RODRIGO CÉSAR PEDROSA SILVA

Orientador: Frederico Gadelha Guimarães

# UM ESTUDO SOBRE A AUTOADAPTAÇÃO DE PARÂMETROS NA EVOLUÇÃO DIFERENCIAL

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

# UM ESTUDO SOBRE A AUTOADAPTAÇÃO DE PARÂMETROS NA EVOLUÇÃO DIFERENCIAL

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

RODRIGO CÉSAR PEDROSA SILVA

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

# Um Estudo sobre a Autoadaptação de Parâmetros na Evolução Diferencial

RODRIGO CÉSAR PEDROSA SILVA

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Frederico Gadelha Guimarães – Orientador
Universidade Federal de Minas Gerais

Dr. André Luis Silva
Universidade Federal de Ouro Preto

Dr. Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto

Dr. Ricardo Sérgio Prado
Instituto Federal Minas Gerais

Ouro Preto, Novembro de 2010

# Resumo

O grande desenvolvimento na área de algoritmos evolutivos nas últimas décadas tem aumentado o domínio de aplicações dessas ferramentas e melhorado seu desempenho em diversos contextos. Em particular, o algoritmo de Evolução Diferencial (DE)tem se mostrado um otimizador simples e eficiente para funções no domínio contínuo. Recentemente ele também tem sido usado para otimização de problemas multi-objetivo e combinatórios em diversas áreas. Apesar desses resultados promissores, a configuração de parâmetros no algoritmo é uma etapa crítica para se obter um desempenho adequado, além disso, a configuração ideal destes parâmetros depende da função que se quer otimizar. Neste contexto, a autoadaptação de parâmetros permite que o algoritmo se adapte, reconfigurando seus parâmetros de acordo com o problema que se está tratando, sem interferência do usuário. Existem alguns estudos sobre este tema na literatura, porém alguns deles não se alinham completamente com os conceitos de autoadaptação. Este trabalho então apresenta uma revisão crítica dos trabalhos da literatura sobre autoadaptação no DE e, em seguida, apresenta uma nova abordagem de autoadaptação para este algoritmo. É feita uma análise experimental do algoritmo proposto e em seguida é feita uma comparação de desempenho com o DE original de parâmetros fixos e com uma estratégia autoadaptativa de sucesso da literatura (o jDE). Os resultados indicam que a proposta é promissora e apresenta rápida convergência.

# Abstract

The great development in the area of evolutionary algorithms in recent decades has increased the range of applications of these tools and improved its performance in different contexts. In particular, the Differential Evolution (DE) algorithm has proven to be a simple and efficient optimizer for continuous functions. Recently it has also been used for multi-objective and combinatorial optimization problems in several areas. Despite these promising results, the configuration of the parameters in the algorithm is a critical step to achieve adequate performance, in addition, the optimal configuration of these parameters depends on the function which will be optimized. In this context, the self-adaptation allows the algorithm to adapt itself according to the problem being treated by reconfiguring its parameters, without user interference. There are some studies on this subject in literature, but some of them do not align completely with self-adaptation concepts. This paper then presents a critical review of the literature on DE self-adaptation and then presents a new approach to self-adapt this algorithm. An experimental analysis of the algorithm is made and then a performance comparison between the original DE, a success self-adaptive DE in the literature (the jDE) and the proposed version is presented. The results indicate that the proposal is promising and has fast convergence.

*Dedico este trabalho aos meu pais, Marcelo e Maria Lúcia pelo apoio incondicional e pela paciência, principalmente, nestes últimos meses.*

# Agradecimentos

Agradeço ao meu orientador Frederico Gadelha Guimarães, pela orientação científica criteriosa e crítica, estimulando e dando o tempo para uma construção pessoal deste trabalho. A disponibilidade que sempre manifestou e a empatia com que recebeu as minhas idéias, foram os estímulos que me permitiram vencer as dificuldades desta etapa.

Agradeço também ao professor Ricardo Prado e aos meus colegas Rodolfo Lopes, Angelo Assis, Pedro Paulo Freitas e Marcus Vinicius Soares pelo apoio e pelas conversas sempre estimulantes, das quais, várias idéias brotaram.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Independente da área, a busca por eficiência é uma constante. O tempo todo queremos solucionar problemas de forma melhor e mais rápida e para tal recorremos à otimização. Otimização é a busca da melhor solução para um dado problema dentro de um conjunto finito ou infinito de possíveis soluções. Este processo pode partir de uma única solução inicial ou de um conjunto delas, realizando melhoramentos progressivos até chegar a um conjunto que contenha a melhor ou as melhores soluções possíveis para o dado problema.

Um problema em otimização é normalmente descrito por uma função, chamada função objetivo, sujeita ou não a uma série de restrições. Encontrar a melhor solução então, significa encontrar o conjunto de variáveis que levam ao ponto de máximo (caso dos problemas de maximização) ou ao ponto de mínimo (caso dos problemas de minimização) da referida função, em outros termos, significa encontrar o valor ótimo da função. Tradicionalmente os métodos mais usados para este tipo de problema baseiam-se em algoritmos de busca local, que frequentemente usam a informação do gradiente da função objetivo como "guia" para a busca do ponto ótimo no espaço de busca. Estes métodos funcionam muito bem para funções contínuas e unimodais, como mostrado na Fig.1.1. Porém, para funções objetivo complexas, que podem ser, multimodais (apresentam vários máximos/mínimos locais), não contínuas, com variáveis de diversos tipos, estes métodos têm sua eficiência bastante reduzida, e tendem a fornecer soluções sub-ótimas (máximos/mínimos locais)(de Sousa, 2003). A Fig.1.2 mostra o que pode acontecer com este tipo de técnica em uma função multimodal.

Neste contexto, os Algoritmos Evolutivos (AEs) se tornaram uma opção para a otimização de problemas complexos demais para serem resolvidos por técnicas determinísticas como os métodos de programação linear e gradiente (Sbalzariniy et al., 2000). Os AEs são um tipo de método probabilístico que imitam o processo de evolução natural e desde os anos 80 têm sido utilizados para a resolução de problemas difíceis de otimização, como relatado em (Bäck et al., 1997).

A idéia básica por trás de todos estes métodos é basicamente a mesma: Dada uma população de indivíduos a pressão ambiental causa seleção natural (sobrevivência do mais adaptado),

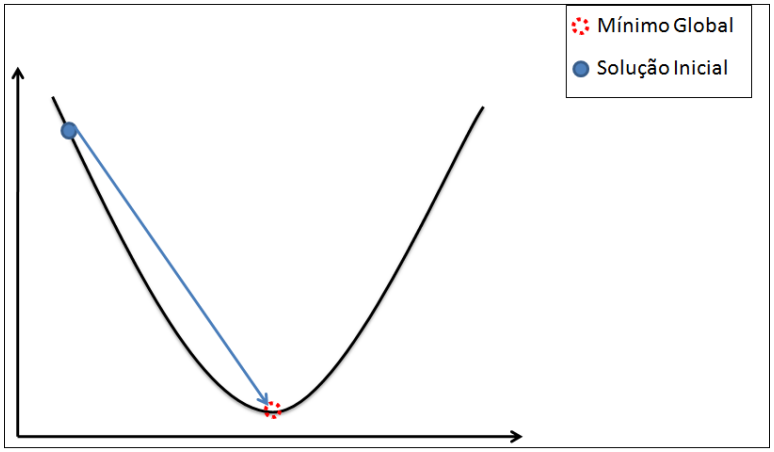


Figura 1.1: Busca Local numa Função Unimodal

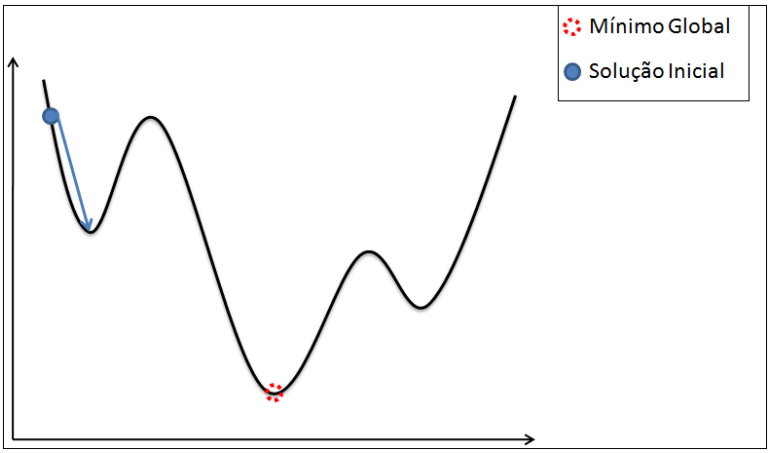


Figura 1.2: Busca Local numa Função Multimodal

ao longo do tempo esta seleção aumenta a aptidão dos indivíduos da população. Adaptando esta idéia para o processo de otimização, temos uma população formada por um conjunto de soluções candidatas do problema, estas soluções podem sofrer algum tipo de variação e, ao longo do tempo, têm sua adaptabilidade medida através da função objetivo. Soluções mais adaptadas, ou seja, soluções que possuem um valor melhor de função objetivo são selecionadas e repetem o processo, soluções ruins são descartadas. O esquema básico de um algoritmo evolutivo pode ser visto na Fig.1.3

A classe de algoritmos evolutivos apresenta várias famílias de métodos, dentre as quais a família dos Algoritmos Genéticos (Goldberg, 1989) é provavelmente a mais popular. Contudo, nos últimos anos, novas famílias de métodos têm sido desenvolvidas, tais como algoritmos de Sistemas Imunes Artificiais (AIS - Artificial Immune Systems)(Castro, 2002), algoritmos de Otimização por Enxame de Partículas (PSO - Particle Swarm Optimization)(Kennedy e Eberhart, 1995) e algoritmos de Evolução Diferencial (DE - Differential Evolution) (Storn e Price,

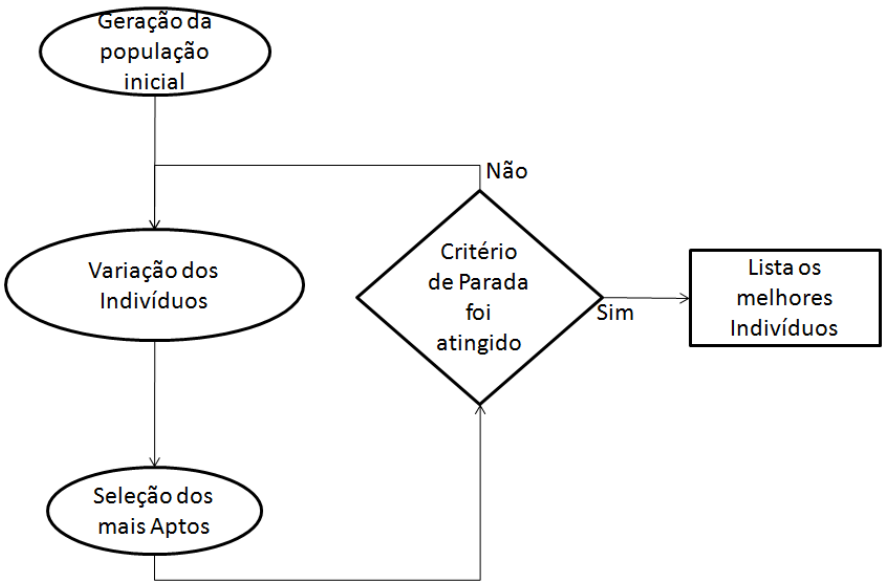


Figura 1.3: Algoritmo Evolutivo Básico

1995). A principal vantagem no uso de AEs está no ganho de flexibilidade e adaptabilidade em relação ao problema em mãos, em combinação com desempenho robusto (apesar desse fator depender da classe do problema) e com suas características de busca global (Bäck et al., 1997).

Nas últimas décadas, os avanços obtidos na área dos AEs aumentaram o domínio de aplicações dessas ferramentas e melhoraram seu desempenho em diversos contextos. Em particular, o algoritmo de Evolução Diferencial (DE - Differential Evolution) proposto em meados da década de 90 é um otimizador simples e versátil para a otimização de funções com variáveis contínuas (Storn e Price, 1995, 1997; Chakraborty, 2008). Graças a seu alto desempenho e facilidade de implementação, o DE se tornou bastante popular quase imediatamente após sua definição. Atualmente tem sido usado com sucesso em diversas aplicações como otimização de sistemas de reservatório (Reddy e Kumar, 2007) e processamento de imagens(Das et al., 2008). Além de se mostrar como um otimizador eficiente em diversos contextos, o algoritmo DE é um método robusto, no sentido de apresentar pequena variância de resultados para um dado número de execuções independentes(Price et al., 2005).

Assim como a maioria dos algoritmos evolutivos o DE é um algoritmo populacional que apresenta operadores de mutação (operador que introduz alterações nas soluções candidatas), recombinação (operador que embaralha as soluções candidatas para obter novas soluções) e seleção (operador que visa manter na população soluções cadidatas com alta aptidão em relação à função que se deseja otimizar). Diante destes operadores alguns parâmetros precisam ser definidos, como o tamanho da população, o grau de pertubação gerado pela mutação e a probabilidade de cruzamento. A fixação destes parâmetros é crítica para o desempenho do

algoritmo e, além disso, é dependente do problema que se quer resolver (Brest e Maucec, 2006). Normalmente a fixação dos parâmetros é feita empiricamente pelo usuário e demanda enorme esforço e tempo, neste contexto a autoadaptação permite que o algoritmo se adapte a qualquer classe de problemas recofigurando-se de acordo, sem a necessidade de interação com o usuário.

Na autoadaptação os parâmetros são acoplados com as variáveis de otimização na representação dos indivíduos, passando a também sofrer ação dos operadores genéticos (mutação, cruzamento e seleção). Supõe-se então que boas soluções são geradas por bons parâmetros, consequentemente estes bons parâmetros serão mantidos e difundidos na população, podendo ainda ser refinados durante o processo evolutivo.

Existem na literatura alguns trabalhos relacionados à autoadaptação de parâmetros no DE (Brest e Maucec, 2006; Qin e Suganthan, 2005; Teo, 2006), porém quando tratamos deste assunto, pressupomos que os parâmetros autoadaptados podem ser avaliados implicitamente pelo operador de seleção do algoritmo, além disso, contempla-se a utilização da experiência adquirida pelo algoritmo durante o processo de otimização na obtenção dos novos parâmetros. Nos trabalhos citados acima estas características não são sempre preservadas, portanto eles não fazem uma autoadaptação verdadeira.

Neste trabalho, é feita uma análise crítica dos esquemas autoadaptativos propostos na literatura para o algoritmo DE. Nesta análise são apontados alguns problemas destes esquemas e uma nova abordagem, que pretende eliminar tais problemas é proposta. Seu comportamento e desempenho são analisados e comparados com a abordagem que até então tem apresentado os melhores resultados.

## 1.1 Objetivos

### 1.1.1 Objetivo geral

Desenvolver uma estratégia de autoadaptação de parâmetros para o algoritmo de Evolução Diferencial. Esta estratégia deve fazer uso de parâmetros adequados e utilizar operadores para a mutação dos parâmetros que levam em consideração o "conhecimento" adquirido, durante o processo de busca.

### 1.1.2 Objetivos específicos

- Estudar o comportamento e desempenho da autoadaptação de parâmetros em algoritmos evolutivos com atenção especial aos algoritmos DE;
- Estudar as abordagens propostas na literatura para autoadaptação de parâmetros no algoritmo de evolução diferencial;
- Verificar quais parâmetros do DE podem ser autoadaptados;

- Propor operadores adequados para a alteração dos parâmetros.

## 1.2 Motivação

AEs são uma categoria de algoritmos de otimização estocásticos inspirados pela biologia. Eles possuem uma vantagem proeminente diante de outros métodos numéricos por apenas requerer a simples descrição matemática do que se quer ver presente na solução. Informações sobre diferenciabilidade e continuidade da função objetivo não são necessárias, o que possibilita sua utilização num espectro maior de problemas.

No uso de um AE, é necessário especificar como as soluções candidatas serão modificadas para gerar novas soluções (Maruo et al., 2005). Isto implica que AEs têm parâmetros que devem ser fixados. Os valores destes prâmetros determinam fortemente a qualidade da solução que vai se obter e a eficiência do processo de busca (Eiben e Smith, 1999).

Um dos atuais estado-da-arte em AEs é o algoritmo de Evolução Diferencial (DE) (Teo, 2006), que tem sido usado com sucesso em diversas aplicações. Contudo como um AE seu desempenho é dependente da configuração de parâmetros, além disso, os parâmetros mais adequados dependem do problema que se deseja tratar, como relatado em (Liu e Lampinen, 2002). Neste caso, a criação de uma boa estratégia de controle automático de parâmetros se mostra importante, pois pode dar ao DE maior flexibilidade e robustez independentemente do problema. Isso permite sua aplicação direta numa maior diversidade de problemas sem a necessidade de interação do usuário e favorece seu uso em situações onde a função objetivo possui características pouco conhecidas ou muda com tempo.

## 1.3 Trabalhos Relacionados

O algoritmo de Evolução Diferencial (DE) foi inicialmente proposto em meados da década de 90 para otimização com variáveis contínuas (Storn e Price, 1995), hoje em dia, é um algoritmo de otimização importante e poderoso em otimização de sistemas mono-objetivo (Mezura-Montes et al., 2006; Price et al., 2005), multiobjetivo (Batista et al., 2009; Xue et al., 2003) e mais recentemente, tem sido usado também em problemas combinatórios (Onwubolu e Davendra, 2009; Prado et al., 2010).

Storn e Price (1997) expuseram que os parâmetros de controle do DE são fáceis de serem fixados, contudo em 2002 Gämperle et al. (Gämperle et al., 2002) relataram que a escolha apropriada dos parâmetros de controle era mais difícil do que se esperava. Em concordância com este estudo Liu e Lampinen, também em 2002, relataram que a eficácia, eficiência e robustez do DE é sensível à configuração dos parâmetros de controle e que, o conjunto mais adequado destes parâmetros, depende da função que se quer otimizar e dos requerimentos de tempo e precisão do problema (Liu e Lampinen, 2002).

Devido à dificuldade na fixação de parâmetros, em 2005 foi proposta por Liu e Lampinen (Liu e Lampinen, 2005), uma versão do DE, onde os parâmetros de controle de mutação e cruzamento são adaptados utilizando-se um mecânismo baseado em lógica fuzzy. Seus resultados mostraram que o DE com este mecanismo tinha uma melhor desempenho se comparado com a versão clássica com parâmetros fixos.

