uma distribuição uniforme de probabilidades e dentro de um intervalo fechado $[x^{min}, x^{max}]$ são simbolizados como $U_I(x^{min}, x^{max})$ quando forem do tipo inteiro e $U_R[x^{min}, x^{max}]$ quando forem do tipo real.

5.2.1 UNM - *Uniform Mutation*

O operador de mutação UNM, também chamado de *Single Point Random Mutation*, é descrito em (MICHALEWICZ T. LOGAN; SWAMINATHAN, 1994). Este operador requer um pai $\vec{x}^{\,p}$ e gera um filho $\vec{x}^{\,f}$, onde a i-ésima variável do indivíduo filho é definida por:

$$x_i^f = \begin{cases} U_R[x_i^L, x_i^U], & \text{se } i = k \\ x_i^p, & \text{caso contrário} \end{cases} \tag{5.7}$$

onde x_i^U é o limite superior e x_i^L é o limite inferior, e k é o ponto de mutação escolhido aleatoriamente: $k = U_I(1,n)$. O valor k é sorteado uma única vez, de forma que tenhamos um único ponto de mutação.

Nas primeiras fases do processo evolutivo, este operador permiti que os pontos de busca se movam por todo o espaço de busca. Já nas fases posteriores, o operador permite que um determinado ponto saía de um mínimo local e busque uma solução melhor.

5.2.2 BDM - *Boundary Mutation*

Construído para otimizar problemas onde a solução ótima está próxima da borda do espaço de busca, o operador de mutação BDM é descrito em (MICHALEWICZ T. LOGAN; SWAMINATHAN, 1994). Este operador requer um pai $\vec{x}^{\,p}$ e gera um filho $\vec{x}^{\,f}$, onde a i-ésima variável do indivíduo filho é definida por:

$$x_i^f = \begin{cases} x_i^L, & \text{se } i = k \text{ e } r = 0 \\ x_i^U, & \text{se } i = k \text{ e } r = 1 \\ x_i^p, & \text{caso contrário} \end{cases} \tag{5.8}$$

onde $k = U_I(1,n)$ é o ponto de mutação e $r = U_I(0,1)$ é uma variável aleatória binária. Os valores k e r são sorteados uma única vez. Este operador é uma variação do UNM onde a componente x_k^f pode assumir o valor x_k^L ou x_k^U com igual probabilidade (bordas do espaço de busca).

5.2.3 NUM - *Non-uniform Mutation*

O operador de mutação NUM, descrito em (MICHALEWICZ T. LOGAN; SWAMINATHAN, 1994), é responsável por executar um ajuste fino nas soluções encontrada para o problema. Este operador requer um pai $\vec{x}^{\,p}$ e gera um filho $\vec{x}^{\,f}$, onde a i-ésima variável do indivíduo filho é definida por:

$$x_i^f = \begin{cases} x_i^p - \Delta(t, x_i^p - x_i^L), & \text{se } i = k \text{ e } r = 0 \\ x_i^p + \Delta(t, x_i^U - x_i^p), & \text{se } i = k \text{ e } r = 1 \\ x_i^p, & \text{caso contrário} \end{cases} \tag{5.9}$$

$$\Delta(t,y) = w \cdot y \cdot \left(1 - \frac{t}{T_{max}}\right)^b$$

onde $k = U_I(1,n)$ é o ponto de mutação, $r = U_I(0,1)$ é uma variável aleatória binária, t é a geração atual da população, T_{max} é o número máximo de gerações, w é um número aleatório tal que $w = U_R[0,1]$, e b é o parâmetro que determina o grau da não uniformidade da mutação (neste trabalho, adotamos $b = 6$ para todas as simulações). Os valores k, r e w são sorteados uma única vez.

No início do processo evolutivo, o operador NUM resulta numa busca uniforme pelo espaço de busca (o valor de t é pequeno), ou seja, numa busca global. Nas fases seguintes da busca, este operador reduz a amplitude das variações e gera uma busca local, fazendo um ajuste fino das soluções encontradas.

5.2.4 BLX - *Blend Crossover*

O operador de recombinação BLX-α, descrito por (WHITACRE, 2007), requer dois pais $\vec{x}_1^{\,p}$ e $\vec{x}_2^{\,p}$ e gera um único filho $\vec{x}^{\,f}$, tal que a i-ésima variável do indivíduo filho é definida por:

$$\begin{aligned} x_i^f &= \begin{cases} U_R[x_{1,i}^p - \alpha \cdot \Delta x_i,\ x_{2,i}^p + \alpha \cdot \Delta x_i], & \text{se } x_{1,i}^p < x_{2,i}^p \\ U_R[x_{2,i}^p - \alpha \cdot \Delta x_i,\ x_{1,i}^p + \alpha \cdot \Delta x_i], & \text{caso contrário} \end{cases} \\ \Delta x_i &= \left| x_{1,i}^p - x_{2,i}^p \right| \end{aligned} \tag{5.10}$$

onde o parâmetro α é um valor real positivo e deve ser ajustado pelo usuário. Neste trabalho, $\alpha = 0,2$ para todas as simulações.

5.2.5 WHX - *Wright's Heuristic Crossover*

O operador de recombinação WHX, descrito por (MICHALEWICZ T. LOGAN; SWAMINATHAN, 1994), requer dois pais $\vec{x}_1^{\,p}$ e $\vec{x}_2^{\,p}$ e gera um único filho $\vec{x}^{\,f}$, tal que:

$$\vec{x}^{\,f} = w \cdot (\vec{x}_2^{\,p} - \vec{x}_1^{\,p}) + \vec{x}_2^{\,p} \tag{5.11}$$

onde w é um valor real aleatório definido por $w = U_R[0,1]$, e o pai $\vec{x}_2^{\,p}$ possui um valor de aptidão melhor que $\vec{x}_1^{\,p}$, ou seja, $f(\vec{x}_2^{\,p}) \geq f(\vec{x}_1^{\,p})$ para problemas de maximização e $f(\vec{x}_2^{\,p}) \leq f(\vec{x}_1^{\,p})$ para problemas de minimização.

O objetivo deste operador é fazer uma interpolação da direção mais promissora. Ele também contribui para um ajuste fino da solução final encontrada.

5.2.6 ELX - *Extended Line Crossover*

O operador de recombinação ELX, descrito por (WHITACRE, 2007), requer dois pais $\vec{x}_1^{\,p}$ e $\vec{x}_2^{\,p}$ e gera um único filho $\vec{x}^{\,f}$, tal que:

$$\vec{x}^{\,f} = w \cdot (\vec{x}_2^{\,p} - \vec{x}_1^{\,p}) + \vec{x}_1^{\,p} \tag{5.12}$$

onde w é um valor real aleatório definido por $w = U_R[-0,25,\ 1,25]$.

Sendo um operador geométrico, este operador gera um ponto no seguimento reta que liga os dois pais e permite uma extrapolação de 25% para cada lado da reta.

5.2.7 UNX - *Uniform Crossover*

O operador de recombinação UNX, descrito por (WHITACRE, 2007), requer dois pais $\vec{x}_1^{\,p}$ e $\vec{x}_2^{\,p}$ e gera um único filho $\vec{x}^{\,f}$, tal que a i-ésima variável do indivíduo filho é definida por:

$$x_i^f = \begin{cases} x_{1,i}^p, & \text{se } r_i = 0 \\ x_{2,i}^p, & \text{se } r_i = 1 \end{cases} \qquad (5.13)$$
$$r_i = U_I(0,1)$$

onde o valor r_i é uma variável aleatória binária sorteada para cada variável i.

5.2.8 DER - Operador *DE/rand/1/bin*

O operador DER foi originalmente descrito por (STORN; PRICE, 1995) como sendo uma estratégia evolutiva para o algoritmo de Evolução Diferencial (DE - *Differential Evolution*). O operador requer quatro pais $\vec{x}_1^{\,p}, \vec{x}_2^{\,p}, \vec{x}_3^{\,p}$ e $\vec{x}_4^{\,p}$ mutuamente exclusivos. Como resultado temos um único filho $\vec{x}^{\,f}$ onde a sua i-ésima variável é definida por:

$$x_i^f = \begin{cases} x_{2,i}^p + F \cdot (x_{3,i}^p - x_{4,i}^p), & \text{se } w_i < CR \\ x_{1,i}^p, & \text{caso contrário} \end{cases} \qquad (5.14)$$
$$w_i = U_R[0,1]$$

onde o valor w_i é uma variável aleatória real sorteada para cada variável i, o parâmetro F é o fator de escala, e o parâmetro CR é a taxa de recombinação adotada. Neste trabalho, $F = 0,7$ e $CR = 0,9$ para todas as simulações.

5.2.9 DEB - Operador *DE/current-to-best/1/bin*

O operador DEB é uma variação do operador DER usando informações do melhor indivíduo da população $\vec{x}^{\,best}$ e do indivíduo base a ser mutado $\vec{x}^{\,base}$. Além dos pais $\vec{x}^{\,best}$ e $\vec{x}^{\,base}$, este operador requer mais dois pais $\vec{x}_1^{\,p}$ e $\vec{x}_2^{\,p}$. Todos os pais devem ser mutuamente exclusivos. Como resultado temos um único filho $\vec{x}^{\,f}$ onde a sua i-ésima variável é definida por:

$$x_i^f = \begin{cases} x_i^{base} + K \cdot (x_i^{best} - x_i^{base}) + F \cdot (x_{1,i}^p - x_{2,i}^p), & \text{se } w_i < CR \\ x_i^{base}, & \text{caso contrário} \end{cases} \qquad (5.15)$$
$$w_i = U_R[0,1]$$

onde o valor w_i é uma variável aleatória real sorteada para cada variável i, os parâmetros F e K são os fatores de escala, e o parâmetro CR é a taxa de recombinação adotada. Neste trabalho, $F = 0,7$, $K = 0,1$ e $CR = 0,9$ para todas as simulações.

5.2.10 CUT - Operador de Reparação

Não sendo um operador evolutivo, este operador é aplicado sobre todos os indivíduos filhos resultados de algum operador de recombinação, pois estes operadores podem gerar indivíduos fora do espaço de busca $\mathbb{X}$. Assim, o operador CUT possui a tarefa de recolocar os indivíduos dentro de $\mathbb{X}$.

$$x_i^f = \begin{cases} x_i^L, & \text{se } (x_i^f < x_i^L) \\ x_i^U, & \text{se } (x_i^f > x_i^U) \\ x_i^f, & \text{caso contrário} \end{cases} \tag{5.16}$$

onde x_i^U é o limite superior e x_i^L é o limite inferior para a variável x_i^f.

5.3 Estratégias para Interpretação da Produtividade dos Operadores

Neste trabalho iremos propor três estratégias de interpretação da produtividade dos operadores: referência local, referência global e posicionamento relativo aos indivíduos da população (*ranking*). Estas três estratégia estão baseadas no uso do comparador ε-*level* mostrado na secção 5.1.1.

Para todas as estratégias, a ideia é recompensar um operador quando o mesmo gera um indivíduo filho melhor que o seu pai, onde a recompensa deve condizer com o ganho de aptidão gerado. Contudo, como estamos propondo um algoritmo para resolver problemas de otimização restrita usando o método ε-*constrained*, os critérios mostrados na seção 5.3 não podem ser usados, uma vez que a aptidão de um indivíduo é definida em função de uma ordem lexográfica (a função $\phi(\vec{x})$ vem antes da função objetivo $f(\vec{x})$).

5.3.1 Estratégia I_{local}

Esta estratégia é uma adaptação da estratégia $I3$ (WHITACRE, 2007), onde a recompensa é calculada com referência local e o crédito é proporcional ao ganho de aptidão gerado. Contudo, se o indivíduo estiver dentro da região de relaxamento definida pelo ε-*level*, o crédito é gerado em função do valor de $\phi(\vec{x})$, caso contrário, o crédito é gerado em função do valor de $f(\vec{x})$.

Dessa forma, considerando um problema (P) de minimização, se aplicarmos o operador Op_i sobre o indivíduo pai $\vec{x}_j^{\,p}$, teremos um indivíduo filho $\vec{x}_j^{\,f}$ e uma recompensa R_i dada por:

$$R_i = \begin{cases} 0, & \text{se } \vec{x}_j^{\,p} \leq_\varepsilon \vec{x}_j^{\,f} \\ \frac{\phi(\vec{x}_j^{\,p}) - \phi(\vec{x}_j^{\,f})}{\phi(\vec{x}_j^{\,p})}, & \text{se } \phi(\vec{x}_j^{\,p}) > \varepsilon \\ \frac{f(\vec{x}_j^{\,p}) - f(\vec{x}_j^{\,f})}{\|f(\vec{x}_j^{\,p})\|}, & \text{caso contrário} \end{cases} \tag{5.17}$$

onde $\leq_\varepsilon$ é o comparador ε-*level* definido em (5.3).

5.3.2 Estratégia I_{global}

Esta estratégia é uma adaptação da estratégia $I7$ (WHITACRE, 2007), onde a recompensa é calculada com referência global e o crédito é proporcional ao ganho de aptidão gerado. De maneira semelhante a estratégia I_{local}, a recompensa a ser usada vai depender do valor da função $\phi(\vec{x})$. Considerando um problema (P) de minimização, se aplicarmos o operador Op_i sobre o indivíduo pai $\vec{x}_j^{\,p}$, teremos um indivíduo filho $\vec{x}_j^{\,f}$ e uma recompensa R_i dada por:

$$R_i = \begin{cases} 0, & \text{se } \vec{x}_j^{\,p} \leq_\varepsilon \vec{x}_j^{\,f} \\ \frac{\|\phi(\vec{x}_{med}) - \phi(\vec{x}_j^{\,f})\|}{\phi(\vec{x}_{med})}, & \text{se } \phi(\vec{x}_{med}) > \varepsilon \\ \frac{\|f(\vec{x}_{med}) - f(\vec{x}_j^{\,f})\|}{\|f(\vec{x}_{med})\|}, & \text{caso contrário} \end{cases} \tag{5.18}$$

onde $\leq_\varepsilon$ é o comparador ε-*level* definido em (5.3), e $\vec{x}_{med}$ é o indivíduo com valor de aptidão mediano, ou seja, indivíduo que é melhor que 50% dos indivíduos da população.

5.3.3 Estratégia I_{rank}

Esta estratégia é uma adaptação da estratégia $I8$ (WHITACRE, 2007), onde a recompensa é calculada em função do posicionamento do indivíduo filho em relação à população (*ranking*). Considerando um problema (P) de minimização, se aplicarmos o operador Op_i sobre o indivíduo pai $\vec{x}_j^{\,p}$, teremos um indivíduo filho $\vec{x}_j^{\,f}$ e uma recompensa R_i dada por:

$$R_i = \sum_{p=1}^{NP} \Omega(j,p) \tag{5.19}$$

$$\Omega(j,p) = \begin{cases} 1, & \text{se } (f_j, \phi_j) <_\varepsilon (f_p, \phi_p) \\ 0, & \text{caso contrário} \end{cases}$$

onde NP é o número de indivíduos da população, (f_j, ϕ_j) são os valores da função objetivo e da função violação de restrição para o j-ésimo indivíduo da população, e $<_\varepsilon$ é o comparador ε-*level* definido em (5.2).

5.4 Algoritmo εMOES

O algoritmo MOES (*Multiple Operators Evolutionary System*) proposto neste trabalho é um algoritmo evolutivo baseado nas estratégias evolutivas e nos conceitos dos algoritmos evolutivos adaptativos. Com objetivo gerar um algoritmo robusto e eficaz para resolver uma grande gama de problemas de otimização, o MOES usa nove operadores evolutivos cujas probabilidades de seleção são adaptadas de forma automática e dinâmica. Buscando otimizar um problema da maneira mais eficiente possível, o algoritmo MOES ajusta as probabilidades dos operadores de forma a usar com mais frequência os operadores mais adequados para se resolver este problema.

A versão εMOES é resultado da substituição dos comparadores ordinais pelos comparadores ε-*level*, de forma que esta versão do algoritmo possa otimizar problemas restritos.

O Algoritmo 9 mostra o pseudo-código do algoritmo εMOES. Dado que queremos resolver um problema de minimização (P) com n dimensões, o passo a passo do algoritmo é explicado a seguir:

1. Na iteração $t = 0$, a função *criarPop* executa a tarefa de criar uma população inicial aleatória de NP indivíduos uniformemente distribuídos pelo espaço de busca $\mathbb{X}$ de (P): $Pop = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{NP}\}$ onde $\vec{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$.

2. A função *avaliarAptidao* avalia todos os NP indivíduos da população Pop, calculando para cada indivíduo $\vec{x}_i$ o valor da função objetivo $f(\vec{x}_i)$ e da função de violação de restrição $\phi(\vec{x}_i)$ (ver expressão (5.1)).

3. O período para a adaptação das probabilidades é definido como $\tau = 3NP$, ou seja, a cada $3NP$ avaliações de (P) um ciclo de adaptação de probabilidades ocorre.

4. As variáveis $\vec{P}$, $\vec{Q}$, $\vec{R}$, $\vec{M}$, P_{min} e P_{max} são inicializadas.

5. O critério de parada do algoritmo é definido como um número máximo de avaliações do problema (P) (cálculo da função objetivo e da função violação de restrição), de forma que o algoritmo realizará $Tmax$ avaliações.

6. A cada iteração do algoritmo atualiza-se o parâmetro ε usado no método ε-*constrained* (ver seção 5.1.3).

7. Para cada indivíduo $\vec{x}_i$ de Pop, um operador Op_s é selecionado e aplicado sobre este indivíduo:

   - A função $SelecionarOperador$ retorna o índice do operador selecionado (sorteio onde a probabilidade de um operador s ser sorteado é P_s). O algoritmo usado é chamado de *roulette wheel*. Os operadores contidos na lista Ops são os nove operadores apresentados na secção 5.2.
   - A função $AplicarOperador$ aplica o operador selecionado sobre o indivíduo $\vec{x}_i$ e usa o interpretador I para gerar a recompensa do operador. A recompensa do operador é então armazenada em R_s. O Algoritmo 10 mostra o pseudo-código desta função. Operadores que requerem mais de um indivíduo pai devem seleciona-los aleatoriamente na população. O interpretador I pode ser do tipo I_{local} ou I_{global} ou I_{rank} (ver seção 5.3).

8. A cada τ avaliações de (P) um ciclo de adaptação ocorre:

   (a) A média das recompensas recebidas é calculada;

   (b) O vetor de qualidade é atualizado (de acordo com a expressão (4.3); e

   (c) O vetor de probabilidades é atualizado através da função $AdaptarProbabilidades$ (ver Algoritmo 11). A estratégia usada para atualizar o vetor $\vec{P}$ é o *Adaptive Pursuit* (de acordo com as expressões (4.6) e (4.7)). Os parâmetros α e β devem ser ajustados pelo usuário. Neste trabalho, adotamos $\alpha = 0,50$ e $\beta = 0,25$ para todos os problemas.

9. Ao fim de um ciclo evolutivo, a função $OrdenarPop$ é executada para ordenar a população em função dos valores de aptidão dos indivíduos (o comparador ε-*level* é usado).

10. Quando o algoritmo realizar T_{max} avaliações de (P) o processo evolutivo é finalizado e o algoritmo retorna o melhor indivíduo da população Pop: esta será a melhor solução encontrada para (P).

Algoritmo 9: εMOES (*ε-constrained + Multiple Operators Evolutionary System*)

Entrada: Ops (lista com K operadores evolutivos), $Prob$ (problema de otimização restrita a ser resolvido), NP (número de indivíduos da população), e I (interpretador de produtividade de operadores).

Dados: $\vec{P}$ (vetor de probabilidades com K componentes), $\vec{Q}$ (vetor de qualidade com K componentes), $\vec{R}$ (vetor de recompensas com K componentes), $\vec{M}$ (vetor de frequências com K componentes), P_{min} (valor mínimo para as probabilidades), P_{max} (valor máximo para as probabilidades), Pop (lista com NP indivíduos - população), s (índice do operador selecionado), t (contador de avaliações de (P)), T_{max} (número máximo de avaliações de (P)), τ (período para adaptação das probabilidades), e ε (valor do parâmetro ε-*level*)

Saída: $\vec{x}_{best}$ (melhor solução encontrada para (P)).

início
  $t \leftarrow 0$
  $\tau \leftarrow 3 \cdot NP$
  $Pop \leftarrow criarPop(NP, Prob)$
  $avaliarAptidao(Pop, Prob)$
  $P_{min} = \frac{1}{2K}$
  $P_{max} \leftarrow 1 - (K-1)P_{min}$
  para $i \leftarrow 1$ *até* K **faça**
    $P[i] \leftarrow \frac{1}{K}$
    $Q[i] \leftarrow 0$
    $R[i] \leftarrow 0$
    $M[i] \leftarrow 0$
  fim
  enquanto $t < T_{max}$ **faça**
    $\varepsilon \leftarrow \varepsilon(t)$
    para $i \leftarrow 1$ *até* NP **faça**
      $s \leftarrow SelecionarOperador(\vec{P})$
      $R[s] \leftarrow R[s] + AplicarOperador(Ops[s], Pop[i], Pop, Prob, I)$
      $M[s] \leftarrow M[s] + 1$
      $t \leftarrow t + 1$
    fim
    se $t \ mod \ \tau = 0$ **então**
      para $i \leftarrow 1$ *até* K **faça**
        $R[i] \leftarrow \frac{R[i]}{M[i]}$
        $Q[i] \leftarrow Q[i] + \alpha(R[i] - Q[i])$
        $R[i] \leftarrow 0$
        $M[i] \leftarrow 0$
      fim
      $AdaptarProbabilidades(\vec{P}, \vec{Q}, P_{min}, P_{max})$
    fim
    $OrdenarPop(Pop)$
  fim
  $Retornar \ Pop[0]$
fim

Algoritmo 10: $R_s \leftarrow AplicarOperador(Op_s, \vec{x}_i, Pop, Prob, I)$

Entrada: Op_s (operador evolutivo selecionado), $\vec{x}_i$ (indivíduo pai), Pop (lista com NP indivíduos - população), $Prob$ (problema de otimização restrita a ser resolvido), e I (interpretador de produtividade de operadores).

