UNIVERSIDADE DE SÃO PAULO
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto
Departamento de Computação e Matemática

Computação Evolutiva em Ambientes Dinâmicos

RENATO TINÓS

Ribeirão Preto
2012

UNIVERSIDADE DE SÃO PAULO
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto
Departamento de Computação e Matemática

Computação Evolutiva em Ambientes Dinâmicos

RENATO TINÓS

Tese submetida à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da UNIVERSIDADE DE SÃO PAULO como parte dos requisitos para a obtenção do título de Livre-Docente na Área de Ciências de Computação, especialidade: Computação Bioinspirada.

Ribeirão Preto, maio de 2012.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE DOCUMENTO, POR MEIO CONVENCIONAL OU ELETRÔNICO PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Tinós, R.
Computação Evolutiva em Ambientes Dinâmicos/ Renato Tinós – Riberão Preto/SP, 2012.
114p .: il.

Tese (Livre-Docente. Área de Ciências de Computação, especialidade: Computação Bioinspirada) – Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da UNIVERSIDADE DE SÃO PAULO.

1. Computação Evolutiva 2. Problemas de Otimização Dinâmica 3. Algoritmos Genéticos

DEDICATÓRIA

Dedico este trabalho à Lúcia, ao Cauê e ao Iago. Pelas alegrias do dia a dia e por me incentivarem a sonhar.

Agradecimentos

Aos orientandos e ex-orientandos, pela troca de conhecimentos: Edward, Fernanda, Vinícius, Gabriela, Luiz Eduardo, Lariza, Luis Henrique, Helder, Gustavo, Luca, Diego, Michel, Ana Lívia e Ariadne. Agradeço ainda a oportunidade de tentar contagiá-los com a paixão pela Ciência.

Aos colegas que propiciaram as mais diversas discussões científicas, em especial, aquelas que me geraram dúvidas.

Aos co-autores, que compartilharam comigo o caminho excitante da descoberta científica. Em especial, aos professores Evandro Ruiz e André Carvalho pelo incentivo, e ao professor Sheng-xiang Yang pela co-autoria de diversos dos trabalhos citados nesta tese.

Aos meus mestres, em especial aos primeiros deles: os meus pais e o meu irmão.

Sumário

Lista de Figuras p. v

Lista de Tabelas p. viii

Normas e convenções p. ix

Resumo p. x

Abstract p. xi

I Introdução aos Algoritmos Evolutivos Aplicados em Otimização Dinâmica 1

1 Introdução p. 2

2 Algoritmos Evolutivos p. 5

3 Problemas de Otimização Dinâmica p. 9

3.1 Problema da Mochila 0-1 Dinâmico . . . . . p. 9

3.2 Escalonamento *Job-Shop* Dinâmico . . . . . p. 10

3.3 Gerador de Problemas Binários Dinâmicos XOR . . . . . p. 10

3.4 Gerador *Moving Peaks* . . . . . p. 11

3.5 Gerador de Problemas Dinâmicos Contínuos GDBG . . . . . p. 12

4 Medidas de Desempenho e Teoria p. 13

4.1 Medidas de Desempenho em AEs Aplicados à Otimização Dinâmica . . . . . p. 13

4.2 Introdução à Teoria . . . . . p. 14

5 Algoritmos Evolutivos Para Problemas de Otimização Dinâmica p. 16

5.1 Geração e Manutenção da Diversidade Durante o Processo Evolutivo . . . . . p. 17

5.2 Uso de Conhecimento Obtido Durante o Processo Evolutivo . . . . . p. 18

5.3 Uso de Múltiplas Populações . . . . . p. 18

II Algoritmos 20

6 Algoritmo Genético com Taxa de Mutação Dependente do Gene p. 21

6.1 Algoritmo . . . . . p. 21

6.2 Determinação dos Parâmetros . . . . . p. 24

6.3 Análise . . . . . p. 25

6.4 Resultados . . . . . p. 28

6.4.1 Seguimento de Padrões Binários . . . . . p. 28

6.4.2 Otimização de Função Unimodal . . . . . p. 29

6.5 Comentários Finais . . . . . p. 29

7 Algoritmo Genético com Imigrantes Aleatórios Auto-Organizado p. 34

7.1 Criticalidade Auto-Organizada . . . . . p. 34

7.2 Algoritmo . . . . . p. 37

7.3 Resultados . . . . . p. 39

7.3.1 Descrição das Simulações . . . . . p. 40

7.3.2 Resultados . . . . . p. 42

7.3.3 Análise dos Resultados . . . . . p. 46

7.4 Comentários Finais . . . . . p. 50

8 Algoritmos Evolutivos com Mutação q-Gaussiana p. 52

8.1 A Distribuição q-Gaussiana . . . . . p. 54

8.2 Auto-adaptação da distribuição de mutações . . . . . p. 56

8.3 ES com Mutação q-Gaussiana . . . . . p. 57

8.4 Análise da mutação q-Gaussiana . . . . . p. 58

8.4.1 O impacto de mudar σ . . . . . p. 59

8.4.2 O impacto de mudar q . . . . . p. 60

8.5 Experimentos . . . . . p. 62

8.5.0.1 Resultados . . . . . p. 63

8.5.0.2 Análise dos Resultados . . . . . p. 64

III Aspectos Teóricos 69

9 Gerador de Problemas Dinâmicos Contínuos Através de Rotação p. 70

9.1 Análise do Gerador de DOPs Binários XOR . . . . . p. 70

9.2 Um Gerador de DOPs Contínuos com Controle de Rotação dos Indivíduos . . p. 71

9.3 Teste do Gerador de DOPs Contínuos Através de Rotação dos Indivíduos . . p. 73

9.4 Comentários Finais . . . . . p. 75

10 Análise por Sistemas Dinâmicos p. 76

10.1 Modelo Exato do GA . . . . . p. 76

10.2 DOPs sob o Ponto de Vista do Modelo Exato . . . . . p. 78

10.3 DOPs Produzidos pelo Gerador XOR . . . . . p. 82

10.4 DOPs em um Problema Simples Envolvendo Robôs Móveis . . . . . p. 87

10.4.1 Robô Móvel Simulado . . . . . p. 87

10.4.2 Análise do Sistema Dinâmico do GA . . . . . p. 88

10.4.3 Simulações . . . . . p. 92

IV Conclusão **95**

Referências p. 99

Lista de Figuras

3.1 Duas superfícies de fitness produzidas pelo Gerador *Moving Peaks*. . . . . p. 12

6.1 Número médio de épocas após o início do ciclo de mudança em que o melhor indivíduo alcança o $fitness$ máximo e respectivo desvio padrão para o exemplo do seguimento de padrões binários. Acima: GA padrão. Abaixo: GAGDM. . . . . p. 30

6.2 Valores médios das probabilidades de mutação dos genes do melhor indivíduo no fim de três ciclos de mudança distintos no exemplo do seguimento de padrões binários. . . . . p. 31

6.3 Número médio de épocas após o início do ciclo de mudança em que o melhor indivíduo alcança o $fitness$ máximo e respectivo desvio padrão para o exemplo da otimização de função unimodal. Acima: GA padrão. Abaixo: GAGDM. . . . . p. 32

6.4 Valores médios das probabilidades de mutação dos genes do melhor indivíduo no fim de três ciclos de mudança distintos no exemplo da otimização de função unimodal. . . . . p. 33

7.1 Fitness dos indivíduos da população corrente nas gerações 27, 28 e 29 em uma execução do SORIGA no exemplo dado. Os 0's e 1's no alto dos gráficos indicam os indivíduos que pertencem à subpopulação. . . . . p. 39

7.2 Disposição dos sensores de distância no robô simulado. . . . . p. 41

7.3 $Fitness$ do melhor indivíduo da geração corrente para a simulação 1. . . . . p. 43

7.4 $Fitness$ do melhor indivíduo da geração corrente para a simulação 2. . . . . p. 44