As Estratégias Evolutivas (Schwefel, 1981), são um tipo de AE que incorpora os parâmetros à representação do indivíduo e faz com que estes também sofram ação dos operadores genéticos. Possivelmente inspirados nesta abordagem, em 2005, Qin e Suganthan (Qin e Suganthan, 2005) propuseram o SaDE (Self-adaptive Differential Evolution). Nele, são incorporadas à representação do indivíduo a estratégia de mutação e os parâmetros F e CR. Durante o processo de evolução, a estratégia e a configuração de parâmetros mais adequada é gradualmente autoadaptada de acordo com a experiência de "aprendizado" do processo de busca.

Na mesma linha, em 2006, Teo (Teo, 2006) apresentou uma tentativa de autoadaptar o tamanho da população, em adição à autoadaptação das taxas de mutação e cruzamento. Aparentemente este esquema não trouxe melhora significativa em relação ao DE clássico. Ainda em 2006, Brest et al.(Brest e Maucec, 2006) propuseram uma nova versão do DE o jDE, que usa um mecanismo, dito autoadaptativo, nos parâmetros de controle F e CR. Ainda tendo os parâmetros acoplados aos indivíduous, esse mecanismo escolhe novos parâmetros aleatoriamente com alguma probabilidade. Seus resultados mostraram que o jDE é melhor ou pelo menos comparável ao DE clássico no conjunto de testes feitos.

Em 2010, numa revisão sobre avanços do DE, Ferrante e Tirronen (Neri e Tirronen, 2010) realizaram testes com diversas estruturas modificadas do DE, incluindo os ditos autoadaptativos jDE e SaDE e no conjunto de funções testadas o jDE apresentou o melhor desempenho na maioria dos casos.

## 1.4 Organização do Trabalho

O restante deste trabalho é organizado da seguinte forma:

No capítulo 2 são apresentados os conceitos básicos de otimização, algoritmos evolutivos e feita uma descrição detalhada do algoritmo de evolução diferencial.

No capítulo 3 é feita uma discussão sobre a configuração de parâmetros em algoritmos evolutivos e é apresentado o método de autoadaptação de parâmetros. Em seguida, é feita uma discução sobre a importância da configuração de parâmetros na Evolução diferencial e são apresentados os principais métodos de autoadaptação para este algoritmo. O capítulo é então concluído com o método proposto.

No capítulo 4 são apresentados os resultados obtidos pelo método proposto num conjunto de funções de teste comparados com a versão autoadaptativa de melhor desempenho até então.

Por fim, no capítulo 5 são apresentadas as conclusões e algumas perspectivas futuras.

# Capítulo 2

# Fundamentos

## 2.1 Otimização

Otimização é a busca da melhor solução para um dado problema dentro de um conjunto finito ou infinito de possíveis soluções. A formulação genérica de um problema de otimização não liner com variáveis contínuas, pode ser dada por:

$$x^* = \arg\min f(x) \tag{2.1}$$

$$\text{sujeito a: } \begin{cases} x \in \mathbb{R}^n \\ g_1(x) \leq 0 \\ \vdots \\ g_m(x) \leq 0 \end{cases} \tag{2.2}$$

$f$ é a a função que representa o processo ou problema, o qual interessa otimizar. Ela pode ser classificada quanto a número de variáveis, dividindo-se em unidimensional e multidimensional, ou mesmo quanto ao número de máximos ou mínimos que possui, dividindo-se em unimodais ou multimodais.

O vetor de variáveis que serão modificadas durante o processo de otimização e representarão uma solução do problema é representado por $x$. O conjunto, espaço ou região que compreende as soluções possíveis ou viáveis sobre as variáveis de projeto $x$, do problema a ser otimizado, é conhecido como espaço de busca. O espaço de busca, por sua vez, é delimitado pelas funções de restrição representadas pela Eq.2.2.

Em otimização temos ainda o conceito de ponto ótimo, que se refere ao ponto no domínio de uma função em que ela atinge o seu valor extremo, ponto mínimo em problemas de minimização e ponto máximo em problemas de maximização. O valor obtido pela função $f$ no ponto ótimo

é chamado de valor ótimo. O valor ótimo pode ser, um *mínimo local* $\hat{f}$, de forma que:

$$\exists \epsilon > 0 \ \forall x \in \mathbb{R}^n : \|x - \hat{x}\| < \epsilon \Rightarrow \hat{f} \leq f(x) \tag{2.3}$$

Ou um *mínimo global* $f^*$ de forma que:

$$\forall x \in \mathbb{R}^n \ : \ f(x) > f(x^*) = f^* \tag{2.4}$$

Isso significa dizer que um ótimo local é a melhor solução dentro de sua vizinhaça e um ótimo global é a melhor solução de um determinado problema descrito ou modelado pela função $f$. Como $\max\{f(x)\} = -\min\{-f(x)\}$ as definições apresentadas nas equações Eq.2.2,2.3 e 2.4 mantêm sua generalidade.

### 2.1.1 Métodos de Otimização

Existe uma vasta gama de classificações e aplicações de métodos de otimização em todo tipo de problema. De forma geral, eles podem ser classificados em dois tipos: Métodos Determinísticos e Método Probabilísticos. Estes tipos serão descritos a seguir.

#### 2.1.1.1 Métodos Determinísticos

Os métodos de otimização baseados nos algoritmos determinísticos, que são a 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 do problema.

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.

Pelos motivos apresentados acima a aplicação deste tipo de método em funções multimodais e/ou não diferenciáveis tem seu desempenho limitado.

#### 2.1.1.2 Métodos Probabilísticos

Os métodos de otimização baseados nos algoritmos probabilísticos usam a avaliação da função objetivo para percorrer de forma "inteligente" o espaço de busca, de forma a encontrar soluções ótimas. Neste tipo de método, introduz-se no processo de otimização dados e parâmetros estocásticos.

Estes métodos apresentam algumas vantagens em relação aos métodos determinísticos, são elas:

- têm menor sensibilidade ao ponto de partida do processo de busca;

- não requerem que a função objetivo seja contínua ou diferenciável;
- trabalham adequadamente, tanto com variáveis contínuas quanto com discretas, ou ainda com uma combinação delas;
- não necessitam de formulações complexas ou reformulações para o problema.

## 2.2 Algoritmos Evolutivos

Algoritmos Evolutivos (AEs) são um tipo de método probabilístico que imitam o processo de evolução natural, processo condutor do surgimento de novas estruturas orgânicas mais complexas e bem adaptadas. Desde os anos 80 AEs tem sido utilizados para a resolução de problemas difíceis de otimização, como relatado em (Bäck et al., 1997).

AEs são algoritmos baseados em população, na qual um indivíduo pode ser afetado por outros indivíduos (e.g. competição por comida e acasalamento), assim como pode ser afetado pelo ambiente. Quanto mais adaptado for o indivíduo nestas condições, maior é a chance dele sobreviver por mais tempo e consequentemente gerar descendentes, que por sua vez herdam informação genética (mais ou menos distorcida) dos pais. No curso da evolução, este cenário leva à disceminação de informação genética de indivíduos acima da média, na população. Este modelo baseado na teoria da Evolução das Espécies de Darwin (Darwin, 1859) leva ao algoritmo evolutivo básico descrito a seguir:

**Algoritmo 2.1**: Algoritmo Evolutivo Básico

```
t ← 1
inicializa P(t)
avalia P(t)
while not condição_de_parada do
   P'(t) ← variação[P(t)]
   avalia [P'(t)]
   P(t+1) ← seleciona[P'(t) ∪ Q]
   t ← t+1
end while
```

Neste algoritmo, $P(t)$ simboliza a população de indivíduos na geração $t$. Geração é cada ciclo de variação + seleção. $Q$ é um conjunto especial de indivíduos que devem ser considerados para a seleção. Uma população de descendentes $P'(t)$ é gerada por meio de operadores de variação como mutação e/ou recombinação (apesar de outros tipos de operadores também serem possíveis) aplicados à população $P(t)$. A população de descendentes $P'(t)$ é avaliada calculando-se o valor da função objetivo $f$ para cada uma das soluções $x$, representadas pelos indivíduos de $P'(t)$. O valor de $f(x)$ dá uma medida do quão adaptado é o indivíduo que representa $x$, essa medida é comumente chamada de *fitness*. Prosseguindo com o algoritmo é feita uma seleção baseada no fitness que guia o processo na direção de soluções melhores.

A recombinação é um operador aplicado a dois ou mais indivíduous (chamados pais), que são "embaralhados" de alguma forma para gerar um ou mais indivíduous (chamados descendentes) e a mutação é um operador que introduz alterações aleatórias no material genético de um indivíduo. O caráter estocástico destes operadores garante a produção permanente de material genético novo, portanto, a criação de novos descendentes.

A classe de algoritmos evolutivos apresenta várias famílias de métodos como, os Algoritmos Genéticos (Goldberg, 1989), algoritmos de Sistemas Imunes Artificiais (AIS - Artificial Immune Systems)(Castro, 2002) e algoritmos de Otimização por Enxame de Partículas (PSO - Particle Swarm Optimization)(Kennedy e Eberhart, 1995), cada uma com suas próprias variações dos operadores de variação e seleção. O grande número de aplicações (Beasley, 1997) e o crescimento contínuo do interesse neste campo se devem ao fato dos AEs serem métodos de otimização global robustos que:

- podem encontrar várias soluções;
- podem otimizar um grande número de parâmetros discretos, contínuos ou combinações deles;
- realizam buscas simultâneas em várias regiões do espaço de busca;
- 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.

Um dos atuais estado-da-arte em AEs é o algoritmo de Evolução Diferencial (DE) (Teo, 2006). Ele será investigado neste trabalho e é descrito a seguir.

## 2.3 Evolução Diferencial

Tanto no meio insdustrial, quanto no meio acadêmico, os usuários de uma técnica de otimização desejam que ela preencha três requerimentos básicos:

1. o método deve achar o ótimo global, independente do estado inicial do processo de busca;
2. a convergência deve ser rápida; e
3. o método deve ter o mínimo de parâmetros de controle, para que ele seja fácil de usar.

Na busca por uma técnica que fosse de encontro aos critérios citados acima, em (Storn e Price, 1995) foi proposto o método conhecido como Evolução Diferencial (DE - Differential Evolution). O DE é um método, que além de ser incrivelmente simples, tem um ótimo desempenho numa grande variedade de problemas, como mostrado por Plagianakos et al. (2008). Além disso, por ser baseado em populações, o DE é inerentemente paralelo possibilitando sua implementação em conjuntos de computadores e processadores.

O sucesso do DE, deve-se principalmente ao seu poderoso e simples mecanismo de busca, o qual usa a diferença entre dois vetores, escolhidos aleatoriamente das soluções candidatas, para produzir novas soluções. À medida que a população evolui, a direção e o tamanho do passo da busca na mutação mudam, ajustando-se de acordo com a distribuição da população no espaço de busca. O DE usa uma abordagem gulosa e , ao mesmo tempo, estocástica para solucionar o problema de otimização, combinando operadores aritméticos simples com as operações clássicas de cruzamento, mutação e seleção, de modo que a população inicialmente aleatória possa evoluir para uma população com soluções de alta qualidade.

Apesar do método de evolução diferencial ser classificado como um algoritmo evolutivo, e como já pode-se perceber, se enquadra no esquema geral de algoritmos evolutivos (ver Fig.1.3), seu operador de mutação não tem inspiração em nenhum processo natural. Este operador, conhecido como *mutação diferencial*, se sustenta em argumentos matemáticos e heurísticos que indicam sua adequação para a otimização de funções, como poderemos ver a seguir.

### 2.3.1 Algoritmo Evolução Diferencial

Consideremos a versão irrestrita do problema genérico de otimização não linear com variáveis contínuas, formulado na Seção 2.1 e $\mathcal{U}_{[a,b]}$ a amostragem de uma variável aleatória com distribuição uniforme entre $a$ e $b$. De acordo com sua definição original, apresentada em (Storn e Price, 1995), o DE consiste no seguinte.

Seja uma população de soluções canditatas, escolhidas aleatoriamente no espaço de busca, representada por $X_t = \{x_{t,i}; i = 1, 2, ..., NP\}$. $t$ é o índice da geração corrente, $i$ é o índice do indivíduo na população e $NP$ é número de indivíduous na população. Cada indivíduo na população corrente é representado por um vetor coluna:

$$x_{t,i} = \begin{bmatrix} x_{t,i,1} \\ x_{t,i,2} \\ \vdots \\ x_{t,i,n} \end{bmatrix} \tag{2.5}$$

onde o terceiro índice indica uma entre as $n$ variáveis do problema de otimização.

O mecanismo de busca do DE utiliza a diferença entre pares de vetores escolhidos aleatoriamente dentre as soluções candidatas da população. O vetor resultante dessa diferença

é adicionado a uma terceira solução também escolhida aleatoriamente. A equação a seguir ilustra este procedimento:

$$v_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t,r3}) \quad r1 \neq r2 \neq r3 \neq i \tag{2.6}$$

$v_{t,i}$ representa a i-ésima solução mutante e $F$ é o fator de escala aplicado ao vetor diferencial e parâmetro do algoritmo DE. O vetor $x_{t,r1}$, ao qual é aplicada a mutação diferencial é denominado *vetor base*.

Após à realização deste procedimento é obtida a população mutante $V_t = \{v_{t,i}; i = 1, 2, ..., NP\}$. $V_t$ então entrará na fase de recombinação com a população corrente $X_t$, produzindo uma população de descendentes, chamados indivíduos teste, $U_t$. Originalmente é empregada a recombinação discreta com probabilidade $CR \in [0, 1]$ ilustrada pela equação a seguir:

$$u_{t,i,j} = \begin{cases} v_{t,i,j}, & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ x_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.7}$$

em que $\delta_i \in 1, \cdots, n$ é um índice aleatório sorteado para o vetor teste $i$ e serve para garantir que pelo menos uma variável da solução teste seja herdada do indivíduo mutante. Notem que o parâmetro $CR$ controla a fração de valores em $u_{t,i}$, que são copiados do vetor mutante $v_{t,i}$. Se $CR = 1$ o vetor teste será idêntico ao vetor mutante.

A Figura2.1 ilustra a geração de um vetor mutante e as possíveis soluções teste obtidas após a recombinação, indicadas por ■. Note que pelo menos uma coordenada $j = \delta_i$ será herdada do vetor mutante, de forma a garantir $u_{t,i} \neq x_{t,i}$. Pode-se observar que as soluções mutantes são geradas a partir de uma perturbação em algum indivíduo da população. A direção e o tamanho dessa perturbação são definidos pela diferença entre as soluções $x_{t,r2}$ e $x_{t,r3}$ , portanto dependem das posições relativas destes indivíduos no espaço de busca.

Finalmente, a solução teste $u_{t,i}$ é avaliada calculando-se seu valor da função objetivo. Cada solução teste $u_{t,i}$ é comparada com seu correspondente $x_{t,i}$ da população corrente e a melhor entre as duas passa para população da próxima geração, a pior dentre as duas é eliminada. Este procedimento é ilustrado pela equação a seguir:

$$x_{t+1,i} = \begin{cases} u_{t,i}, & \text{se } f(u_{t,i}) < f(x_{t,i}) \\ x_{t,i}, & \text{caso contrário} \end{cases} \tag{2.8}$$

Podemos observar ainda que as equações 2.6 e 2.7, podem ser escritas de forma compacta como segue:

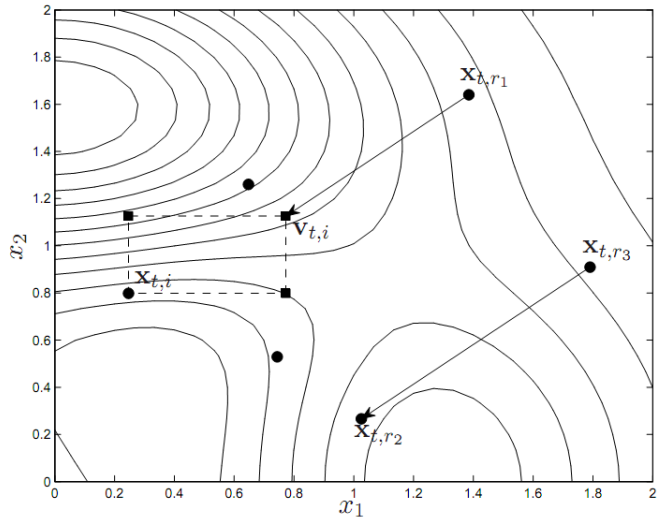


Figura 2.1: Ilustração do procedimento de geração de uma solução mutante (Guimarães, 2009)

$$u_{t,i,j} = \begin{cases} x_{t,r1} + F(x_{t,r2} - x_{t_r3}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ x_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.9}$$

logo, as equações 2.9 e 2.8 descrevem todas as operações do algoritmo de evolução diferencial em sua versão original. O pseudocódigo do DE é apresentado no Algoritmo2.2.

### 2.3.2 Comportamento da Mutação Diferencial

Apesar de sua simplicidade, como pudemos notar na seção anterior, para entender melhor o funcionamento do algoritmo de evolução diferencial, é necessário compreender o funcionamento do seu mecanismo busca sustentado pela mutação diferencial.

O princípio básico de funcionamento do algoritmo de evolução diferencial é perturbar soluções da população corrente gerando vetores mutantes. Essas perturbações, também chamadas de *vetores diferenciais*, são proporcionais à diferença entre pares de soluções escolhidas aleatoriamente na população. Portanto, sua distribuição depende da distribuição espacial dos indivíduos da população no problema em questão. À medida que a população se distribui de acordo com o "relevo" da função, a distribuição e o tamanho dos vetores diferenciais também se ajusta. Para entender melhor o comportamento do algoritmo verifiquemos a distribuição dos possíveis vetores diferenciais em diversos instantes do processo de otimização.

As Figuras 2.2 a 2.4 ilustram a propriedade de autoajuste dos vetores diferenciais em uma função quadrádica, cujas curvas de nível correspondem a elipsóides rotacionados de $\pi/4$ no sentido anti-horário em relação aos eixos coordenados. Além disso, um dos eixos desse elipsóide é maior do que o outro, tornando o elipsóide "alongado" numa dada direção.

**Algoritmo 2.2**: Algoritmo de Evolução Diferencial

$t \leftarrow 1$
Inicializar População $X_t = \{x_{t,i}; i = 1, 2, ..., NP\}$
Avalia $X_t$
**while not** condição_de_parada **do**
  **for** $i = 1$ to $NP$ **do**
    Selecione Aleatoriamente $r_1, r_2, r_3 \in 1, \cdots, NP$
    Selecione Aleatoriamente $\delta_i \in 1, \cdots, n$
    **for** $j = 1$ to $n$ **do**
      **if** $\mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i$ **then**
        $u_{t,i,j} = x_{t,r1} + F(x_{t,r2} - x_{t_r3})$
      **else**
        $u_{t,i,j} = x_{t,i,j}$
      **end if**
    **end for**
  **end for**
  **for** $i = 1$ to $NP$ **do**
    **if** $f(u_{t,i}) < f(x_{t,i})$ **then**
      $x_{t+1,i} \leftarrow u_{t,i}$
    **else**
      $x_{t+1,i} \leftarrow x_{t,i}$
    **end if**
  **end for**
  $t \leftarrow t + 1$
**end while**

A Figura 2.2-(a) mostra a configuração inicial da população $X_t$ no espaço de busca. Neste estágio as soluções estão distribuídas aleatoriamente com distribuição uniforme na região retangular correspondentes aos limites inferiores e superiores de cada variável. A Figura 2.2-(b) mostra que a distribuição de vetores diferencias correspondente, não possui nenhum tipo de polarização e é formada por vetores de diversos tamanhos.