Dados: $\vec{x}_{filho}$ (indivíduo filho gerado pelo operador Op_s).

Saída: R (valor da recompensa gerada por I).

início
  $R \leftarrow 0$
  $\vec{x}_{filho} \leftarrow Op_s(\vec{x}_i,\ Pop)$
  $avaliarAptidao(\vec{x}_{filho}, Prob)$
  se $(f_{filho}, \phi_{filho}) <_\varepsilon (f_i, \phi_i)$ **então**
    $R \leftarrow I(\vec{x}_{filho})$
    $\vec{x}_i \leftarrow \vec{x}_{filho}$
  fim
  $Retornar\ R$
fim

Algoritmo 11: $AdaptarProbabilidades(\vec{P}, \vec{Q}, P_{min}, P_{max})$

Entrada: $\vec{P}$ (vetor de probabilidades com K componentes), $\vec{Q}$ (vetor de qualidade com K componentes), P_{min} (valor mínimo para as probabilidades) e P_{max} (valor máximo para as probabilidades).

Dados: Max (índice do operador com maior qualidade).

início
  $Max \leftarrow RandInt(1, K)$
  para $i \leftarrow 1$ *até* K **faça**
    se $Q[i] > Q[Max]$ **então**
      $Max \leftarrow i$
    fim
  fim
  para $i \leftarrow 1$ *até* K **faça**
    se $i = Max$ **então**
      $P[i] \leftarrow P[i] + \beta(P_{max} - P[i])$
    senão
      $P[i] \leftarrow P[i] + \beta(P_{min} - P[i])$
    fim
  fim
fim

6 *Resultados e Discussões*

Neste capítulo temos os resultados obtidos da aplicação do algoritmo proposto aos 24 problemas testes selecionados. Temos uma comparação dos resultados obtidos com as quatro versões do algoritmo εMOES e com outros três algoritmos evolutivos tradicionais não adaptativos: εGA, εPSO e εDE. Além dos resultados, apresentamos também as configurações utilizadas, os parâmetros usados, a metodologia de análise e alguns comentários sobre as dificuldades encontradas na resolução de alguns destes problemas. Por fim, uma análise de complexidade dos algoritmos é apresentada e discutida.

6.1 Problemas Testes (*Benchmark*)

Para avaliar a performance do algoritmo proposto e fazer uma comparação com outros algoritmos evolutivos, utilizou-se um conjunto com 24 problemas de otimização restrita proposto para a CEC'2006 (LIANG et al., 2006). No Anexo A temos uma descrição completa destes 24 problemas e os seus valores ótimos (ou melhores valores conhecidos). As Tabelas 6.1 e 6.2 mostram um resumo das principais características dos problemas usados.

Na Tabela 6.1, temos que n é o número de variáveis de controle (dimensão do problema), ρ é a relação estimada entre a região de factibilidade $\mathbb{F}$ e o espaço de busca $\mathbb{X}$, DL é o número de restrições de desigualdade linear, DN é o número de restrições de desigualdade não lineares, EL é o número de restrições de igualdade lineares e EN é o número de restrições de igualdade não lineares. NA é o número de restrições ativas no ponto ótimo $\vec{x}^*$.

O valor do parâmetro ρ pode ser estimado da seguinte maneira:

$$\rho = \frac{|\mathbb{F}|}{|\mathbb{X}|} \tag{6.1}$$

onde, $|\mathbb{F}|$ é o número de soluções factíveis e $|\mathbb{X}|$ é o número de soluções aleatoriamente geradas. Em (LIANG et al., 2006), o número total de soluções para $|\mathbb{X}|$ é de 1.000.000. Quanto menor for o valor de ρ, mais difícil será gerar uma solução dentro de $\mathbb{F}$.

Tabela 6.1: Resumo das principais características dos 24 problemas testes.

Prob.	**n**	**Tipo de** $f(\vec{x})$	ρ	**DL**	**DN**	**EL**	**EN**	**NA**
g01	13	Quadrática	0,0111%	9	0	0	0	6
g02	20	Não linear	99,997%	0	2	0	0	1
g03	10	Polinomial	0,0000%	0	0	0	1	1
g04	5	Quadrática	52,123%	0	6	0	0	2
g05	4	Cúbica	0,0000%	2	0	0	3	3
g06	2	Cúbica	0,0066%	0	2	0	0	2
g07	10	Quadrática	0,0003%	3	5	0	0	6
g08	2	Não linear	0,8560%	0	2	0	0	0
g09	7	Polinomial	0,5121%	0	4	0	0	2
g10	8	Linear	0,0010%	3	3	0	0	6
g11	2	Quadrática	0,0000%	0	0	0	1	1
g12	3	Quadrática	4,7713%	0	1	0	0	0
g13	5	Não linear	0,0000%	0	0	0	3	3
g14	10	Não linear	0,0000%	0	0	3	0	3
g15	3	Quadrática	0,0000%	0	0	1	1	2
g16	5	Não linear	0,0204%	4	34	0	0	4
g17	6	Não linear	0,0000%	0	0	0	4	4
g18	9	Quadrática	0,0000%	0	13	0	0	6
g19	15	Não linear	33,476%	0	5	0	0	0
g20	24	Linear	0,0000%	0	6	2	12	16
g21	7	Linear	0,0000%	0	1	0	5	6
g22	22	Linear	0,0000%	0	1	8	11	19
g23	9	Linear	0,0000%	0	2	3	1	6
g24	2	Linear	79,656%	0	2	0	0	2

Analisando-se as tabelas 6.1 e 6.2 podemos perceber claramente a variedade de características dos 24 problemas testes utilizados. Estas variedades estão relacionadas com os diferentes tipos de função objetivo (quadrática, não-linear, linear, polinomial e cúbica) e os diferentes tipos de restrições (desigualdades não-lineares e lineares, e igualdades lineares e não-lineares). Para todos os 24 problemas aplicou-se o algoritmo proposto utilizando a mesma estrutura apresentada no capítulo anterior.

De acordo com o Anexo A, cada problema (P) é composto de uma função objetivo $f(\vec{x})$, q restrições de desigualdade $g_j(\vec{x}) \leq 0$ e $m-q$ restrições de igualdade $h_j(\vec{x}) = 0$, onde todo ponto $\vec{x} = [x_1, \ldots, x_n]$ do espaço de busca deve satisfazer as restrições de limite inferior x_i^L e limite superior x_i^U. Contudo, neste trabalho as restrições de igualdade foram relaxadas por uma tolerância de $\xi = 0,0001$. Assim, todas as restrições de igualdade foram substituídas por restrições de desigualdades com o seguinte formato:

$$|h_j(\vec{x})| - \xi \leq 0, \; para \; j = q+1, \ldots, m \tag{6.2}$$

Tabela 6.2: Os valores de $f(\vec{x}^*)$ e os limites do espaço de busca para os 24 problemas testes.

Prob.	n	$\mathbf{f}(\vec{x}^*)$	Limites para $\mathbb{X}$
g01	13	-15,00000	$0 \leq x_i \leq 1$ $(i = 1,\ldots,9)$,$0 \leq x_i \leq 100$ $(i = 10,11,12)$ e $0 \leq x_{13} \leq 1$
g02	20	-0,803619	$0 < x_i \leq 10$ $(i = 1,\ldots,n)$
g03	10	-1,00050010	$0 \leq x_i \leq 1$ $(i = 1,\ldots,n)$
g04	5	-30665,53867178	$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ e $27 \leq x_i \leq 45$ $(i = 3,4,5)$
g05	4	5126,49671401	$0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0,55 \leq x_3 \leq 0,55$ e $-0,55 \leq x_4 \leq 0,55$
g06	2	-6961,81387558	$13 \leq x_1 \leq 100$ e $0 \leq x_2 \leq 100$
g07	10	24,30620907	$-10 \leq x_i \leq 10$ $(i = 1,\ldots,10)$
g08	2	-0,09582504	$0 \leq x_1 \leq 10$ e $0 \leq x_2 \leq 10$
g09	7	680,63005737	$-10 \leq x_i \leq 10$ for $(i = 1,\ldots,7)$
g10	8	7049,24802055	$100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2,3)$ e $10 \leq x_i \leq 1000$ $(i = 4,\ldots,8)$
g11	2	0,74990000	$-1 \leq x_1 \leq 1$ e $-1 \leq x_2 \leq 1$
g12	3	-1,00000	$0 \leq x_i \leq 10$ $(i = 1,2,3)$e $p,q,r = 1,2,\ldots,9$
g13	5	0,05394151	$-2,3 \leq x_i \leq 2,3$ $(i = 1,2)$ e $-3,2 \leq x_i \leq 3,2$ $(i = 3,4,5)$
g14	10	-47,76488846	$0 < x_i \leq 10$ $(i = 1,\ldots,10)$
g15	3	961,71502229	$0 \leq x_i \leq 10$ $(i = 1,2,3)$
g16	5	-1,90515526	$704,4148 \leq x_1 \leq 906,3855,68,6 \leq x_2 \leq 288,88$, $0 \leq x_3 \leq 134,75$, $193 \leq x_4 \leq 287,0966$, $25 \leq x_5 \leq 84,1988$
g17	6	8853,53967481	$0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ e $0 \leq x_6 \leq 0,5236$
g18	9	-0,86602540	$-10 \leq x_i \leq 10$ $(i = 1,\ldots,8)$ e $0 \leq x_9 \leq 20$
g19	15	32,65559295	$0 \leq x_i \leq 10$ $(i = 1,\ldots,15)$
g20	24	0,14742926	$0 \leq x_i \leq 10$ $(i = 1,\ldots,24)$
g21	7	193,72451007	$0 \leq x_1 \leq 1000$,$0 \leq x_2,x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6,3 \leq x_5 \leq 6,7$, $5,9 \leq x_6 \leq 6,4$ e $4,5 \leq x_7 \leq 6,25$
g22	22	236,43097550	$0 \leq x_1 \leq 20000$, $0 \leq x_2,x_3,x_4 \leq 1 \times 10^6$, $0 \leq x_5,x_6,x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299,99$,$100 \leq x_9 \leq 399,99$, $100,01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13},x_{14},x_{15} \leq 500$,$0,01 \leq x_{16} \leq 300$, $0,01 \leq x_{17} \leq 400$, $-4,7 \leq x_{18},x_{19},x_{20},x_{21},x_{22} \leq 6,25$
g23	9	-400,05509978	$0 \leq x_1,x_2,x_6 \leq 300$, $0 \leq x_3,x_5,x_7 \leq 100$, $0 \leq x_4,x_8 \leq 200$ e $0,01 \leq x_9 \leq 0,03$
g24	2	-5,50801327	$0 \leq x_1 \leq 3$ e $0 \leq x_2 \leq 4$

6.2 Algoritmos Testados

Para avaliar o desempenho do algoritmo proposto (εMOES), sete algoritmos evolutivos foram implementados e aplicados aos problemas de otimização:

1. **εMOES-R1**: Versão do algoritmo εMOES cujos valores de probabilidade são adaptados usando o método *Adaptive Pursuit* e o interpretador de produtividade I_{local}.

2. **εMOES-R2**: Versão do algoritmo εMOES cujos valores de probabilidade são adaptados usando o método *Adaptive Pursuit* e o interpretador de produtividade I_{global}.

3. **εMOES-R3**: Versão do algoritmo εMOES cujos valores de probabilidade são adaptados usando o método *Adaptive Pursuit* e o interpretador de produtividade I_{rank}.

4. **εMOES-FX**: Versão do algoritmo εMOES cujos valores de probabilidade não são adaptados, ou seja, os valores iniciais permanecem fixos durante todo o processo evolutivo.

5. **εGA**: Proposto em (TAKAHAMA; SAKAI, 2006), este algoritmo pode ser classificado como um Algoritmo Genético (GA) convencional usando a metodologia ε-*constrained* para fazer a manipulação de restrições. Este algoritmo faz uso dos operadores evolutivos *Blend Crossover* e *Gaussian Mutation*.

6. **εPSO**: Proposto em (TAKAHAMA; SAKAI, 2006), este algoritmo foi baseado na metodologia de Otimização por Enxame de Partículas e usa a metodologia ε-*constrained* para fazer a manipulação de restrições. Nesta versão, as velocidades das partículas são controladas.

7. **εDE**: Proposto em (TAKAHAMA; SAKAI, 2006), este algoritmo foi baseado nos algoritmos de Evolução Diferencial e usa a metodologia ε-*constrained* para fazer a manipulação de restrições. A estratégia implementada é a "DE/rand/l/exp".

Todos os algoritmos listados acima foram implementados na linguagem *C#* e compartilham as mesmas classes base. Assim, as implementações dos comparadores ε-*level* e dos problemas testes são as mesmas para todos os algoritmos.

Os algoritmos εGA, εPSO e εDE foram escolhidos por usar a metodologia ε-*constrained* para fazer a manipulação de restrições e pelos bons resultados apresentados nos artigos publicados pelo autor. Estes algoritmos foram implementados e aplicados aos 24 problemas propostos.

6.3 Ajuste de Parâmetros

Os parâmetros estáticos do algoritmo εMOES que foram usados para gerar os resultados apresentados neste capítulo são mostrados na Tabela 6.3.

Tabela 6.3: Parâmetros usados no Algoritmo εMOES. n é a dimensão do problema.

Símbolo	Descrição	Valor
T_{max}	Número máximo de avaliações ao problema	500.000
T_c	Fim do relaxamento das restrições	100.000
cp	Velocidade de redução do relaxamento ε	100
NP	Número de indivíduos da população	$2.n+60$
α	Taxa de adaptação para as qualidades	$0,50$
β	Taxa de adaptação para as probabilidades	$0,25$
τ	Período do clico adaptativo	$3.NP$

Como mostrado no Algoritmo 9, o processo evolutivo executa T_{max} avaliações ao problema em questão. Como forma de comparar os resultados, esse critério de parada ($t < T_{max}$) foi

adicionado aos algoritmos εGA, εPSO e εDE. Assim, todos os algoritmos testados realizaram no máximo 500.000 avaliações das funções objetivo e violação de restrição a cada execução. Também temos que os valores de NP, cp e T_c são os mesmos para todos os algoritmos.

6.4 Metodologia Experimental

Nesta seção, vamos apresentar a metodologia experimental usada para avaliar o desempenho do algoritmo proposto. Contudo, para entender as medidas de desempenho usadas para avaliar os algoritmos testados, algumas definições se fazem necessárias:

- **Avaliação do Problema**: É o cálculo do valor da função objetivo e dos valores das restrições para uma solução (no contexto dos EAs, uma solução é chamada de ponto de busca ou indivíduo). O número total de avaliações de um algoritmo é uma importante medida de custo computacional. Na Tabela 6.3 temos que cada algoritmo executa no máximo 500.000 avaliações do problema.

- **Solução Factível**: É uma solução que satisfaz todas as restrições do problema. No método ε-*constrained*, uma solução é dita factível quando $\phi(\vec{x}) = 0$.

- **Execução Factível**: É uma execução de um dado algoritmo em que pelo menos a melhor solução (melhor indivíduo da população) é uma solução factível para o problema tratado. Vale lembrar que o método ε-*constrained* usa uma relação lexográfica onde a factibilidade de uma solução é mais importante que a otimização da função objetivo.

- **Solução Bem-sucedida**: Solução factível que é igual ou melhor que a solução ótima (ou melhor solução conhecida) do problema. Na classificação de uma solução $\vec{x}$ como bem-sucedida, verificamos se ela é factível ($\phi(\vec{x}) = 0$) e se $(f(\vec{x}) - f(\vec{x}^*)) < 0,0001$, onde $\vec{x}^*$ é o ótimo global para o problema. O número de avaliações do problema necessário para se obter a primeira solução bem-sucedida de uma execução será simbolizado por **PE**.

- **Execução Bem-sucedida**: É uma execução de um dado algoritmo em que pelo menos a melhor solução (melhor indivíduo da população) é uma solução bem-sucedida.

Os algoritmos de otimização global geram ao final de sua execução uma solução candidata a solução do problema tratado, tendo como objetivo encontrar uma solução que corresponda ao valor de ótimo global para o problema. Neste trabalho, para cada execução de um algoritmo, a solução final retornada pelo mesmo será o melhor indivíduo da população (melhor valor de aptidão).

Para produzir os resultados apresentados neste capítulo, os sete algoritmos listados na seção 6.2 foram executados 30 vezes para cada um dos 24 problemas, sendo que estas execuções foram feitas de forma independente uma da outra (para cada execução o gerador de números aleatórios recebeu uma *seed* diferente). Assim, um problema (P) gerou sete conjuntos de 30 soluções cada, sendo um para cada algoritmo.

Os dados apresentados nas tabelas da seção 6.5, são descritos a seguir:

- $\mathbf{f_{melhor}}$: Valor da função objetivo para a melhor solução entre as 30 soluções encontradas. Entre colchetes temos o número total de restrições violadas por esta solução.
- $\mathbf{f_{pior}}$: Valor da função objetivo para a pior solução entre as 30 soluções encontradas. Entre colchetes temos o número total de restrições violadas por esta solução.
- $\mathbf{f}_{\text{médio}}$: Valor médio para a função objetivo, calculado com as 30 soluções encontradas.
- $\sigma_{\mathbf{f(x)}}$ Desvio padrão para os valores da função objetivo gerados com as 30 soluções encontradas.
- $\phi_{\text{médio}}$: Valor médio para a função violação de restrição (expressão (5.1)), calculado com as 30 soluções encontradas.
- $\mathbf{PE_{melhor}}$: Das 30 execuções realizadas, é o menor número de avaliações que foi necessário para se obter a primeira solução bem-sucedida.
- $\mathbf{PE_{pior}}$: Das 30 execuções realizadas, é o maior número de avaliações que foi necessário para se obter a primeira solução bem-sucedida.
- $\mathbf{PE}_{\text{médio}}$: Média dos valores de PE para as 30 execuções realizadas.
- $\sigma_{\mathbf{PE}}$: Desvio padrão para os valores de PE encontrados nas 30 execuções.
- **FR**: Taxa de Factibilidade, calculada da seguinte forma:

$$FR = \frac{\text{número total de soluções factíveis}}{\text{número total de execuções}}(\%)$$

- **SR**: Taxa de Sucesso, calculada da seguinte forma:

$$SR = \frac{\text{número total de soluções bem-sucedidas}}{\text{número total de execuções}}(\%)$$

- $\mathbf{FR}_{\text{médio}}$: Valor médio da taxa de factibilidade:

$$FR_{\text{médio}} = \frac{1}{24}\sum_{i=1}^{24} FR_i(\%)$$

onde FR_i é a taxa de factibilidade para o problema i.

- $\mathbf{SR}_{\text{médio}}$: Valor médio da taxa de sucesso:

$$SR_{\text{médio}} = \frac{1}{24}\sum_{i=1}^{24} SR_i(\%)$$

onde SR_i é a taxa de sucesso para o problema i.

6.5 Resultados Obtidos

Nesta seção apresentamos 24 tabelas (uma para cada problema) comparando os resultados obtidos para os sete algoritmos listando na seção 6.2. Estes algoritmos foram implementados com a linguagem *C#* e compartilham as mesmas classes base. Os dados apresentados nas tabelas de 6.4 a 6.27 estão detalhados na seção 6.4.