7.5 Simulação de um individuo obtido na geração anterior à mudança no ambiente na sétima execução da simulação 1 utilizando o SORIGA. O robô é representado através de um círculo de cor branca e sua trajetória através de uma linha tracejada. A área de recarga de bateria, representada por um semicírculo no canto superior da arena, possui uma cor diferente no chão. Uma torre de iluminação, representada por um pequeno círculo hachurado, se encontra no canto superior da arena. Nos gráficos à direita, os sinais de saída para os motores (o0 e o1) e as entradas da rede provenientes dos sensores (I0 a I11) são representados nos eixos verticais e o tempo no eixo horizontal. Esta figura foi obtida através do simulador EVOROBOT. . . . . p. 45

7.6 Simulação de um individuo obtido na geração final na sétima execução da simulação 1 utilizando o SORIGA. A arena simulada tem tamanho diferente daquela apresentada na Figura 7.5, além de possuir cinco obstáculos cilíndricos, representados na figura por círculos hachurados. . . . . p. 46

7.7 Simulação de um individuo sem falhas obtido na geração anterior à mudança no problema na décima execução da simulação 2 utilizando o SORIGA. . . . p. 47

7.8 Simulação de um individuo com falhas obtido na geração final na décima execução da simulação 2 utilizando o SORIGA. . . . . p. 47

7.9 *Fitness* do melhor indivíduo e duração das extinções na décima execução da simulação 2 (reconfiguração após falhas). . . . . p. 50

7.10 Número de ocorrências para cada tamanho das extinções na décima execução da simulação 2 (reconfiguração após falhas). . . . . p. 50

8.1 Função q-Gaussiana para diferentes valores de q. As funções Gaussiana e de Cauchy são também apresentadas. . . . . p. 55

8.2 Regiões positivas (em cinza) e negativas (em branco) das derivadas da q-exponencial dada pela Eq. 8.24 para $\sigma = 3$. . . . . p. 61

8.3 Mediana do erro da média do melhor fitness em cada ciclo de mudança para diferentes valores de ρ (GES: triângulo; CES: "+"; qGES: quadrado). . . . . p. 67

8.4 Média da norma Euclidiana do vetor de parâmetros de força de mutação e do parâmetro de distribuição q do melhor indivíduo nos últimos 10 ciclos de mudança do experimento com $\tau = 300$ e $\rho = 0,1$ ou $\rho = 0,2$ (GES: linha sólida azul; CES: linha tracejada verde; qGES: linha traço-ponto vermelha). . p. 68

9.1 Fitness médio (sobre 30 execuções) do melhor indivíduo da população no DOP criado a partir da função esfera (SGA: linha sólida; RIGA: linha tracejada; HGA: linha pontilhada). . . . . p. 74

9.2 Fitness médio (sobre 30 execuções) do melhor indivíduo da população no DOP criado a partir da função de Griewank generalizada (SGA: linha sólida; RIGA: linha tracejada; HGA: linha pontilhada). . . . . p. 75

10.1 Trajetória da população durante 6 ciclos de mutança para $\tau = 70$ e $rho = 0,875$ no DOP criado pelo gerador XOR: (a) fitness médio da população; (b) distância para os atuais estados metaestáveis principal (linha sólida) e secundário (linha pontilhada). . . . . p. 84

10.2 Fitness normalizado (a) e distância para o estado metaestável principal (b) no DOP criado pelo Gerador XOR para diferentes valores de τ e ρ. Os valores são relativos à média (sobre 30 ciclos de mudança) obtidos pelo vetor de população na geração antes da mudança. . . . . p. 86

10.3 Espaço de fitness (vetor $\mathbf{f}$) para o modelo 1 ($l = 4$). As soluções são apresentadas de acordo com o valor inteiro de $\mathbf{x}$. . . . . p. 91

10.4 Média do fitness da população e distância para três estados metaestáveis em simulações para os três tipos de falhas. A distância para o atual estado metaestável principal é apresentada pela linha sólida. . . . . p. 94

Lista de Tabelas

7.1 Resultados do SORIGA no problema de falhas em robôs móveis . . . . . p. 43

8.1 Comparação estatística em relação ao erro da média do melhor fitness em cada ciclo de mudança. . . . . p. 64

Normas e convenções

Este documento foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza a classe *ieeetr* do BibTeX, que segue as recomendações do IEEE (*Institute of Electrical and Electronics Engineers*) para publicação em periódicos da instituição.

A formatação da capa, folha de rosto, folha de aprovação, resumo e abstract segue as "diretrizes para apresentação de dissertações e teses da USP", disponivel em <*http://www.teses.usp.br*>.

A formatação de sumário, lista de figuras e tabelas, lista de abreviaturas e siglas, espaçamento entre linhas, numeração de páginas e cabeçalhos de páginas segue a norma ABNT NBR 14724 para "Apresentação de trabalhos acadêmicos".

A formatação de títulos e capítulos de seções segue a norma ABNT NBR 6024 para "Numeração progressiva das seções de um documento".

Todas as formatações que seguem a norma ABNT foram geradas automaticamente utilizando as macros da classe abntex disponíneis em <*http://abntex.codigolivre.org.br/*>.

Resumo

TINÓS, R.. **Computação Evolutiva em Ambientes Dinâmicos**. Tese (Livre-Docente) – Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2012.

Algoritmos Evolutivos (AEs) são meta-heurísticas populacionais inspiradas em princípios básicos da evolução natural e de outros paradigmas biológicos. Diversos aplicações de AEs ocorrem em ambientes dinâmicos, nos quais a função de avaliação, as variáveis de decisão e/ou as restrições do problema mudam durante o processo de otimização. Em tais problemas, AEs tradicionais geralmente não apresentam desempenho satisfatório. Esta tese trata do problema do uso de Computação Evolutiva em ambientes dinâmicos sob diversos ângulos. Na primeira parte do trabalho, uma visão geral sobre o problema tratado é apresentada. As duas partes seguintes desta tese aprofundam os temas estudados, trazendo com detalhes exemplos de técnicas práticas e teóricas, todas propostas pelo autor desta tese. Três AEs especialmente desenvolvidos para ambientes dinâmicos são apresentados: o Algoritmo Genético com Taxa de Mutação Dependente do Gene, o Algoritmo Genético com Imigrantes Aleatórios Auto-Organizado; e os Algoritmos Evolutivos com Mutação q-Gaussiana. Com relação aos aspectos teóricos do problema estudado, é apresentada, entre outras, a análise de Algoritmos Genéticos pelo enfoque dos sistemas dinâmicos.

Palavras-chave: Computação Evolutiva, Problemas de Otimização Dinâmica, Algoritmos Genéticos.

Abstract

TINÓS, R.. **Evolutionary Computation in Dynamic Environments**. Thesis (Livre-Docente) – Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2012.

Evolutionary Algorithms (EAs) are population meta-heuristics inspired by the principles of natural evolution and other biological paradigms. Many real world problems occur in dynamic environments, where the evaluation function, decision variables and/or the constraints of the problem change during the optimization process. When the optimization problem changes during the evolutionary process, traditional EAs generally do not present good performance. This thesis addresses the problem of the use of Evolutionary Computation in dynamic environments from different aspects. In the first part of the work, an overview of the problem addressed is presented. The next two parts present in details examples of practical and theoretical techniques for the investigated topic, all proposed by the author of this thesis. Three EAs specially developed for dynamic environments are presented: the Genetic Algorithm with Gene Dependent Mutation Rate, the Genetic Algorithm with Self-Organized Random Immigrants, and the Evolutionary Algorithms with q-Gaussian Mutation . Concerning the theoretical aspects of the problem, it is presented, among others, the analysis of Genetic Algorithms for dynamic environments using the dynamical systems approach.

Keywords: Evolutionary Computation, Dynamic Optimization Problems, Genetic Algorithms.

"O universo (que outros chamam a Biblioteca) é composto de um número indefinido, e talvez infinito, de galerias hexagonais , com vastos poços de ventilação no meio . . . A distribuição das galerias é invariável. Vinte prateleiras, com cinco longas prateleiras por lado, cobrem todos os lados menos dois; sua altura, que é a dos andares, mal ultrapassa a de um bibliotecário normal. Uma das faces livres dá para um corredor apertado, que desemboca noutra galeria, idêntica à primeira e a todas. . . A cada um dos muros de cada hexágono correspondem cinco prateleiras; cada prateleira contém trinta e dois livros de formato uniforme; cada livro tem quatrocentas e dez páginas; cada página, quarenta linhas; cada linha, umas oitenta letras de cor negra. Também há letras no dorso de cada livro; essas letras não indicam ou prefiguram o que dirão as páginas.