À medida que o processo de otimização continua, novas soluções vão sendo geradas. Soluções ruins, com maior valor de função objetivo, vão sendo substituidas por soluções mais próximas do mínimo, alterando a distribuição das soluções da população $X_t$.

A Figura 2.3-(a) ilustra a distribuição espacial 10 gerações após a distribuição inicial. Observe que neste estágio a distribuição dos vetores diferencias correspondente, mostrada na Figura 2.3-(b), se alinha à forma da função, possuindo direções mais favoráveis à sua minimização. Outro fator importante a ser observado é que o tamanho do vetores, que darão os tamanhos das pertubações a serem aplicadas aos demais indivíduos, diminuiram devido à aglomeração dos indivíduos em torno do ponto de mínimo.

A Figura 2.4 ilustra a distribuição espacial da população e a distribuição dos vetores diferenciais 20 gerações após a distribuição inicial. A população está agora mais próxima do

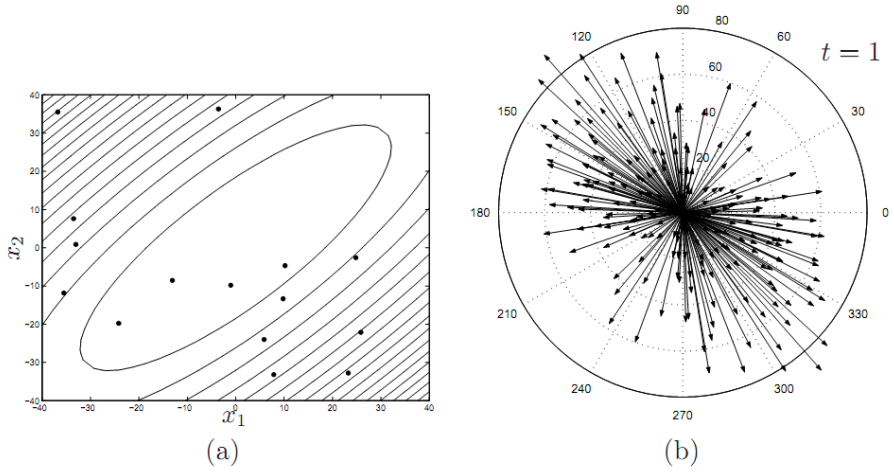


Figura 2.2: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 1$ (b) Distribuição dos vetores diferenciais na geração $t = 1$ (Guimarães, 2009)

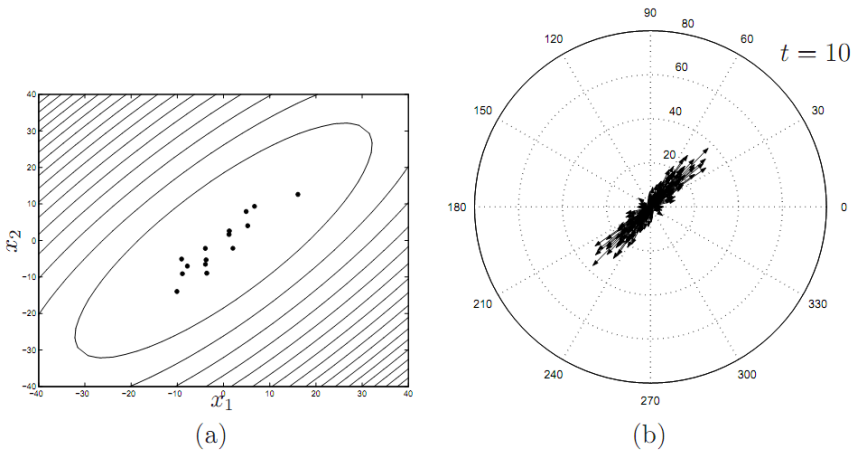


Figura 2.3: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 10$ (b) Distribuição dos vetores diferenciais na geração $t = 10$ (Guimarães, 2009)

ponto de mínimo e ocupa um volume reduzido em relação á distribuição espacial em $t = 1$. A distribuição dos vetores diferenciais continua alinhada com os elipsóides que formam as curvas de nível da função, porém os tamanhos desses vetores estão bem reduzidos, favorecendo a intensificação da busca. Nesse momento, o algoritmo converge para o ótimo global.

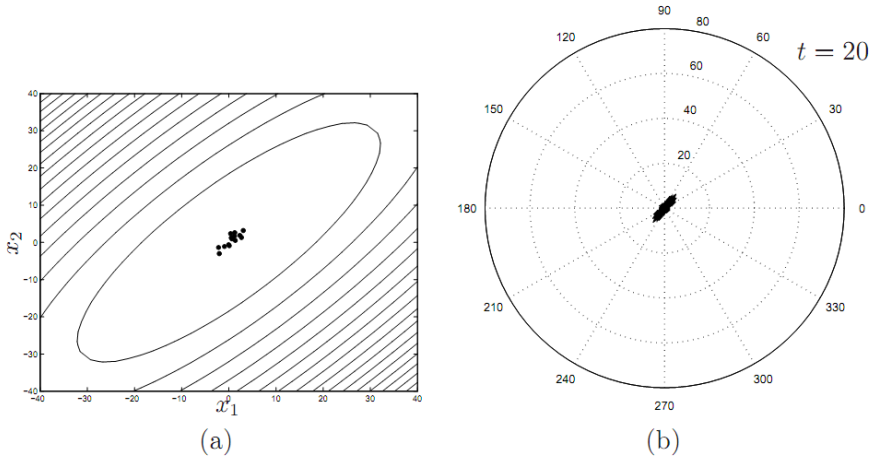


Figura 2.4: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 20$ (b) Distribuição dos vetores diferenciais na geração $t = 20$ (Guimarães, 2009)

Este exemplo ilustra o comportamento geral do algoritmo de evolução diferencial no processo de otimização, mostrando claramente a adaptação dos tamanhos e das direções das mutações.Como pudemos notar o DE possui qualidades importantes como, a capacidade de adaptar-se à estrutura da função objetivo e simplicidade de implementação, características essas que o tornam o importante otimizador que é hoje.

# Capítulo 3

# Configuração de Parâmetros

A definição da representação das soluções e a definição da função objetivo são itens que formam a ponte entre o contexto original do problema que deseja-se tratar e o método de resolução. Quando este método é um algoritmo evolutivo, precisamos definir seus componentes, como operadores de variação (mutação e recombinação) que sejam adequados à representação, o mecanismo de seleção, responsável por manter no processo evolutivo boas soluções e a população inicial. Cada um destes componentes deve possuir parâmetros, em princípio: a probabilidade de mutação, o número de indivíduos envolvidos na seleção e o tamanho da população. O valor destes parâmetros determinam fortemente a qualidade das soluções encontradas e o tempo gasto para encontrá-las Eiben et al. (1999).

Normalmente a escolha dos parâmetros é feita por tentativa e erro numa bateria de testes preliminares, que por sua vez demandam um tempo considerável. Para tratar esta questão métodos de configuração automática de parâmetros têm sido desenvolvidos. Estes métodos podem ser classificado de três formas (Eiben et al., 1999):

- Determinístico: Ocorre quando o valor dos parâmetros é alterado por alguma regra determinística sem uso de nenhum feedback do processo de busca.
- Adaptativo: Ocorre quando alguma forma de feedback do processo de busca é usada para determinar a direção e/ou magnitude da alteração do parâmetro.
- Autoadaptativo: Ocorre quando os parâmetros a serem configurados são codificados na representação do indivíduo e passam a sofrer ação dos operadores de variação (mutação e recombinação). Quanto melhores forem os valores dos parâmetros codificados, melhores as soluções geradas por eles. Estas soluções por sua vez possuem uma maior chance de sobreviver e produzir descendentes, propagando estes bons valores de parâmetro.

Neste trabalho é investigada a terceira forma, a autoadaptação de parâmetros. Esta forma, em princípio não exige a formulação de heurísticas específicas para determinar quando um

determinado valor de parâmetro é bom ou não e o que fazer em relação ao valor do parâmetro dada sua qualidade. Na autoadaptação o próprio processo evolutivo se encarrega de alterar e propagar bons valores de parâmetros o que torna esta abordagem mais simples e genérica. Uma análise mais detalhada desta forma de controle de parâmetro e sua aplicação no algoritmo de evolução diferencial é feita a seguir.

## 3.1 Autoadaptação de Parâmetros

Algoritmos Evolutivos são processos inerentementes dinâmicos e adaptativos, logo o uso de parâmetros rígidos que não têm seus valores alterados ao longo do tempo parece ir contra estes princípios. Adicionalmente parece intuitivo que bons valores de parâmetros podem depender do estágio no qual o algoritmo se encontra. Em princípio, grandes pertubações nas soluções podem ser boas nas primeiras gerações, por aumentarem o poder de exploração da busca. Pertubações menores devem ser necessárias em gerações mais avançadas para intensificar a busca, refinando as boas soluções já encontradas.

Contudo, encontrar bons valores de parâmetros para um algoritmo evolutivo é um problema complexo, mal definido e fracamente estruturado. Talvez por coincidência é este o tipo de problema no qual AEs costumam ter um desempenho melhor do que outros métodos (Eiben et al., 1999). Torna-se natural então usar o próprio AE para controlar os valores dos parâmetros automaticamente.

Nestas bases em (Schwefel, 1981), foi introduzido um mecanismo interno de controle da magnitude da perturbação, introduzindo os parâmetros na representação do indivíduos, facilitando a autoadaptação dos parâmetros pelo próprio processo evolutivo. Desta forma, a busca era feita no espaço de soluções e no espaço de parâmetros simultaneamente, permitindo a adequação dos parâmetros à circunstância atual do processo de busca. A codificação do indivíduo é apresentada na Eq.3.1

$$\langle x_1, x_2, \cdots, x_n, \sigma \rangle \tag{3.1}$$

onde $\{x_1, x_2, \cdots, x_n\}$ são as variáveis do problema e $\sigma$ é o parâmetro que determina a magnitude da perturbação. Agora tanto as variáveis do problema ($x_i$) quanto o valor do parâmetro passam a sofrer ação do operador de variação. Um variação típica seria:

$$\sigma' = \sigma + e^{(\mathcal{N}_{[0,\tau_0]})} \tag{3.2}$$

onde $\mathcal{N}_{[a,b]}$ representa a amostragem de uma variável aleatória com distribuição normal e média em $a$ e desvio padrão igual a $b$ e $\tau_0$ é um parâmetro do método. Após feita a variação

do parâmetro modificam-se as variáveis do problema com o seguinte esquema:

$$x_i{}' = x_i + \mathcal{N}_{[0,\sigma']} \tag{3.3}$$

Um estudo feito por Gehlhaar e Fogel (1996) indica que a ordem das variações tem um forte impacto na efetividade da autoadaptação. Parece ser importante variar os parâmetros primeiro e depois usá-los para modificar as variáveis do problema. Como os autores apontam neste estudo, o mecanismo contrário sofre por gerar descendentes que possuem bons vetores de variáveis mas possuem valores de parâmetro ruins. Isso ocorre pois os valores de parâmetro herdados não foram os valores utilizados para gerar o vetor de variáveis do indivíduo ao qual pertencem.

Tipicamente os valores iniciais dos parâmetros são selecionados aleatoriamente. Cada indivíduo agora passa então a ter seu próprio conjunto de parâmetros, os quais serão usados para sua variação. O que definirá se esses parâmetros se propagaram para as gerações seguintes será apenas a qualidade das soluções geradas por eles.

## 3.2 Autoadaptação de Parâmetros na Evolução Diferencial

Como visto na Seção2.3.1 o algoritmo de evolução diferencial (DE) é um algoritmo para otimização global em espaços contínuos (Storn e Price, 1995, 1997), que também pode ser usado para variáveis discretas (Prado et al., 2010; Onwubolu e Davendra, 2009). O DE cria novas soluções candidatas pertubando os indivíduous da população corrente com vetores criados pela diferença de duas outras soluções, ver Eq.2.6.

Como visto anteriormente, este esquema à primeira vista parece ser muito eficiente, porém ele esconde uma limitação. Se por alguma razão o algoritmo falhar em gerar soluções que superam seus pais, a busca será repetida novamente com os mesmos tamanhos de pertubação, que provavelmente também irão falhar caindo numa condição não desejada de estagnação (Lampinen e Zelinka, 2000). Estagnação é o fenômeno que ocorre quando um algoritmo baseado em população não converge para uma solução (nem sub-ótima) e a diversidade da população continua alta. No caso do DE, a estagnação ocorre quando o algoritmo não consegue melhorar nennhuma solução da população por um número considerável de gerações. Em outras palavras, o principal problema do DE é que ele possui um número limitado de movimentos exploratórios e se estes movimentos não forem suficientes para gerar novas boas soluções, a busca pode ser fortemente comprometida.

Fica claro então, que o sucesso do DE depende de uma boa configuração de seus três parâmetros de controle, sendo eles:

- $F$: Fator de escala das perturbações geradas na mutação;
- $CR$: A probablidade de cruzamento; e

- $NP$: O tamanho da população.

Como relatado em (Liu e Lampinen, 2002), o conjunto mais adequado destes parâmetros depende da função que se quer otimizar e dos requerimentos de tempo e precisão do problema. O problema de configuração de parâmetros é ainda enfatizado quando o DE é empregado para tratar de dificuldades de problemas do mundo real como alta dimensionalidade e ruído. Funções com muitas dimensões exigem um conjunto maior de movimentos possíveis para aumentar a capacidade do DE em encontrar novas boas soluções. Já em problemas com ruído, o DE emprega muita lógica determinística na busca e por isso tem dificuldade para lidar com ambientes ruidosos, nos quais pode haver estagnação (Neri e Tirronen, 2010).

Neste contexto a autoadaptação surge como uma opção para dar ao DE original este número extra de movimentos que ele precisa para evitar a estagnação e se tornar robusto numa maior gama de problemas. A seguir serão apresentadas e analisadas, as três principais versões autoadativas do DE. Após esta análise será apresentada a versão proposta neste trabalho que pretende eliminar os problemas apontados nas outras versões.

### 3.2.1 Autoadaptação para o Ajuste de Parâmetros na Evolução Diferencial - jDE

Para evitar o ajuste manual dos parâmetros F e CR do DE, em (Brest e Maucec, 2006) foi proposta a estratégia "Autoadaptação para o Ajuste de Parâmetros na Evolução Diferencial". O algoritmo DE utilizando esta estratégia, nomeada de jDE funciona da seguinte forma.

Quando a população inicial é gerada, dois valores extras no intervalo de 0 a 1 também são gerados para cada indivíduo. Esses valores representam o F e o CR relacionados ao indivíduo em análise. Sendo assim, cada indivíduo será representado da seguinte forma:

$$x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}, F_i, CR_i \rangle \tag{3.4}$$

De acordo com a lógica da autoadaptação, ver (Gehlhaar e Fogel, 1996), a variação dos parâmetros precede a execução das operações, de forma que, a cada geração o i-ésimo indivíduo $x_i$ em análise tem seus parâmetros F e CR alterados de acordo com o seguinte esquema:

$$F_i = \begin{cases} F_l + F_u * \mathcal{U}_{[0,1]}, & \text{se } \mathcal{U}_{[0,1]} < \tau_1 \\ F_i, & \text{caso contrário} \end{cases} \tag{3.5}$$

$$CR_i = \begin{cases} \mathcal{U}_{[0,1]}, & \text{se } \mathcal{U}_{[0,1]} < \tau_2 \\ CR_i, & \text{caso contrário} \end{cases} \tag{3.6}$$

onde $\mathcal{U}_{[0,1]}$, é a amostragem de uma variável aleatória com distribuição uniforme entre 0 e 1. $\tau_1$ e $\tau_2$ são valores constantes (0.1) que representam a probabilidade de alteração dos parâmetros $F$ e $CR$ respectivamente, $F_l$ e $F_u$ são constantes que representam o valor mínimo e máximo

que podem ser atribuídos ao parâmetro $F$ , respectivamente. O novos valores calculados de $F_i$ e $CR_i$ são então usados para a variação de $x_i$.

Apesar de aparentemente melhorar as propriedades de robustez do algoritmo DE original (Brest e Maucec, 2006; Zamuda et al., 2009; Brest, 2009), este esquema não é verdadeiramente autoadaptativo. A alteração dos parâmetros não leva em consideração os valores anteriores, não utilizando a experiência adquirida pelo algorimo durante o processo de otimização. A escolha aleatória de novos parâmetros com probabilidades $\tau_1$ e $\tau_2$, pode acabar por destruir bons valores.

### 3.2.2 Evolução Diferencial Autoadaptativa - SaDE

A primeira versão do algoritmo de Evolução Diferencial Autoadaptativo (SaDE - Self-adaptive Differential Evolution) foi proposta em (Qin e Suganthan, 2005) e uma versão mais sofisticada, a qual será descrita nesta seção, foi proposta recentemente em (Qin et al., 2009). A principal característica dessa abordagem é o emprego de múltiplas variações da estratégia de mutação do DE (no caso quatro) sendo elas:

- rand/1

$$v_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t,r3}) \tag{3.7}$$

- rand-to-best/2

$$v_{t,i} = x_{t,i} + F(x_{t,best} - x_{t,i}) + F(x_{t,r1} - x_{t,r2}) + F(x_{t,r3} - x_{t,r4}) \tag{3.8}$$

- rand/2

$$v_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t,r3}) + F(x_{t,r4} - x_{t,r5}) \tag{3.9}$$

- current-to-rand/1

$$v_{t,i} = x_{t,i} + F(x_{t,r1} - x_{t,i}) + F(x_{t,r2} - x_{t,r3}) \tag{3.10}$$

onde *best* é o índice do melhor indivíduo da população.

Neste caso, teremos codificados no indivíduo probabilidades relacionadas à aplicação das estratégias de mutação, probabilidades de cruzamento $CR$, para cada uma das estratégias e um parâmetro $F$ geral. Mais especificamente um indivíduo da população terá a seguinte configuração:

$$x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}, F_i, CR_i^1, CR_i^2, CR_i^3, CR_i^4, p_i^1, p_i^2, p_i^3, p_i^4 \rangle \tag{3.11}$$

onde $p_i^k$ para $k = 1, 2, 3, 4$ é a probabilidade da estratégia de mutação 1,2,3 ou 4 ser empregada na variação do indivíduo $x_i$. É importante ressaltar que $\sum_{k=1}^{4} p_i^k = 1$.