Tabela 6.4: Resultados obtidos para o problema g01, onde $f(\vec{x}^*) = -15,0000$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f}_{\mathbf{melhor}}$	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]
$\mathbf{f}_{\mathbf{pior}}$	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-15,0000$[0]	$-11,2717$[0]	$-8,0000$[0]	$-15,0000$[0]
$\mathbf{f}_{\text{médio}}$	$-15,0000$	$-15,0000$	$-15,0000$	$-15,0000$	$-13,2468$	$-11,7453$	$-15,0000$
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	1,2018	1,7412	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
$\mathbf{PE}_{\mathbf{melhor}}$	18880	17600	18560	20240	86480	18720	23200
$\mathbf{PE}_{\mathbf{pior}}$	22160	20320	21040	25120	-	-	26720
$\mathbf{PE}_{\text{médio}}$	20800	18933	19883	23016	203184	20933	24859
$\sigma_{\mathbf{PE}}$	980,01	733,02	705,86	1121,74	95412,54	1736,00	889,15
FR	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
SR	100,0%	100,0%	100,0%	100,0%	16,7%	10,0%	100,0%

Tabela 6.5: Resultados obtidos para o problema g02, onde $f(\vec{x}^*) = -0,8036$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f}_{\mathbf{melhor}}$	$-0,8036$[0]	$-0,8036$[0]	$-0,8036$[0]	$-0,8036$[0]	$-0,5303$[0]	$-0,8036$[0]	$-0,8036$[0]
$\mathbf{f}_{\mathbf{pior}}$	$-0,7926$[0]	$-0,7926$[0]	$-0,7926$[0]	$-0,7926$[0]	$-0,3952$[0]	$-0,7721$[0]	$-0,8035$[0]
$\mathbf{f}_{\text{médio}}$	$-0,7968$	$-0,7980$	$-0,7959$	$-0,7969$	$-0,4397$	$-0,7903$	$-0,8035$
$\sigma_{\mathbf{f(x)}}$	0,0049	0,0040	0,0046	0,0045	0,0366	0,0081	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
$\mathbf{PE}_{\mathbf{melhor}}$	75960	73920	81360	78000	-	114840	473280
$\mathbf{PE}_{\mathbf{pior}}$	-	-	-	-	-	-	-
$\mathbf{PE}_{\text{médio}}$	110220	98040	138531	94455	-	177930	481296
$\sigma_{\mathbf{PE}}$	27746,70	18508,12	35686,57	13043,27	-	50513,33	9447,41
FR	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
SR	33,3%	30,0%	23,3%	26,7%	0,0%	13,3%	16,7%

Tabela 6.6: Resultados obtidos para o problema g03, onde $f(\vec{x}^*) = -1,0005$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-1,0005$[0]	$-1,0005$[0]	$-1,0005$[0]	$-1,0005$[0]	$-1,0003$[0]	$-1,0005$[0]	$-1,0005$[0]
$\mathbf{f_{pior}}$	$-1,0005$[0]	$-1,0005$[0]	$-1,0005$[0]	$-1,0005$[0]	$-0,9991$[0]	$-0,9329$[0]	$-1,0005$[0]
$\mathbf{f}_{\text{médio}}$	$-1,0005$	$-1,0005$	$-1,0005$	$-1,0005$	$-0,9999$	$-0,9956$	$-1,0005$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0003$	$0,0151$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	61180	61180	61390	51170	-	298830	72100
$\mathbf{PE_{pior}}$	64120	64330	79870	53060	-	-	84700
$\mathbf{PE}_{\text{médio}}$	62046	62298	69328	52010	-	421182	78113
$\sigma_{\mathbf{PE}}$	$667,14$	$815,83$	$5543,85$	$505,26$	-	$69157,71$	$2938,80$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$60,0\%$	$100,0\%$

Tabela 6.7: Resultados obtidos para o problema g04, onde $f(\vec{x}^*) = -30665,5387$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-30665,5$[0]	$-30665,5$[0]	$-30665,5$[0]	$-30665,5$[0]	$-30665,2$[0]	$-30665,5$[0]	$-30665,5$[0]
$\mathbf{f_{pior}}$	$-30665,5$[0]	$-30665,5$[0]	$-30665,5$[0]	$-30665,5$[0]	$-30660,8$[0]	$-30665,5$[0]	$-30665,5$[0]
$\mathbf{f}_{\text{médio}}$	$-30665,5$	$-30665,5$	$-30665,5$	$-30665,5$	$-30663,8$	$-30665,5$	$-30665,5$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$1,1112$	$0,0000$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	8640	7260	7800	11460	-	13560	10860
$\mathbf{PE_{pior}}$	11940	10020	10440	16380	-	19620	25920
$\mathbf{PE}_{\text{médio}}$	10638	9066	9430	13932	-	16560	13814
$\sigma_{\mathbf{PE}}$	$834,86$	$604,06$	$653,44$	$1246,87$	-	$1629,45$	$3365,70$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$100,0\%$	$100,0\%$

Tabela 6.8: Resultados obtidos para o problema g05, onde $f(\vec{x}^*) = 5126,4967$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$5126,4967$[0]	$5126,4967$[0]	$5126,4967$[0]	$5126,4967$[0]	$5150,4873$[1]	$5126,4967$[0]	$5126,4967$[0]
$\mathbf{f_{pior}}$	$5126,4967$[0]	$5126,4967$[0]	$5126,4967$[0]	$5126,4967$[0]	$5136,2734$[1]	$3552,0000$[2]	$5126,4967$[0]
$\mathbf{f}_{\text{médio}}$	$5126,4967$	$5126,4967$	$5126,4967$	$5126,4967$	$5138,2204$	$5088,0582$	$5126,4967$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$14,4635$	$287,1455$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,2791$	$15,9796$	$0,0000$
$\mathbf{PE_{melhor}}$	80040	80220	79620	74400	-	82440	79920
$\mathbf{PE_{pior}}$	81240	81840	81720	75660	-	-	82260
$\mathbf{PE}_{\text{médio}}$	80732	81054	81042	75231	-	82440	81186
$\sigma_{\mathbf{PE}}$	$304,66$	$531,12$	$491,16$	$304,92$	-	-	$590,60$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$90,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$3,3\%$	$100,0\%$

Tabela 6.9: Resultados obtidos para o problema g06, onde $f(\vec{x}^*) = -6961,8139$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-6961,81[0]$	$-6961,81[0]$	$-6961,81[0]$	$-6961,81[0]$	$-6943,93[0]$	$-7973,00[1]$	$-6961,81[0]$
$\mathbf{f_{pior}}$	$-6961,81[0]$	$-6961,81[0]$	$-6961,81[0]$	$-6961,81[0]$	$-6905,73[0]$	$-7973,00[1]$	$-6961,81[0]$
$\mathbf{f}_{\text{médio}}$	$-6961,81$	$-6961,81$	$-6961,81$	$-6961,81$	$-6924,61$	$-7973,00$	$-6961,81$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$8,1886$	$0,0000$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$11,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	15700	12300	14300	21200	-	-	11100
$\mathbf{PE_{pior}}$	18700	17100	16900	24900	-	-	15700
$\mathbf{PE}_{\text{médio}}$	17343	15960	15880	22925	-	-	13277
$\sigma_{\mathbf{PE}}$	$845,25$	$1014,76$	$672,01$	$899,37$	-	-	$965,98$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$0,0\%$	$100,0\%$

Tabela 6.10: Resultados obtidos para o problema g07, onde $f(\vec{x}^*) = 24,3062$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$24,3062[0]$	$24,3062[0]$	$24,3062[0]$	$24,3062[0]$	$24,7036[0]$	$25,0995[0]$	$24,3062[0]$
$\mathbf{f_{pior}}$	$24,3062[0]$	$24,3062[0]$	$24,3063[0]$	$24,3062[0]$	$28,0166[0]$	$345,5775[2]$	$24,3062[0]$
$\mathbf{f}_{\text{médio}}$	$24,3062$	$24,3062$	$24,3062$	$24,3062$	$26,1612$	$130,9489$	$24,3062$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,9202$	$190,9549$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$2,8758$	$0,0000$
$\mathbf{PE_{melhor}}$	97760	82720	106640	93600	-	-	163360
$\mathbf{PE_{pior}}$	287760	167520	441520	167760	-	-	188000
$\mathbf{PE}_{\text{médio}}$	187805	120520	261013	123380	-	-	173275
$\sigma_{\mathbf{PE}}$	$53727,08$	$22370,52$	$91878,35$	$23498,78$	-	-	$6598,03$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$80,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$0,0\%$	$100,0\%$

Tabela 6.11: Resultados obtidos para o problema g08, onde $f(\vec{x}^*) = -0,0958$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$
$\mathbf{f_{pior}}$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0958[0]$	$-0,0272[0]$	$0,0000[1]$	$0,0000[1]$
$\mathbf{f}_{\text{médio}}$	$-0,0958$	$-0,0958$	$-0,0958$	$-0,0958$	$-0,0616$	$-0,0543$	$-0,0511$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0342$	$0,0475$	$0,0478$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,4333$	$0,4667$
$\mathbf{PE_{melhor}}$	3420	3420	3300	3540	3480	3840	3660
$\mathbf{PE_{pior}}$	5100	4740	4800	5400	-	-	-
$\mathbf{PE}_{\text{médio}}$	4378	4242	4334	4626	5248	4902	6510
$\sigma_{\mathbf{PE}}$	$434,74$	$326,12$	$318,88$	$498,36$	$965,70$	$633,90$	$3964,03$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$56,7\%$	$53,3\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$50,0\%$	$56,7\%$	$53,3\%$

Tabela 6.12: Resultados obtidos para o problema g09, onde $f(\vec{x}^*) = 680,6301$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	680,6301[0]	680,6301[0]	680,6301[0]	680,6301[0]	680,7143[0]	680,6311[0]	680,6301[0]
$\mathbf{f_{pior}}$	680,6301[0]	680,6301[0]	680,6301[0]	680,6301[0]	681,7554[0]	680,6536[0]	680,6301[0]
$\mathbf{f}_{\text{médio}}$	680,6301	680,6301	680,6301	680,6301	681,2287	680,6361	680,6301
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	0,2758	0,0046	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
$\mathbf{PE_{melhor}}$	25840	22800	25760	29600	-	-	41040
$\mathbf{PE_{pior}}$	35120	33280	44880	42640	-	-	48240
$\mathbf{PE}_{\text{médio}}$	30469	28765	34856	35212	-	-	45176
$\sigma_{\mathbf{PE}}$	2759,69	3111,76	5143,12	3230,58	-	-	1710,85
FR	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
SR	100,0%	100,0%	100,0%	100,0%	0,0%	0,0%	100,0%

Tabela 6.13: Resultados obtidos para o problema g10, onde $f(\vec{x}^*) = 7049,2480$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	7049,2480[0]	7049,2480[0]	7049,2480[0]	7049,2480[0]	7104,0139[0]	7264,1900[0]	7049,2480[0]
$\mathbf{f_{pior}}$	7049,2481[0]	7049,2480[0]	7049,2480[0]	7049,2480[0]	14376,85[0]	2100,0000[1]	7250,9673[0]
$\mathbf{f}_{\text{médio}}$	7049,2480	7049,2480	7049,2480	7049,2480	9638,2342	7817,7620	7069,4200
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	1865,1081	3136,5650	60,5158
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,3150	0,0000
$\mathbf{PE_{melhor}}$	168960	142240	184320	173360	-	-	203440
$\mathbf{PE_{pior}}$	472160	274480	453840	423200	-	-	-
$\mathbf{PE}_{\text{médio}}$	331440	216779	352664	243520	-	-	220839
$\sigma_{\mathbf{PE}}$	92467,15	37050,99	91622,80	64628,90	-	-	8046,98
FR	100,0%	100,0%	100,0%	100,0%	100,0%	70,0%	100,0%
SR	100,0%	100,0%	100,0%	100,0%	0,0%	0,0%	90,0%

Tabela 6.14: Resultados obtidos para o problema g11, onde $f(\vec{x}^*) = 0,7499$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	0,7499[0]	0,7499[0]	0,7499[0]	0,7499[0]	0,7499[0]	0,7499[0]	0,7499[0]
$\mathbf{f_{pior}}$	0,7499[0]	0,7499[0]	0,7499[0]	0,7517[0]	0,7620[0]	1,0000[0]	0,7499[0]
$\mathbf{f}_{\text{médio}}$	0,7499	0,7499	0,7499	0,7501	0,7506	0,8928	0,7499
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0005	0,0023	0,1205	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
$\mathbf{PE_{melhor}}$	6480	3600	5760	4480	7360	8800	6960
$\mathbf{PE_{pior}}$	23200	11280	17120	-	-	-	7840
$\mathbf{PE}_{\text{médio}}$	11421	7699	8907	13415	214792	170152	7461
$\sigma_{\mathbf{PE}}$	5266,96	1515,18	3004,45	9941,00	150902,08	97270,80	276,31
FR	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
SR	100,0%	100,0%	100,0%	85,0%	66,7%	33,3%	100,0%

Tabela 6.15: Resultados obtidos para o problema g12, onde $f(\vec{x}^*) = -1,0000$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]
$\mathbf{f_{pior}}$	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-1,0000$[0]	$-0,9694$[0]	$-1,0000$[0]	$-1,0000$[0]
$\mathbf{f}_{\text{médio}}$	$-1,0000$	$-1,0000$	$-1,0000$	$-1,0000$	$-0,9954$	$-1,0000$	$-1,0000$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0064$	$0,0000$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	900	780	900	1020	180	60	720
$\mathbf{PE_{pior}}$	1740	1800	1680	2880	-	1620	6900
$\mathbf{PE}_{\text{médio}}$	1432	1348	1406	1821	381	750	5148
$\sigma_{\mathbf{PE}}$	$239,37$	$276,14$	$252,52$	$462,15$	$66,53$	$323,39$	$1303,02$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$46,7\%$	$100,0\%$	$100,0\%$

Tabela 6.16: Resultados obtidos para o problema g13, onde $f(\vec{x}^*) = 0,0539$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$0,0539$[0]	$0,0539$[0]	$0,0539$[0]	$0,0539$[0]	$0,0540$[0]	$0,0539$[0]	$0,0539$[0]
$\mathbf{f_{pior}}$	$0,0539$[0]	$0,0539$[0]	$0,0539$[0]	$0,0539$[0]	$0,5166$[1]	$0,0564$[0]	$0,4388$[0]
$\mathbf{f}_{\text{médio}}$	$0,0539$	$0,0539$	$0,0539$	$0,0539$	$0,3101$	$0,0545$	$0,0796$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,3283$	$0,0006$	$0,0960$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0002$	$0,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	68000	67680	63440	59680	84560	63760	67120
$\mathbf{PE_{pior}}$	68880	68800	68800	61440	-	-	-
$\mathbf{PE}_{\text{médio}}$	68504	68408	68125	60372	170240	67040	94780
$\sigma_{\mathbf{PE}}$	$268,65$	$308,70$	$1260,80$	$420,49$	$85680,00$	$1808,07$	$83702,45$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$83,3\%$	$100,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$6,7\%$	$16,7\%$	$93,3\%$

Tabela 6.17: Resultados obtidos para o problema g14, onde $f(\vec{x}^*) = -47,7649$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-47,7649$[0]	$-47,7649$[0]	$-47,7649$[0]	$-47,7649$[0]	$-47,6074$[0]	$-47,5628$[0]	$-47,7649$[0]
$\mathbf{f_{pior}}$	$-47,7649$[0]	$-47,7649$[0]	$-47,7649$[0]	$-47,7649$[0]	$-44,2926$[0]	$-318,4418$[1]	$-47,7649$[0]
$\mathbf{f}_{\text{médio}}$	$-47,7649$	$-47,7649$	$-47,7649$	$-47,7649$	$-46,3320$	$-97,5808$	$-47,7649$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,9086$	$101,6423$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$3,8667$	$0,0000$
$\mathbf{PE_{melhor}}$	134000	116200	174600	149200	-	-	391400
$\mathbf{PE_{pior}}$	215800	233600	341200	183600	-	-	431400
$\mathbf{PE}_{\text{médio}}$	178707	199307	260753	167300	-	-	410627
$\sigma_{\mathbf{PE}}$	$24749,72$	$27673,71$	$51021,90$	$9931,47$	-	-	$9631,99$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$80,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$0,0\%$	$100,0\%$

Tabela 6.18: Resultados obtidos para o problema g15, onde $f(\vec{x}^*) = 961,7150$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	961,7150[0]	961,7150[0]	961,7150[0]	961,7150[0]	961,8223[0]	961,7176[0]	961,7150[0]
$\mathbf{f_{pior}}$	961,7150[0]	961,7150[0]	961,7150[0]	961,7150[0]	968,0000[1]	967,9999[1]	961,7150[0]
$\mathbf{f}_{\text{médio}}$	961,7150	961,7150	961,7150	961,7150	964,6959	963,5612	961,7150
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	2,9258	2,9711	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	3,0015	0,6917	0,0000
$\mathbf{PE_{melhor}}$	11920	12240	12080	10640	-	-	11840
$\mathbf{PE_{pior}}$	16800	13280	13440	154640	-	-	12960
$\mathbf{PE}_{\text{médio}}$	12899	12693	12752	44988	-	-	12555
$\sigma_{\mathbf{PE}}$	852,84	217,95	296,90	38414,56	-	-	239,94
FR	100,0%	100,0%	100,0%	100,0%	3,3%	90,0%	100,0%
SR	100,0%	100,0%	100,0%	100,0%	0,0%	0,0%	100,0%

Tabela 6.19: Resultados obtidos para o problema g16, onde $f(\vec{x}^*) = -1,9052$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	−1,9052[0]	−1,9052[0]	−1,9052[0]	−1,9052[0]	−1,8976[0]	−1,9017[0]	−1,9052[0]
$\mathbf{f_{pior}}$	−1,9052[0]	−1,9052[0]	−1,9052[0]	−1,9052[0]	−1,0507[0]	−2,5799[3]	−2,5799[3]
$\mathbf{f}_{\text{médio}}$	−1,9052	−1,9052	−1,9052	−1,9052	−1,3782	−2,5134	−2,5104
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	0,2215	0,1828	0,2020
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	124,9115	123,8567
$\mathbf{PE_{melhor}}$	18400	19680	17840	21600	-	-	19280
$\mathbf{PE_{pior}}$	23760	23360	22960	30640	-	-	-
$\mathbf{PE}_{\text{médio}}$	21811	21792	21024	25832	-	-	20693
$\sigma_{\mathbf{PE}}$	1439,92	1023,79	1389,24	2158,21	-	-	1041,71
FR	100,0%	100,0%	100,0%	100,0%	100,0%	6,7%	10,0%
SR	100,0%	100,0%	100,0%	100,0%	0,0%	0,0%	10,0%

Tabela 6.20: Resultados obtidos para o problema g17, onde $f(\vec{x}^*) = 8853,5397$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	8853,5397[0]	8853,5397[0]	8853,5397[0]	8853,5397[0]	8927,6051[0]	8853,5397[0]	8853,5397[0]
$\mathbf{f_{pior}}$	8853,5397[0]	8853,5397[0]	8927,5976[0]	8936,6938[0]	8927,4547[2]	9281,4872[0]	8853,5397[0]
$\mathbf{f}_{\text{médio}}$	8853,5397	8853,5397	8900,4431	8856,3115	8947,6571	9049,2729	8853,5397
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	35,6881	14,9266	60,1870	168,8551	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0293	0,0000	0,0000
$\mathbf{PE_{melhor}}$	81120	80400	87600	99720	-	130680	159840
$\mathbf{PE_{pior}}$	91560	117960	-	-	-	-	266040
$\mathbf{PE}_{\text{médio}}$	86284	93412	119422	127812	-	130680	213316
$\sigma_{\mathbf{PE}}$	2476,67	8823,75	45364,35	35928,72	-	-	30730,62
FR	100,0%	100,0%	100,0%	100,0%	13,3%	100,0%	100,0%
SR	100,0%	100,0%	36,7%	96,7%	0,0%	3,3%	100,0%

Tabela 6.21: Resultados obtidos para o problema g18, onde $f(\vec{x}^*) = -0,8660$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-0,8660$[0]	$-0,8660$[0]	$-0,8660$[0]	$-0,8660$[0]	$-0,8658$[0]	$-0,8660$[0]	$-0,8660$[0]
$\mathbf{f_{pior}}$	$-0,8660$[0]	$-0,8660$[0]	$-0,8660$[0]	$-0,8660$[0]	$-0,8458$[0]	0,0000[8]	$-0,8660$[0]
$\mathbf{f}_{\text{médio}}$	$-0,8660$	$-0,8660$	$-0,8660$	$-0,8660$	$-0,8595$	$-0,7197$	$-0,8660$
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0000	0,0000	0,0045	0,3219	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	33,4881	0,0000
$\mathbf{PE_{melhor}}$	83360	80960	84320	74080	-	99600	161680
$\mathbf{PE_{pior}}$	102560	106080	106800	109360	-	-	249360
$\mathbf{PE}_{\text{médio}}$	90541	94387	96160	89128	-	109960	207432
$\sigma_{\mathbf{PE}}$	5294,71	7090,79	7073,56	10950,24	-	10360,00	19673,22
FR	100,0%	100,0%	100,0%	100,0%	100,0%	83,3%	100,0%
SR	100,0%	100,0%	100,0%	100,0%	0,0%	6,7%	100,0%

Tabela 6.22: Resultados obtidos para o problema g19, onde $f(\vec{x}^*) = 32,6556$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	32,6556[0]	32,6556[0]	32,6556[0]	32,6556[0]	39,6372[0]	33,3010[0]	32,6556[0]
$\mathbf{f_{pior}}$	32,6557[0]	32,6556[0]	32,6558[0]	32,6558[0]	95,9866[0]	49,8845[0]	32,6557[0]
$\mathbf{f}_{\text{médio}}$	32,6556	32,6556	32,6557	32,6556	73,8040	39,4752	32,6556
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0000	0,0001	0,0000	15,1251	5,6805	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
$\mathbf{PE_{melhor}}$	128000	167800	156200	175200	-	-	411000
$\mathbf{PE_{pior}}$	481100	311400	-	-	-	-	477300
$\mathbf{PE}_{\text{médio}}$	352843	247740	346844	291562	-	-	458410
$\sigma_{\mathbf{PE}}$	100031,37	42176,66	97290,28	70437,26	-	-	13751,42
FR	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
SR	100,0%	100,0%	60,0%	96,7%	0,0%	0,0%	100,0%