. . . quero rememorar alguns axiomas. . . . O segundo: o número de símbolos ortográficos é vinte e cinco. Esta constatação permitiu, há trezentos anos, formular uma teoria geral da Biblioteca e resolver satisfatoriamente o problema que nenhuma conjectura tinha decifrado: a natureza informe e caótica de quase todos os livros. Um, que meu pai viu no hexágono do circuito quinze noventa e quatro, constava das letras M C V perversamente repetidas desde a primeira linha à última. Outro (muito consultado nesta zona) é um mero labirinto de letras, mas a página penúltima diz - Oh tempo tuas pirâmides -. . .

Dessas premissas inconvertíveis deduziu que a Biblioteca é total e que suas prateleiras registram todas as possíveis combinações dos vinte e tantos símbolos ortográficos (número, ainda que vastíssimo, não infinito), ou seja, tudo o que é dado expressar: em todos os idiomas . . .

Naquele tempo falou-se muito das Vindicações: livros de apologia e de profecia, que para sempre vindicavam os atos de cada homem do universo e guardavam arcanos prodigiosos para o seu futuro. Milhares de cobiçosos abandonaram o doce hexágono natal e precipitaram-se escadas acima, premidos pelo vão propósito de encontrar sua Vindicação . . . Também se esperou então o esclarecimento dos mistérios básicos da humanidade: a origem da Biblioteca e do tempo . . .

Em alguma estante de algum hexágono (raciocinaram os homens) deve existir um livro que seja a cifra e o compêndio perfeito de todos os demais: algum bibliotecário o consultou e é análogo a um deus.... Muitos peregrinaram à procura d´Ele. Durante um século trilharam em vão os mais diversos rumos. Como localizar o venerado hexágono secreto que o hospedava? Alguém propôs um método regressivo: Para localizar o livro A, consultar previamente um livro B, que indique o lugar de A; para localizar o livro B, consultar previamente um livro C, assim até o infinito . . .

Em aventuras como essas, prodigalizei e consumi meus anos. Não me parece inverossímil que em alguma prateleira do universo haja um livro total;. . . Se a honra e a sabedoria e a felicidade não são para mim, que sejam para outros. . . "

"A biblioteca de babel", livro "Ficções", de Jorge Luis Borges (3a ed., 2001, Ed. Globo).

Parte I

Introdução aos Algoritmos Evolutivos Aplicados em Otimização Dinâmica

1 Introdução

A Natureza possui mecanismos eficientes para tratar diversos tipos de problemas complexos que, quando descritos computacionalmente, são difíceis de serem manipulados por técnicas de computação tradicionais. Inspiradas em tais mecanismos, diversas soluções computacionais têm sido propostas nas últimas décadas. Entre estas fontes de inspiração da Natureza, podemos citar o sistema nervoso, como no caso das Redes Neurais Artificiais [1], o comportamento de grupos de indivíduos, como nos casos da Otimização por Colônias de Formigas [2] e da Otimização por Enxame de Partículas [3], o sistema imunológico, como no Sistemas Imunológicos Artificiais [4], e a genética e a evolução natural, como na Computação Evolutiva [5].

Algoritmos Evolutivos (AEs) têm sido aplicados com sucesso em um grande número de problemas de otimização nas últimas décadas. AEs são meta-heurísticas populacionais inspiradas por princípios básicos da evolução das espécies. Como em outras partes do mundo, um crescente número de pesquisas em Computação Evolutiva têm sido feita no Brasil recentemente [6]. No Brasil, assim como no resto do mundo, a maioria dos problemas de otimização evolutiva tem sido descrita como problemas estacionários, nos quais a função de avaliação, as variáveis de decisão e as restrições do problema são mantidas fixas durante o processo de otimização. Contudo, em diversas aplicações, os problemas de otimização são dinâmicos, nos quais a função de avaliação, as variáveis de decisão e/ou as restrições do problema mudam durante o processo evolutivo. Diversos fatores são responsáveis por tais mudanças, como, por exemplo, falhas, alterações no ambiente, chegada de novos requisitos e demandas, objetivos que se alteram, mudanças econômicas e variáveis humanas.

Como exemplo, considere o caso da otimização das leis de controle de um robô móvel em um ambiente não-estruturado [7]. Se a avaliação de uma solução é feita online, ou seja, o robô com leis de controle ditadas pela solução que está sendo avaliada deve executar uma tarefa e, de acordo com seu desempenho, o valor da aptidão é gerado, o processo de otimização será em geral extremamente longo [8]. Por exemplo, considerando que a avaliação do indivíduo demore 1 minuto, o tempo de execução para um processo com 100 gerações e uma população do AE com 50 indivíduos excederá 3 dias. Neste tempo, diversas alterações poderão ocorrem

no ambiente e no robô, como mudanças na potência das baterias devido ao consumo da energia, alterações na luminosidade do ambiente, entre outras [7].

Quando mudanças ocorrem no problema depois de um processo de otimização ter sido executado, a solução encontrada pode não ser mais eficaz, sendo que uma nova solução deve ser encontrada. A solução mais simples para lidar com a mudança no problema é iniciar um novo processo de otimização quando alterações no desempenho das melhores soluções ocorrem. Contudo, como no exemplo dado no parágrafo anterior, o processo de otimização muitas vezes requer um longo tempo e um substancial esforço computacional. Se a nova solução depois da mudança é de alguma forma relacionada às soluções prévias, estas podem ser utilizadas para se encontrar a solução requerida pelas condições atuais do processo, o que pode representar uma enorme economia de tempo e esforço computacional. Deve-se ressaltar que estes requisitos são de vital importância em várias situações, como, por exemplo, quando a nova solução deve ser obtida em uma janela curta de tempo.

AEs são particularmente atrativos para tais problemas. Indivíduos da população anterior à mudança no problema podem gerar a nova população após a mudança no problema. Entretanto, após várias gerações, os indivíduos da população geralmente convergem no espaço de busca para pontos vizinhos ao melhor indivíduo atual. Se a superfície de fitness muda abruptamente, a população atual pode ficar presa em ótimos locais vizinhos à melhor solução encontrada antes da mudança no problema. Além disso, mesmo em problemas que mudam suavemente, o conhecimento obtido durante o processo evolutivo pode ser útil para se encontrar as novas soluções após as mudanças. Assim, o estudo do desempenho de AEs tradicionais e/ou o desenvolvimento de novos AEs para tais problemas são necessários. Desta forma, problemas de otimização que ocorrem em ambientes dinâmicos, ou Problemas de Otimização Dinâmica (*Dynamic Optimization Problems* - DOPs), têm atraído crescente atenção da comunidade de Computação Evolutiva nos últimos anos [9], [10], [11], [12].

Esta tese trata o problema do uso de Computação Evolutiva em ambientes dinâmicos sob diversos ângulos. Na primeira parte do trabalho, uma visão geral sobre o problema tratado é apresentada. Primeiramente, no Capítulo 2, AEs são brevemente apresentados. Em seguida (Capítulo 3), DOP é definido formalmente, bem como são apresentados exemplos destes problemas de forma a introduzir o leitor à temática estudada de forma prática. No Capítulo 4, algumas medidas de desempenho utilizadas em AEs aplicados em DOPs e aspectos teóricos destes são introduzidos. Em geral, técnicas evolutivas para DOPs podem ser classificados em três categorias principais: 1) Geração e manutenção da diversidade durante o processo evolutivo; 2) Uso de conhecimento obtido durante o processo de otimização; 3) Uso de múltiplas

populações. Esta classificação é comentada no Capítulo 5, bem como são apresentadas algumas técnicas típicas de cada categoria.