Quando a população inicial é gerada, as probabilidades $p_i^k$ de cada indivíduo são fixadas em 0.25. Durante as PA gerações seguintes (PA se refere a período de aprendizado), para cada indivíduo, o número de gerações bem sucedidas $n_s^k$ relativa a uma determinda estratégia de mutação , i.e., o número de descendentes gerados pela k-ésima estratégia que possuem um melhor valor de função objetivo que o pai, é salvo. De maneira análoga é salvo o número de gerações mal sucedidas $n_f^k$.

No final do período de aprendizado (depois de PA gerações), as probabilidades são atualizadas para cada indivíduo $x_i$, em cada geração $G$, de acordo com a fórmula:

$$p_i^k = \frac{S_i^k}{\sum_{k=1}^{4} S_i^k} \tag{3.12}$$

onde:

$$S_i^k = \frac{\sum_{g=G-PA}^{G-1} n_s^k}{\sum_{g=G-PA}^{G-1} n_s^k + \sum_{g=G-PA}^{G-1} n_f^k} + \epsilon \tag{3.13}$$

onde $g$ é o índice da geração e $\epsilon$ é uma constante de valor 0.01 cujo objetivo é garantir a estabilidade numérica do algoritmo caso $S_i^k = 0$ para todas as estratégias. Portando $S_i^k$ representa a taxa de sucesso da descendência gerada em termos da k-ésima estratégia. A divisão feita na Eq.3.12, serve para garantir a normalização das probabilidades. Dadas as quatro probabilidades a estratégia a ser realizada será escolhida através do procedimento de amostragem universal estocástica, ver (Baker, 1987).

O ajuste do parâmetro $CR$ também é "autoadaptado". Quando a população inicial é gerada, $CR_i^k$ é fixado em 0.5 para todos os indivíduos para todas as estratégias. Durante as primeiras PA gerações, cada $CR_i^k$ é atualizado da seguinte forma:

$$CR_i^k = \mathcal{N}_{[CR_i^k, 0.1]} \tag{3.14}$$

Durante o período de aprendizado, para cada estratégia de mutação, o conjunto de taxas de cruzamento que levam à geração de descendentes bem sucedidos (indivíduos que possuem melhor valor de função que seu pai) $CR_s^k$ são salvos. Depois do período de aprendizado, $CR_i^k$ é atualizado usando-se a mediana do conjunto $CR_s^k$ no esquema a seguir:

$$CR_i^k = \mathcal{N}_{[\text{mediana}(CR_s^k), 0.1]} \tag{3.15}$$

A atualização do parâmetro $F$ não utiliza nenhuma forma de feedback e são feitas de acordo com a equação abaixo:

$$F_i = \mathcal{N}_{[0.5, 0.3]} \tag{3.16}$$

Apesar dos parâmetros estarem codificados nos indivíduos essa abordagem não é verdadeiramente autoadaptativa, o termo mais correto a ser usado neste caso seria que essa abordagem é adaptativa, uma vez que ela usa feedback do algoritmo na atualização dos parâmetros. A

adaptação exige a criação de uma heurística para determinar que atitudes tomar em relação aos valores dos parâmetros, e o desempenho da estratégia fica dependente da heurística aplicada.

Alguns problemas do SaDE são:

1. Não adaptação ou autoadaptação do parâmetro F ao qual o DE é bastante sensível como exposto por (Brest e Maucec, 2006);

2. A criação de novos parâmetros, como os parâmetros das normais aplicadas para variação de $F$ e $CR$. Apesar de em sua proposta os autores manterem estes valores fixos, não se conhece sua influência no processo; e

3. A utilização de um único $F$ para todas as estratégias. Uma vez que, as características dos vetores de pertubação criados em cada estratégia é diferente, é possível que diferentes estratégias exijam fatores de escala $F$ diferentes.

### 3.2.3 Evolução Diferencial com População Autoadaptativa - DESAP

O algoritmo de Evolução Diferencial com População autoadaptativa (DESAP - Diffrential Evolution with Self-adapting Population) foi proposto por (Teo, 2006). A principal característica deste algoritmo é autoadaptar o tamanho da população, dado pelo parâmetro $NP$, de forma que ela se ajuste dinamicamente. Além disso, uma estratégia autoadaptativa também é usada para controlar as taxas de recombinação e mutação do algoritmo proposto. Sendo assim, a codificação do indivíduo ficará da seguinte forma:

$$x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}, M_i, CR_i, NP_i \rangle \tag{3.17}$$

O autor propõe duas versões do DESAP, onde a primeira utiliza uma codificação absoluta para o tamanho da população e a segunda utiliza uma codificação relativa para o tamanho da população. Estas versões são chamadas de DESAP-Abs e DESAP-Rel respectivamente. Na inicialização da população o parâmetro NP será fixado da seguinte forma.
No DESAP-Abs:

$$NP = round(10 * D + \mathcal{N}_{[0,1]}) \tag{3.18}$$

No DESAP-Rel:

$$NP = \mathcal{U}_{[-0.5,0.5]} \tag{3.19}$$

onde $D$ é o número de dimensões do problema.

Uma vez inicializada a população, a cada geração cada indivíduo em análise entra numa fase chamada pelo autor de recombinação, descrita no Algoritmo3.1. Nesta fase tanto o vetor de variáveis do problema quanto os parâmetros codificados nos indivíduos são alterados pelo uso da Eq.2.6 da própria mutação diferencial, note que segundo a proposta dos autores o

parâmetro $F$ é fixo com valor 1. A notação $A'$, representa o valor gerado pela variação do parâmetro ou do vetor de variáveis $A$ e $round(B)$ representa o resultado do arredondamento do número $B$.

**Algoritmo 3.1**: Fase de Recombinação do DESAP

**if** $\mathcal{U}_{[0,1]} < CR_i$ **then**
  $x'_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t_r3})$
  $CR'_{t,i} = CR_{t,r1} + F(CR_{t,r2} - CR_{t_r3})$
  $M'_{t,i} = M_{t,r1} + F(M_{t,r2} - M_{t_r3})$
  **DESAP - Abs:**
  $NP'_{t,i} = NP_{t,r1} + round(F(NP_{t,r2} - NP_{t_r3}))$
  **DESAP - Rel:**
  $NP'_{t,i} = NP_{t,r1} + F(NP_{t,r2} - M_{NP_r3})$
**else**
  $x'_{t,i} = x_{t,i}$
  $CR'_{t,i} = CR_{t,i}$
  $M'_{t,i} = M_{t,i}$
  $NP'_{t,i} = NP_{t,i}$
**end if**

Depois da fase de recombinação o indivíduo em análise entra na fase chamada pelo autor de mutação, fase esta que não existe no DE original, e é descrita pelo Algoritmo3.2.

**Algoritmo 3.2**: Fase de Mutação do DESAP

**if** $\mathcal{U}_{[0,1]} < M_i$ **then**
  $x'_{t,i} = x_{t,i} + \mathcal{N}_{[0,M_i]}$
  $CR'_{t,i} = \mathcal{N}_{[0,1]}$
  $M'_{t,i} = \mathcal{N}_{[0,1]}$
  **DESAP - Abs:**
  $NP'_{t,i} = NP_{t,i} + round(\mathcal{N}_{[0.5,1]})$
  **DESAP - Rel:**
  $NP'_{t,i} = NP_{t,i} + \mathcal{N}_{[0,M_i]}$
**else**
  $x'_{t,i} = x_{t,i}$
  $CR'_{t,i} = CR_{t,i}$
  $M'_{t,i} = M_{t,i}$
  $NP'_{t,i} = NP_{t,i}$
**end if**

Ralizadas as duas fases para todos os indivíduos da população, passa-se para a atualização do tamanho da população da próxima geração. O cálculo do novo valor de $NP$ é feito como segue:

DESAP-Abs:

$$NP = round\left(\sum_{i=1}^{NP_{atual}} \frac{NP_i}{NP_{atual}}\right) \tag{3.20}$$

DESAP-Rel:

$$NP = round(NP_i + (NP_{atual} * NP_i)) \tag{3.21}$$

Caso a população aumente, são feitos clones do melhor indivíduo até que ela fique completa, se a população diminui são selecionados os $NP$ melhores indivíduos da população anterior.

Apesar desta abordagem fazer uma autoadaptação verdadeira nos parâmetros $M$ e $CR$ na fases de recombinação ela falha em alguns pontos, são eles:

1. A alteração dos parâmetros é feita depois da realização das operações o que pode ser improdutivo, gerando boas soluções com parâmetros ruins, como relatado em (Gehlhaar e Fogel, 1996);

2. O parâmetro $NP$ não parece ser uma boa escolha para ser um parâmetro autoadaptado, uma vez que ele não interfere diretamente no valor de função de um único indivíduo, ou seja, numa população com 100 indivíduos as soluções geradas sofrem ação de $NP = 100$ independente do valor que elas possuem em sua codificação. Esta afirmação ainda é reforçada pelo fato de que para atualizar a população precisa-se dos parâmetros de todos os indivíduous, demonstrando a falta de importância do parâmetro tamanho da população num indivíduo isolado; e

3. Não adaptação do parâmetro $F$, ao qual o desempenho do DE é bastante sensível (Brest e Maucec, 2006).

### 3.2.4 Evolução Diferencial com Mutação Autoadaptativa (SaMDE) - Abordagem Proposta

Para lidar com os problemas do DE original com parâmetros fixos, como estagnação, convergência lenta e dificuldade para configurar bons parâmetros para um determinado problema, é proposta neste trabalho uma nova abordagem autoadaptativa para o DE a Evolução Diferencial com Mutação Autoadaptativa (SaMDE - Self-adaptive Mutation Differential Evolution).

O SaMDE faz autoadaptação dos parâmetros $F$ e $CR$ para múltiplas estratégias de mutação, inclusive autoadaptando as probabilidades de aplicação destas estratégias. Com essa nova abordagem pretende-se também eliminar ou pelo menos minimizar os problemas apontados nas outras estratégias. As estratégias de mutação foram escolhidas de forma a cobrir a maior variedade de problemas possível. Elas são apresentadas abaixo:

1. **rand/1** - É a estratégia do DE original e normalmente apresenta convergência lenta e forte capacidade de exploração. Graças a estas características, ela é considerada a

estratégia mais adequada para resolução de problemas multimodais (Qin et al., 2009).

$$v_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t,r3}) \tag{3.22}$$

2. **best/1** - Esta estratégia normalmente apresenta convergência rápida e possui bom desempenho em funções unimodais.

$$v_{t,i} = x_{t,best} + F(x_{t,r1} - x_{t,r2}) \tag{3.23}$$

3. **rand/2** - Neste tipo de estratégia a distribuição estatística da soma de todos os pares de vetores diferença têm a forma de sino (Qin et al., 2009), o que dá uma forma diferente de pertubação das soluções candidatas.

$$v_{t,i} = x_{t,r1} + F(x_{t,r2} - x_{t,r3}) + F(x_{t,r4} - x_{t,r5}) \tag{3.24}$$

4. **current-to-rand/1** - É uma estratégia invariante a rotação, sem recombinação, que teve sua eficiência verificada quando foi aplicada em problemas de otimização multi-objetivo (Iorio e Li, 2004).

$$v_{t,i} = x_{t,i} + F(x_{t,r1} - x_{t,i}) + F(x_{t,r2} - x_{t,r3}) \tag{3.25}$$

Daqui por diante as diferentes estratégias de mutação serão refenciadas pelos números da enumeração acima. Temos então que no SaMDE um indivíduo é codificado da seguinte forma:

$$x_i = \langle x_{i,1}, ..., x_{i,D}, V_i^1, V_i^2, V_i^3, V_i^4, F_i^1, CR_i^1, F_i^2, CR_i^2, F_i^3, CR_i^3, F_i^4, CR_i^4 \rangle \tag{3.26}$$

onde $x_{i,d}$ são as variáveis de otimização, $V_i^k$ para $k = 1, 2, 3, 4$ é um valor no intervalo $[0, 1]$ que confere maior ou menor chance de aplicação da estratégia $k$. Os pares $F_i^k \in [0, 1], CR_i^k \in [0, 1]$ representam os parâmetros $F$ e $CR$ da estratégia $k$. Como foi observado anteriormente o parâmetro que indica o tamanho da população ($NP$) não deve ser autoadaptado, portanto este não foi inserido na codificação do indivíduo. Para gerar perturbações nos parâmetros é usada a própria mutação diferencial, que, como foi visto anteriormente, fornece uma boa estratégia de busca.

Na criação da população os parâmetros são inicializados de forma aleatória nos intervalos mencionados acima. Então para cada indivíduo da população, primeiramente altera-se cada um dos parâmetros $V$ da seguinte forma:

$$V_i^k = V_{r1}^k + F'(V_{r2}^k - V_{r3}^k) \tag{3.27}$$

Com os valores dos $Vs$ atualizados determina-se a estratégia de mutação vencedora, ou seja a estratégia que será aplicada ao indivíduo em análise. A determinação da estratégia vencedora é feita através do algoritmo de seleção por roleta, de forma que as estratégias com maiores valores de $V$ tem maior probabilidade de serem selecionadas.

Feita a seleção da estratégia vencedora apenas os $F$ e $CR$ da estratégia correspondente serão alterados, pois somente seus valores poderão ser avaliados durante a fase de seleção. A alteração de $F$ e $CR$ é feita como segue.

$$F_i^w = F_{r1}^k + F'(F_{r2}^w - F_{r3}^w) \tag{3.28}$$

$$CR_i^w = CR_{r1}^w + F'(CR_{r2}^w - CR_{r3}^w) \tag{3.29}$$

onde $w$ representa o número da estratégia vencedora e $F'$ é sorteado aleatoriamente no intervalo $[0.8, 1]$, antes da modificação dos $Vs$.

Atualizados o $F$ e o $CR$ correspondentes aplicam-se às variáveis do problema, os operadores mutação (utilizando a estratégia vencedora) e recombinação, lembrando que a estratégia current-to-rand/1 não possui recombinação. Antecedendo a fase de seleção recombina-se os parâmetros originais com os parâmetros alterados, utilizando-se também o $CR$ da estratégia vencedora. Para finalizar, o indivíduo gerado por mutação e recombinação passa pela fase de seleção com o indivíduo em análise (Ver Eq.2.8). Este procedimento é feito para todos os indivíduos a cada geração até que algum critério de parada seja atingido.

Para manter tanto as variáveis do problema quanto os parâmetros dentro dos limites pré-estabelecidos, foi usado o método proposto por Ronkkonen et al. (2005). Este método sugere que a violação dos limites deve ser refletida nos limites. Este esquema é apresentado abaixo:

$$x = \begin{cases} 2 * x_{inferior} - x & \text{se } x < x_{inferior} \\ 2 * x_{superior} - x & \text{se } x > x_{superior} \end{cases} \tag{3.30}$$

onde $x$ pode representar uma variável do problema ou um parâmetro, $x_{inferior}$ e $x_{superior}$ referem-se ao limite inferior e ao limite superior da variável ou parâmetro em análise, respectivamente.

Com o objetivo de tornar o SaMDE mais claro seu pseudo-código é apresentado no Algortimo 3.3.

A estratégia proposta é completamente autoadaptativa, não utiliza nenhum tipo de feedback explícito do processo de busca e possui um único parâmetro extra $F'$ que não precisa ser modificado externamente, como mostram os resultados apresentados na Seção 4.3.1.

**Algoritmo 3.3**: Algoritmo de Evolução Diferencial com Mutação Autoadaptativa

$t \leftarrow 1$
Inicializar População $X_t = \{x_{t,i}; i = 1, 2, ..., NP\}$
Avalia $X_t$
**while not** condição_de_parada **do**
  **for** $i = 1$ to $NP$ **do**
    Selecione Aleatoriamente $r_1, r_2, r_3, r_4, r_5 \in 1, \cdots, NP$
    Selecione Aleatoriamente $\delta_i \in 1, \cdots, n$
    Selecione Aleatoriamente $F' \in [0.8, 1]$
    $**$ Variação dos Vs $**$
    **for** $k = 1$ to Número_de_Estratégias **do**
      $V_i^k = V_{r1}^k + F'(V_{r2}^k - V_{r3}^k)$
    **end for**
    $w \leftarrow$ Estratégia escolhida pela roleta
    $**$ Variação dos Parâmetros da Estratégia Escolhida $**$
    $F_i^w = F_{r1}^w + F'(F_{r2}^w - F_{r3}^w)$
    $CR_i^w = CR_{r1}^w + F'(CR_{r2}^w - CR_{r3}^w)$
    $**$ Alteração das Variáveis dos Problema $**$
    **for** $j = 1$ to $n$ **do**
      **if** $\mathcal{U}_{[0,1]} \leq CR_i^w \vee j = \delta_i \vee w = 4$ **then**
        **if** $w = 1$ **then**
          $u_{t,i,j} \leftarrow x_{t,r1,j} + F_i^w(x_{t,r2,j} - x_{t,r3,j})$
        **else if** $w = 2$ **then**
          $u_{t,i,j} \leftarrow x_{t,best,j} + F_i^w(x_{t,r1,j} - x_{t,r2,j})$
        **else if** $w = 3$ **then**
          $u_{t,i,j} \leftarrow x_{t,r1,j} + F_i^w(x_{t,r2,j} - x_{t,r3,j}) + F_i^w(x_{t,r4,j} - x_{t,r5,j})$
        **else if** $w = 4$ **then**
          $u_{t,i,j} \leftarrow x_{t,i,j} + F_i^w(x_{t,r1,j} - x_{t,i,j}) + F_i^w(x_{t,r2,j} - x_{t,r3,j})$
        **end if**
      **else**
        $u_{t,i,j} \leftarrow x_{t,i,j}$
      **end if**
    **end for**
    $**$ Recombinação dos Parâmetros $**$
    **for** $j = n$ to $n +$ Número_de_Estratégias **do**
      **if** $\mathcal{U}_{[0,1]} \leq CR_i^w$ **then**
        $v_{t,i,j} \leftarrow$ Variação$(v_{t,i,j})$
      **end if**
    **end for**
    $**$ Recombinação dos Parâmetros $**$
    **for** $i = 1$ to $NP$ **do**
      **if** $f(u_{t,i}) < f(x_{t,i})$ **then**
        $x_{t+1,i} \leftarrow u_{t,i}$
      **else**
        $x_{t+1,i} \leftarrow x_{t,i}$
      **end if**
    **end for**
  **end for**
  $t \leftarrow t + 1$
**end while**

# Capítulo 4

# Experimentos Computacionais

Neste capítulo serão apresentados os resultados obtidos, nos experimentos computacionais.