Tabela 6.23: Resultados obtidos para o problema g20, onde $f(\vec{x}^*) = 0,1474$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	0,1474[0]	0,1474[0]	0,1474[0]	0,1474[0]	0,1690[0]	0,0000[2]	0,0000[2]
$\mathbf{f_{pior}}$	0,1690[0]	0,1474[0]	0,1474[0]	0,2984[0]	0,2653[10]	0,6930[3]	0,0000[2]
$\mathbf{f}_{\text{médio}}$	0,1489	0,1474	0,1474	0,1589	0,1822	0,0231	0,0000
$\sigma_{\mathbf{f(x)}}$	0,0054	0,0000	0,0000	0,0291	0,0433	0,1244	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,0522	2,9600	2,6710
$\mathbf{PE_{melhor}}$	94080	92640	93840	91320	-	-	-
$\mathbf{PE_{pior}}$	-	145320	123000	-	-	-	-
$\mathbf{PE}_{\text{médio}}$	109787	113168	105548	137708	-	-	-
$\sigma_{\mathbf{PE}}$	14696,04	14152,03	7490,07	27264,80	-	-	-
FR	100,0%	100,0%	100,0%	100,0%	6,7%	0,0%	0,0%
SR	93,3%	100,0%	100,0%	76,7%	0,0%	0,0%	0,0%

Tabela 6.24: Resultados obtidos para o problema g21, onde $f(\vec{x}^*) = 193,7245$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	193,7245[0]	193,7245[0]	193,7245[0]	193,7245[0]	322,0782[2]	0,0000[2]	0,0000[2]
$\mathbf{f_{pior}}$	193,7245[0]	193,7546[0]	193,7245[0]	320,1136[1]	119,6469[2]	0,0000[1]	0,0000[2]
$\mathbf{f}_{\text{médio}}$	193,7245	193,7275	193,7245	225,3157	215,6787	0,0000	0,0000
$\sigma_{\mathbf{f(x)}}$	0,0000	0,0090	0,0000	50,4225	87,2635	0,0000	0,0000
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,0000	0,2397	0,7542	0,6930
$\mathbf{PE_{melhor}}$	155000	149400	173400	197600	-	-	-
$\mathbf{PE_{pior}}$	231000	-	243600	-	-	-	-
$\mathbf{PE}_{\text{médio}}$	198453	191207	208333	240181	-	-	-
$\sigma_{\mathbf{PE}}$	18939,13	25032,10	20428,50	17686,90	-	-	-
FR	100,0%	100,0%	100,0%	96,7%	0,0%	0,0%	0,0%
SR	100,0%	90,0%	100,0%	70,0%	0,0%	0,0%	0,0%

Tabela 6.25: Resultados obtidos para o problema g22, onde $f(\vec{x}^*) = 236,4310$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	485,3028[5]	6961,0595[4]	8099,0375[4]	18236,94[8]	2748,8373[19]	0,0000[11]	0,0000[20]
$\mathbf{f_{pior}}$	7664,0676[4]	5173,9028[13]	20000,00[5]	9252,9355[17]	2319,9403[20]	0,0000[10]	0,0000[20]
$\mathbf{f}_{\text{médio}}$	8393,3241	9094,8717	10558,48	6865,8616	3513,7771	0,0000	0,0000
$\sigma_{\mathbf{f(x)}}$	5832,3500	5917,5730	6656,5271	5237,1046	2453,6376	0,0000	0,0000
$\phi_{\text{médio}}$	23,1218	16,9056	19,9433	29,4687	24131,94	2392673	1812159
$\mathbf{PE_{melhor}}$	-	-	-	-	-	-	-
$\mathbf{PE_{pior}}$	-	-	-	-	-	-	-
$\mathbf{PE}_{\text{médio}}$	-	-	-	-	-	-	-
$\sigma_{\mathbf{PE}}$	-	-	-	-	-	-	-
FR	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
SR	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%

Tabela 6.26: Resultados obtidos para o problema g23, onde $f(\vec{x}^*) = -400,0551$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	−400,0551[0]	−400,0551[0]	−400,0551[0]	−400,0551[0]	−894,0980[2]	900,0000[0]	−400,0459[0]
$\mathbf{f_{pior}}$	−395,6020[0]	−384,2961[0]	−389,8207[0]	−2100,00[2]	−1423,88[2]	−2100,00[2]	0,0000[0]
$\mathbf{f}_{\text{médio}}$	−399,6397	−398,3794	−399,3298	−269,9103	−1232,84	−1970,00	−69,7428
$\sigma_{\mathbf{f(x)}}$	1,1077	3,8185	2,2658	392,6765	150,2299	540,4627	147,1744
$\phi_{\text{médio}}$	0,0000	0,0000	0,0000	0,1167	1,7824	3,3332	0,0000
$\mathbf{PE_{melhor}}$	288400	222600	242480	258860	-	-	-
$\mathbf{PE_{pior}}$	-	-	-	-	-	-	-
$\mathbf{PE}_{\text{médio}}$	378914	367115	393314	354360	-	-	-
$\sigma_{\mathbf{PE}}$	58054,74	62171,13	81717,15	52797,50	-	-	-
FR	100,0%	100,0%	100,0%	96,7%	0,0%	3,3%	100,0%
SR	63,3%	66,7%	60,0%	23,3%	0,0%	0,0%	0,0%

Tabela 6.27: Resultados obtidos para o problema g24, onde $f(\vec{x}^*) = -5,5080$.

Dados	**εMOES-R1**	**εMOES-R2**	**εMOES-R3**	**εMOES-FX**	**εGA**	**εPSO**	**εDE**
$\mathbf{f_{melhor}}$	$-5,5080[0]$	$-5,5080[0]$	$-5,5080[0]$	$-5,5080[0]$	$-5,5076[0]$	$-5,5080[0]$	$-5,5080[0]$
$\mathbf{f_{pior}}$	$-5,5080[0]$	$-5,5080[0]$	$-5,5080[0]$	$-5,5080[0]$	$-5,4987[0]$	$-5,5080[0]$	$-5,5080[0]$
$\mathbf{f}_{\text{médio}}$	$-5,5080$	$-5,5080$	$-5,5080$	$-5,5080$	$-5,5051$	$-5,5080$	$-5,5080$
$\sigma_{\mathbf{f(x)}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0024$	$0,0000$	$0,0000$
$\phi_{\text{médio}}$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$	$0,0000$
$\mathbf{PE_{melhor}}$	4880	4240	4480	5520	-	6320	4320
$\mathbf{PE_{pior}}$	6480	5680	6000	8320	-	9600	5520
$\mathbf{PE}_{\text{médio}}$	6019	5192	5349	7160	-	8061	5005
$\sigma_{\mathbf{PE}}$	$356,39$	$364,48$	$424,70$	$829,26$	-	$827,70$	$301,24$
FR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$
SR	$100,0\%$	$100,0\%$	$100,0\%$	$100,0\%$	$0,0\%$	$100,0\%$	$100,0\%$

Das tabelas apresentadas nesta seção, podemos dizer que algoritmo proposto εMOES (nas versões R1, R2, R3 e FX) foi capaz de encontrar os pontos de ótimo global para quase todos os problemas testados, não convergindo apenas para o problema g22. Para os problemas g02, g11, g17, g19, g20, g21 e g23 o algoritmo εMOES não convergiu para o ótimo global em todas as 30 execuções realizadas (casos onde a taxa SR é menor que 100%), mas obteve resultados superiores ao dos algoritmos εGA, εPSO e εDE.

O problema g22 foi considerado com sendo o mais difícil dos problemas testados. Nenhum dos sete algoritmos usados conseguiu encontrar o ponto de ótimo global deste problema. Observamos que para resolvê-lo usando o algoritmo εMOES necessitamos de uma população grande (com mais de 160 indivíduos) e um maior número de iterações (em torno de 650.000 avaliações do problema).

Observamos que o desempenho dos algoritmos εPSO e εDE, quando aplicados aos 24 problemas testes, é muito dependente do parâmetro NP (tamanho da população), onde para cada problema existe um número bom de indivíduos para resolvê-lo. Com o objetivo de comparar os resultados, usamos o mesmo número de indivíduos para todos os algoritmos (ver seção 6.3), o que prejudicou o desempenho dos algoritmos εPSO e εDE.

O bom desempenho do algoritmo εMOES pode ser observado nas taxas de factibilidade (FR) e sucesso (SR) obtidas. No geral, as versões R1, R2, R3 e FX do algoritmo εMOES apresentam um desempenho superior ao dos algoritmos εGA, εPSO e εDE. A Tabela 6.28 mostra uma comparação dos valores médios para estas taxas.

Em relação ao processo de busca pelo ótimo global, comparando os resultados obtidos para o algoritmo εMOES (nas versões R1, R2, R3 e FX) com os dos algoritmos εGA, εPSO e εDE, podemos concluir que o uso de múltiplos operadores torna o algoritmo evolutivo mais robusto e capaz de resolver um maior número de problemas.

Tabela 6.28: Comparação dos valores médios das taxas de factibilidade (FR) e sucesso (SR) para todos os algoritmos testados. O problema `g22` não foi considerado no cálculo.

Algoritmos	$\mathbf{FR}_{\text{médio}}$	$\mathbf{SR}_{\text{médio}}$
εMOES-R1	$\mathbf{100,00\%}$	$\mathbf{95,22\%}$
εMOES-R2	$\mathbf{100,00\%}$	$95,07\%$
εMOES-R3	$\mathbf{100,00\%}$	$90,43\%$
εMOES-FX	$99,71\%$	$90,22\%$
εGA	$74,20\%$	$8,12\%$
εPSO	$72,17\%$	$21,88\%$
εDE	$85,36\%$	$76,67\%$

Da Tabela 6.28, observamos que os algoritmos εMOES-R1, εMOES-R2 e εMOES-R3 obtiveram melhores resultados no processo de busca pela região factível, gerando soluções factíveis em todas as 30 execuções realizadas para cada um dos problema testados. Contudo, o valor de $FR_{\text{médio}}$ para o algoritmo εMOES-FX (εMOES com probabilidades fixas) é quase 100%, indicando que fazer a adaptação das probabilidades dos operadores não gera grandes melhorias no processo de busca por soluções factíveis.

Em relação a taxa de sucesso das soluções obtidas (SR), podemos observar na Tabela 6.28 que a versão εMOES-R1 (critério I_{local}) obteve os melhores resultados. Contudo, quando comparamos os algoritmos εMOES-R1 e εMOES-R2 (critérios I_{local} e I_{global}, respectivamente) observamos que seus resultados estão muito próximos, o que indica que os dois critérios de cálculo de produtividade geram resultados semelhantes. No geral, os resultados obtidos com as quatros versões do algoritmo εMOES foram superiores aos dos outros algoritmos testados (εGA, εPSO e εDE).

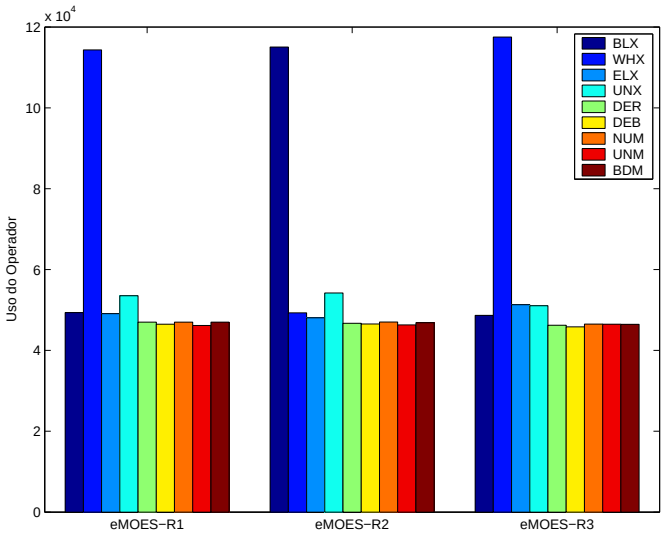
Em relação a velocidade de convergência dos algoritmos testados, podemos observar nas tabelas apresentadas acima que o algoritmo εMOES (nas versões R1, R2, R3 e FX) gerou os melhores resultados para todos os 24 problemas, ou seja, convergiu com um menor número de avaliações do problema (PE). Isto demostra a eficiência do algoritmo εMOES, quando comparado com os algoritmos εGA, εPSO e εDE.

O Anexo B apresenta 24 gráficos de convergência (um para cada problema) comparando os sete algoritmos testados. Estes resultados foram obtidos com a melhor das 30 execuções realizadas. Analisando estes gráficos podemos concluir que o uso do algoritmo de adaptação de probabilidades diminui o tempo necessário para a convergência do algoritmo: o algoritmo realiza menos avaliações do problema. Esta seria a grande vantagem de se adaptar as probabilidades de seleção dos operadores, pois os operadores que geram melhores resultados são usados com maior frequência. Contudo, observamos que se os parâmetros α e β forem mal ajustados

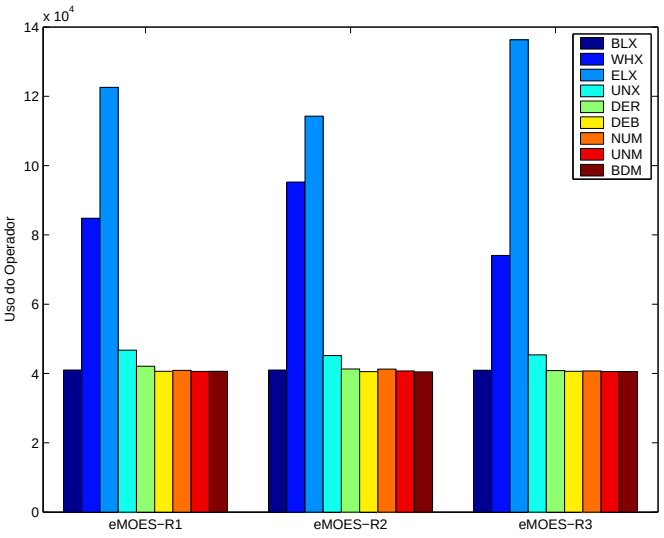
o algoritmo εMOES pode se tornar instável (reduz as taxas SR e FR do algoritmo).

6.6 Uso dos Operadores

Neste seção apresentamos 24 gráficos de barras mostrando o uso dos operadores no processo de otimização (quantas vezes o operador foi executado pelo sistema). Cada gráfico possui três conjunto de nove barras cada, um para cada algoritmo adaptativo: εMOES-R1, εMOES-R2 e εMOES-R3. Os valores mostrados são relativos a melhor das 30 execuções realizadas.

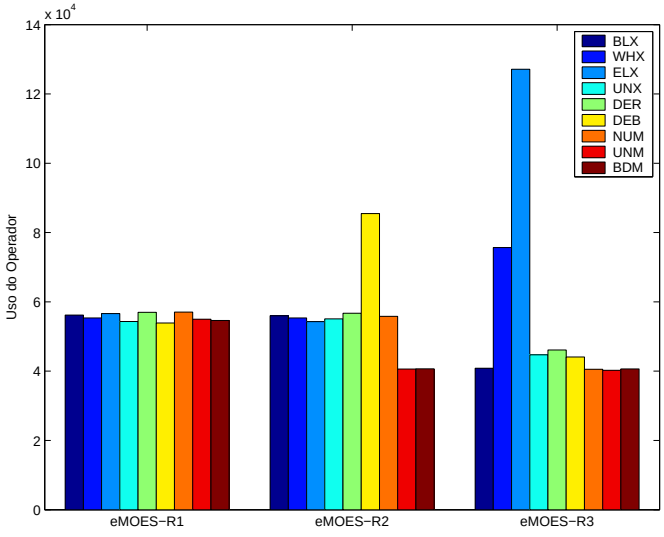


(a) Problema g01

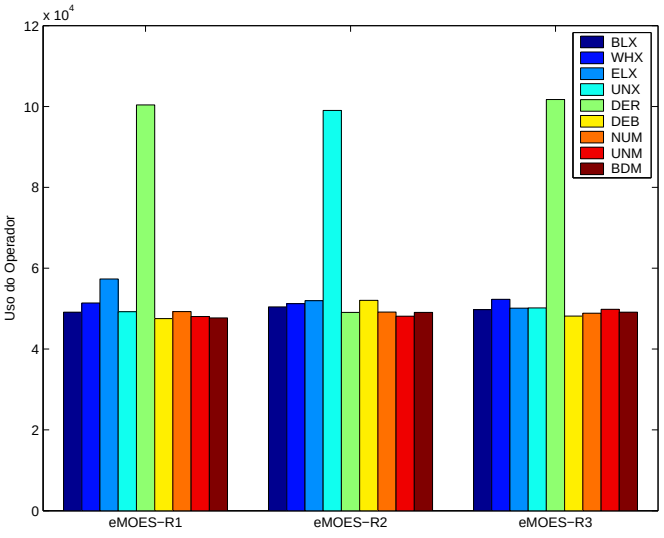


(b) Problema g02

Figura 6.1: Gráficos com o uso dos operadores evolutivos para os problemas g01 e g02.

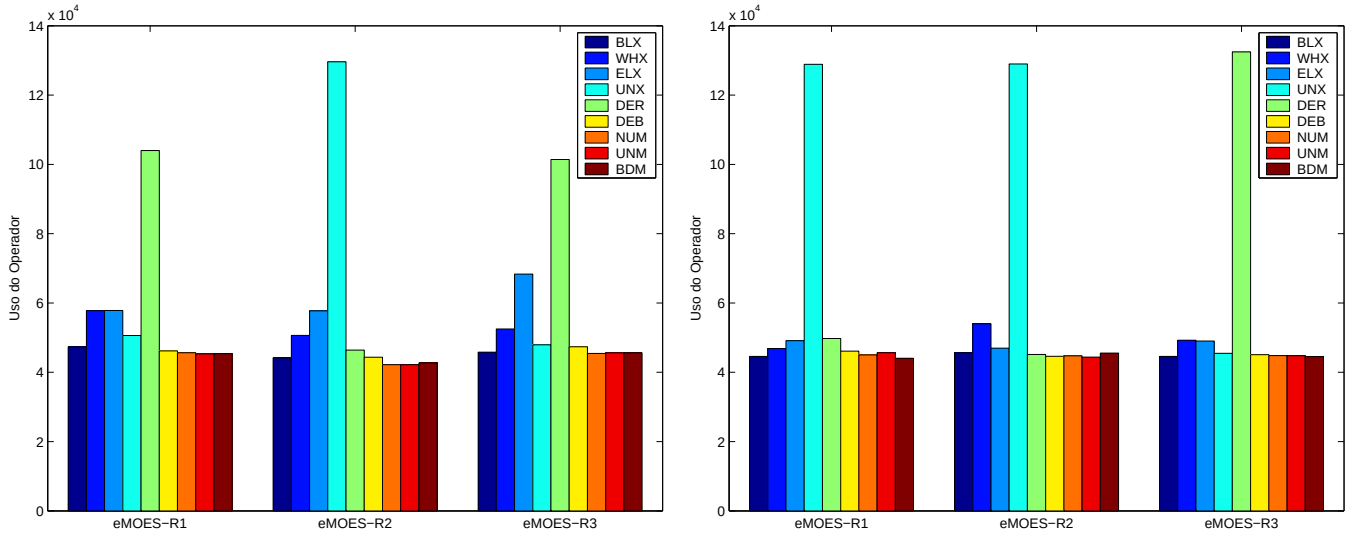


(a) Problema g03



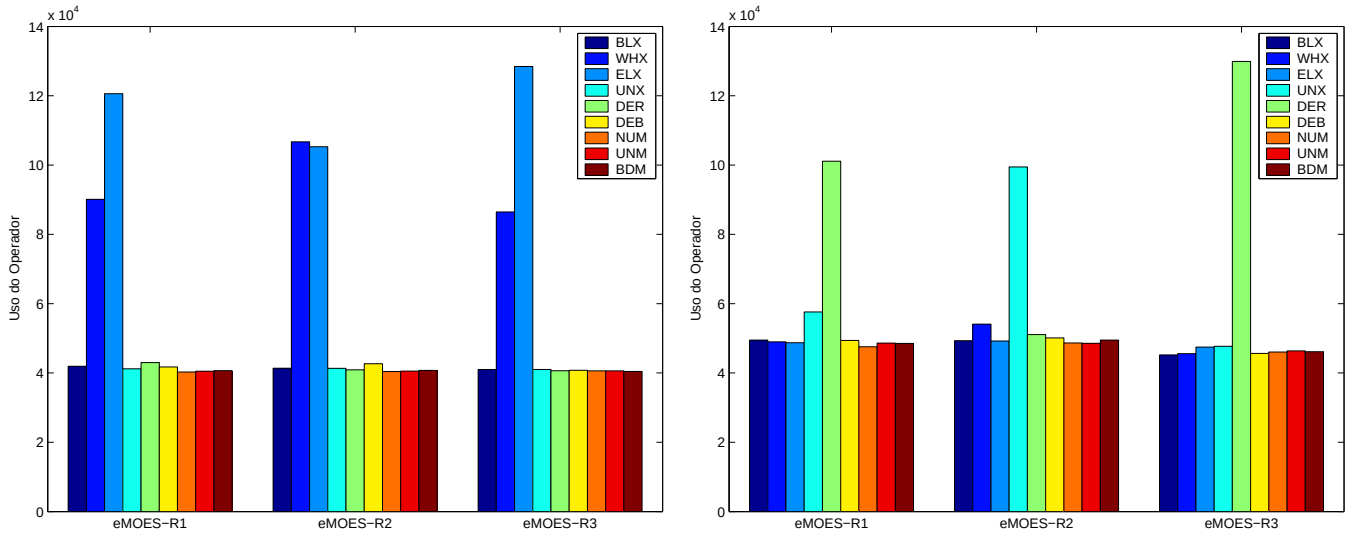
(b) Problema g04

Figura 6.2: Gráficos com o uso dos operadores evolutivos para os problemas g03 e g04.