As duas partes seguintes desta tese aprofundam os temas estudados nos capítulos 4 e 5, trazendo com detalhes exemplos de técnicas práticas e teóricas, todas propostas em artigos científicos cujo primeiro autor é o autor desta tese. Na Parte II, três AEs especialmente desenvolvidos para DOPs são apresentados. São eles: Algoritmo Genético com Taxa de Mutação Dependente do Gene (Capítulo 6); Algoritmo Genético com Imigrantes Aleatórios Auto-Organizado (Capítulo 7); e Algoritmos Evolutivos com Mutação q-Gaussiana (Capítulo 8).

A inspiração biológica e os bons resultados alcançados em diversos problemas práticos não nos eximem de buscar o entendimento teórico para o funcionamento e desempenho dos AEs. Pelo contrário, reforçam ainda mais a necessidade de investigar AEs de um ponto de vista teórico. Desta forma, na Parte III, aspectos teóricos do problema estudado são apresentados (aspectos teóricos dos algoritmos propostos são também discutidos ao longo dos capítulos da Parte II). Primeiramente, o Gerador de Problemas Dinâmicos Contínuos Através de Rotação, obtido através da análise do Gerador de Problemas Binários XOR [13], é apresentado no Capítulo 9. Em seguida, a análise de Algoritmos Genéticos pelo enfoque dos sistemas dinâmicos, desenvolvida para problemas estacionários por Vose [14], é estendida para DOPs no Capítulo 10.

Finalmente, as observações finais deste trabalho aparecem na Parte IV.

2 Algoritmos Evolutivos

Uma importante técnica para otimização é a dos métodos iterativos, empregados largamente quando métodos exaustivos e de cálculo analítico não podem ser aplicados ou apresentam um custo computacional inviável. Em um método iterativo, transforma-se a solução (ou conjunto de soluções) atual(is) do problema iterativamente através de um operador determinado por uma regra pré-definida, partindo-se de uma solução (ou conjunto de soluções) inicial(is). Após cada solução ser gerada, sua qualidade é avaliada através da função de avaliação. As novas soluções geradas podem ou não ser aceitas, o que é feito de acordo com um operador, determinado por uma regra também pré-definida, que aqui, por conveniência, é chamada de operador de seleção. Basicamente, os métodos iterativos são diferenciados de acordo com os operadores de transformação e seleção empregados [1].

Por exemplo, considere o método do Gradiente Descendente [15]. Neste caso, o operador de trasformação da solução atual é definido através do gradiente local da função de avaliação, enquanto que o operador de seleção especifica que toda nova solução gerada é aceita. Note que ambos operadores não apresentam caráter estocástico, o que faz com que o método seja classificado com determinístico (não contando, é claro, a forma de inicialização das soluções).

Já no método do Recozimento Simulado (*Simulated Annealing*) [16], as soluções, que são reais, são trasformadas através da adição de um desvio aleatório gerado a partir de uma distribuição Gaussiana, enquanto que a nova solução é aceita ou não de acordo com o critério de seleção de Boltzmann, cujo parâmetro que define o rigor com que uma solução pior que a atual seja aceita decai ao longo do processo de otimização. O Recozimento Simulado é um método iterativo com operadores estocásticos, classificado como meta-heurística por ser uma técnica heurística genérica e adaptável para resolver problemas de otimização.

Algoritmos Evolutivos (AEs) são uma classe de meta-heurísticas populacionais utilizadas em otimização. A idéia principal por trás da Computação Evolutiva é criar um conjunto de

[1] Os critérios de inicialização das soluções e parada, bem como outros detalhes como a seleção de parâmetros dos métodos, podem também ser importantes em muitos casos, mas geralmente não assumem a mesma relevância para a dinâmica do processo de otimização dos operadores de transformação e seleção

soluções candidatas para um determinado problema e otimizá-las usando operadores de seleção e trasformação (aqui chamados de reprodução) inspirados naqueles encontrados na evolução por Seleção Natural e na Genética [17]. Na terminologia da Computação Evolutiva, as soluções candidatas são chamadas de indivíduos e o conjunto atual de soluções candidatas (ou um subconjunto deste) é conhecido como população. As soluções são codificadas nos indivíduos através de um conjunto de objetos, chamado de cromossomo.

Na classe dos AEs, é utilizada uma grande variedade de operadores de transformação e seleção, o que os tornam uma ferramenta bastante versátil. Por exemplo, se o operador de transformação utilizado é a mutação Gaussiana e o critério de seleção de Boltzmann com adaptação dos parâmetros é empregado, o AE reproduz o Recozimento Simulado para uma população com apenas um indivíduo. Esta versatilidade explica em parte a grande utilização de AEs em problemas de otimização em que técnicas tradicionais não têm apresentado bons resultados.

Os algoritmos 1 e 2 apresentam os pseudo-códigos com os formatos básicos geralmente encontrados em AEs. Nos pseudocódigos apresentados, $\mathbf{P}(t)$ é a população de indivíduos na iteração t, também conhecida como geração t [18], [19]. Na função "*inicializePopulacao*", os indivíduos da população inicial são muitas vezes produzidos aleatoriamente a partir da distribuição uniforme. No esquema geracional, uma nova população é formada inteiramente por descendentes gerados a partir dos pais da geração anterior, enquanto que no esquema não-geracional, somente alguns indivíduos são substituídos em cada geração [20].

Algoritmo 1 Estutrutura básica do Algoritmo Evolutivo Geracional

1: $t \leftarrow 1$
2: inicializePopulacao($\mathbf{P}(t)$)
3: avaliePopulacao($\mathbf{P}(t)$)
4: **enquanto** (critério de parada não é satisfeito) **faça**
5: $\quad \mathbf{P}(t+1) \leftarrow$ selecao($\mathbf{P}(t)$)
6: $\quad$ trasformePopulacao($\mathbf{P}(t+1)$)
7: $\quad$ avaliePopulacao($\mathbf{P}(t+1)$)
8: $\quad t \leftarrow t+1$
9: **fim enquanto**

AEs diferem de técnicas de otimização tradicionais principalmente porque [17]:

- AEs trabalham com uma população de soluções candidatas em paralelo. O uso da população de soluções candidatas muitas vezes reduz as chances de aprisionamento em ótimos locais, principalmente nas iterações iniciais do processo de otimização.
- Como Recozimento Simulado e outros métodos iterativos estocásticos, AEs utilizam regras de transição probabilísticas ao invés de regras determinísticas.

Algoritmo 2 Estutrutura básica do Algoritmo Evolutivo Não-Geracional (*Steady State*)

1: $t \leftarrow 1$
2: inicializePopulacao($\mathbf{P}(t)$)
3: avaliePopulacao($\mathbf{P}(t)$)
4: **enquanto** (critério de parada não é satisfeito) **faça**
5: $\mathbf{Q}(t) \leftarrow$ trasformePopulacao($\mathbf{P}(t)$)
6: avaliePopulacao($\mathbf{Q}(t)$)
7: $\mathbf{P}(t+1) \leftarrow$ selecao($\mathbf{P}(t) \cup \mathbf{Q}(t)$)
8: $t \leftarrow t+1$
9: **fim enquanto**

- AEs utilizam somente a informação dos valores de custo das soluções avaliadas para guiar o processo de otimização, não sendo necessário conhecimento auxiliar como, por exemplo, as derivadas da função de custo. Esta característica torna o processo mais lento em muitos problemas em que informações adicionais sobre o espaço de busca são disponíveis e úteis. Entretanto, é extremamente interessante em problemas em que estas informações não estão disponíveis, ou não são confiáveis, ou quando o uso das informações adicionais pode levar a decisões ineficazes (por exemplo, o uso de derivadas em espaços de busca ruidosos).

- A otimização pode ocorrer no espaço das soluções candidatas codificadas, o que pode representar uma vantagem em alguns problemas [15], reduzindo, por exemplo, o número de ótimos locais.

As três principais áreas tradicionais de pesquisa em AEs são em Algoritmos Genéticos (*Genetic Algorithms* - GAs), Estratégias Evolutivas (*Evolution Strategies* - ESs) e Programação Evolutiva (*Evolutionary Programming* - EP) [20], [19]. É importante observar que as delimitações que definiam tais áreas no passado estão sendo rompidas, já que um crescente número de pesquisadores de determinadas áreas está adotando operadores e estratégias empregadas tradicionalmente em outras áreas.