## 4.1 Funções de Teste

Para a realização dos experimentos foram escolhidas funções de teste comumente usadas na literatura (Brest e Maucec, 2006; Liu e Lampinen, 2002; Qin e Suganthan, 2005; Qin et al., 2009), são elas:

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \ , \quad D = 30, \quad x_i \in [-100, 100] \tag{4.1}$$

$$f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \ , \quad D = 30, \quad x_i \in [-10, 10] \tag{4.2}$$

$$f_3(x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{i} x_j \right)^2 \ , \quad D = 30, \quad x_i \in [-100, 100] \tag{4.3}$$

$$f_4(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{|x_i|}) \ , \quad D = 30, \quad x_i \in [-500, -500] \tag{4.4}$$

$$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10) \ , \quad D = 30, \quad x_1 \in [-5.12, 5.12] \tag{4.5}$$

$$f_6(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \ , \quad D = 30, \quad x_i \in [-600, 600] \tag{4.6}$$

onde $D$ representa o número de dimensões da função. As funções $f_1$ a $f_3$ possuem muitas dimensões e são unimodais, as funções $f_4$ a $f_6$ são funções multimodais e o número de mínimos

locais cresce exponencialmente com o número de dimensões do problema. O mínimo global de $f_4$ é $-12569.5$ e o de todas as outras funções é 0.

## 4.2 Configuração dos Experimentos

Em todos os experimentos foi utilizada uma população de 100 indivíduous ($NP = 100$). Quando se queria verificar o melhor valor obtido, o critério de parada utilizado foi somente o número máximo de gerações. Para as funções unimodais o número máximo de gerações foi fixado em 3000 e para as função multimodais 5000. Quando se queria verificar o número de gerações para convergência, além do número máximo de gerações, os algoritmos também paravam quando encontravam o mínimo global com a precisão de $1.e^{-6}$.

## 4.3 Análise do Algoritmo Proposto

Esta seção tem por objetivo apresentar os resultados de experimentos que pretendem mostrar o comportamento do algoritmo proposto, o SaMDE.

### 4.3.1 Análise do Parâmetro $F'$

Abaixo são mostrados os boxplot[1] criados a partir de 30 execuções independentes do SaMDE para cada uma das função de teste.

A Fig.4.1 mostra o desempenho do algoritmo em relação ao número de gerações gastos nas funções unimodais. Fig.4.2 mostra o desempenho do algoritmo em relação ao melhor valor obtido nas funções unimodais e a Fig.4.3 nas multimodais. Podemos observar que levando-se em consideração todos os casos, os resultados obtidos com $F' \in [0.8, 1]$ foram os melhores. Podemos notar também que valores muito pequenos de $F'$ (valores entre 0.1 e 0.4) normalmente levam a um baixo desempenho, possivelmente, devido à baixa capacidade de alterar os parâmetros nestes casos. Foi tendo em vista estas observações que na abordagem proposta os valores de $F'$ sempre variam entre 0.8 e 1.

A análise em relação ao número de gerações gastas não foi feita para as funções multimodais pois independente do valor de $F'$ o algoritmo não conseguia convergir para o critério de convergência fixado.

[1] O boxplot é um gráfico que possibilita representar a distribuição de um conjunto de dados com base em alguns de seus parâmetros descritivos, quais sejam: a mediana, o quartil inferior (q1), o quartil superior (q3) e do intervalo interquartil (IQR = q3 - q1) . Sua representação gráfica é formada por uma caixa e por duas hastes. A linha central da caixa marca a mediana do conjunto de dados. A parte inferior da caixa é delimitada pelo quartil inferior e a parte superior pelo quartil superior. As hastes inferiores e superiores delimitam, respectivamente, as cercas inferior (q1 - 1.5IQR) e superior (q3 + 1.5IQR) e constituem limites para além dos quais, os dados passam a ser considerados outliers.

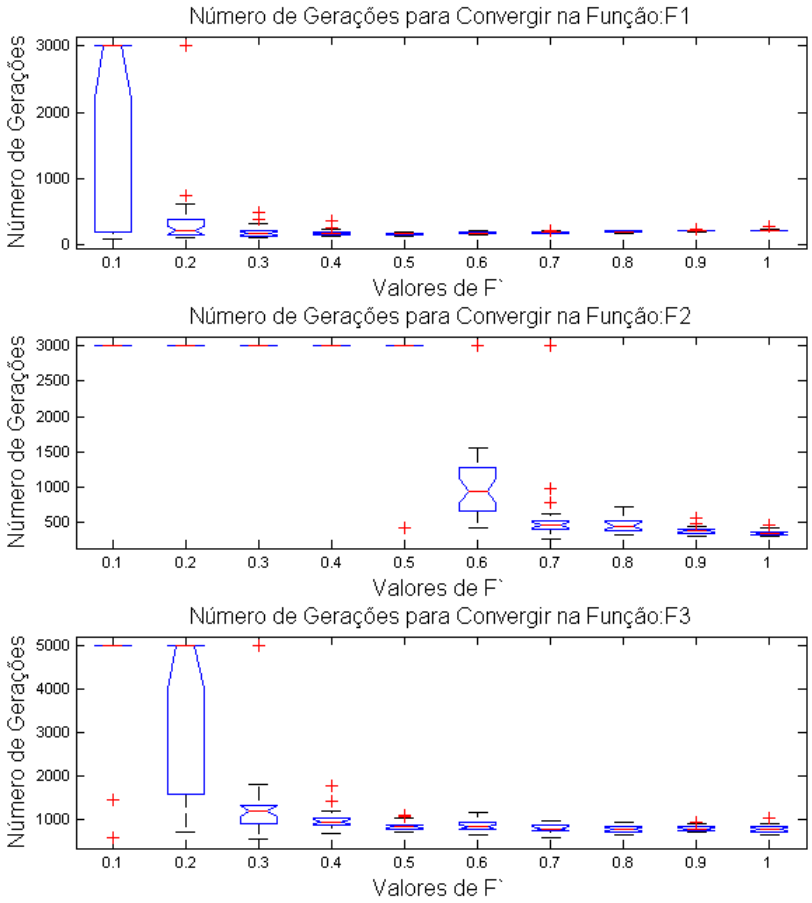


Figura 4.1: Número de Gerações Gastos em Média para Convergência nas Funções Unimodais Variando F'

### 4.3.2 Análise dos Parâmetros Autoadaptados

Esta seção tem por objetivo mostrar o processo evolutivo que envolve os parâmetros autoadaptados. Para fazer essa análise foram tiradas as médias dos valores dos parâmetros de todos os indivíduous a cada geração.

#### 4.3.2.1 Parâmetro V

Os gráficos mostrados na Fig.4.4 mostram o valor médio de cada um dos $Vs$ a cada geração nas funções unimodais. Podemos observar claramente que a estratégia de mutação representada por $V_2$ (best/1) possui os maiores valores por quase todo o processo de busca, o que já era esperado uma vez que reconhecidamente na literatura esta estratégia possui o melhor desempenho neste tipo de função. Isso indica que o SaMDE é sensível à estratégia de mutação,

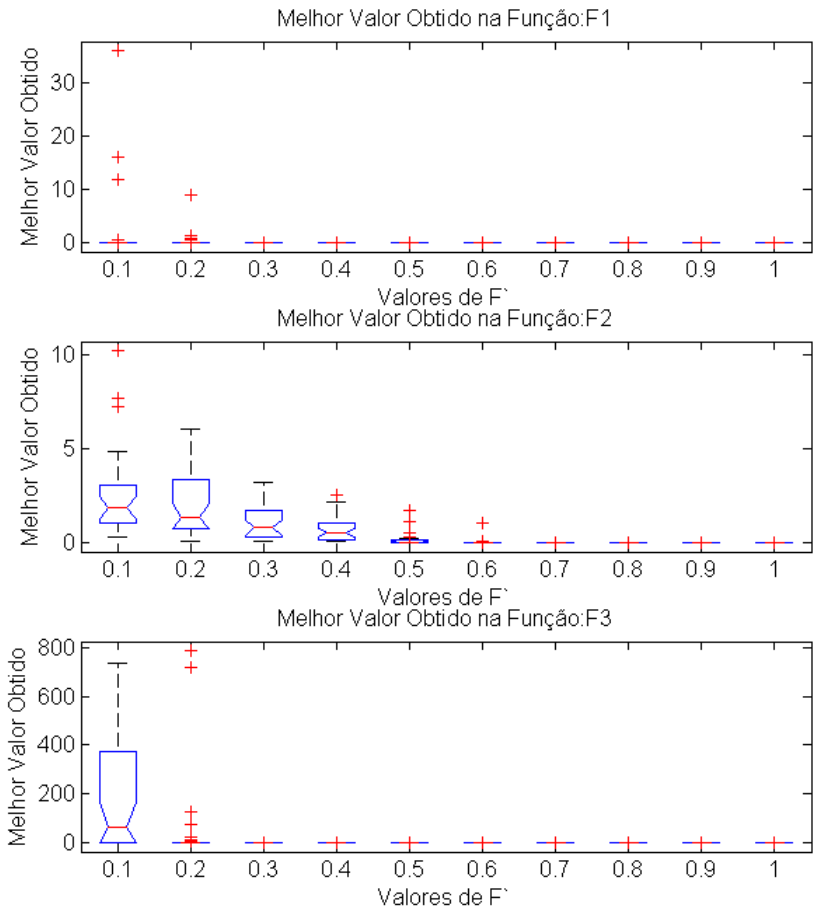


Figura 4.2: Melhor Valor Encontrado em Média nas Funções Unimodais Variando F'

conseguindo distinguir a melhor estratégia, dando a ela melhores chances para ser aplicada novamente.

Os gráficos mostrados na Fig.4.5 mostram os mesmos dados para as funções multimodais. Em todas elas podemos perceber uma domínio da estratégia representada por $V_1$ (best/1) nos estágios inicias do processo de busca. Contudo com o avançar das gerações outras estratégias passam a dominar. Isso mostra que o SaMDE tem o poder de mudar de estratégias à medida que elas perdem potencial de melhorar as soluções.

Na Fig4.6 e na Fig.4.7 é mostrado o número de vitórias que cada uma das estratégias de mutação teve em cada geração, isto é, o número de vezes em que cada uma delas foi selecionada para ser aplicada a cada geração. Podemos perceber que o número de vitórias de cada estratégia segue a grandeza do seu valor $V$ correspondente (Figuras 4.4 e 4.5), o que mostra a adequação da abordagem de seleção de estratégia utilizada.

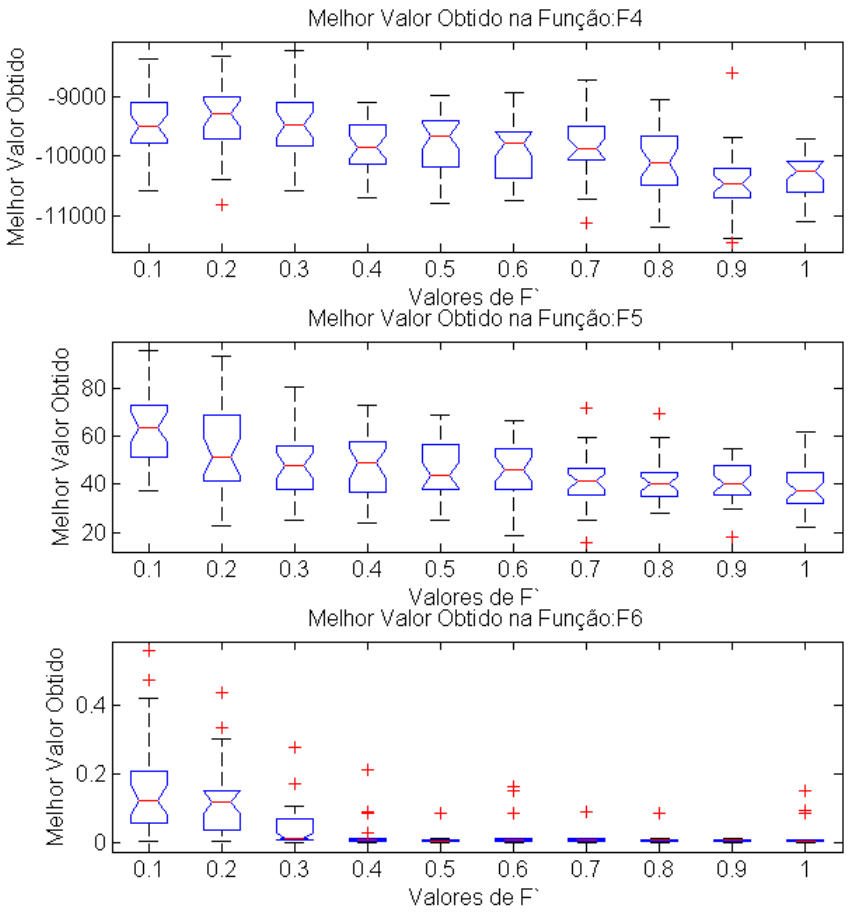


Figura 4.3: Melhor Valor Encontrado em Média nas Funções Multimodais Variando F'

#### 4.3.2.2 Parâmetros CR e F

Os gráficos mostrados na Fig.4.4 mostram o valor médio dos $CRs$ correspondentes a cada uma das estratégias de mutação, representadas pelos respectivos $Vs$, a cada geração nas funções unimodais. Podemos ver que nas funções $f_1$ e $f_2$ a diferença entre os valores médios de CR é bem pequena, contudo, na função $f_3$ já pode-se obervar uma diferenciação do comportamento do parâmetro na estratégia representada por $V4$. Na Fig.4.5, que apresenta estes resultados nas funções multimodais, também podemos notar a diferença nos comportamentos dos $CRs$ nas funções $f4$ e $f5$, o que demonstra a necessidade de diferentes $CRs$ para as diferentes estratégias. Observando ainda as funções $f4$ e $f5$ podemos notar que o comportamento de $CR$ pode ser diferente dependendo do estágio do processo de otimização. No início podemos ver um crescimento dos valores e posteriormente notamos seu decrescimento na maioria das estratégias de mutação, o que mostra as vantegens de não se usar o DE de parâmetros fixos.