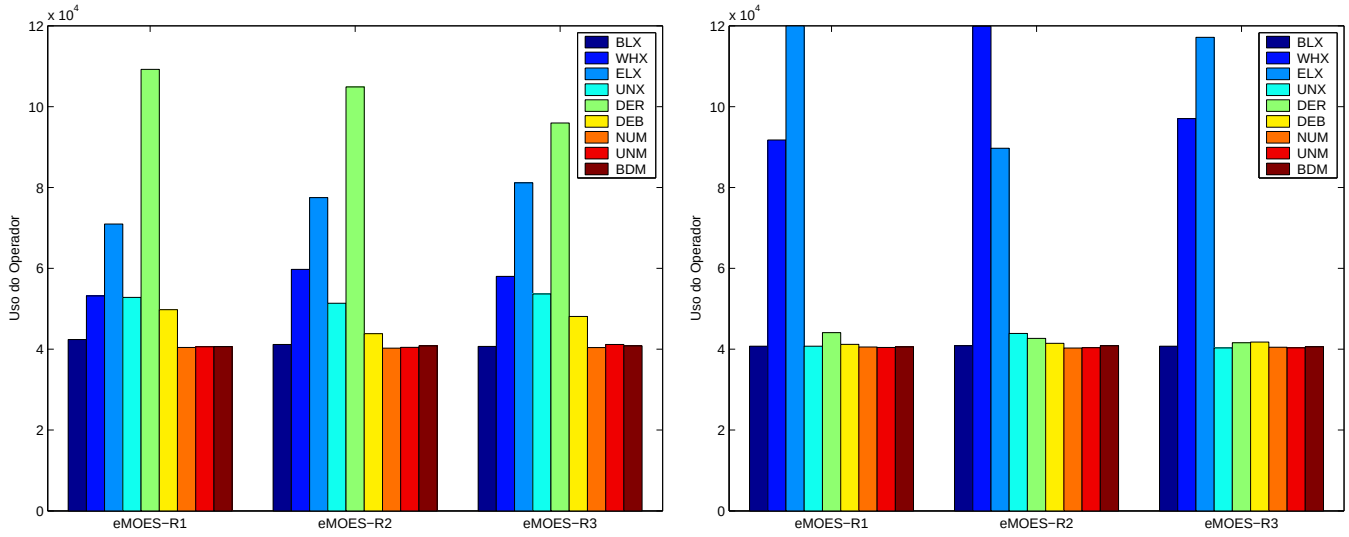
(a) Problema g05 (b) Problema g06

Figura 6.3: Gráficos com o uso dos operadores evolutivos para os problemas g05 e g06.



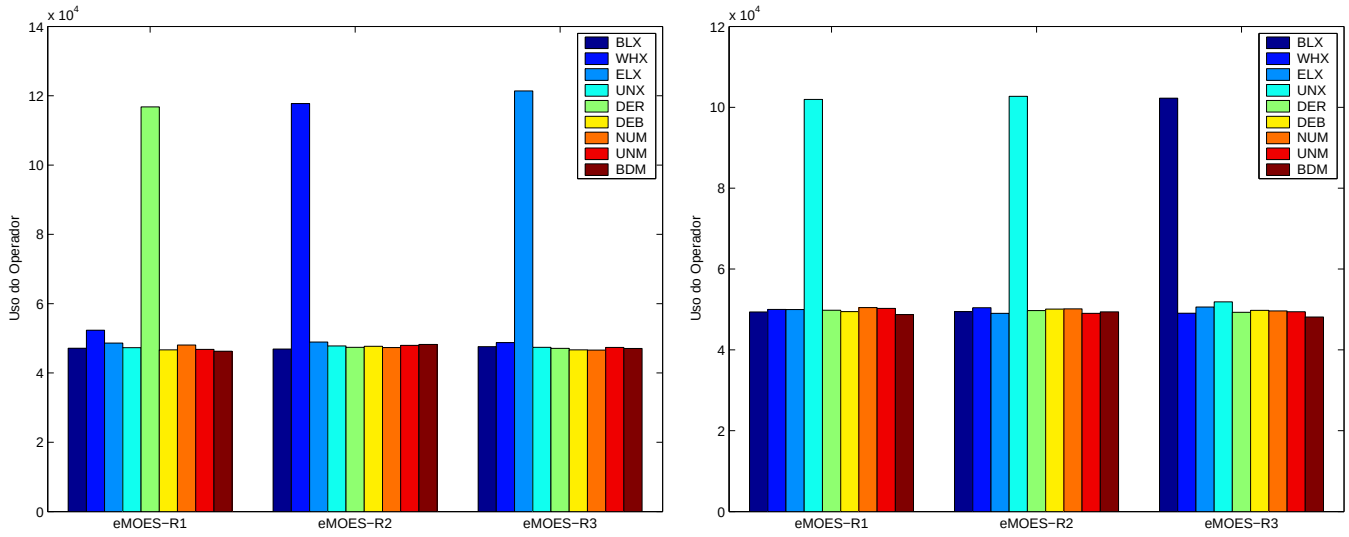
(a) Problema g07 (b) Problema g08

Figura 6.4: Gráficos com o uso dos operadores evolutivos para os problemas g07 e g08.



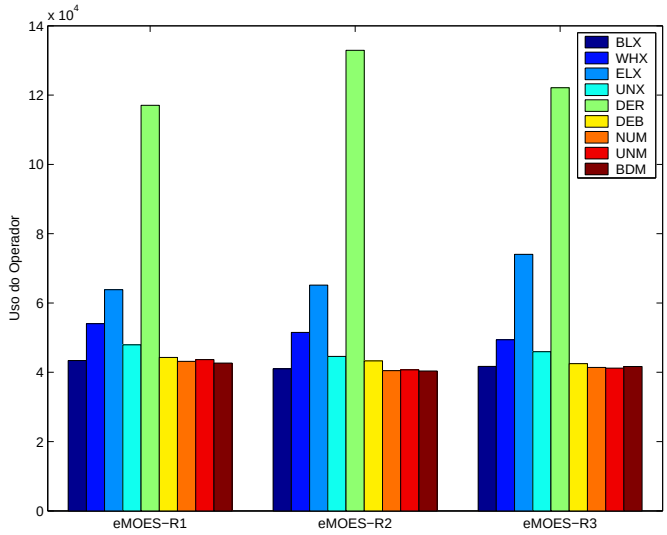
(a) Problema g09 (b) Problema g10

Figura 6.5: Gráficos com o uso dos operadores evolutivos para os problemas g09 e g10.

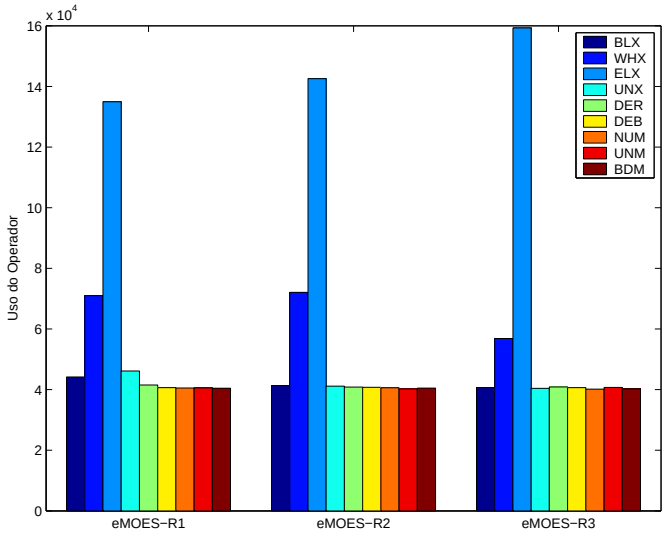


(a) Problema g11 (b) Problema g12

Figura 6.6: Gráficos com o uso dos operadores evolutivos para os problemas g11 e g12.

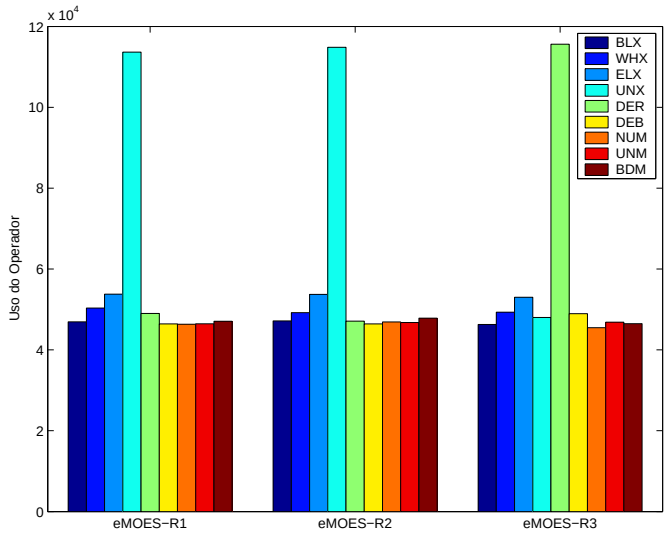


(a) Problema g13

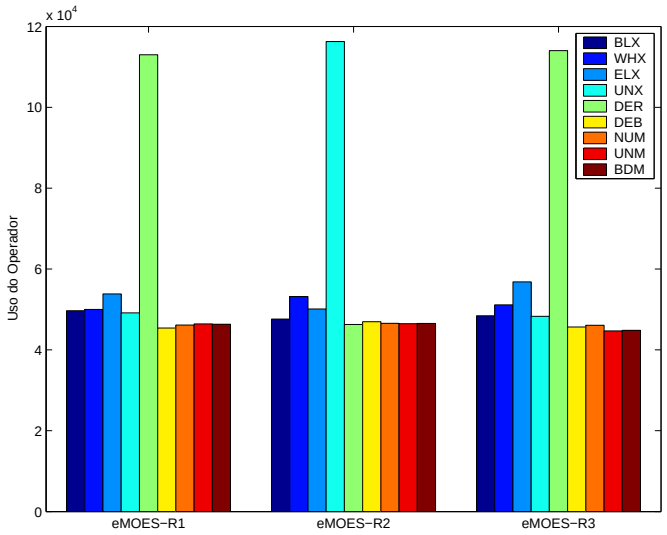


(b) Problema g14

Figura 6.7: Gráficos com o uso dos operadores evolutivos para os problemas g13 e g14.

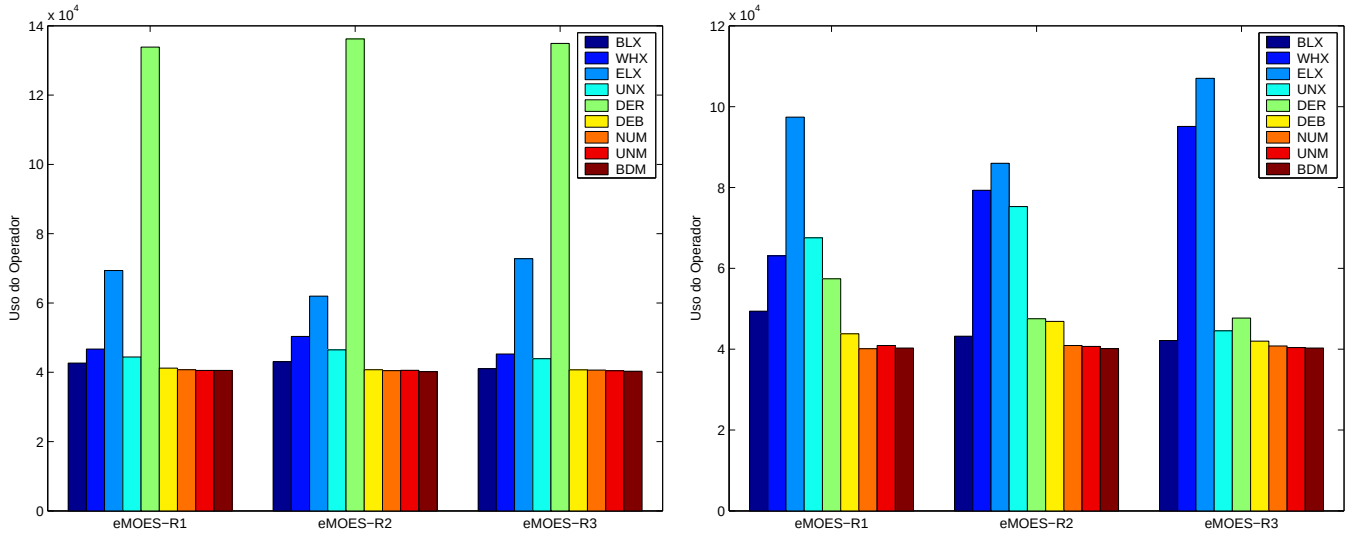


(a) Problema g15



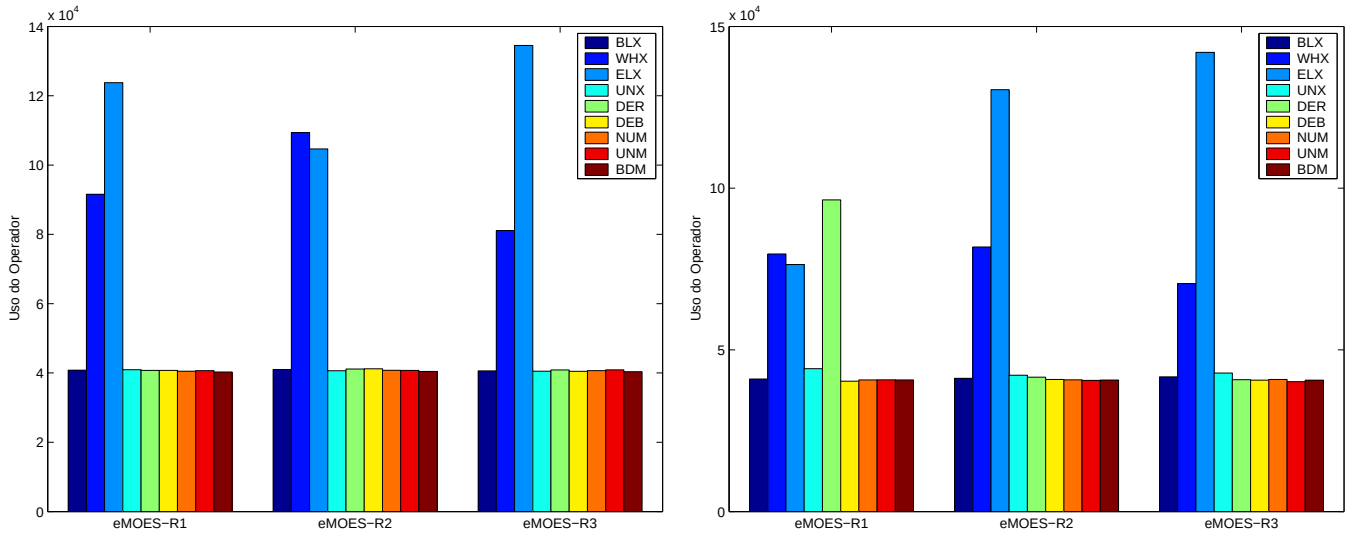
(b) Problema g16

Figura 6.8: Gráficos com o uso dos operadores evolutivos para os problemas g15 e g16.



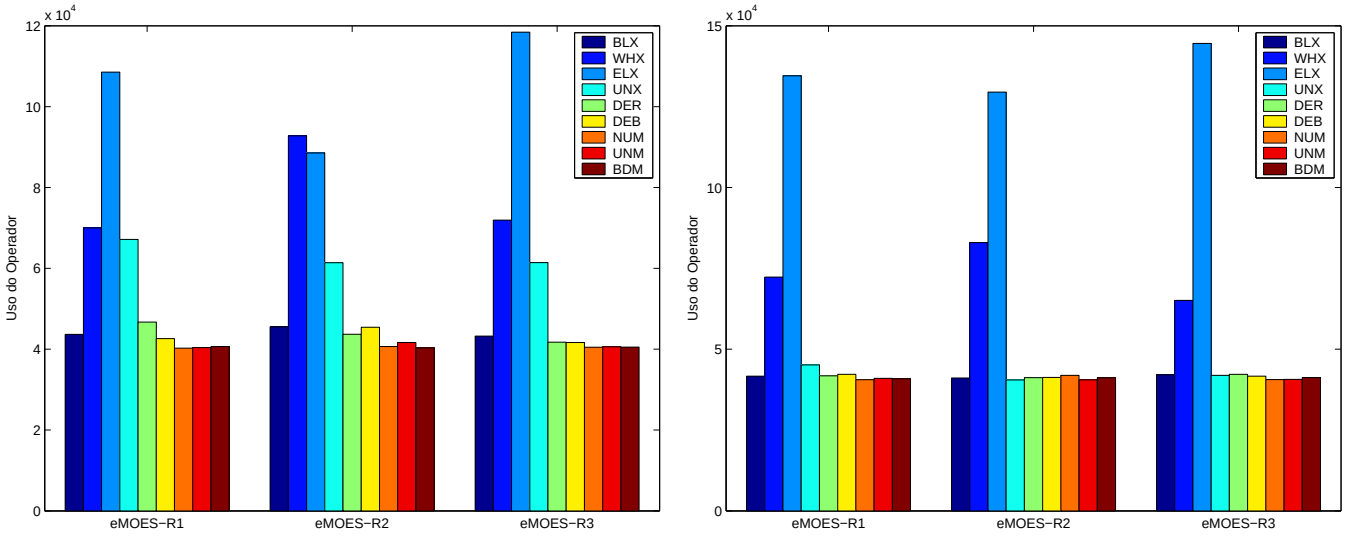
(a) Problema g17 (b) Problema g18

Figura 6.9: Gráficos com o uso dos operadores evolutivos para os problemas g17 e g18.



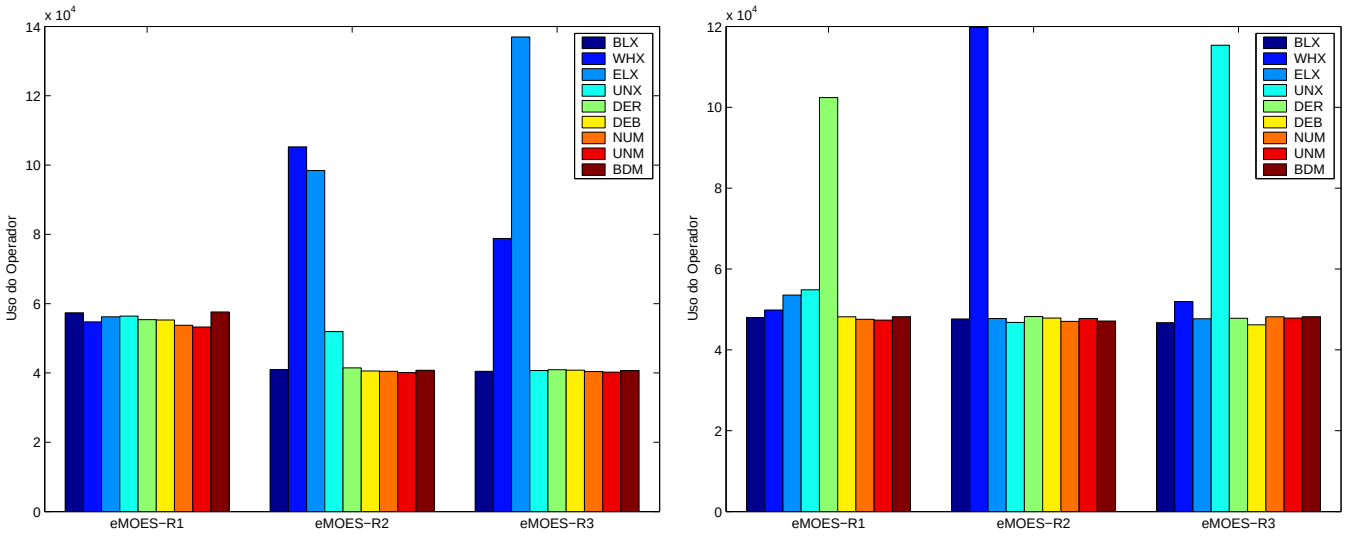
(a) Problema g19 (b) Problema g20

Figura 6.10: Gráficos com o uso dos operadores evolutivos para os problemas g19 e g20.



(a) Problema g21 (b) Problema g22

Figura 6.11: Gráficos com o uso dos operadores evolutivos para os problemas g21 e g22.



(a) Problema g23 (b) Problema g24

Figura 6.12: Gráficos com o uso dos operadores evolutivos para os problemas g23 e g24.