GAs tradicionalmente utilizam um vetor (cromossomo) binário para codificar a solução, apesar de outras representações poderem ser encontradas. GAs são geralmente geracionais, sendo que em um primeiro momento, indivíduos da população corrente são selecionados através de um dado critério de seleção, por exemplo, definindo a probabilidade de seleção de um indivíduo proporcionalmente ao seu fitness, como na seleção por roleta. Então, operadores genéticos de reprodução são aplicados nos indivíduos selecionados. Os operadores genéticos mais populares são o crossover, no qual componentes dos cromossomos de dois indivíduos selecionados (pais) são trocados, e a mutação, na qual um gene tem seu valor alterado de acordo com

uma regra pré-definida. Em ambos os casos, os operadores são aplicados probabilisticamente de acordo com taxas definidas a priori.

No começo da década de 1990, Koza propôs a Programação Genética (*Genetic Programming* - GP) para evoluir programas automaticamente através de algoritmos baseados em GAs [21], [20]. Na PG, os cromossomos são estruturas não-lineares, diferentemente dos AGs que utilizam estruturas lineares. Desde então, PG se tornou bastante popular, representando uma fértil área de pesquisa na Computação Evolutiva.

ESs foram propostas para otimizar soluções candidatas com valores reais [22]. Na (μ,λ)-ES, que representa uma das mais usadas formas de ESs, uma população de μ pais cria $\lambda \geq \mu$ descendentes através de recombinação e mutação. Os melhores μ descendentes são então selecionados para compor a nova população. De forma diferente, na $(\mu+\lambda)$-ES, a nova população é composta pelos melhores μ indivíduos obtidos pela união dos μ pais e λ descendentes (veja Algoritmo 2). Pode-se observar que a versão $(\mu+\lambda)$ é elitista, enquanto que a versão (μ,λ)-ES é não elitista, o que a torna, geralmente, mais apropriada para ambientes dinâmicos. Na mutação, geralmente se utiliza uma distribuição normal com média zero e desvio-padrão pré-definido para alterar a solução candidata que está sendo mutada [19]. A recombinação pode ser discreta (similar ao crossover uniforme nos GAs), ou intermediária, na qual a média aritmética entre os valores dos genes dos pais é utilizada para gerar os genes de um descendente.

Na forma inicial da EP, em uma tentativa de criar inteligência artificial inspirada na evolução por Seleção Natural [23], as soluções candidatas eram representadas através de máquinas de estados finitos que eram evoluídas através de mutações aleatórias e seleção elitista. Atualmente, representações de EP mais usuais utilizam números reais, o que resulta em algoritmos similares às ESs. No entanto, diferentemente destas que podem utilizar recombinação, EP é baseada exclusivamente em mutação.

3 Problemas de Otimização Dinâmica

Considerando o problema de otimização como um problema de minimização, podemos definir o problema de otimização dinâmica como:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}(t),t) \\ \text{sujeito a} \quad & g_i(\mathbf{x}(t),t) \geq 0 \end{aligned} \tag{3.1}$$

na qual $f(\mathbf{x}(t),t)$ é a função objetivo (função de fitness) no instante t, g_i é a i-ésima função de restrição, sendo $i = 1,\ldots,m(t)$, $m(t)$ é o número de restrições no instante t, $\mathbf{x}(t) \in \Omega(t)$ é a solução candidata representada por um vetor com dimensão $n(t)$, e $\Omega(t)$ é um conjunto de candidatos a pontos ótimos no instante t.

Repare que é possível que várias características do problema mudem com o tempo [9]. Na maioria dos problemas encontrados na literatura, apenas a função de fitness é dinâmica, mas é possível que as restrições do problema e o número de variáveis da solução candidata ($n(t)$) também se alterem com o tempo.

A seguir, são apresentados alguns DOPs de referência e geradores de DOPs utilizados por pesquisadores na área de Computação Evolutiva em otimização dinâmica.

3.1 Problema da Mochila 0-1 Dinâmico

O problema da mochila 0-1 é um problema combinatório do tipo NP-completo no qual o objetivo é selecionar o subconjunto de itens com máxima soma de valores sem que um dado limite da soma de pesos (capacidade da mochila) seja atingido. Na forma dinâmica, os pesos e valores dos itens, o número máximo de itens e/ou a capacidade da mochila podem variar com o tempo. Dado que existem $n(t)$ itens, o problema dinâmico pode ser descrito como:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n(t)} p_i(t)x_i \\ \text{sujeito a} \quad & \textstyle\sum_{i=1}^{n(t)} w_i(t)x_i \leq c(t) \end{aligned} \tag{3.2}$$

sendo $x_i \in \{0,1\}$, $w_i(t)$ e $p_i(t)$ respectivamente o peso e o valor do item i no instante t, e $c(t)$ a capacidade da mochila no instante t. Em geral, a função de fitness é definida como a soma de valores na mochila se a capacidade desta for respeitada [5]. Caso contrário, uma punição proporcional ao valor do peso excedido é aplicada.

Apesar de todos os parâmetros poderem ser alterados durante o processo de otimização, geralmente são descritos na literatura experimentos no qual apenas poucos (em geral um) parâmetros são modificados durante o processo evolutivo. Por exemplo, em [24], apenas a capacidade da mochila $c(t)$ é alterada, alternando-se a cada 20000 avaliações de fitness entre os valores 50%, 30% e 80% da soma total dos pesos de todos os itens.

3.2 Escalonamento *Job-Shop* Dinâmico

Na versão geral deste problema, deve-se planejar como n tarefas com tamanhos e prioridades diferentes devem ser executadas em $m < n$ máquinas idênticas. O objetivo é encontrar o melhor escalonamento que satisfaça as prioridades. Na versão estática, as tarefas e o número de máquinas são conhecidos antes do processo de otimização ser iniciado. Na versão dinâmica, também chamada estocástica, o planejamento é feito online, conforme os processos chegam [9]. Este é um problema de grande interesse para diversas áreas, como em sistemas de computação nos quais as tarefas chegam a todo instante. Repare que no caso dinâmico, o número de variáveis de decisão, $n(t)$, depende do tempo, sendo que uma realocação dos processos geralmente tem que ser feita quando um novo processo chega. Neste caso, a codificação utilizada pelo AE deve levar em conta as mudanças em $n(t)$. Pode-se também considerar que o número de máquinas, $m(t)$, muda ao longo do processo de otimização.

3.3 Gerador de Problemas Binários Dinâmicos XOR

Em [13], um gerador que permite criar DOPs a partir de qualquer problema de otimização binário estacionário é apresentado. Geradores de DOPs são interessantes pois permitem comparar diferentes AEs em problemas com diferentes características. No Gerador de DOPs XOR, dado um problema estacionário com função de fitness $f(\mathbf{x})$ e soluções candidatas $\mathbf{x} \in \{0,1\}^n$, a função de fitness de um problema, que é periodicamente mudada a cada τ gerações, é dada por:

$$f(\mathbf{x}, e) = f(\mathbf{x} \oplus \mathbf{m}(e)), \tag{3.3}$$

na qual $\oplus$ é o operador XOR, $e = \lceil t/\tau \rceil$ é o índice do ciclo de mudança (i.e., o conjunto de τ gerações no qual a função de fitness não muda), t é o índice da geração, e $\mathbf{m}(e)$ é a máscara binária para o ciclo de mudança e, a qual é incrementada de acordo com:

$$\mathbf{m}(e) = \mathbf{m}(e-1) \oplus \mathbf{r}(e), \tag{3.4}$$

sendo $\mathbf{r}(e)$ um template binário aleatório para o ciclo de mudança e contendo $\lfloor \rho \times n \rfloor$ uns, e $\{\rho \in \mathbb{R} \mid 0,0 \leq \rho \leq 1,0\}$ controla o grau de mudança para o DOP. Se $\rho = 0$, o problema permanece estacionário, enquanto que se $\rho = 1$, a máxima mudança na distância de Hamming ocorre. No primeiro ciclo de mudança, $\mathbf{m}_1$ é igual a um vetor de zeros. Ajustando τ é possível controlar a frequência das mudanças, enquanto que o ajuste de ρ possibilita o controle da severidade das mudanças.