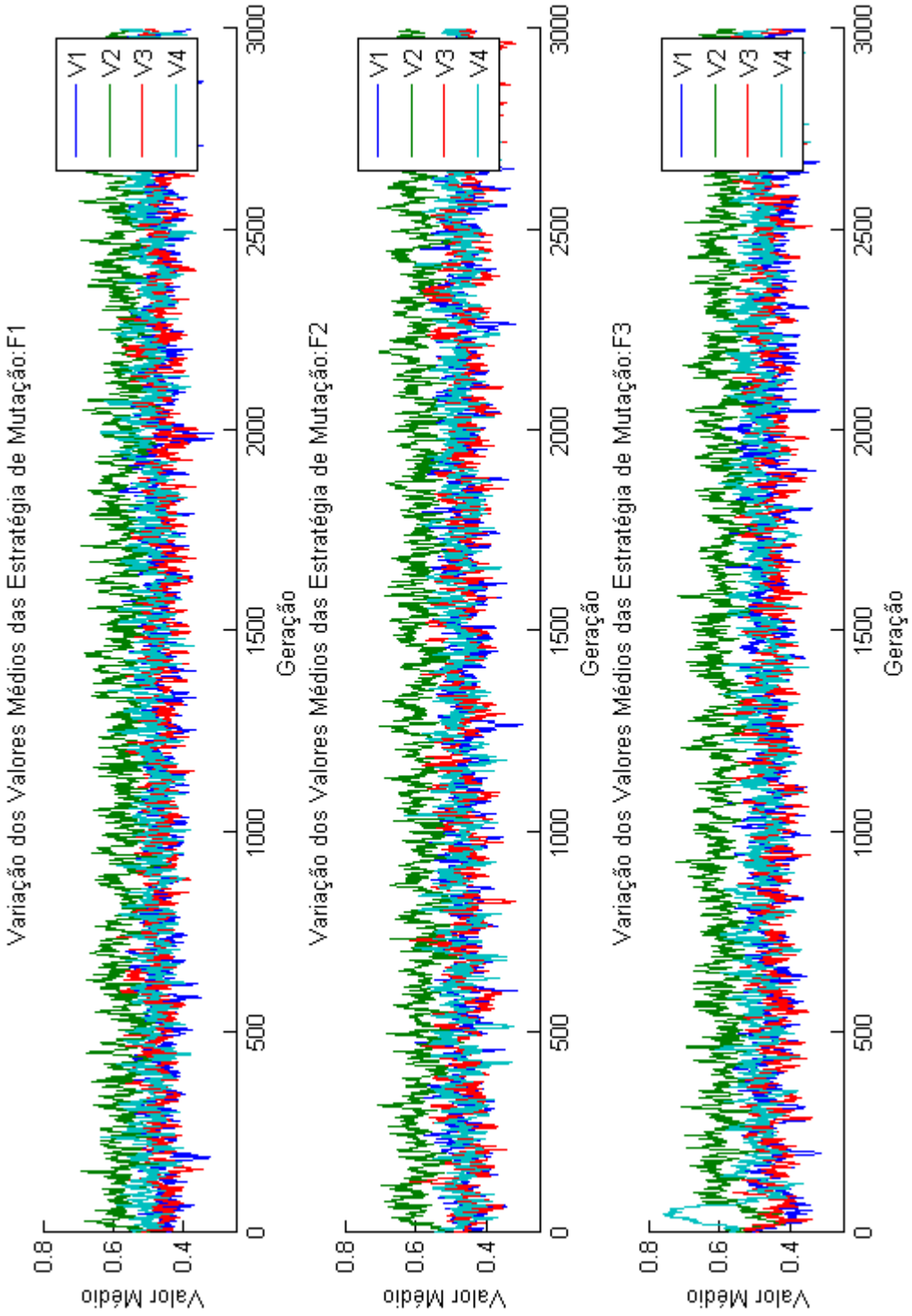


Figura 4.4: Valor Médio dos Vs nas Funções Unimodais

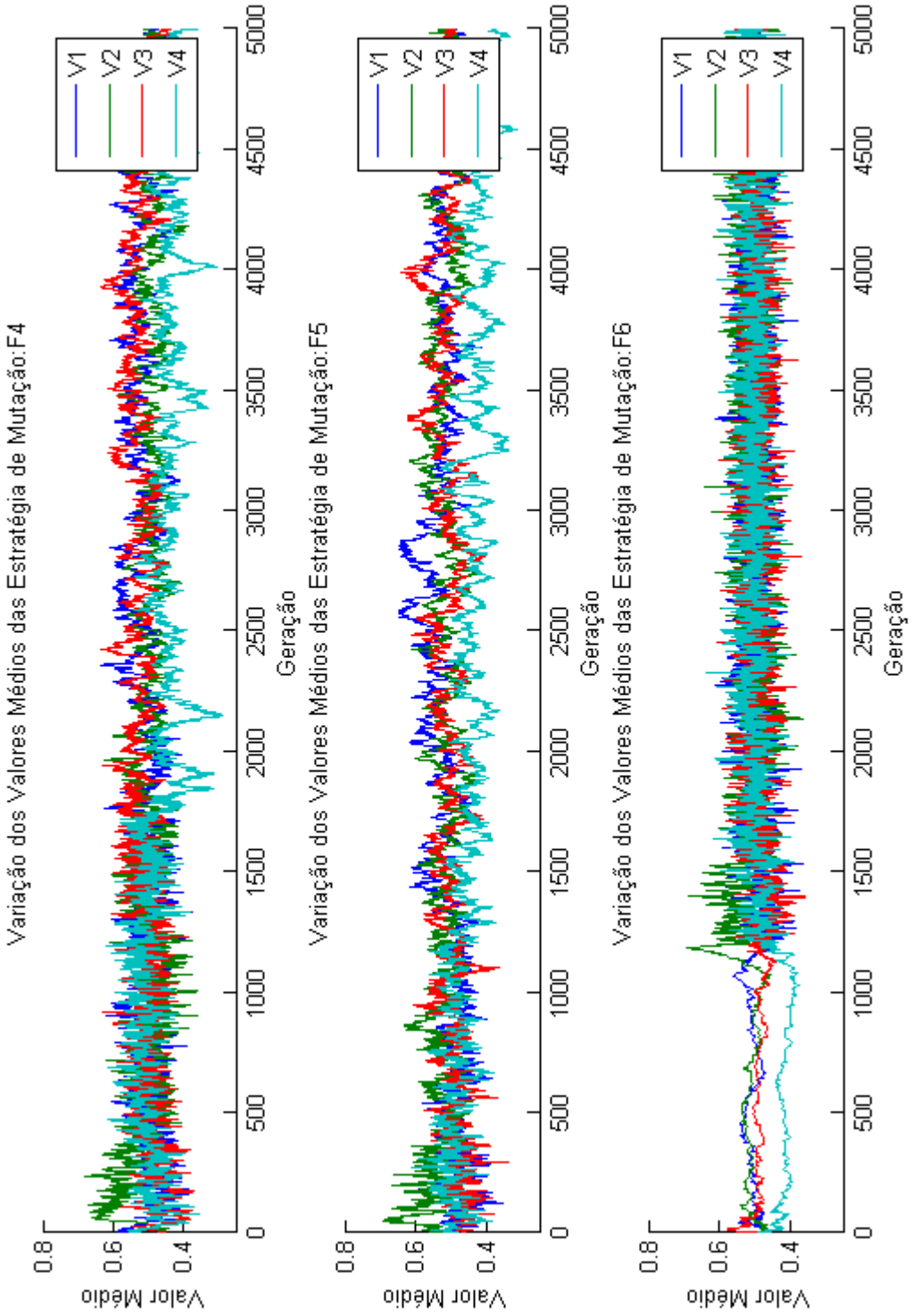


Figura 4.5: Valor Médio dos Vs nas Funções Multimodais

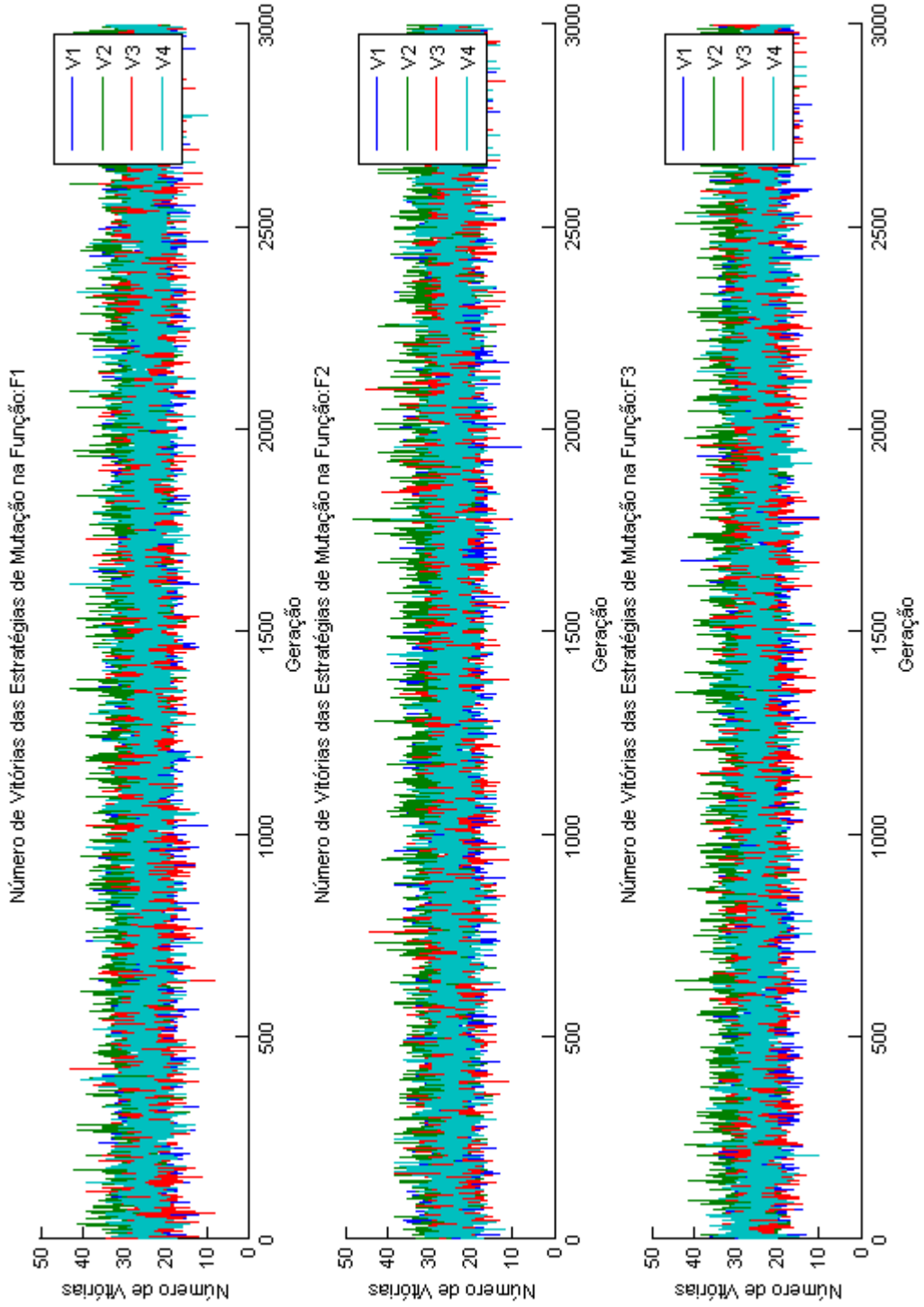


Figura 4.6: Número de vezes em que cada uma das Estratégia de Mutação foi aplicada em cada Geração nas Funções Unimodais

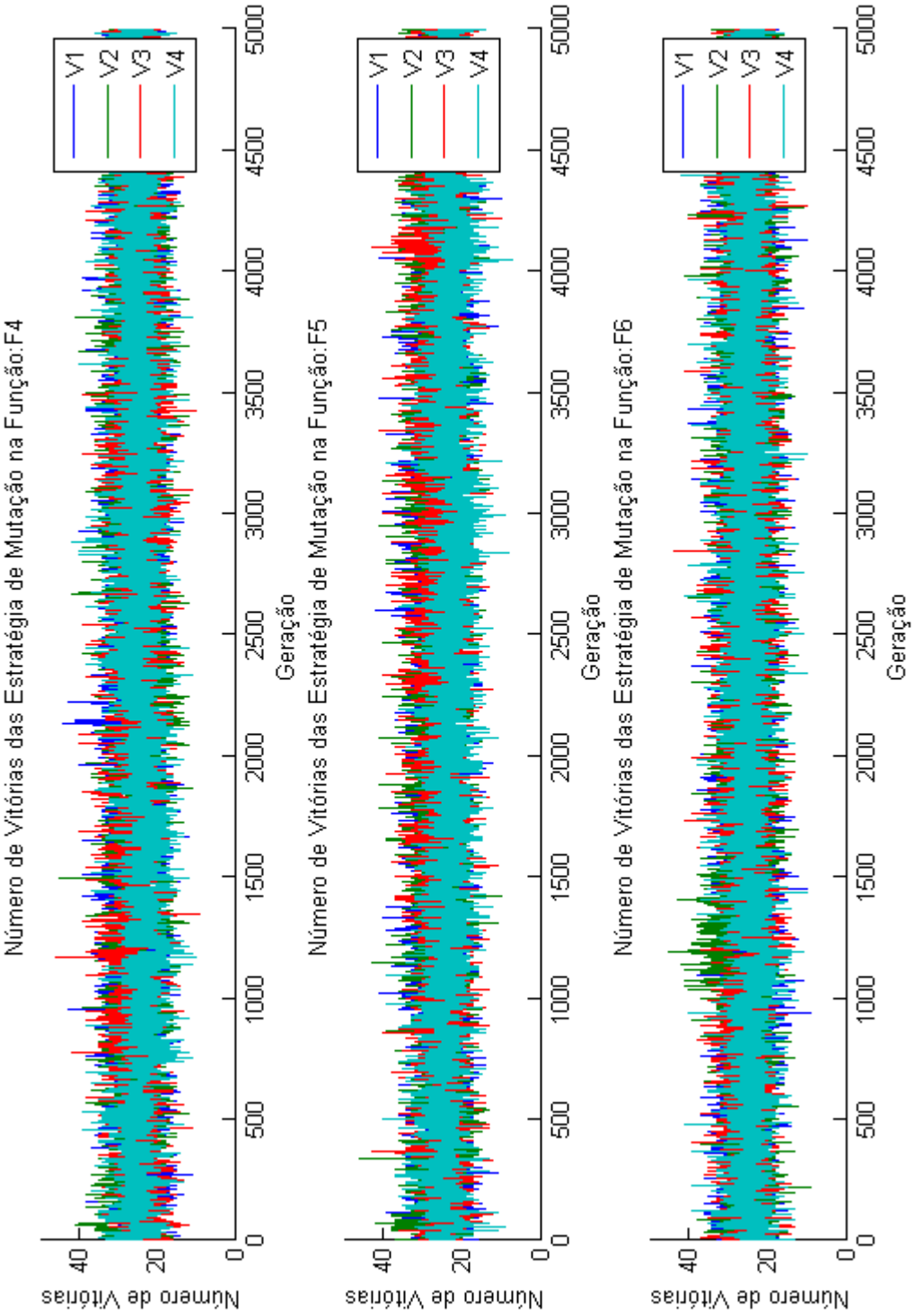


Figura 4.7: Número de vezes em que cada uma das Estratégia de Mutação foi aplicada em cada Geração nas Funções Multimodais

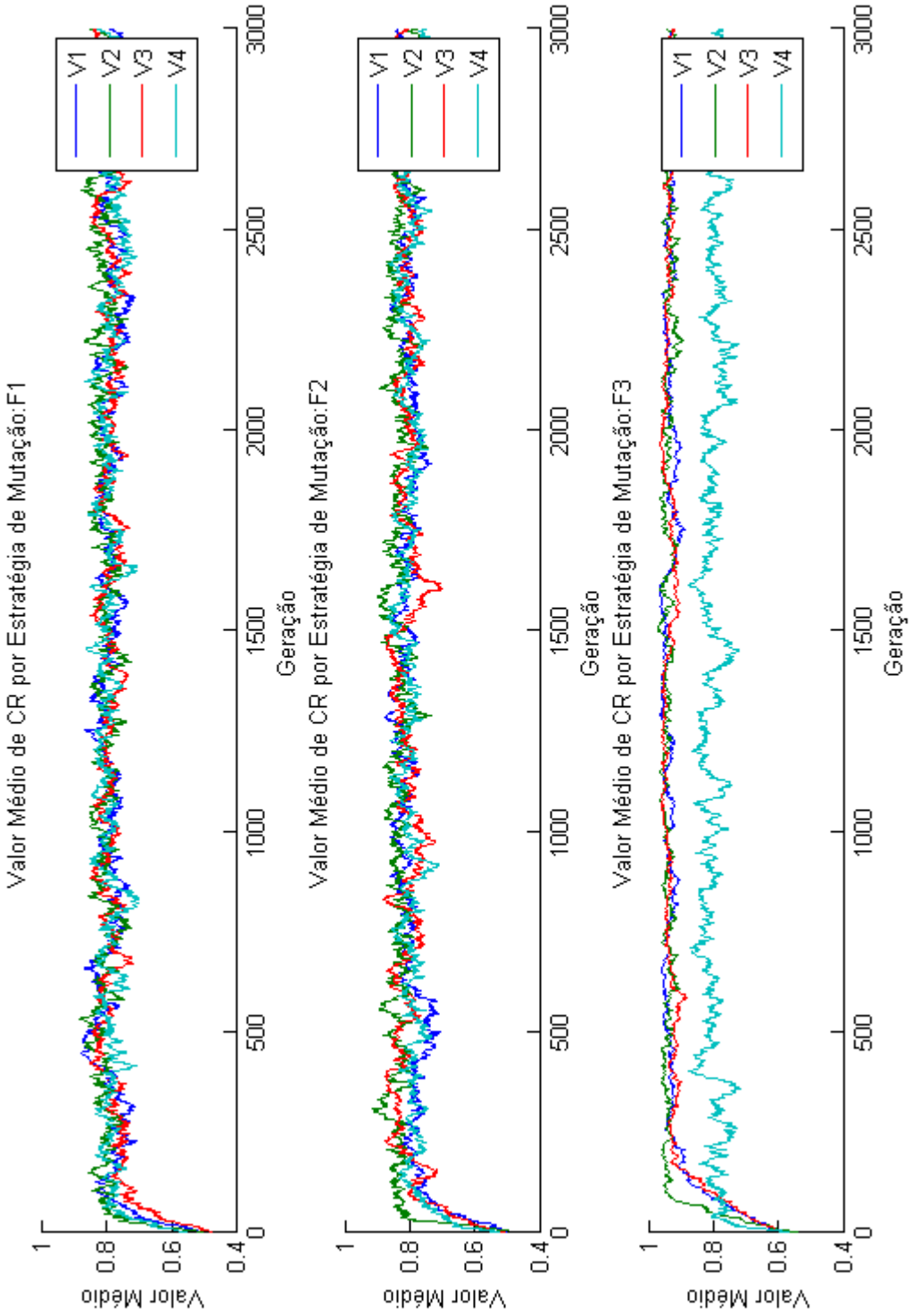


Figura 4.8: Valor Médio dos CRs nas Funções Unimodais

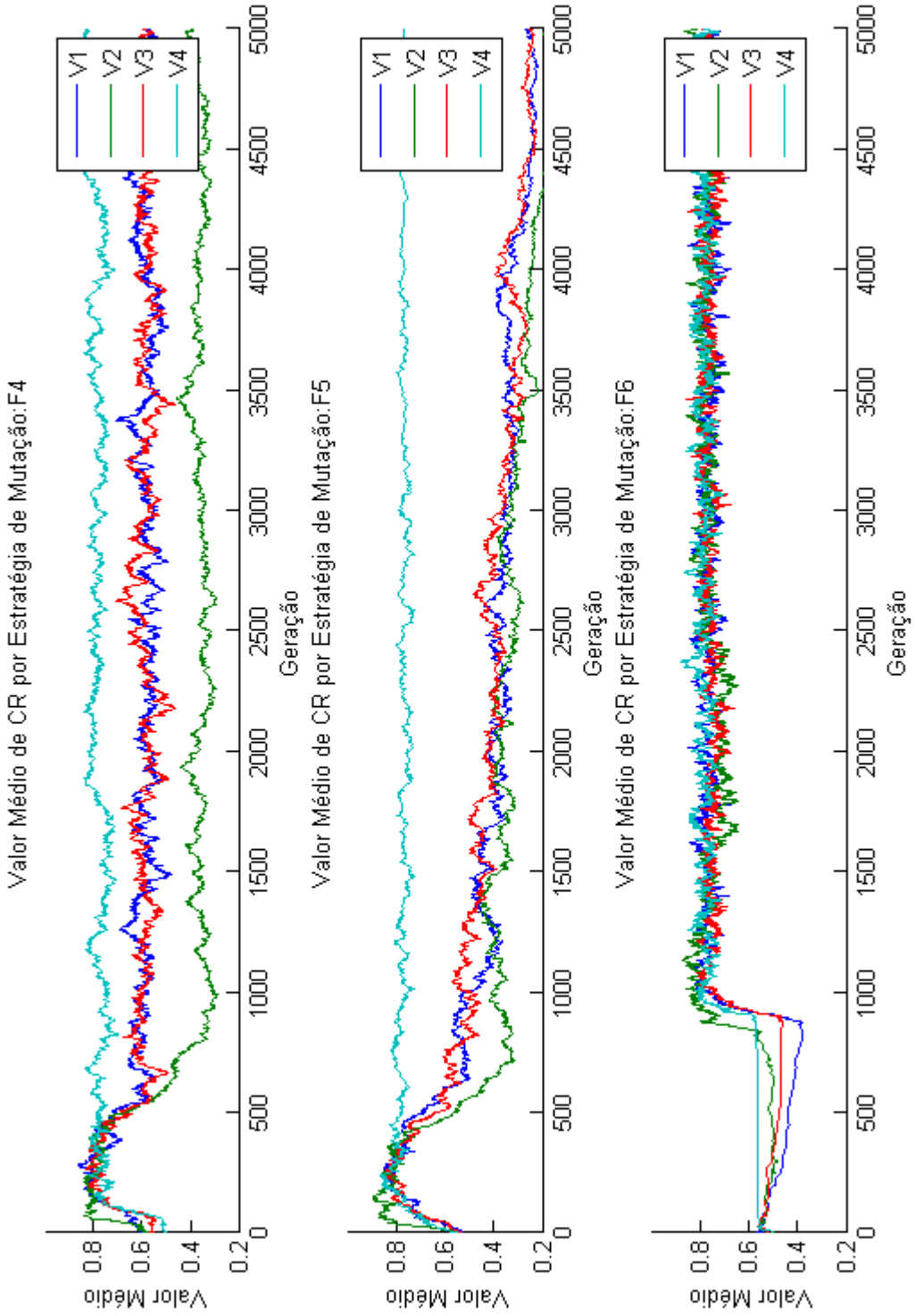


Figura 4.9: Valor Médio dos CRs nas Funções Multimodais

Quanto aos valores de $F$ apresentados nas Figuras 4.10 e 4.11, temos uma diferença menor nas médias em cada uma das estratégias, contudo podemos notar com clareza que normalmente a estratégia de mutação representada por $V3$ (rand/2) possui menores valores de $F$ que a estratégia representada por $V2$, o que reforça a hipótese de que diferentes estratégias requerem diferentes valores de $F$.