6.7 Análise de Complexidade

A complexidade dos algoritmos testados será avaliada através dos seguintes parâmetros: $T1$, $T2$ e $(T2-T1)/T1$. Sendo assim, temos que:

$$T1 = \frac{1}{24}\sum_{i=1}^{24} tp_i \tag{6.3}$$

onde tp_i é o tempo computacional necessário para se realizar 10000 avaliações do problema i. Este parâmetro será igual para todos os algoritmos, pois eles compartilham a mesma implementação dos problemas.

$$T2 = \frac{1}{24}\sum_{i=1}^{24} ta_i \tag{6.4}$$

onde ta_i é o tempo computacional total que o algoritmo requer para resolver o problema i com no máximo 10000 avaliações.

Na Tabela 6.29 podemos ver que os algoritmos εMOES possuem uma maior complexidade computacional que os algoritmos com probabilidades estática (εGA, εDE e εPSO). Isso se deve ao algoritmo de adaptação de probabilidades. No geral, o εDE é o algoritmo com a menor complexidade computacional e o εMOES-R3 é o algoritmo com a maior complexidade computacional.

Tabela 6.29: Complexidade computacional dos Algoritmos Testados.

Algoritmo	$T1$	$T2$	$(T2-T1)/T1$
εMOES-R1	515	922	0,79
εMOES-R2	515	938	0,82
εMOES-R3	515	1031	1,00
εMOES-FX	515	895	0,74
εGA	515	891	0,73
εPSO	515	875	0,70
εDE	515	734	0,43

7 Conclusões

Neste trabalho, foi proposto um novo algoritmo de otimização baseado nos conceitos dos algoritmos evolutivos adaptativos e usando múltiplos operadores de busca. Denominado por εMOES, o algoritmo usa nove operadores de busca que foram selecionados de forma a aumentar a quantidade de classes de problemas que o algoritmo é capaz de resolver. Com objetivo de aumentar a eficiência do algoritmo, o método *Adaptive Pursuit* (THIERENS, 2005) foi usado para adaptar as probabilidades de seleção dos operadores. As quatro versões do εMOES foram aplicadas aos 24 problemas testes propostos para a CEC'2006 (LIANG et al., 2006).

Tendo em vista o Teorema *No Free Lunch* (NFL) (WOLPERT; MACREADY, 1997), se para cada classe de problema existe uma boa heurística para resolvê-lo, então um algoritmo que incorpore diversas heurísticas (através do uso de vários operadores de busca) e consiga selecionar de forma automática qual delas é a mais adequada para resolver um determinado problema será capaz de resolver um maior número de classes de problemas de otimização.

Analisando os resultados apresentados no Capítulo 6, podemos dizer que o algoritmo de otimização global εMOES mostrou ser bastante eficiente, gerando bons resultados para quase todos os problemas testados (os algoritmos não convergiram apenas para o problema `g22`). Assim, observamos que o uso de múltiplos operadores de busca em um único algoritmo evolutivo torna o processo de busca mais robusto e aumenta a gama de problemas que este algoritmo é capaz de resolver. Contudo, o processo adaptativo aumenta a complexidade computacional do algoritmo (ver Tabela 6.29).

No geral, comparando os resultados obtidos com as quatro versões do εMOES (εMOES-R1, εMOES-R2, εMOES-R3 e εMOES-FX), adaptar as probabilidades dos operadores não representou uma vantagem significativa sobre não adaptar. O grande impacto do método adaptativo usado no algoritmo εMOES se dá na velocidade de convergência, pois gerou um algoritmo capaz de convergir mais rapidamente (ver os gráficos de convergência no Anexo B). Contudo, observamos que as estratégias usadas no cálculo de produtividade dos operadores (I_{local}, I_{global} e I_{rank}) geram resultados semelhantes.

Outros autores (WHITACRE, 2007; BARBOSA; Sá, 2000; XIAO et al., 1997), testando outras técnicas de adaptação de parâmetros, observaram melhoras no processo de busca, tanto em relação à velocidade de convergência quanto no desempenho do algoritmo (aumento da qualidade da solução final e diminuição da dispersão dos resultados).

Sabe-se que procurar probabilidades iniciais fixas eficientes para os operadores de busca é um problema cuja complexidade cresce muito com o aumento do número de operadores, onde para cada problema teremos um conjunto ótimo de probabilidades. Sendo assim, o algoritmo εMOES libera o usuário do trabalho de ter que encontrar os valores ótimos de probabilidade para um determinado problema.

O método de manipulação de restrições ε-*constrained*, quando incorporado ao algoritmo MOES, mostrou ser uma forma simples, eficiente e robusta de se otimizar problemas com restrições. Dos resultados podemos concluir que este método foi capaz de convergir a população para a região de factibilidade em todos os problemas testados. A estratégia usada para controle do parâmetro ε-*level* é um dos motivos para os bons resultados encontrados, visto que o relaxamento das restrições dos problemas altamente restritos facilita o processo de busca de soluções factíveis.

Como proposta para trabalhos futuros podemos destacar os seguintes pontos de interesse:

- O conjunto de operadores de busca pode ser melhorado pela adição de novos operadores. Uma maior variedade de heurísticas de busca em um mesmo algoritmo aumenta a robustez do processo de busca pela solução ótima.
- Uma analise do comportamento de cada operador seria uma forma de detectar operadores com comportamentos semelhantes. Retirar operadores redundantes pode melhorar o desempenho do algoritmo.
- A metodologia usada para fazer a adaptação das probabilidades pode ser melhorada através da pesquisa de novas técnicas de adaptação.
- Com os resultados obtidos neste trabalho, não fica claro que usar apenas o ganho de aptidão da população basta para quantificar a qualidade dos operadores.
- Aumentar a quantidades de problemas testes de forma a avaliar o desempenho do algoritmo em outras classes de problemas. Uma ideia é aplicar o algoritmo εMOES em problemas de Fluxo Ótimo de Potência.

Referências Bibliográficas

BARBOSA, H.; Sá, A. On adaptive operator probabilities in real coded genetic algorithms. In: CITESEER. *Proc. XX International Conference of the Chilean Computer Science Society*. [S.l.], 2000.

CASTRO, L. D.; ZUBEN, F. V. et al. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, Citeseer, v. 6, n. 3, p. 239–251, 2002.

EIBEN, A. et al. Parameter control in evolutionary algorithms. *Intelligence (SCI)*, Springer, v. 54, p. 19–46, 2007.

FIALHO, A. et al. *Exploração de relações entre as técnicas nebulosas e evolutivas da inteligência computacional.* [S.l.]: Biblioteca Digital de Teses e Dissertações da USP, 2007.

HANSEN, N.; OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, MIT Press, v. 9, n. 2, p. 159–195, 2001.

HOLTZ, G. *Traçado Automático de Envoltórias de Esforços em Estruturas Planas Utilizando um Algoritmo Evolucionário*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, 2005.

HONORIO, L. et al. Intelligent optimal power flow system development using aspect-oriented modeling. *IEEE Transactions on Power Systems*, v. 22, n. 4, p. 1826–1834, 2007.

HONORIO, L.; SILVA, A. Leite da; BARBOSA, D. A gradient-based artificial immune system applied to optimal power flow problems. In: *ICARIS*. [S.l.: s.n.], 2007. p. 1–12.

LIANG, J. et al. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. *Nanyang Technological University, Tech. Rep*, 2006.

MICHALEWICZ T. LOGAN, F. L. Z.; SWAMINATHAN, S. Evolutionary operators for continuous convex parameter spaces. In: *Proceedings of the 3rd Annual conference on Evolutionary Programming*. [S.l.: s.n.], 1994. p. 84–97.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. [S.l.]: Springer, 1996.

MUHLENBEIN, H.; PAASS, G. From recombination of genes to the estimation of distributions: I. binary parameters. *Lecture notes in computer science*, Citeseer, v. 1141, p. 178–187, 1996.

OLIVEIRA, G. *Estudo e Aplicações da Evolução Diferencial*. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2006.

Sá, A.; BARBOSA, H. Adaptação das probabilidades dos operadores genéticos: um problema de decisão. *Anais do V seminário sobre Elementos Finitos e Métodos Numéricos em Engenharia*, UFJF - Juiz de Fora/MG, p. 31–44, 1998.

STORN, R.; PRICE, K. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *INTERNATIONAL COMPUTER SCIENCE INSTITUTE-PUBLICATIONS-TR*, Citeseer, 1995.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by alpha constrained genetic algorithm. *Systems and Computers in Japan*, v. 35, n. 5, p. 11–22, 2004.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by applying the alpha constrained method to the nonlinear simplex method with mutations. *IEEE Trans. Evolutionary Computation*, v. 9, n. 5, p. 437–451, 2005.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by the alpha constrained particle swarm optimizer. *JACIII*, v. 9, n. 3, p. 282–289, 2005.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. [S.l.: s.n.], 2006. p. 308–315.

TAKAHAMA, T.; SAKAI, S. Constrained Optimization by the ε Constrained Genetic Algorithm. *Transactions*, v. 47, n. 6, p. 1861–1871, 2006.

TAKAHAMA, T.; SAKAI, S. Solving Constrained Optimization Problems by the ε Constrained Particle Swarm Optimizer with Adaptive Velocity Limit Control. In: *2006 IEEE Conference on Cybernetics and Intelligent Systems*. [S.l.: s.n.], 2006. p. 1–7.

TAKAHAMA, T.; SAKAI, S. Solving Difficult Constrained Optimization Problems by the ε Constrained Differential Evolution with Gradient-Based Mutation. *Constraint-handling in Evolutionary Optimization*, Springer, p. 51, 2009.

THATHACHAR, M.; SASTRY, P. A class of rapidly converging algorithms for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, v. 15, p. 168–175, 1985.

THIERENS, D. An adaptive pursuit strategy for allocating operator probabilities. In: ACM. *Proceedings of the 2005 conference on Genetic and evolutionary computation*. [S.l.], 2005. p. 1546.

VERMAAS, L. et al. Learning Fuzzy Systems by a Co-Evolutionary Artificial-Immune-Based Algorithm. In: SPRINGER. *Proceedings of the 8th International Workshop on Fuzzy Logic and Applications*. [S.l.], 2009. p. 319.

WEISE, T. *Global Optimization Algorithms - Theory and Application*. [s.n.], 2008. Disponível em: <http://www.it-weise.de/>.

WHITACRE, J. *Adaptation and Self-Organization in Evolutionary Algorithms*. Tese (Doutorado) — The University of New South Wales, 2007.

WOLPERT, D.; MACREADY, W. No Free Lunch Theorems for Optimization. *IEEE transactions on evolutionary computation*, Citeseer, v. 1, n. 1, p. 67–82, 1997.

WRIGHT, A. Genetic algorithms for real parameter optimization. *Foundations of genetic algorithms*, San Francisco: Morgan Kaufmann, v. 1, p. 205–218, 1991.

XIAO, J. et al. Adaptive Evolutionary Planner/Navigator for Mobile Robots. *IEEE transactions on evolutionary computation*, Citeseer, v. 1, n. 1, p. 18–28, 1997.

ANEXO A – Problemas Testes

Neste anexo são descritos 24 problemas testes de otimização restrita. Estes problemas podem ser encontrados em (www.ntu.edu.sg/home/EPNSugan) e foram utilizados para realização de testes de desempenho no algoritmo proposto. Além das descrições, apresentam-se as melhores soluções já encontradas. Na realização das simulações, eles foram transformados para o seguinte formato:

$$\textit{Minimizar} \quad f(\vec{x}),\ \vec{x} = [x_1, x_2, \ldots, x_n]$$

Sujeito a:

$$g_i(\vec{x}) \leq 0,\ i = 1, \ldots, q$$
$$h_j(\vec{x}) = 0,\ j = q+1, \ldots, m$$

Neste trabalho, as restrições de igualdade foram transformadas em inequações com o seguinte formato:

$$|h_j(\vec{x})| - \xi \leq 0,\ \textit{para}\ j = q+1, \ldots, m$$

Assim, uma solução $\vec{x}$ é considerada **factível** se $g_i(\vec{x}) \leq 0$, para $j = 1, \ldots, q$ e $|h_j(\vec{x})| - \xi \leq 0$, para $j = q+1, \ldots, m$. Em todos os problemas, ξ foi ajustado para $0,0001$.

Problema `g01`

Minimizar:

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{A.1}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(\vec{x}) &= -8x_1 + x_{10} \leq 0 \\
g_5(\vec{x}) &= -8x_2 + x_{11} \leq 0 \\
g_6(\vec{x}) &= -8x_3 + x_{12} \leq 0 \\
g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned} \tag{A.2}$$

onde os limites são $0 \leq x_i \leq 1$ ($i = 1, \ldots, 9$), $0 \leq x_i \leq 100$ ($i = 10, 11, 12$) e $0 \leq x_{13} \leq 1$. O mínimo global está em $\vec{x}^*$ = (1,1,1,1,1,1,1,1,1,3,3,3,1), onde seis restrições estão ativas (g_1, g_2, g_3, g_7, g_8 e g_9) e $f(\vec{x}^*) = -15$.

Problema `g02`

Minimizar:

$$f(\vec{x}) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{A.3}$$

Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\ g_2(\vec{x}) &= \sum_{i=1}^{n} x_i - 7.5n \leq 0 \end{aligned} \tag{A.4}$$

onde $n = 20$ e $0 < x_i \leq 10$ ($i = 1, \ldots, n$). O mínimo global está em $\vec{x}^*$ = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865, 0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317); o melhor valor encontrado é $f(\vec{x}^*) = -0.80361900$ e a restrição g_1 está próxima de ser ativada.

Problema `g03`

Minimizar:

$$f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i \tag{A.5}$$

Sujeito a:

$$h_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \tag{A.6}$$

onde $n = 10$ e $0 \leq x_i \leq 1$ ($i = 1, \ldots, n$). O mínimo global está em $\vec{x}^*$ = (0.31624357647283069, 0.316243577414338339, 0.316243578012345927, 0.316243575664017895, 0.316243578205526066, 0.31624357738855069, 0.316243575472949512, 0.316243577164883938, 0.316243578155920302, 0.316243576147374916) onde $f(\vec{x}^*)$ = -1.00050010001000.

Problema `g04`

Minimizar:

$$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{A.7}$$

Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\ g_2(\vec{x}) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\ g_3(\vec{x}) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\ g_4(\vec{x}) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\ g_5(\vec{x}) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\ g_6(\vec{x}) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0 \end{aligned} \tag{A.8}$$

onde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ e $27 \leq x_i \leq 45$ ($i = 3,4,5$). O mínimo global está em $\vec{x}^*$ = 78, 33, 29.9952560256815985, 45, 36.7758129057882073) onde $f(\vec{x}^*)$ = -3.066553867178332e+04. Duas restrições estão ativas (g_1 e g_6).

Problema g05

Minimizar:

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3 \tag{A.9}$$

Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= -x_4 + x_3 - 0.55 \leq 0 \\ g_2(\vec{x}) &= -x_3 + x_4 - 0.55 \leq 0 \\ h_3(\vec{x}) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\ h_4(\vec{x}) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\ h_5(\vec{x}) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \end{aligned} \tag{A.10}$$

onde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ e $-0.55 \leq x_4 \leq 0.55$. O mínimo global está em $\vec{x}^*$ = (679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826) onde $f(\vec{x}^*)$ = 5126.4967140071.

Problema g06

Minimizar:

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{A.11}$$

Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\ g_2(\vec{x}) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \end{aligned} \tag{A.12}$$

onde $13 \leq x_1 \leq 100$ e $0 \leq x_2 \leq 100$. O mínimo global está em $\vec{x}^*$ = (14.09500000000000064, 0.8429607892154795668) onde $f(\vec{x}^*)$ = -6961.81387558015. As duas restrições, g_1 e g_2, estão ativadas.

Problema g07

Minimizar:

$$\begin{aligned} f(\vec{x}) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \end{aligned} \tag{A.13}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\
g_2(\vec{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
g_3(\vec{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
g_4(\vec{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
g_5(\vec{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
g_6(\vec{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
g_7(\vec{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
g_8(\vec{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0
\end{aligned} \tag{A.14}$$

onde $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. O mínimo global está em $\vec{x}^*$ = (2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347) onde $f(\vec{x}^*)$ = 24.30620906818. Seis restrições estão ativas (g_1, g_2, g_3, g_4, g_5 e g_6).

Problema g08

Minimizar:

$$f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \tag{A.15}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= x_1^2 - x_2 + 1 \leq 0 \\
g_2(\vec{x}) &= 1 - x_1 + (x_2 - 4)^2 \leq 0
\end{aligned} \tag{A.16}$$

onde $0 \leq x_1 \leq 10$ e $0 \leq x_2 \leq 10$. O mínimo global está em $\vec{x}^*$ = (1.22797135260752599, 4.24537336612274885) onde $f(\vec{x}^*)$ = -0.0958250414180359.

Problema g09

Minimizar:

$$\begin{aligned}
f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
+ 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7
\end{aligned} \tag{A.17}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\
g_2(\vec{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\
g_3(\vec{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\
g_4(\vec{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{aligned} \tag{A.18}$$

onde $-10 \leq x_i \leq 10$ para $(i = 1, \ldots, 7)$. O mínimo global está em $\vec{x}^*$ = (2.33049935147405174, 1.95137236847114592, -0.477541399510615805, 4.36572624923625874, -0.624486959100388983, 1.03813099410962173, 1.5942266780671519) onde $f(\vec{x}^*)$ = 680.630057374402. Duas restrições estão ativas (g_1 e g_4).

Problema `g10`

Minimizar:

$$f(\vec{x}) = x_1 + x_2 + x_3 \tag{A.19}$$

Sujeito a:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(\vec{x}) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0$$
$$g_5(\vec{x}) = -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0$$
$$g_6(\vec{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0$$

onde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ ($i = 2,3$) e $10 \leq x_i \leq 1000$ ($i = 4,\ldots,8$). O mínimo global está em $\vec{x}^*$ = (579.306685017979589, 1359.97067807935605, 5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852, 395.601173702746735), onde $f(\vec{x}^*)$ = 7049.24802052867.Todas as restrições estão ativas (g_1, g_2 e g_3).

Problema `g11`

Minimizar:

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2 \tag{A.20}$$

Sujeito a:

$$h(\vec{x}) = x_2 - x_1^2 = 0 \tag{A.21}$$

onde $-1 \leq x_1 \leq 1$ e $-1 \leq x_2 \leq 1$. O mínimo global está em $\vec{x}^*$ = (-0.707036070037170616, 0.500000004333606807) onde $f(\vec{x}^*)$ = 0.7499.

Problema `g12`

Minimizar:

$$f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \tag{A.22}$$

Sujeito a:

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

onde $0 \leq x_i \leq 10$ ($i = 1,2,3$) e $p,q,r = 1,2,\ldots,9$. A região factível do espaço de busca consiste de 9^3 esferas disjuntas. Um ponto (x_1,x_2,x_3) é um ponto factível se e somente se existe p,q,r para os quais a inequação acima seja verdadeira. O mínimo global está em $\vec{x}^* = (5,5,5)$ onde $f(\vec{x}^*) = -1$.

Problema `g13`

Minimizar:

$$f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5} \tag{A.23}$$

Sujeito a:

$$\begin{aligned} h_1(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2(\vec{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\ h_3(\vec{x}) &= x_1^3 + x_2^3 + 1 = 0 \end{aligned} \tag{A.24}$$

onde $-2.3 \leq x_i \leq 2.3$ ($i = 1,2$) e $-3.2 \leq x_i \leq 3.2$ ($i = 3,4,5$). O mínimo global está em $\vec{x}^*$ = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498) onde $f(\vec{x}^*) = 0.053941514041898$.

Problema `g14`

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right) \tag{A.25}$$

Sujeito a:

$$\begin{aligned} h_1(\vec{x}) &= x_1 + 2x_2 + +2x_3 + x_6 + x_{10} - 2 = 0 \\ h_2(\vec{x}) &= x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\ h_3(\vec{x}) &= x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \end{aligned} \tag{A.26}$$

onde $0 < x_i \leq 10$ ($i = 1, \ldots, 10$), e $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$. O mínimo global está em $\vec{x}^*$ = (0.0406684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845) onde $f(x^*)$ = -47.7648884594915.

Problema `g15`

Minimizar:

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3 \tag{A.27}$$

Sujeito a:

$$\begin{aligned} h_1(\vec{x}) &= x_1^2 + x_2^2 + +x_3^2 - 25 = 0 \\ h_2(\vec{x}) &= 8x_1 + 14x_2 + 7x_3 - 56 = 0 \end{aligned} \tag{A.28}$$

onde $0 \leq x_i \leq 10$ ($i = 1,2,3$). O mínimo global está em $\vec{x}^*$ = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921) onde $f(x^*) = 961.715022289961$.