Os problemas produzidos pelo Gerador de DOPs XOR serão analisados mais tarde nesta tese (Parte III).

3.4 Gerador *Moving Peaks*

O Gerador *Moving Peaks* foi sugerido por Branke [9] para gerar problemas dinâmicos contínuos nos quais o espaço de busca é constituído por diversos picos com altura, largura e posição ajustáveis. Um gerador de problemas similar foi proposto por Morrison [11]. No Gerador *Moving Peaks*, m picos são definidos em um espaço n-dimensional, sendo a função de fitness dada por:

$$f(\mathbf{x}, e) = \max\Big(B(\mathbf{x}), \max_{i=1,\ldots,m} P(\mathbf{x}, h_i(e), w_i(e), \mathbf{p}_i(e))\Big), \tag{3.5}$$

sendo $e = \lceil t/\tau \rceil$ o índice do ciclo de mudança, τ o parâmetro que controla a frequência das mudanças, $B(\mathbf{x})$ uma superfície estacionária (base) e $P(\mathbf{x}, h_i(e), w_i(e), \mathbf{p}_i(e))$ uma função definindo a forma do pico i com altura, largura e posição no ciclo de mudança e respectivamente dadas por $h_i(e)$, $w_i(e)$ e $\mathbf{p}_i(e)$. Em geral, os parâmetros que definem os picos são alterados através de uma função linear, com tamanho de passo aleatório para cada mudança.

A Figura 3.1 mostra a modificação de uma superfície (esquerda) formada por três picos através da alteração das alturas dos picos. Observe que, neste caso, a posição do ótimo global pode ser alterada apesar de a posição de cada pico manter-se constante.

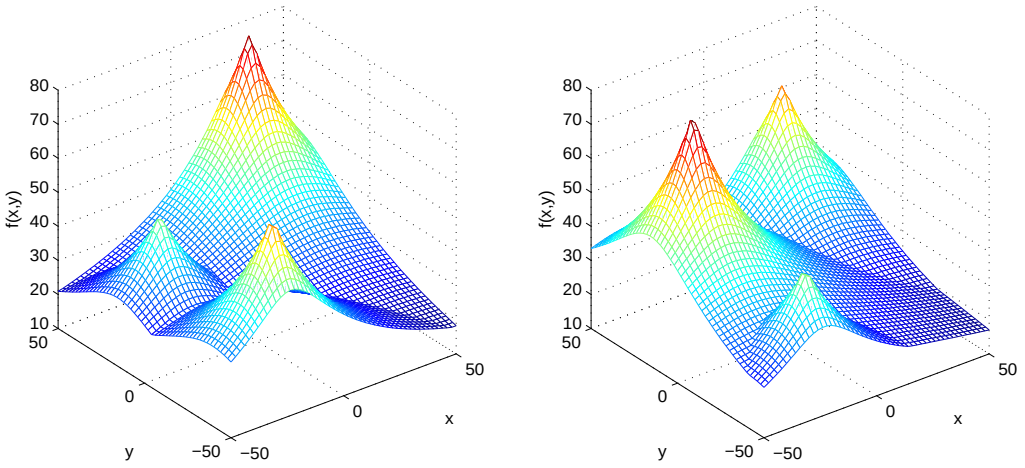

Figura 3.1: Duas superfícies de fitness produzidas pelo Gerador *Moving Peaks*.

3.5 Gerador de Problemas Dinâmicos Contínuos GDBG

Em [25], um gerador de problemas contínuos englobando algumas das idéias utilizadas no Gerador de Problemas Dinâmicos Contínuos Através de Rotação (Capítulo 9) foi proposto. No GDBG, é possível utilizar várias funções contínuas para criar a função de fitness do DOP da seguinte maneira:

$$f(\mathbf{x},e) = \sum_{j=1}^{m} \Big(w_j \cdot (f'_j(\mathbf{M}_j \cdot (\mathbf{x} + \mathbf{p}_j(e) + \mathbf{o}_j)/\lambda_j) + \mathbf{H}_j(e)) \Big), \tag{3.6}$$

sendo m o número de funções utilizadas para compor a função de fitness, e o índice do ciclo de mudança, $\mathbf{x} \in \mathbb{R}^{n(t)}$, f'_j obtida através da j-ésima função utilizada para compor a função de fitness, $\mathbf{M}_j$ uma matriz de rotação, λ_j o fator de normalização das funções, $\mathbf{o}_j$ a posição do ótimo global da função f_j original, e w_j, $\mathbf{H}_j$, $\mathbf{p}_j(e)$, respectivamente, largura, altura, e posição do ótimo global da função f'_j (obtidos de modo similar àquele utilizado pelo Gerador *Moving Peaks*).

Em [25], sugere-se que alguns parâmetros definidos anteriormente sejam mudados regularmente para produzir DOPs. As mudanças podem ser lineares, aleatórias, caóticas, recorrentes e com ruído. Sugere-se também que o número de dimensões do espaço de busca, $n(t)$, possa ser alterado durante o processo de otimização. Este gerador foi utilizado em duas competições envolvendo otimização evolutiva dinâmica que ocorreram no evento *IEEE Congress on Evolutionary Computation* nos anos de 2009 [25] e 2012 [26].

4 Medidas de Desempenho e Teoria

4.1 Medidas de Desempenho em AEs Aplicados à Otimização Dinâmica

Avaliar o desempenho de um AE experimentalmente é fundamental já que, salvo para alguns poucos problemas simples, ferramentas teóricas geralmente não permitem dizer qual entre dois AEs apresentará melhor resultado. Em problemas de otimização estacionários, uma medida de desempenho usualmente empregada é aquela que armazena o melhor valor encontrado entre todas as avaliações da função de aptidão realizadas durante o processo de otimização. Esta medida é claramente imprópria em DOPs, já que mudanças no problema podem acarretar valores diferentes de aptidão para uma mesma solução quando a avaliação é feita em momentos distintos. No entanto, duas outras medidas empregadas em ambientes estacionários podem ser utilizadas em DOPs.

A primeira delas é o Desempenho Online, calculado como a média sobre todas as avaliações de fitness realizadas em uma determinada execução k pelo algoritmo utilizado, i.e.,

$$F_k = \frac{1}{N}\sum_{t=1}^{n_{ger}} \frac{1}{N(t)} \sum_{j=1}^{N(t)} f(\mathbf{x}_j(t), t), \tag{4.1}$$

sendo t o índice da geração, n_{ger} o número máximo de gerações na execução k, j o índice do indivíduo $\mathbf{x}_j(t)$ da população $\mathbf{P}(t)$, e $N(t)$ o tamanho da população $\mathbf{P}(t)$.

Como em geral estamos interessados nos melhores resultados alcançados, o Desempenho Offline, que é feito utilizando-se apenas o melhor indivíduo da população, é mais apropriado. O desempenho Offline na execução k é calculado como:

$$F_k = \frac{1}{n_{ger}} \sum_{t=1}^{n_{ger}} f(\mathbf{x}^*(t), t), \tag{4.2}$$

sendo $\mathbf{x}^*(t)$ o melhor indivíduo da população $\mathbf{P}(t)$. O Desempenho Offline é bastante apropriado em ambientes que se modificam em intervalos bastante curtos. No entanto, o valor de fitness

após uma mudança pode piorar muito e depois melhorar significativamente. Assim, a média dos valores de fitness do melhor indivíduo durante um ciclo de mudança (período entre duas mudanças no qual a função de fitness é estacionária) pode não refletir o real desempenho do algoritmo.

Desta forma, propõe-se que em DOPs que mudam abruptamente [9], o melhor fitness calculado em cada ciclo de mudança seja utilizado para o cálculo da média, i.e.,

$$F_k = \frac{1}{n_e} \sum_{e=1}^{n_e} f(\mathbf{x}^*(e), e), \tag{4.3}$$

sendo n_e o número de ciclos de mudanças e $\mathbf{x}^*(e)$ o melhor indivíduo encontrado no ciclo de mudança e.