## 4.4 Comparação de Desempenho

Nesta seção será comparado o desempenho do SaMDE em relação ao DE original com $F = 0.5$ e $CR = 0.9$ como sugerido por Brest et al. (2007) e em relação ao jDE que apresentou o melhor desempenho quando comparado com outras estruturas modificadas do DE na maior parte dos testes feitos em (Neri e Tirronen, 2010). Os gráficos a seguir apresentam a síntese dos dados de 30 execuções independentes de cada estratégia em cada função. As três abordagens obrigatoriamente começam com a mesma população inicial.

A Fig.4.12 mostra que nas funções unimodais o SaMDE normalmente converge mais rápido necessitando de um menor número de gerações. Os piores resultados são apresentados pelo DE de parâmetros fixos que inclusive não consegue convergir no número máximo de gerações predefinidos na função $f_3$. O jDE apresenta desempenho intermediário.

Em relação aos melhorese valores encontrados o SaMDE e o jDE possuem desempenho comparável e novamente o DE original possui o pior desempenho, como podemos observar na Fig.4.13.

Nas funções multimodais o desempenho de todas as abordagens fica pior. Como mostra a Fig.4.14 na função $f_4$ nenhuma das estratégias consegue convergir dentro do número máximo de gerações estipulado. Na função $f_5$ apenas o jDE consegue convergir para o ótimo global. Já na função $f_6$ apenas o SaMDE consegue convergir, mas faz isso apenas algumas vezes.

Em relação aos melhores valores encontrados, podemos observar na Fig.4.15, o melhor desempenho é apresentado pelo jDE. Nas funções $f_5$ e $f_6$ o pior desempenho é apresentado pelo DE original deixando o SaMDE com o desempenho intermediário. Contudo, na $f_4$ o pior desempenho é apresentado pelo SaMDE.

A Fig.4.16 mostra o valor de função objetivo do melhor indivíduo a cada geração, de cada uma das abordagens nas funções unimodais. Os gráficos apresentados confirmam a maior velocidade de convergência do SaMDE neste tipo de função.

A Fig.4.17 mostra o valor de função objetivo do melhor indivíduo a cada geração, de cada uma das abordagens nas funções multimodais. Os gráficos apresentados mostram que o SaMDE converge rapidamente, porém para um valor sub-ótimo. Esse fenômeno é conhecido como convergência prematura. O DE de parâmetros fixos mostrou velocidade de convergência baixa, não conseguindo alcançar os valores encontrados pelo jDE, que por sua vez, apesar de apresentar convergência mais lenta que o SaMDE consegue encontrar melhores soluções. É

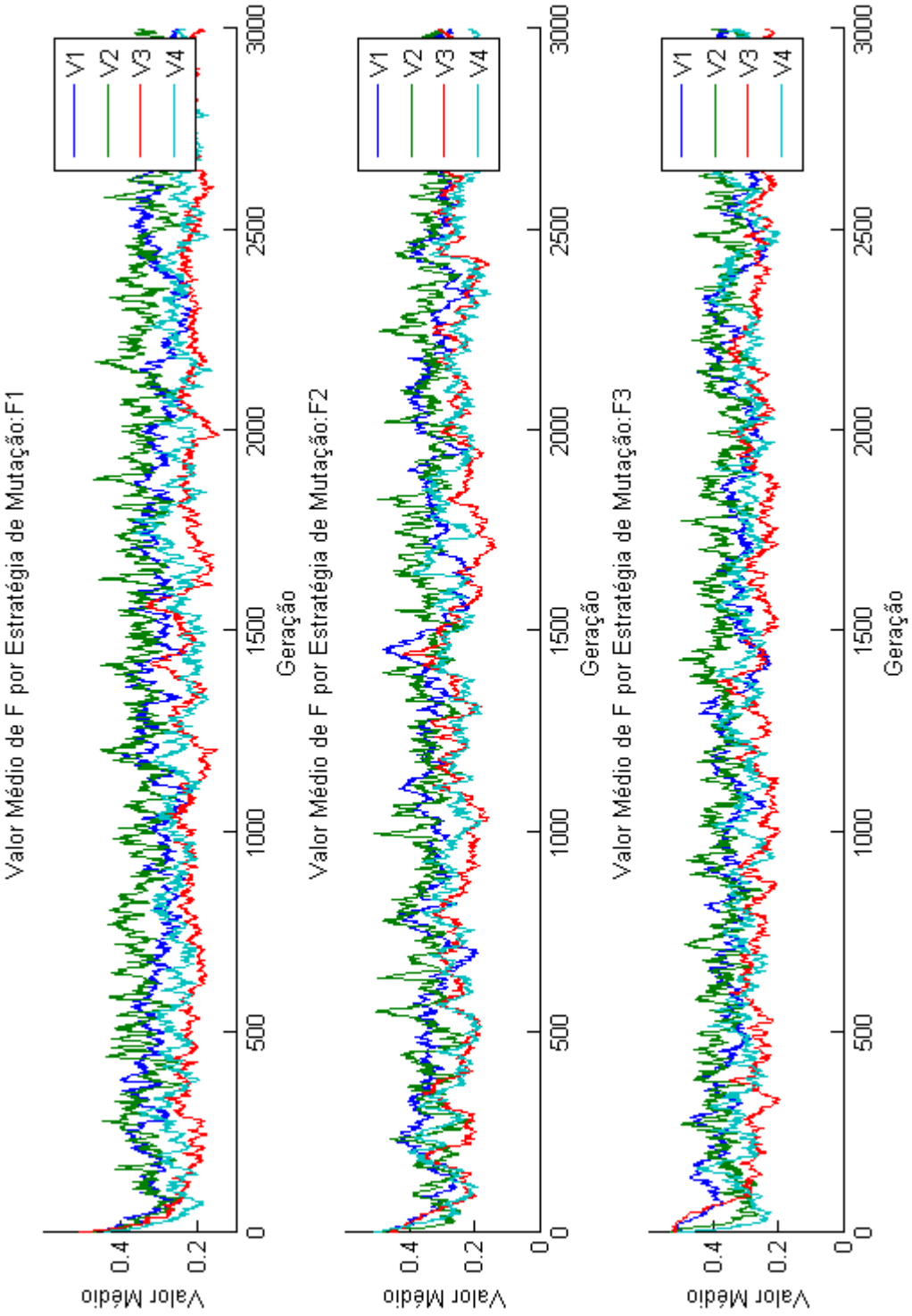


Figura 4.10: Valor Médio dos Fs nas Funções Unimodais

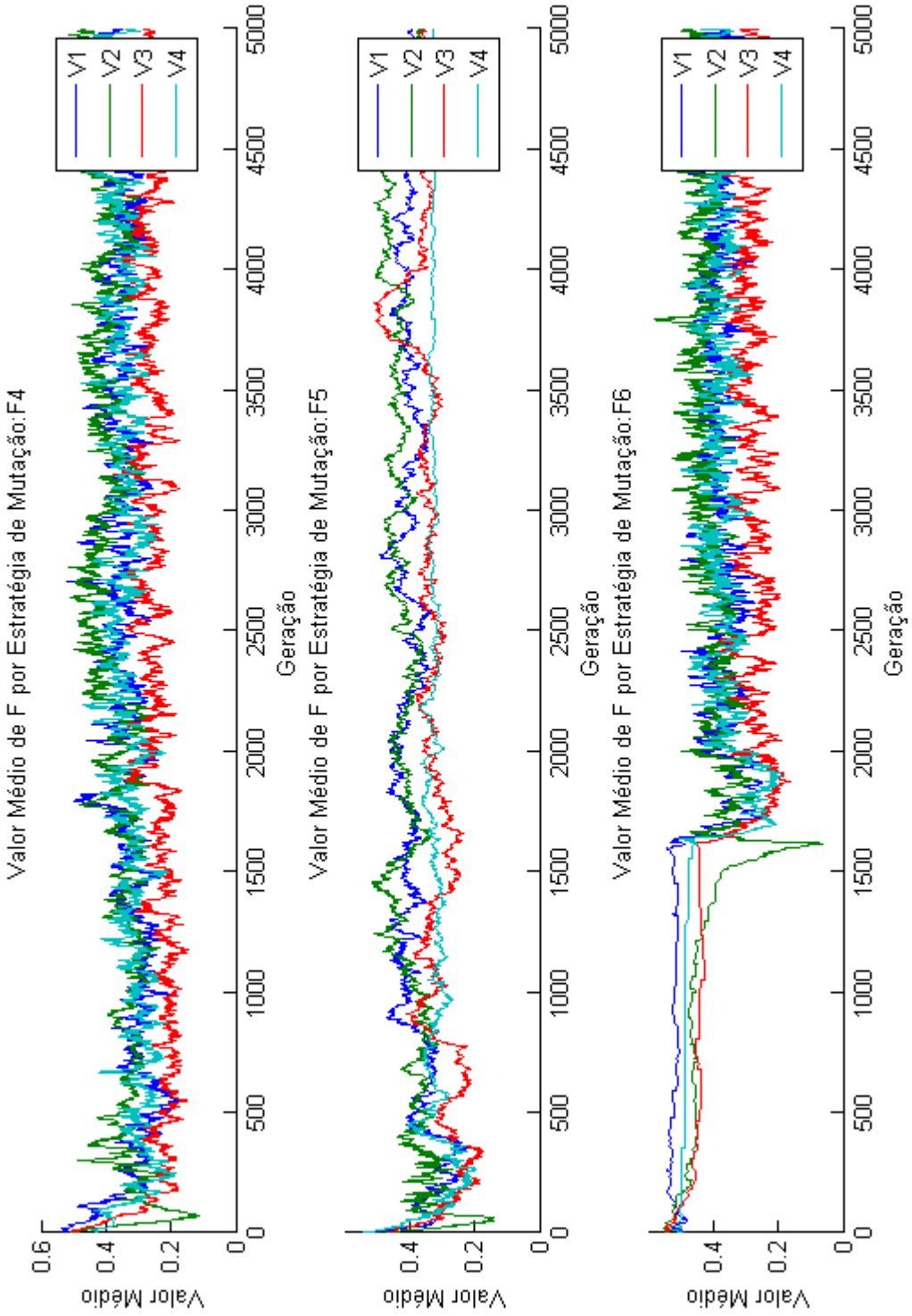


Figura 4.11: Valor Médio dos Fs nas Funções Multimodais

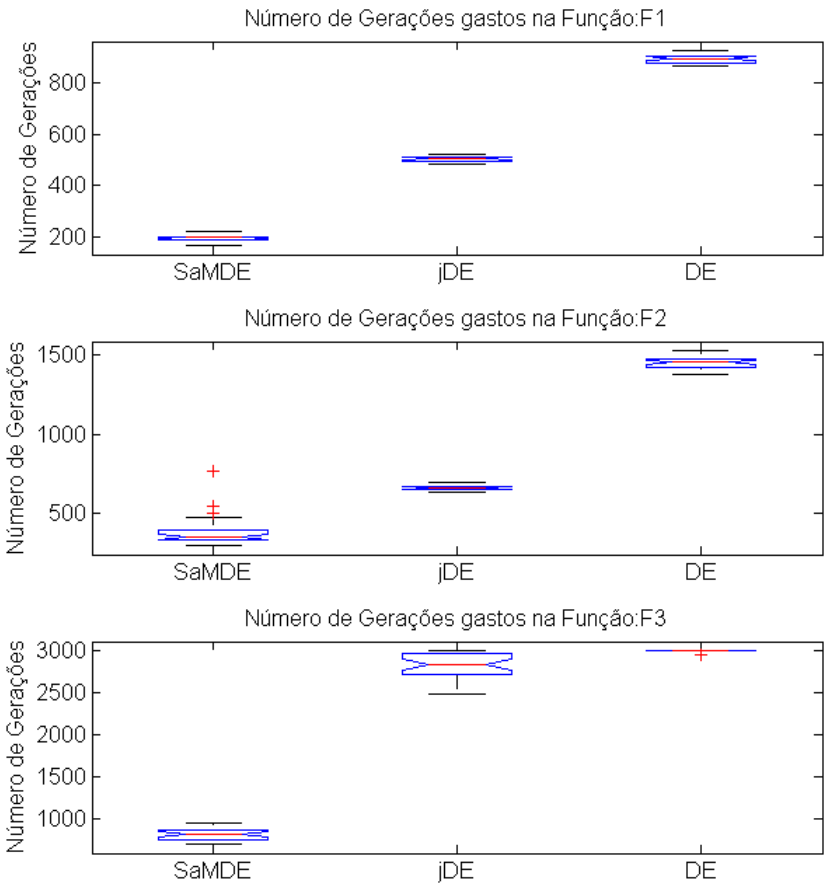


Figura 4.12: Número de Gerações gastos em Média nas Funções Unimodais

bom observar que mesmo encontrando melores soluções o jDE também sofreu de convergência prematura uma vez que não conseguiu atingir o mínimo global com a precisão desejada.

A estratégia usada pelo jDE de escolher novos parâmetros aleatoriamente com determinada probabilidade apesar de aparentemente diminuir sua velocidade de convergência, dá a ele maior capacidade de exploração. Isso explica seus melhores resultados no que diz respeito à qualidade da solução obtida nas funções multimodais e a convergência mais lenta em relação ao SaMDE na funções uniomodais.

O problema de convergência prematura no SaMDE provavelmente ocorre, pois os parâmetros tendem a convergir para valores muito próximos (valores estes que são apontados pelo processos evolutivo como bons) rapidamente, diminuindo a capacidade do algoritmo de diversificar os parâmetros novamente. Para resolver este problema pode-se usar a reinicialização dos parâmetros de tempos em tempos, como feito no próprio jDE em (Brest et al., 2007), ou fazer o uso de mutações neutras, onde os parâmetrso possuem cópias a princípio inativas, que

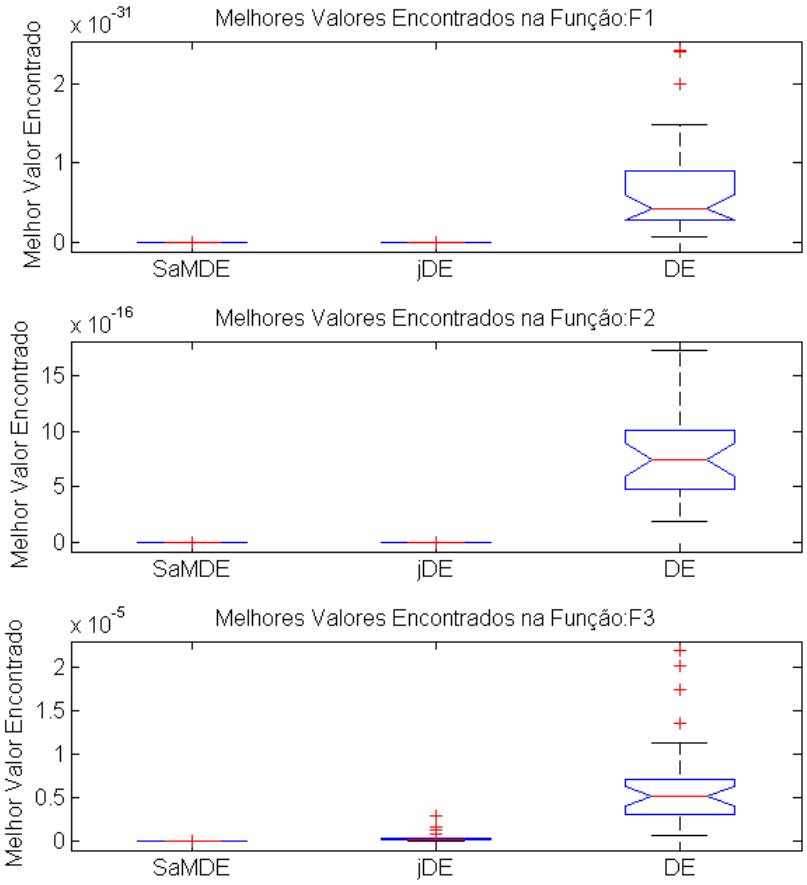


Figura 4.13: Melhor Valor Encontrado em Média nas Funções Unimodais

podem ser ativadas em algum momento. Este tipo de abordagem já foi aplicado com sucesso em Estratégias Evolutivas por Ohkura et al. (2000).