Problema `g16`

Minimizar:

$$f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17} \tag{A.29}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= \frac{0.28}{0.72}y_5 - y_4 \leq 0 \\
g_2(\vec{x}) &= x_3 - 1.5x_2 \leq 0 \\
g_3(\vec{x}) &= 3496\frac{y_2}{c_{12}} - 21 \leq 0 \\
g_4(\vec{x}) &= 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0 \\
g_5(\vec{x}) &= 213.1 - y_1 \leq 0 \\
g_6(\vec{x}) &= y_1 - 405.23 \leq 0 \\
g_7(\vec{x}) &= 17.505 - y_2 \leq 0 \\
g_8(\vec{x}) &= y_2 - 1053.6667 \leq 0 \\
g_9(\vec{x}) &= 11.275 - y_3 \leq 0 \\
g_{10}(\vec{x}) &= y_3 - 35.03 \leq 0 \\
g_{11}(\vec{x}) &= 214.228 - y_4 \leq 0 \\
g_{12}(\vec{x}) &= y_4 - 665.585 \leq 0 \\
g_{13}(\vec{x}) &= 7.458 - y_5 \leq 0 \\
g_{14}(\vec{x}) &= y_5 - 584.463 \leq 0 \\
g_{15}(\vec{x}) &= 0.961 - y_6 \leq 0 \\
g_{16}(\vec{x}) &= y_6 - 265.916 \leq 0 \\
g_{17}(\vec{x}) &= 1.612 - y_7 \leq 0 \\
g_{18}(\vec{x}) &= y_7 - 7.046 \leq 0 \\
g_{19}(\vec{x}) &= 0.146 - y_8 \leq 0 \\
g_{20}(\vec{x}) &= y_8 - 0.222 \leq 0 \\
g_{21}(\vec{x}) &= 107.99 - y_9 \leq 0 \\
g_{22}(\vec{x}) &= y_9 - 273.366 \leq 0 \\
g_{23}(\vec{x}) &= 922.693 - y_{10} \leq 0 \\
g_{24}(\vec{x}) &= y_{10} - 1286.105 \leq 0 \\
g_{25}(\vec{x}) &= 926.832 - y_{11} \leq 0 \\
g_{26}(\vec{x}) &= y_{11} - 1444.046 \leq 0 \\
g_{27}(\vec{x}) &= 18.766 - y_{12} \leq 0 \\
g_{28}(\vec{x}) &= y_{12} - 537.141 \leq 0 \\
g_{29}(\vec{x}) &= 1072.163 - y_{13} \leq 0 \\
g_{30}(\vec{x}) &= y_{13} - 3247.039 \leq 0
\end{aligned} \tag{A.30}$$

$$
\begin{aligned}
g_{31}(\vec{x}) &= 8961.448 - y_{14} \leq 0 \\
g_{32}(\vec{x}) &= y_{14} - 26844.086 \leq 0 \\
g_{33}(\vec{x}) &= 0.063 - y_{15} \leq 0 \\
g_{34}(\vec{x}) &= y_{15} - 0.386 \leq 0 \\
g_{35}(\vec{x}) &= 71084.33 - y_{16} \leq 0 \\
g_{36}(\vec{x}) &= -140000 + y_{16} \leq 0 \\
g_{37}(\vec{x}) &= 2802713 - y_{17} \leq 0 \\
g_{38}(\vec{x}) &= y_{17} - 12146108 \leq 0
\end{aligned}
$$

onde:

$$
\begin{aligned}
y_1 &= x_2 + x_3 + 41.6 \\
c_1 &= 0.024x_4 - 4.62 \\
y_2 &= \frac{12.5}{c_1} + 12 \\
c_2 &= 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 &= 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 &= \frac{c_2}{c_3} \\
y_4 &= 19y_3 \\
c_4 &= 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
c_5 &= 100x_2 \\
c_6 &= x_1 - y_3 - y_4 \\
c_7 &= 0.950 - \frac{c_4}{c_5} \\
y_5 &= c_6c_7 \\
y_6 &= x_1 - y_5 - y_4 - y_3 \\
c_8 &= (y_5 + y_4)0.995 \\
y_7 &= \frac{c_8}{y_1} \\
y_8 &= \frac{c_8}{3798} \\
c_9 &= y_7 - \frac{0.0663y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321y_1 \\
y_{10} &= 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
y_{11} &= 1.71x_1 - 0.452y_4 + 0.580y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75y_2)(0.995x_1) \\
c_{12} &= 0.995y_{10} + 1998 \\
y_{12} &= c_{10}x_1 + \frac{c_{11}}{c_{12}}
\end{aligned}
\tag{A.31}
$$

$$
\begin{aligned}
y_{13} &= c_{12} - 1.75y_2 \\
y_{14} &= 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5} \\
c_{13} &= 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \\
y_{15} &= \frac{y_{13}}{c_{13}} \\
y_{16} &= 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13} \\
c_{14} &= 2324y_{10} - 28740000y_2 \\
y_{17} &= 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} &= \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52} \\
c_{16} &= 1.104 - 0.72y_{15} \\
c_{17} &= y_9 + x_5
\end{aligned}
$$

e onde os limites são $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ e $25 \leq x_5 \leq 84.1988$. O mínimo global está em $\vec{x}^*$ = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832) onde $f(x^*) = -1.90515525853479$.

Problema `g17`

Minimizar:

$$f(\vec{x}) = f(x_1) + f(x_2) \tag{A.32}$$

onde

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

Sujeito a:

$$
\begin{aligned}
h_1(\vec{x}) &= -x_1 + 300 - \frac{x_3 x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588) \\
h_2(\vec{x}) &= -x_2 - \frac{x_3 x_4}{131.078}\cos((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588) \\
h_3(\vec{x}) &= -x_5 - \frac{x_3 x_4}{131.078}\sin((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\sin(1.47588) \\
h_4(\vec{x}) &= 200 - \frac{x_3 x_4}{131.078}\sin((1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\sin(1.47588)
\end{aligned} \tag{A.33}
$$

onde os limites são $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ e $0 \leq x_6 \leq 0.5236$. O mínimo global está em $\vec{x}^*$ = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128) onde $f(\vec{x}^*)$ = 8853.53967480648.

Problema `g18`

Minimizar:

$$f(\vec{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7) \tag{A.34}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= x_3^2 + x_4^2 - 1 \leq 0 \\
g_2(\vec{x}) &= x_9^2 - 1 \leq 0 \\
g_3(\vec{x}) &= x_5^2 + x_6^2 - 1 \leq 0 \\
g_4(\vec{x}) &= x_1^2 + (x_2 - x_9)^2 - 1 \leq 0 \\
g_5(\vec{x}) &= (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0 \\
g_6(\vec{x}) &= (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0 \\
g_7(\vec{x}) &= (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0 \\
g_8(\vec{x}) &= (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0 \\
g_9(\vec{x}) &= x_7^2 + (x_8 - x_9)^2 - 1 \leq 0 \\
g_{10}(\vec{x}) &= x_2x_3 - x_1x_4 \leq 0 \\
g_{11}(\vec{x}) &= -x_3x_9 \leq 0 \\
g_{12}(\vec{x}) &= x_5x_9 \leq 0 \\
g_{13}(\vec{x}) &= x_6x_7 - x_5x_8 \leq 0
\end{aligned} \tag{A.35}$$

onde os limites são $-10 \leq x_i \leq 10$ ($i = 1, \ldots, 8$) e $0 \leq x_9 \leq 20$. O mínimo global está em $\vec{x}^*$ = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542) onde $f(\vec{x}^*) = -0.866025403784439$.

Problema `g19`

Minimizar:

$$f(\vec{x}) = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij}x_{(10+i)}x_{(10+j)} + 2\sum_{j=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i \tag{A.36}$$

Sujeito a:

$$g_j(\vec{x}) = -2\sum_{i=1}^{5} c_{ij}x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij}x_i \leq 0 \qquad j = 1, \ldots, 5 \tag{A.37}$$

onde $\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$ e o restante dos dados estão na Tabela A.1. Os limites são $0 \leq x_i \leq 10$ ($i = 1, \ldots, 15$). O mínimo global é $\vec{x}^*$ = (1.66991341326291344e-17, 3.95378229282456509e-16, 3.94599045143233784, 1.06036597479721211e-16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333e-17, 1.2026194599794709e-17, 2.50706276000769697e-15, 2.24624122987970677e-15, 0.370764847417013987, 0.278456024942955571, 0.523834876722411 71, 0.388620152510322781, 0.298156764974678579) onde $f(\vec{x}^*)$ = 32.6555929502463.

j	1	2	3	4	5
e_j	−15	−27	−36	−18	−12
c_{1j}	30	−20	−10	32	−10
c_{2j}	−20	39	−6	−31	32
c_{3j}	−10	−6	10	−6	−10
c_{4j}	32	−31	−6	39	−20
c_{5j}	−10	32	−10	−20	30
d_j	4	8	10	6	2
a_{1j}	−16	2	0	1	0
a_{2j}	0	−2	0	0.4	2
a_{3j}	−3.5	0	2	0	0
a_{4j}	0	−2	0	−4	−1
a_{5j}	0	−9	−2	1	−2.8
a_{6j}	2	0	−4	0	0
a_{7j}	−1	−1	−1	−1	−1
a_{8j}	−1	−2	−3	−2	−1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

Tabela A.1: Conjunto de dados para o problema g19

Problema g20

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i \tag{A.38}$$

Sujeito a:

$$\begin{aligned}
g_i(\vec{x}) &= \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \qquad i = 1,2,3 \\
g_i(\vec{x}) &= \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \qquad i = 4,5,6 \\
h_i(\vec{x}) &= \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \qquad i = 1,\ldots,12 \\
h_{13}(\vec{x}) &= \sum_{i=1}^{24} x_i - 1 = 0 \\
h_{14}(\vec{x}) &= \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0
\end{aligned} \tag{A.39}$$

onde $k = (0.7302)(530)(\frac{14.7}{40})$ e o conjunto de dados está detalhado na Tabela A.2. Os limites são $0 \leq x_i \leq 10$ ($i = 1,\ldots,24$). O mínimo global está em $\vec{x}^*$ = (1.28582343498528086e-18, 4.83460302526130664e-34, 0.0, 0.0, 6.30459929660781851e-18, 7.57192526201145068e-34, 5.03350698372840437e-34, 9.28268079616618064e-34, 0.0, 1.76723384525547359e-17, 3.55686101822965701e-34, 2.99413850083471346e-34, 0.158143376337580827, 2.29601774161699833e-19, 1.06106938611042947e-18, 1.31968344319506391e-18, 0.530902525044209539, 0.0, 2.89148310257773535e-18, 3.34892126180666159e-18, 0.0, 0.310999974151577319, 5.41244666317833561e-05, 4.84993165246959553e-16).

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

Tabela A.2: Conjunto de dados para o problema g20

Problema `g21`

Minimizar:

$$f(\vec{x}) = x_1 \tag{A.40}$$

Sujeito a:

$$\begin{aligned}
g_1(\vec{x}) &= -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0 \\
h_1(\vec{x}) &= -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0 \\
h_2(\vec{x}) &= 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0 \\
h_3(\vec{x}) &= -x_5 + \ln(-x_4 + 900) = 0 \\
h_4(\vec{x}) &= -x_6 + \ln(x_4 + 300) = 0 \\
h_5(\vec{x}) &= -x_7 + \ln(-2x_4 + 700) = 0
\end{aligned} \tag{A.41}$$

onde $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ e $4.5 \leq x_7 \leq 6.25$. O mínimo global está em $\vec{x}^*$ = (193.724510070034967, 5.56944131553368433e-27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451) onde $f(\vec{x}^*)$ = 193.724510070035.

Problema `g22`

Minimizar:

$$f(\vec{x}) = x_1 \tag{A.42}$$

Sujeito a:

$$
\begin{aligned}
&g_1(\vec{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0 \\
&h_1(\vec{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0 \\
&h_2(\vec{x}) = x_6 + 100000x_8 - 100000x_9 = 0 \\
&h_3(\vec{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0 \\
&h_4(\vec{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0 \\
&h_5(\vec{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0 \\
&h_6(\vec{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0 \\
&h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0 \\
&h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0 \\
&h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0 \\
&h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0 \\
&h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0 \\
&h_{12}(\vec{x}) = -x_{18} + \ln(x_{10} - 100) = 0 \\
&h_{13}(\vec{x}) = -x_{19} + \ln(-x_8 + 300) = 0 \\
&h_{14}(\vec{x}) = -x_{20} + \ln(x_{16}) = 0 \\
&h_{15}(\vec{x}) = -x_{21} + \ln(-x_9 + 400) = 0 \\
&h_{16}(\vec{x}) = -x_{22} + \ln(x_{17}) = 0 \\
&h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0 \\
&h_{18}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0 \\
&h_{19}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0
\end{aligned}
\tag{A.43}
$$

onde os limites são $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. O mínimo global está em $\vec{x}^*$ = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395) onde $f(\vec{x}^*)$ = 236.430975504001.

Problema g23

Minimizar:

$$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7) \tag{A.44}$$

Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0 \\ g_2(\vec{x}) &= x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0 \\ h_1(\vec{x}) &= x_1 + x_2 - x_3 - x_4 = 0 \\ h_2(\vec{x}) &= 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0 \\ h_3(\vec{x}) &= x_3 + x_6 - x_5 = 0 \\ h_4(\vec{x}) &= x_4 + x_7 - x_8 = 0 \end{aligned} \tag{A.45}$$

onde os limites são $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ e $0.01 \leq x_9 \leq 0.03$. O mínimo global está em $\vec{x}^*$ = (0.00510000000000259465, 99.9947000000000514, 9.01920162996045897e-18, 99.9999000000000535, 0.000100000000027086086, 2.75700683389584542e-14, 99.9999999999999574, 200 0.0100000100000100008) onde $f(\vec{x}^*)$ = -400.055099999999584.

Problema g24

Minimizar:

$$f(\vec{x}) = -x_1 - x_2 \tag{A.46}$$

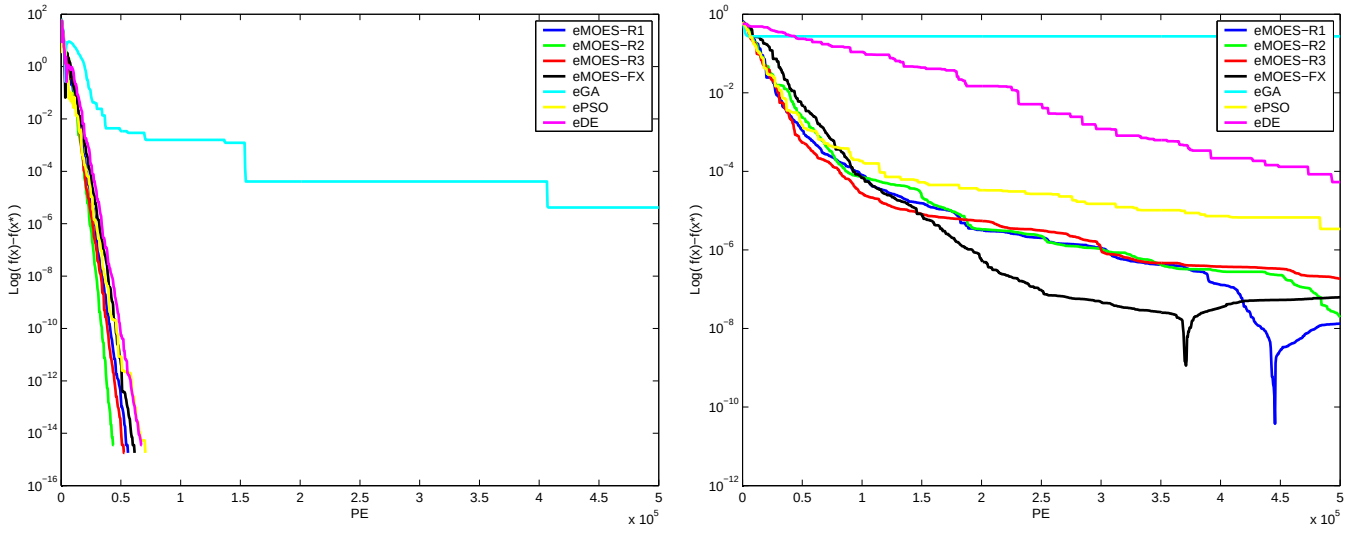
Sujeito a:

$$\begin{aligned} g_1(\vec{x}) &= -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ g_2(\vec{x}) &= -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \end{aligned} \tag{A.47}$$

onde $0 \leq x_1 \leq 3$ e $0 \leq x_2 \leq 4$. O mínimo global está em $\vec{x}^*$ = (2.32952019747762, 3.17849307411774) onde $f(\vec{x}^*)$ = -5.50801327159536.

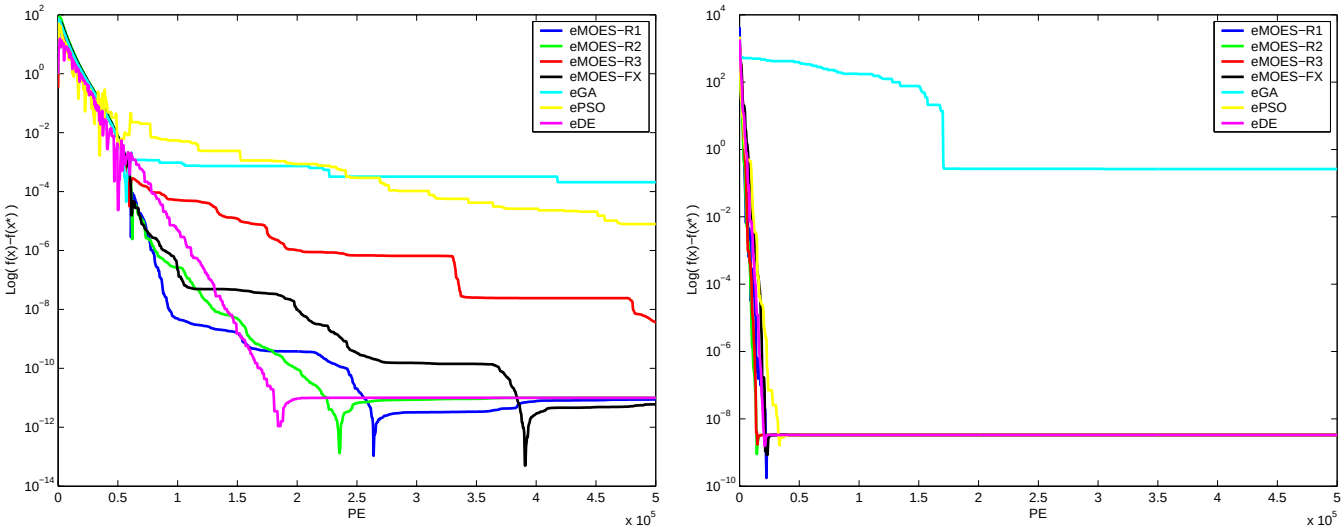
ANEXO B – Gráficos de Convergência

Neste anexo apresentamos 24 gráficos de convergência comparando os sete algoritmos listando na seção 6.2. Os valores plotados são relativos à melhor rodada de cada algoritmo.