4.2 Introdução à Teoria

Assim como em outras áreas da Computação Evolutiva, a grande maioria dos artigos científicos em AEs aplicados em DOPs analisa os algoritmos experimentalmente. Isso ocorre, entre outras, pela falta de ferramentas teóricas que possam ser aplicadas a várias classes de problemas e algoritmos. Em geral, o conhecimento do espaço de fitness é um requisito para uma análise teórica do AE, sendo que, assim, ferramentas teóricas têm sido utilizadas apenas em problemas bem definidos, na maioria das vezes com poucas variáveis de decisão.

Em DOPs, a análise dos aspectos teóricos é ainda mais difícil devido ao fato de mudanças poderem ocorrer na superfície de fitness. Assim, por exemplo, estudos teóricos que verificam a convergência do AE não podem ser aplicados, com exceção em algumas classes de problemas específicas [27]. No entanto, entender o funcionamento do algoritmo do ponto de vista teórico é de extrema importância para, entre outros, comparar algoritmos já existentes e para desenvolver novos algoritmos. Desta forma, um número cada vez maior de pesquisadores investiga os aspectos teóricos dos AEs em DOPs, alguns obtendo grande sucesso em problemas bem definidos. A seguir, alguns destes trabalhos são comentados.

Muitos dos trabalhos teóricos geralmente estendem a análise de AEs desenvolvida para otimização estacionária para DOPs simples. Em [28], os autores analisaram o (1+1)-GA no problema *onemax* dinâmico. Rohlfshagen et al. [29] analisaram como a magnitude e frequência das mudanças afetam o desempenho de um (1+1)-AE aplicado em problemas gerados a partir de funções pseudo-booleanas tornados dinâmicos através de procedimento similar ao adotado no Gerador de DOPs XOR (ver Seção 3.3).

Arnold e Beyer [30] estudaram a capacidade da (μ, λ)-ES, com recombinação e adaptação cumulativa dos parâmetros de mutação, em seguir um simples pico movendo-se no espaço de busca (problema da esfera dinâmico). Em [27], o GA com mutação e seleção proporcional é investigado em um DOP com mudanças regulares através do enfoque por sistemas dinâmicos, ou modelo exato do GA [14]. Apesar de demandar um grande número de equações para descrever todas as possíveis soluções representadas pelos indivíduos do GA, o uso do modelo exato é atrativo pois permite uma descrição completa da dinâmica populacional do GA [31]. No Capítulo 10 desta tese, o modelo exato é utilizado para investigar os efeitos das mudanças produzidas em diversos DOPs.

5 *Algoritmos Evolutivos Para Problemas de Otimização Dinâmica*

Como dito anteriormente, procedimentos utilizados para problemas estacionários são muitas vezes insatisfatórios em DOPs. Uma clara exceção é a dos algoritmos que utilizam o mecanismo de auto-adaptação empregado em ESs e EP, que fornece uma forma natural para lidar com mudanças no problema [32]. No Capítulo 7 desta tese, é apresentada uma estratégia auto-adaptativa para alterar a forma das distribuições de mutação que se enquadra nesta classe de algoritmos.

Desta forma, uma das estratégias utilizadas em DOPs é alterar os procedimentos utilizados em problemas estacionários de forma a adaptá-los às novas condições dinâmicas sem que os algoritmos sejam modificados. Por exemplo, uma solução simples para lidar com uma eventual mudança no problema é iniciar um novo processo de otimização quando alterações no desempenho das melhores soluções ocorrerem. Este procedimento, conhecido como *restart* e utilizado, por exemplo, em [33], pode ser no entanto extremamente custoso por não utilizar o conhecimento já adquirido (ver Seção 1). Entretanto, se o novo ótimo global após a mudança no ambiente não é relacionado às soluções ótimas prévias, o *restart* é a opção mais atraente.

Como alternativa, procura-se desenvolver novos algoritmos baseados nos AEs tradicionais para lidar com a característica dinâmica do ambiente. Vale ressaltar que estas estratégias são interessantes quando a busca pelas novas soluções é, de alguma forma, relacionada à busca já realizada anteriormente. Nos últimos anos, um grande número de novas estratégias baseadas em AEs têm sido desenvolvidas. Esta seção não visa fazer uma revisão detalhada de tais técnicas, sendo que excelentes referências podem ser encontradas na literatura [9], [10], [11], [34], [12], e em repositórios online [35]. Na Parte II desta tese, três diferentes AEs para DOPs são apresentados de forma mais aprofundada.

A seguir, os enfoques mais comuns utilizando AEs tradicionais para lidar com DOPs são classificados, sendo apresentados alguns exemplos de algoritmos de cada classe. Vale ressaltar que diversas outras meta-heurísticas populacionais, como a Otimização por Colônias de Formi-

gas [36], a Otimização por Enxame de Partículas [37], e os Sistemas Imunológicos Artificiais [38], têm sido utilizadas na forma padrão ou modificada em DOPs.

5.1 Geração e Manutenção da Diversidade Durante o Processo Evolutivo

Em AEs, devido aos operadores de cruzamento e seleção, a diversidade da população geralmente diminui ao longo das gerações. Este fenômeno é um sério problema em diversas aplicações, por exemplo, em ambientes multimodais. A perda da diversidade faz com que a população se concentre próxima da melhor solução atual, o que geralmente prejudica o desempenho do processo de otimização quando o ambiente muda em DOPs. Muitas vezes, o ótimo em que a população se encontra antes da mudança se torna um ótimo local que aprisiona a população por várias gerações. Além disso, a diversidade é necessária para que regiões do espaço de busca antes desinteressantes, mas agora promissoras, sejam exploradas.

Assim, uma das mais usuais estratégias em AEs para DOPs é utilizar ferramentas que controlem implícita ou explicitamente a diversidade da população. Em um primeiro enfoque, ações são tomadas de forma a aumentar a diversidade da população após a detecção da mudança no problema. Uma estratégia representativa deste enfoque é a hipermutação [39], na qual a taxa de mutação é aumentada drasticamente por algumas gerações após a detecção de perda de desempenho durante o processo de otimização.

A detecção das mudanças no problema muitas vezes é complicada, principalmente em estratégias não-elitistas, sendo esta uma área que carece de maiores investigações. Alternativamente, procura-se manter a diversidade da população ao longo das gerações, preparando assim a população para eventuais mudanças no problema. Uma estratégia clássica deste enfoque é o uso de imigrantes aleatórios [39], no qual uma parcela fixa da população escolhida aleatoriamente é substituida em cada geração por novos indivíduos criados aleatoriamente (imigrantes). Já no GA termodinâmico [40], modifica-se o mecanismo de seleção dos GAs de forma a escolher bons indivíduos que mais preservam a diversidade da população.

Uma dificuldade destas estratégias é que inserindo muita diversidade na população, o nível de exploração das melhores soluções atuais diminui, o que prejudica a convergência do algoritmo. Desta forma, deve-se buscar o equilíbrio entre exploração de novas regiões e exploração das melhores soluções atuais, o que nem sempre é uma tarefa fácil. Atualmente, diversas estratégias que tentam controlar o nível de diversidade da população de forma automática têm sido desenvolvidas [41], [42]. A estratégia proposta em [41] é apresentada no Capítulo 7.

5.2 Uso de Conhecimento Obtido Durante o Processo Evolutivo

Uma das mais importantes habilidades na adaptação às mudanças é o uso do conhecimento obtido a partir das experiências passadas. Por exemplo, em DOPs nos quais os ótimos retornam repetidamente para as mesmas vizinhanças, o uso do conhecimento adquirido é claramente vantajoso. O conhecimento útil para o novo processo de busca pode estar, por exemplo, na forma de memória de soluções que foram interessantes no passado.

Estratégias baseadas em memória podem ser classificadas como explícitas ou implícitas [34]. No primeiro caso, as soluções são armazenadas explicitamente, sendo que estratégias de armazenamento e recuperação devem ser utilizadas. Em [43], por exemplo, uma estratégia baseada no filtro de Kalman é utilizada para escolher se um novo indivíduo deve ser gerado através de operadores tradicionais ou copiando-se uma solução vista no passado. Já em [44], raciocínio baseado em casos é utilizado para identificar semelhanças entre ambientes em diferentes ciclos de mudança. Desta forma, indivíduos que foram úteis em um determinado ambiente passado são reintroduzidos na população quando a semelhança com o novo ambiente é identificada.