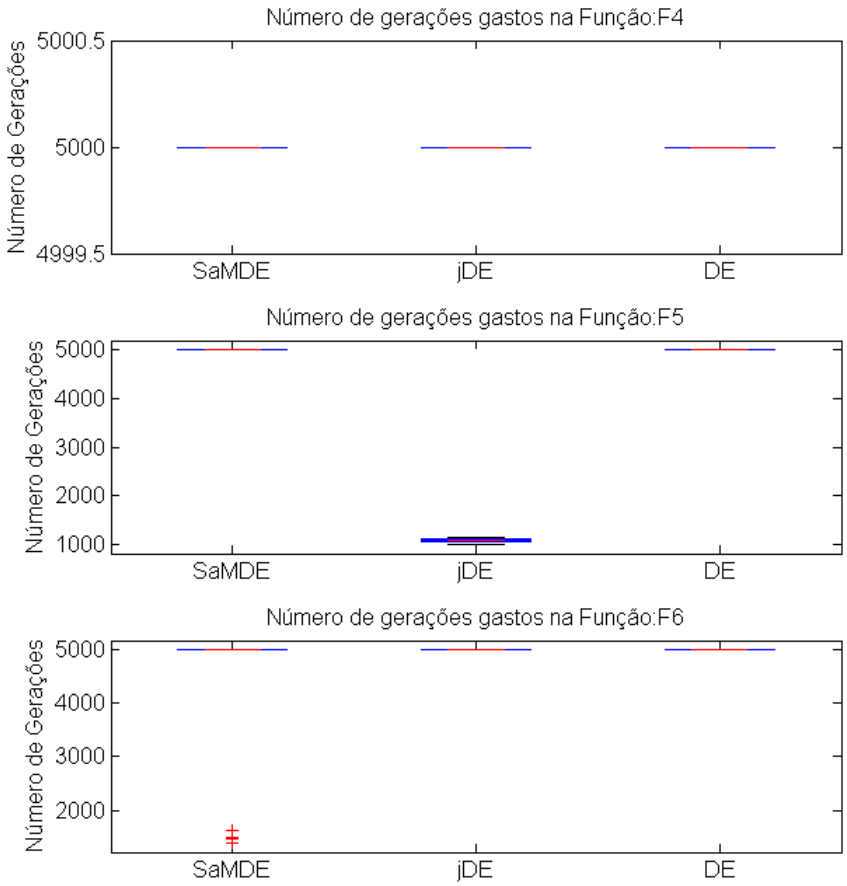


Figura 4.14: Número de Gerações gastos em Média nas Funções Multimodais

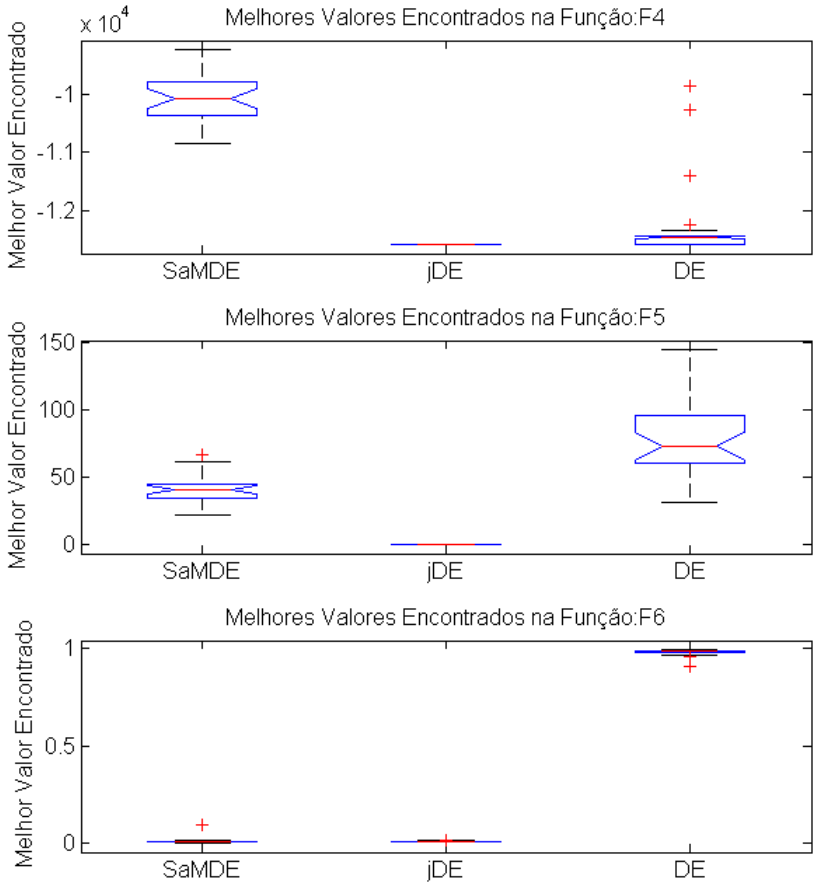


Figura 4.15: Melhor Valor Encontrado em Média nas Funções Multimodais

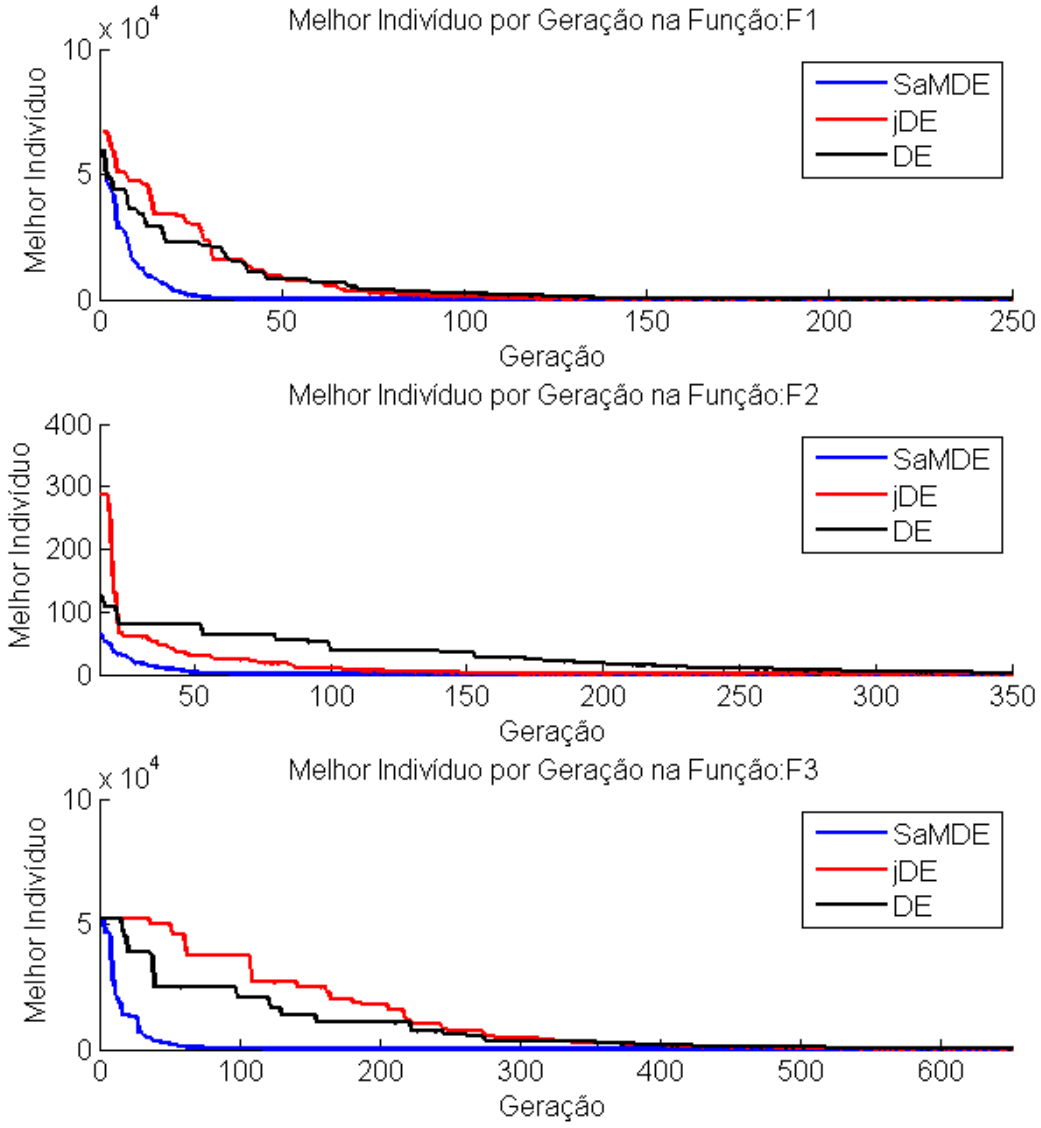


Figura 4.16: Comportamento do Melhor Indivíduo da População nas Funções Unimodais

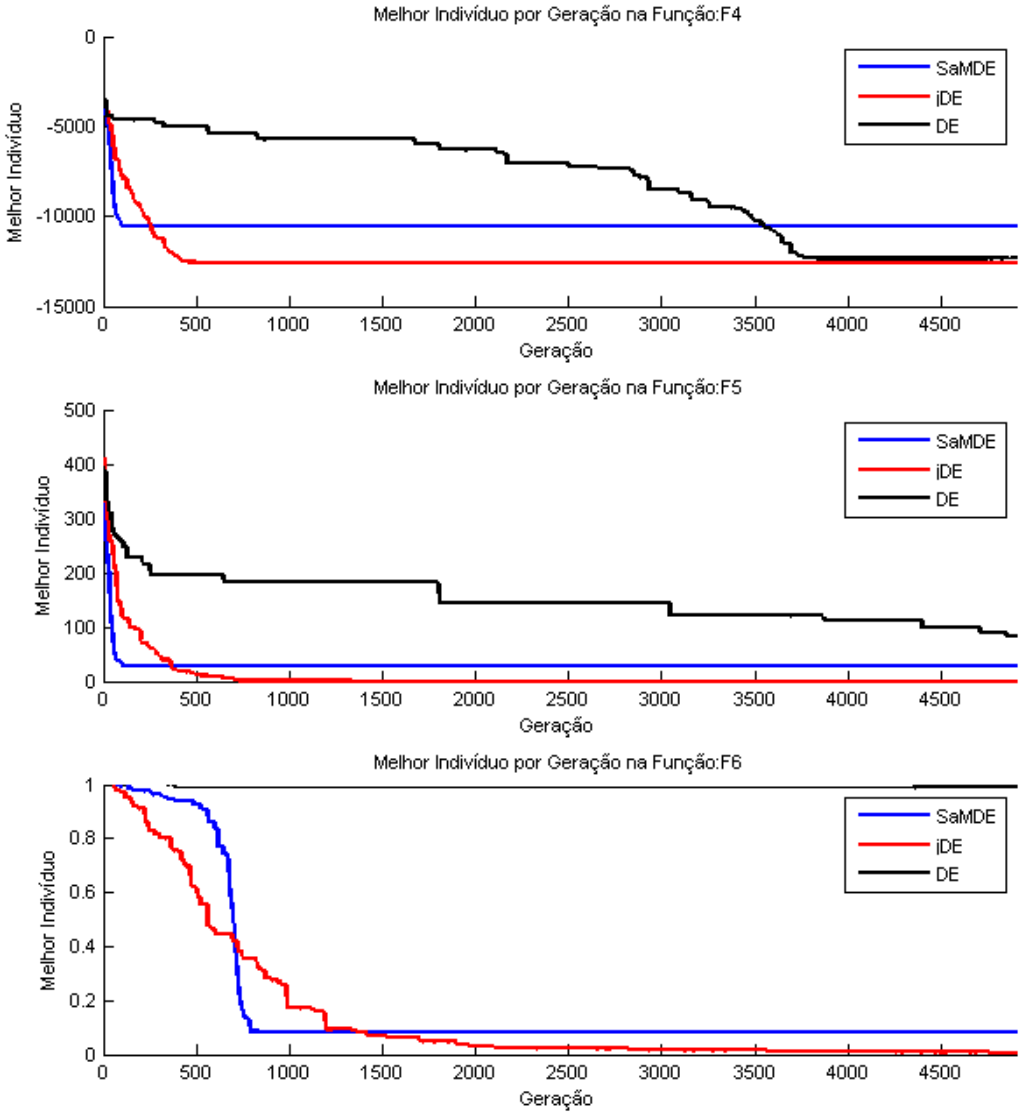


Figura 4.17: Comportamento do Melhor Indivíduo da População nas Funções Multimodais

# Capítulo 5

# Conclusões e Perspectivas Futuras

## 5.1 Conclusão

Algoritmos Evolutivos têm sido usados desde os anos 80 como uma ferramenta importante para a resolução de problemas difíceis de otimização. Em especial, o algoritmo de Evolução Diferencial proposto em meados da década de 90, tem sido usado com muito sucesso na otimização de problemas com variáveis contínuas e mais recentemente em problemas com variáveis discretas.

Apesar deste sucesso, estes algoritmos possuem parâmetros que precisam ser definidos. Estes parâmetros têm grande impacto no desempenho dessas técnicas e seu conjunto ótimo depende do problema que se quer tratar. Normalmente, a definição destes parâmetros é feita via tentativa e erro, na qual é gasta uma grande quantidade de tempo e recurso computacional. Para minimizar este esforço faz-se o uso da autoadaptação, na qual os parâmetros são acoplatos à representação do indivíduo e o próprio processo evolutivo cuida de ajustá-los.

Existem na literatura algumas abordagens ditas autoadaptativas para o DE, contudo estas abordagens possuem alguns problemas. Para minimizar estes problemas foi proposto neste trabalho uma nova estratégia, chamada SaMDE.

No geral, o SaMDE melhora as características de robustez do DE original convergindo num menor número de gerações com menor variação, além de conseguir soluções de melhor qualidade na maioria dos casos testados. Em relação a outra abordagem autoadaptativa testada, o jDE, o SaMDE obteve melhor desempenho nas funções unimodais e apesar de convergir rapidamente nas funções multimodais, obtêm soluções de menor qualidade se comparadas às soluções obtidas pelo jDE.

Contudo, a estratégia proposta se mostra promissora, pois tem boa capacidade de escolher bons parâmetros e de selecionar a estratégia de mutação mais adequada para determinado problema, sem a necessidade de interferência de um usuário externo. Ela ainda tem boas perspectivas de melhora se forem aplicadas técnicas que evitam a estagnação do parâmetros, ja aplicadas em outros algoritmos autoadaptativos.

## 5.2 Perspectivas Futuras

Como perspectivas futuras temos:

- A aplicação de uma técnica para evitar a convergência prematura do algoritmo, aumentando sua capaciade de diversificar os parâmetros, prejudicando o mímino possível a velocidade de convergência.
- Realizar experimentos em uma maior diversidade de problemas mono-objetivo e multi-objetivo.
- Utilizar o SaMDE em algoritmos DE para problemas combinatórios, tanto para melhorar suas propriedades de convergência, quanto para obter um melhor entendimento do comportamento destes parâmetros durante o processo de busca.

# Referências Bibliográficas


Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pp. 14–21, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Batista, L. S.; Guimarães, F. G. e Ramírez, J. A. (2009). A differential mutation operator for the archive population of multi-objective evolutionary algorithms. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pp. 1108–1115, Piscataway, NJ, USA. IEEE Press.

Bäck, T.; Hammel, U. e Schwefel, H.-P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1:3–17.

Beasley, D. (1997). Possible applications of evolutionary computation. *Handbook of Evolutionary Computation*.

Brest, J. (2009). Constrained Real-Parameter Optimization with $\epsilon$-Self-Adaptive Differential Evolution. In Mezura-Montes, E., editor, *C*onstraint-Handling in Evolutionary Computation, chapter 4, pp. 73–93. Springer. Studies in Computational Intelligence, Volume 198, Berlin. ISBN 978-3-642-00618-0.

Brest, J.; Bokovic, B.; Greiner, S.; umer, V. e Maucec, M. (2007). Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11:617–629. 10.1007/s00500-006-0124-0.

Brest, J. e Maucec, M. S. (2006). Control parameters in sel-adaptive diffrential evolution.

Castro, L. N. D. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London.

Chakraborty, U. K. (2008). *Advances in Differential Evolution*. Springer Publishing Company, Incorporated.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*.

Das, S.; Dasgupta, S.; Biswas, A. e Abraham, A. (2008). Automatic circle detection on images with annealed differential evolution. *Hybrid Intelligent Systems, International Conference on*, 0:684–689.

de Sousa, F. L. (2003). *Otimização Extrema Generalizada: Um Novo Algoritmo Estocástico para o Projeto Ótimo*. Ph.d. dissertation, Instituto Nacional de Pesquisas Espaciais - INPE.

Eiben, A. e Smith, J. (1999). *Introdution to Evolutionary Computing*. Springer-Verlag, Berlin, Germany.

Eiben, A. E.; Michalewicz, Z.; Schoenauer, M. e Smith, J. E. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*.

Gämperle, R.; Müller, S. D. e Koumoutsakos, P. (2002). A parameter study for differential evolution. In *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pp. 293–298. Press.

Gehlhaar, D. K. e Fogel, D. B. (1996). Tuning evolutionary programming for conformationally flexible molecular docking. In *Evolutionary Programming*, pp. 419–429.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1 edição.

Guimarães, F. G. (2009). *Algoritmos de Evolução Diferencial*. Manual de Algoritmos Evolutivos e Metaheurísticas.

Iorio, A. W. e Li, X. (2004). Solving rotated multi-objective optimization problems using differential evolution. In *In AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence*, pp. 861–872. press.

Kennedy, J. e Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948.

Lampinen, J. e Zelinka, I. (2000). On stagnation of the differential evolution algorithm.

Liu, J. e Lampinen, J. (2002). On setting the control parameter of the differential evolution method. In *8 th Int. Conf. Soft Computing (MENDEL2002)*, pp. 11–18.

Liu, J. e Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Comput.*, 9(6):448–462.

Maruo, M.; Lopes, H. e M.R., D. (2005). Self-adapting evolutionary parameters: Encoding aspects for combinatorial optimization problems. In *Proceedings of Evolutionary Computing Combinatorial Optimization*, pp. 155–166. Springer-Verlag.

Mezura-Montes, E.; Velázquez-Reyes, J. e Coello Coello, C. A. (2006). A comparative study of differential evolution variants for global optimization. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 485–492, New York, NY, USA. ACM.

Neri, F. e Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artif. Intell. Rev.*, 33(1-2):61–106.

Ohkura, K.; Matsumura, Y.; Ueda, K. e Yao, X. (2000). Robust evolution strategies. *Applied Intelligence*, 1585:10–17.

Onwubolu, G. C. e Davendra, D. (2009). *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. Springer Publishing Company, Incorporated.

Plagianakos, V.; Tasoulis, D. e Vrahatis, M. (2008). A review of major application areas of differential evolution. In Chakraborty, U., editor, *A*dvances in Differential Evolution, volume 143 of *S*tudies in Computational Intelligence, pp. 197–238. Springer Berlin / Heidelberg. 10.1007/978-3-540-68830-$3_8$.

Prado, R. S.; Silva, R. C. P.; Guimarães, F. G. e Neto, O. M. (2010). A new differential evolution based metaheuristic for discrete optimization. *International Journal of Natural Computing Research (IJNCR)*, 1:15–32.

Price, K. V.; Storn, R. M. e Lampinen, J. A. (2005). *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany.

Qin, A. K.; Huang, V. L. e Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *Trans. Evol. Comp*, 13:398–417.

Qin, A. K. e Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *IEEE Congress on Evolutionary Computation (CEC 2005)*, pp. 1785–1791. IEEE Press.

Reddy, M. J. e Kumar, D. N. (2007). Multiobjective differential evolution with application to reservoir system optimization.

Ronkkonen, J.; Kukkonen, S. e Price, K. V. (2005). Real-parameter optimization with differential evolution. volume 1, pp. 506–513 Vol.1.

Sbalzariniy, I. F.; Müller, S. e Koumoutsakosyz, P. (2000). Multiobjective optimization using evolutionary algorithms. In *Proceedings of Center for Turbulence Research Summer Program 2000*, pp. 63–74.

Schwefel, H. (1981). *Numerical optimization of computer models*. Wiley, Chichester, WS, UK.

Storn, R. e Price, K. (1995). Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report.

Storn, R. e Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359.

Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.*, 10(8):673–686.

Xue, F.; Sanderson, A. C. e Graves, R. J. (2003). Pareto-based multi-objective differential evolution. In *Evolutionary Computation, 2003. CEC '03*, volume 2, pp. 862–869.

Zamuda, A.; Brest, J.; Boskovic, B. e Zumer, V. (2009). Differential Evolution with Self-Adaptation and Local Search for Constrained Multiobjective Optimization. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pp. 195–202, Trondheim, Norway. IEEE Service Center.