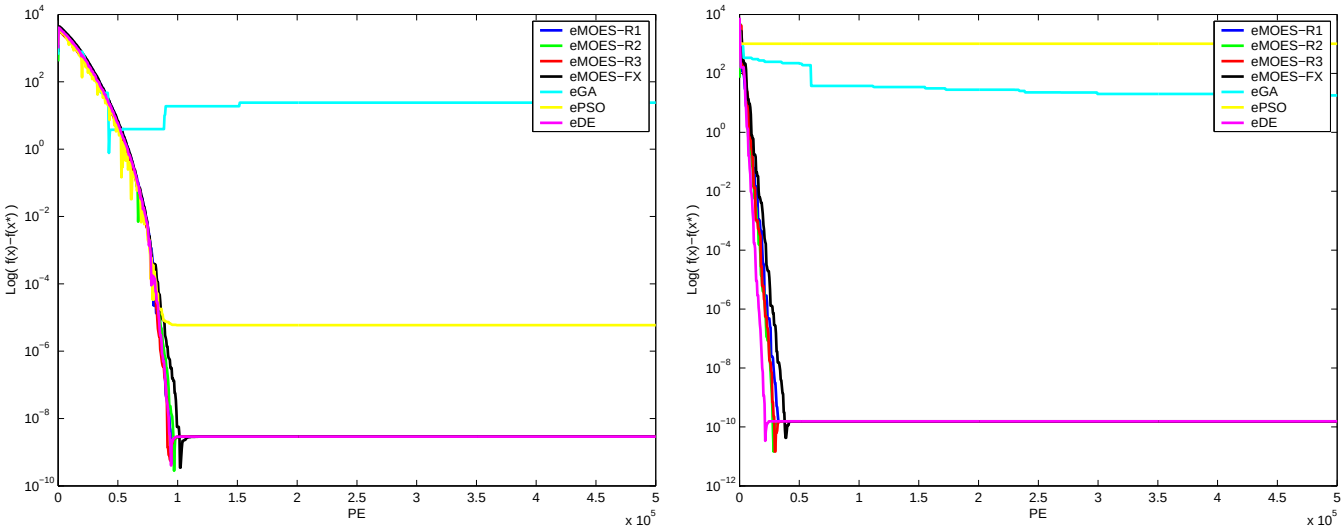
(a) Problema g01 (b) Problema g02

Figura B.1: Gráficos de convergência para os problemas g01 e g02



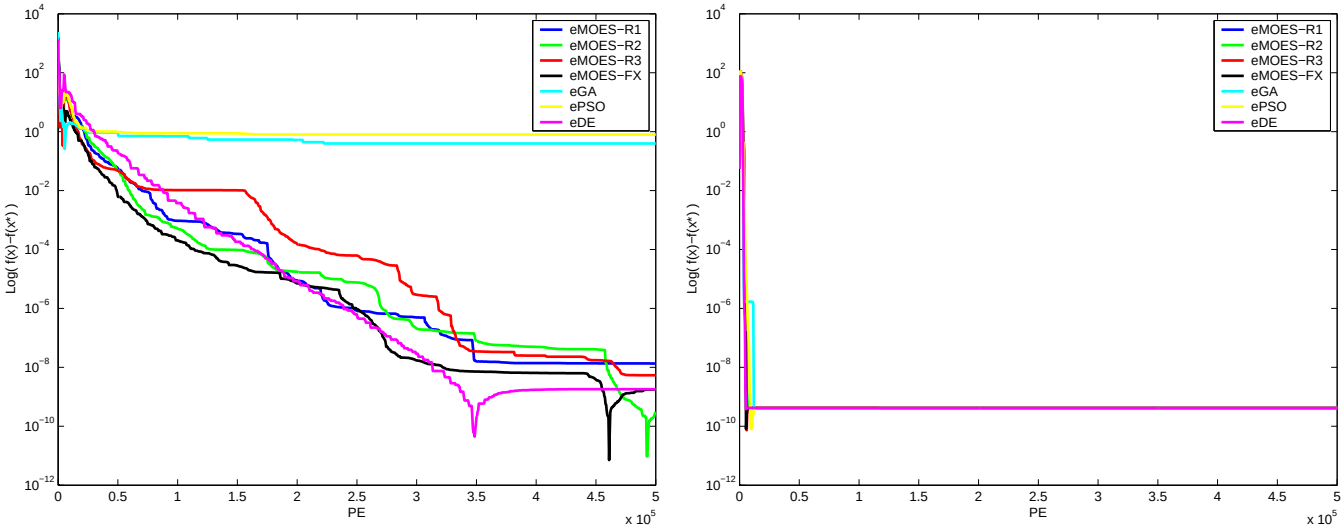
(a) Problema g03 (b) Problema g04

Figura B.2: Gráficos de convergência para os problemas g03 e g04



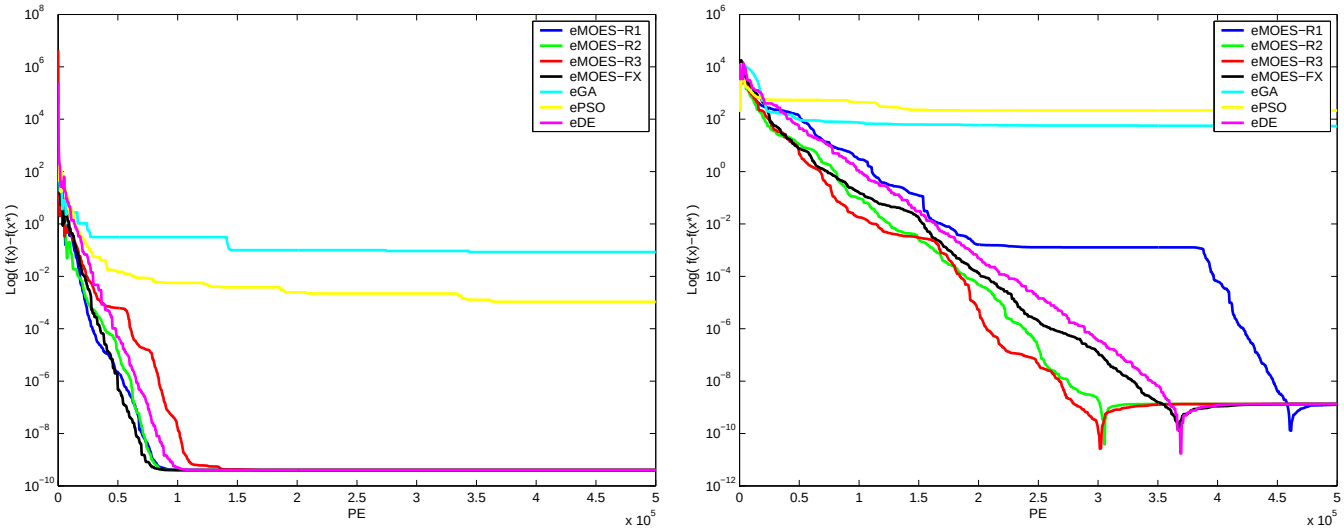
(a) Problema g05 (b) Problema g06

Figura B.3: Gráficos de convergência para os problemas g05 e g06



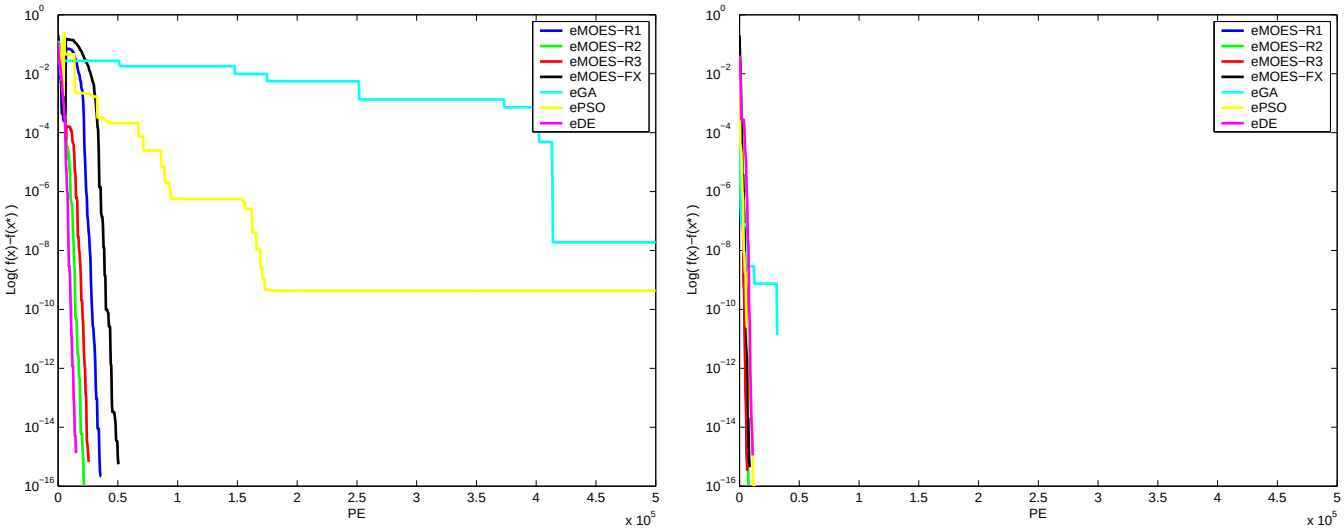
(a) Problema g07 (b) Problema g08

Figura B.4: Gráficos de convergência para os problemas g07 e g08



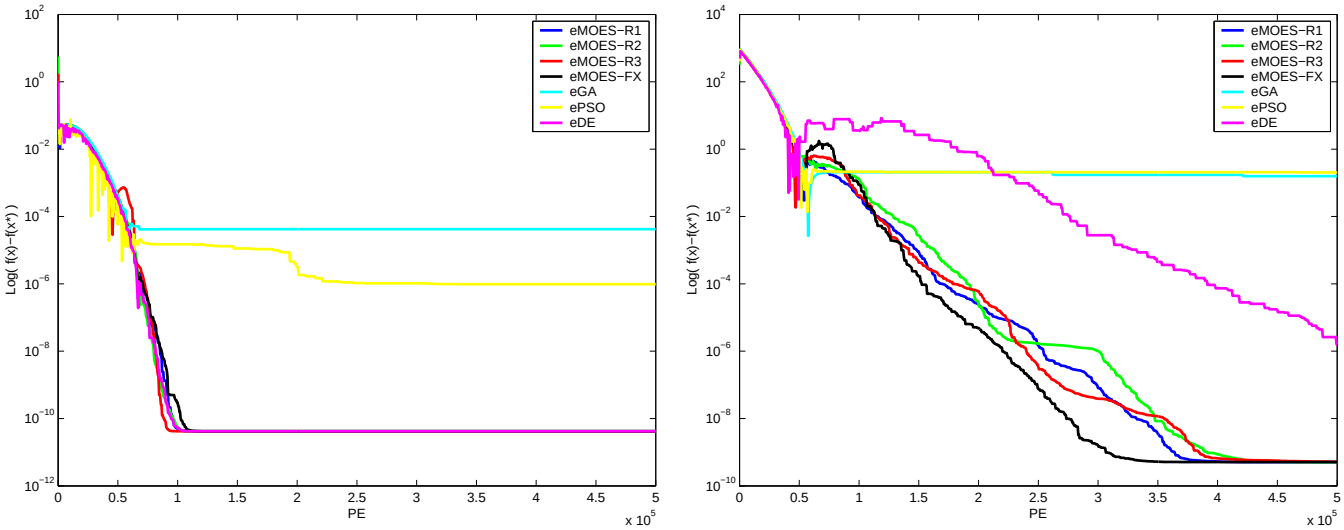
(a) Problema g09 (b) Problema g10

Figura B.5: Gráficos de convergência para os problemas g09 e g10



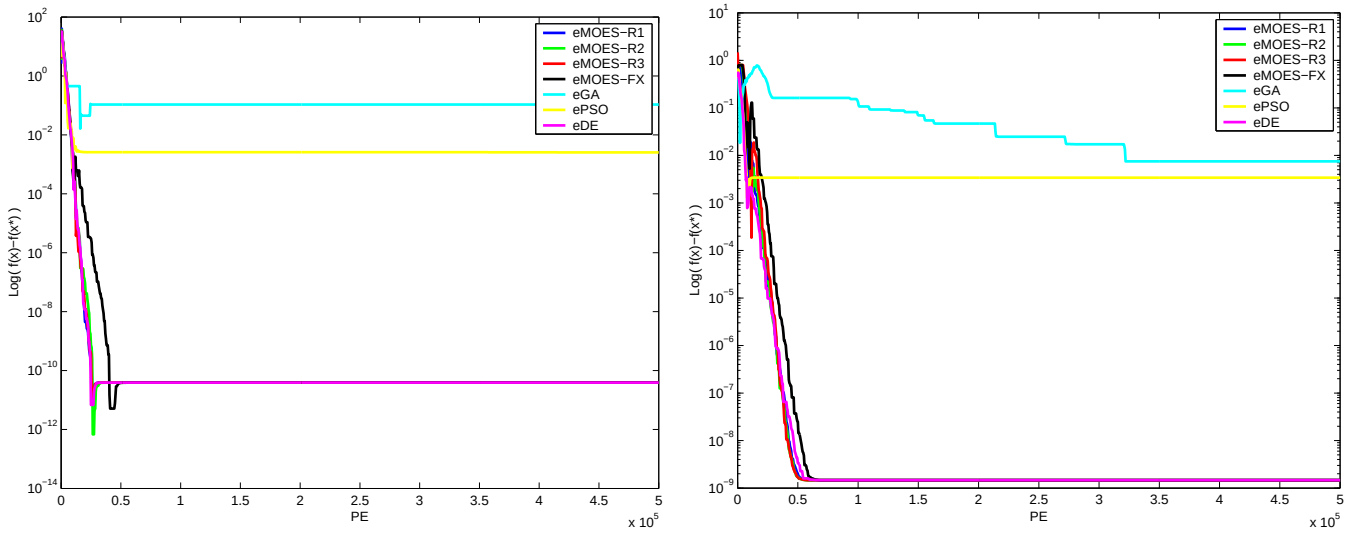
(a) Problema g11 (b) Problema g12

Figura B.6: Gráficos de convergência para os problemas g11 e g12



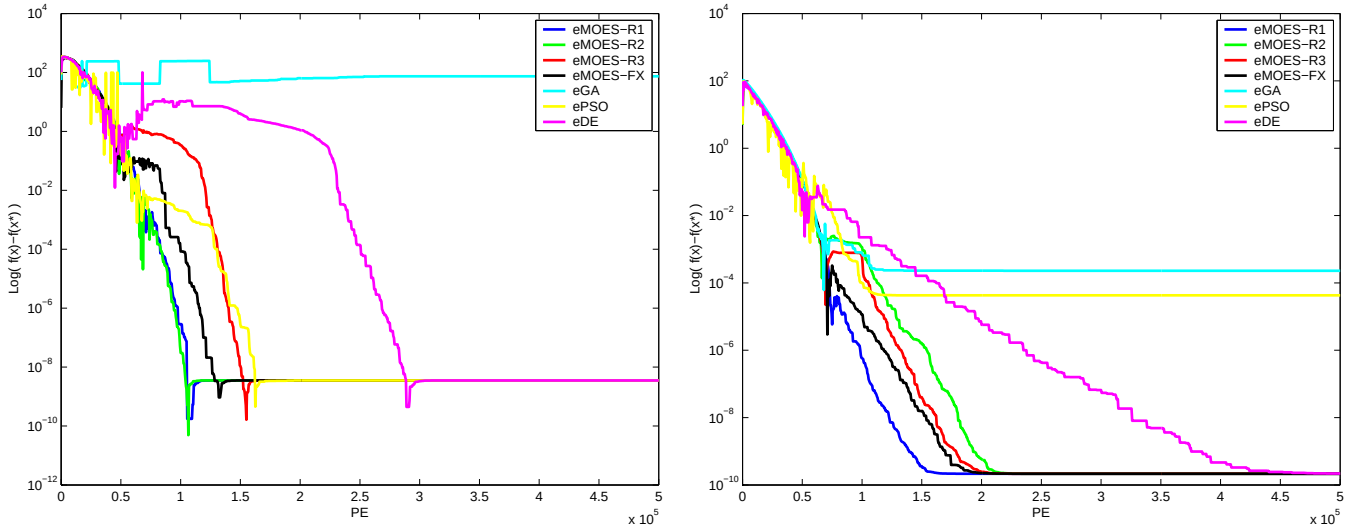
(a) Problema g13 (b) Problema g14

Figura B.7: Gráficos de convergência para os problemas g13 e g14



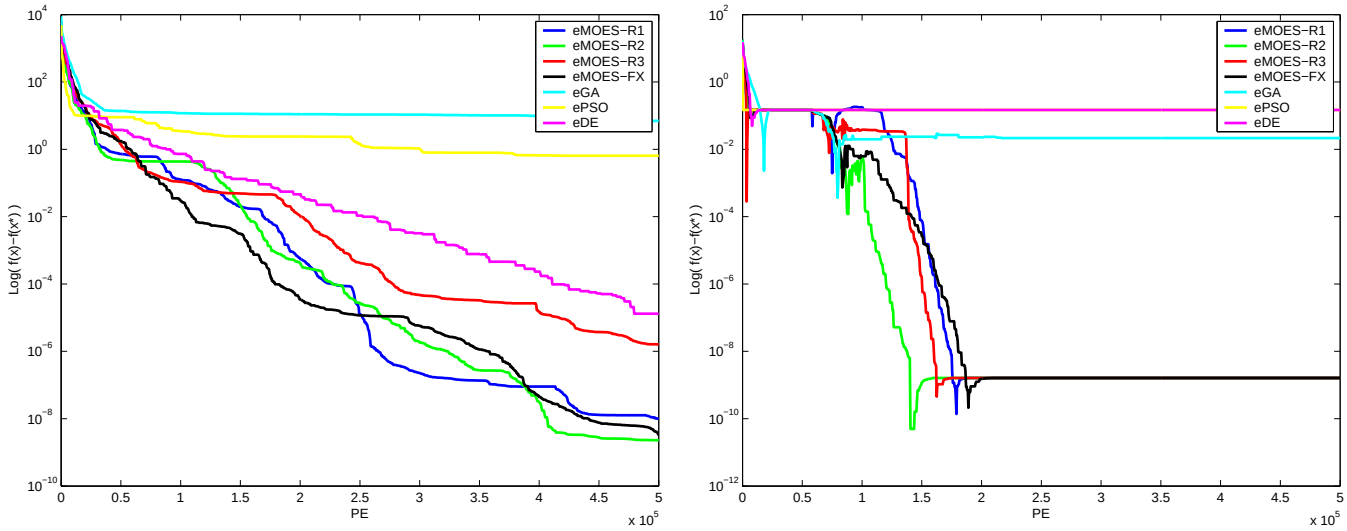
(a) Problema g15 (b) Problema g16

Figura B.8: Gráficos de convergência para os problemas g15 e g16



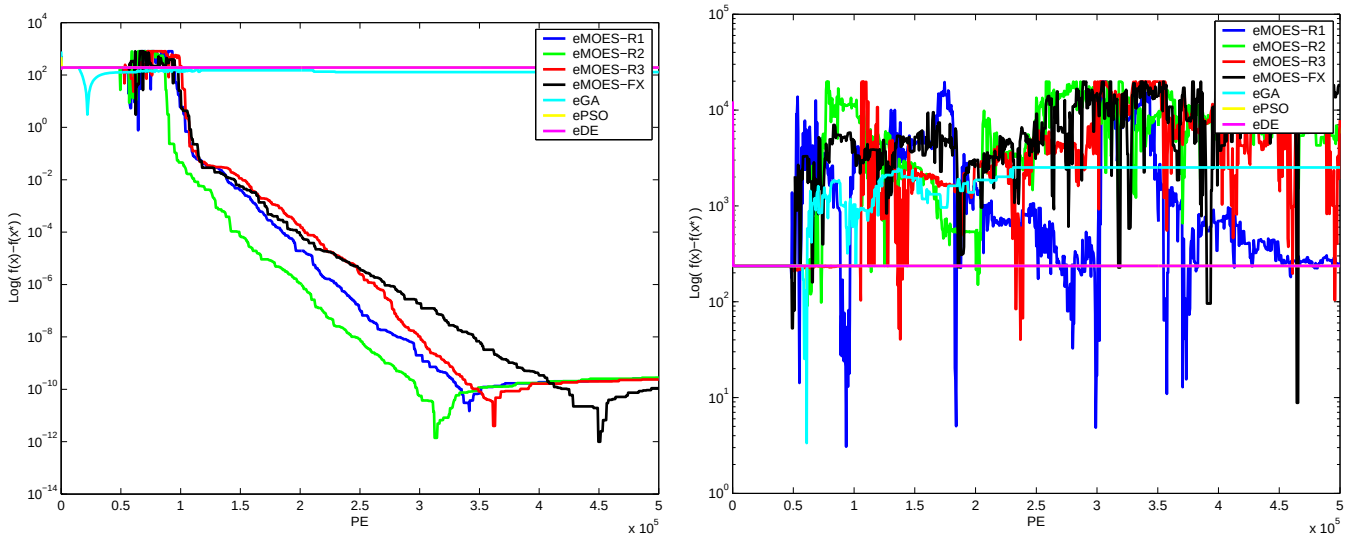
(a) Problema g17 (b) Problema g18

Figura B.9: Gráficos de convergência para os problemas g17 e g18



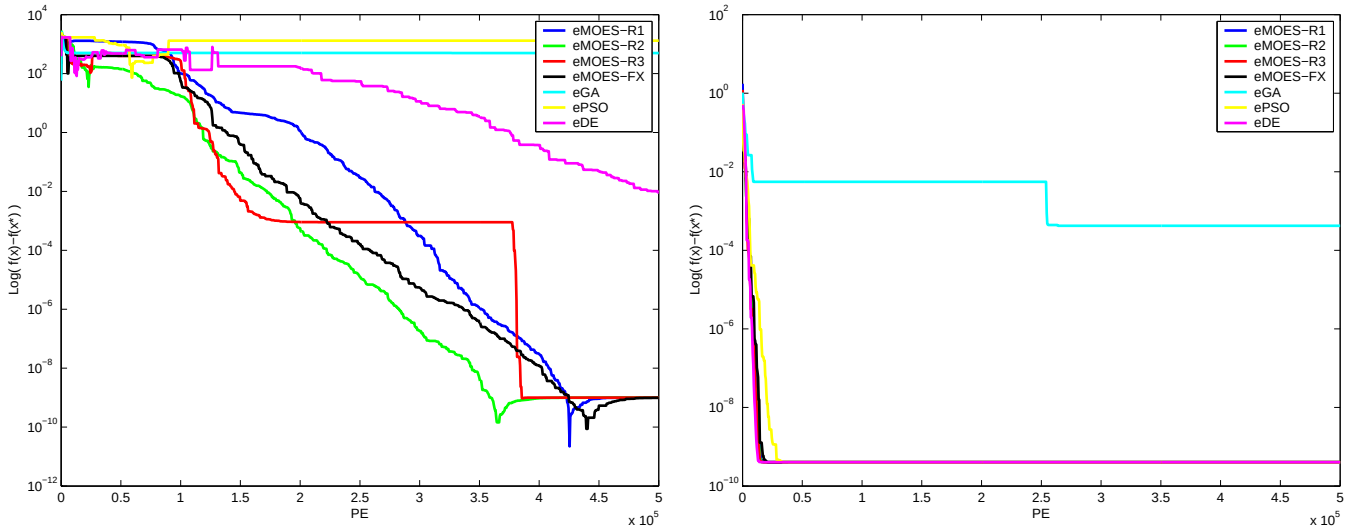
(a) Problema g19 (b) Problema g20

Figura B.10: Gráficos de convergência para os problemas g19 e g20



(a) Problema g21 (b) Problema g22

Figura B.11: Gráficos de convergência para os problemas g21 e g22



(a) Problema g23 (b) Problema g24

Figura B.12: Gráficos de convergência para os problemas g23 e g24