Já nas estratégias que utilizam memória implícita, as informações úteis são armazenadas de forma indireta durante o processo evolutivo. Em geral, o uso da informação passada é feito de modo auto-adaptativo pelo algoritmo. Exemplos marcantes de tais estratégias são o uso de diploidia [45] e poliploidia [46], nas quais partes das soluções passadas ficam armazenadas em cromossomos recessivos que eventualmente podem ficar dominantes. Uma estratégia em que o conhecimento adquirido (sobre o efeito das mudanças do ambiente nos genes dos indivíduos) fica armazenado na forma de probabilidades de mutações dos genes é apresentada nesta tese no Capítulo 6.

5.3 Uso de Múltiplas Populações

Em diversos DOPs, existe a necessidade de exploração de novas regiões do espaço de busca após a mudança no problema. Tal necessidade ocorre especialmente em problemas multimodais nos quais a população pode ficar aprisionada por diversas gerações em um ótimo local. Assim, vários AEs têm sido criados com múltiplas populações, utilizadas para explorar simultaneamente diferentes partes do espaço de busca. Quando mudanças ocorrem no problema, uma população que não vinha apresentando bom desempenho pode eventualmente encontrar soluções promissoras.

Um exemplo de tal enfoque é a estratégia *self-organizing scouts* [9], no qual uma população é dividida sempre que um pico é identificado no espaço de busca. Enquanto que uma pequena população formada por indivíduos da população atual permanece explorando o pico já identificado, uma nova população é gerada espalhando-se soluções pelo espaço de busca com o intuito de explorar novas regiões.

Parte II

Algoritmos

6 *Algoritmo Genético com Taxa de Mutação Dependente do Gene*

Nos seres vivos, as sequências de nucleotídeos que codificam os genes podem ter probabilidades de mutação diferentes. Isto pode ocorrer por diversos motivos, tais como tamanho da sequência de nucleotídeos, tipo da sequência, uso de enzimas diferentes durante o processo de replicação e uso de códons sinônimos [47].

Em [7], Tinós e Carvalho propõem uma nova solução para tratar o problema de GAs aplicados em DOPs: alterar a robustez de cada gene para permitir que a função por ele codificada seja mais ou menos suscetível a mudanças. Para isso, cada gene é associado com taxas de mutação diferentes, similarmente ao que ocorre em outros AEs como Estratégias Evolutivas e Programação Evolutiva [5]. No entanto, diferentemente das soluções adotadas nestes AEs, o conhecimento adquirido durante os períodos em que ocorrem as mudanças nos DOPs é utilizado para alterar tais taxas de mutação. Assim, se a variação de um conjunto de genes propicia uma melhor adaptação às mudanças do problema, as probabilidades de mutação deste conjunto aumentam em relação às probabilidades dos outros genes.

Desta forma, a busca no espaço das soluções concentra-se mais nas regiões correspondentes a variações do conjunto de genes com maiores probabilidades de mutação e, portanto, a velocidade de adaptação frente às alterações aumenta. Salienta-se, entretanto, que a busca não se restringe a tais regiões já que as probabilidades de mutação dos genes são sempre maiores que zero. Este AE, conhecido como GA com taxa de mutação dependente do gene (*GA with gene dependent mutation probabilities* - GAGDM), é aqui apresentado.

6.1 Algoritmo

No GA padrão, todos os genes (aqui chamamos de gene cada elemento do cromossomo) do indivíduo têm a mesma taxa de mutação p_m. No GAGDM, a probabilidade de mutação de cada gene é independente e suscetível a mudanças. Cada gene g ($g = 1, \ldots, l$) de cada indivíduo i

$(i = 1, \ldots, N)$ é associado com uma variável υ_{i_g}, o qual controla a probabilidade de mutação do gene. A probabilidade de mutação do g-ésimo gene do i-ésimo indivíduo é dada por:

$$\gamma_{i_g} = N p_m \frac{\upsilon_{i_g}}{\sum_{j=1}^{l} \upsilon_{i_j}} \tag{6.1}$$

na qual $\left\{\upsilon_{i_g} \in \mathbb{R} \mid 0 < \upsilon_{min} \leq \upsilon_{i_g} \leq \upsilon_{max}\right\}$, sendo υ_{min} e υ_{max} constantes reais. Se υ_{i_g} é igual para todos os genes do indivíduo, então $\gamma_{i_g} = p_m$ para $g = 1, \ldots, l$ e o GAGDM reproduz o GA padrão.

Inicialmente, todas variáveis υ_{i_g} são iguais a υ_{min}, ou seja, todos os genes têm a mesma probabilidade de serem escolhidos. Na natureza, e nos esquemas auto-adaptativos em ESs, a variação da estabilidade do gene é alcançada através da seleção natural. Nos casos em que se conhece quais genes devem ter probabilidades de mutação maiores, as variáveis υ_{i_g} podem ser estabelecidas *a priori*. No GAGDM, o conhecimento adquirido através do processo de evolução é utilizado para alterar as variáveis υ_{i_g}. A seguir, o processo de atualização da variável υ_{i_g} em uma época é apresentado.

Caso crossover seja empregado, os descendentes gerados por *crossover* herdam as variáveis υ_{i_g} do pai. Já nos que foram gerados por mutação, três casos podem ser considerados. Quando ocorrer uma mutação que tenha como consequência um descendente com $fitness$ maior do que aquele do indivíduo mutado, a varíavel υ_{i_g}, sendo g o gene escolhido para mutação, será aumentada tanto no pai quanto no descendente. Vale salientar que as varíaveis de controle da probabilidade de mutação do outros genes que não foram mutados permanecem inalteradas. Assim, as sucessivas mutações através do ciclos de mudanças levam a um aumento da probabilidade de mutação daqueles genes que devem mudar quando a função de $fitness$ se altera.

Quando o $fitness$ é menor no descendente, a variável υ_{i_g}, sendo g o gene escolhido para mutação, do pai é diminuída enquanto que a do filho é aumentada. Assim, a mutação daquele gene no filho é estimulada enquanto que a do pai é desestimulada. Quando o $fitness$ é igual no descendente e no pai, a variável υ_{i_g} não é alterada. Ou seja, quando o indivíduo i sofre mutação, a variação em υ_{i_g} para o gene mutado g do indivíduo i (pai) é dada por:

$$\Delta\upsilon_{i_g} = \begin{cases} +\beta\alpha & \text{se } f(\mathbf{x}_f) > f(\mathbf{x}_i) \\ -\alpha & \text{se } f(\mathbf{x}_f) < f(\mathbf{x}_i) \\ 0 & \text{caso contrário} \end{cases} \tag{6.2}$$

na qual o índice f sinaliza o descendente e os parâmetros β e α controlam o tamanho da varia-

ção. Já a variação em υ_{f_g} para o gene g do indivíduo descendente é dada por:

$$\Delta\upsilon_{f_g} = \begin{cases} +\beta\alpha & \text{se } f(\mathbf{x}_f) > f(\mathbf{x}_i) \\ +\alpha & \text{se } f(\mathbf{x}_f) < f(\mathbf{x}_i) \\ 0 & \text{caso contrário} \end{cases} \tag{6.3}$$

O GAGDM baseado no GA geracional é apresentado no Algoritmo 3. Comparado com o GA padrão, o GAGDM necessita de mais memória para armazenar o vetor l-dimensional de parâmetros υ_{i_g} de cada indivíduo da população e mais esforço computacional para atualizar tais parâmetros em cada geração.

Algoritmo 3 GAGDM

1: inicializePopulacao($\mathbf{P}(t)$)
2: faça $\upsilon_{i_g} = \upsilon_{min}$ para os genes $g = 1,\ldots,l$ e indivíduos $i = 1,\ldots,N$ de $\mathbf{P}(t)$
3: avaliePopulacao($\mathbf{P}(t)$)
4: **enquanto** (critério de parada não é satisfeito) **faça**
5: **enquanto** ($\mathbf{P}(t+1)$ não está completa) **faça**
6: selecione dois indivíduos de $\mathbf{P}(t)$ de acordo com o critério de seleção
7: aplique crossover com taxa p_c
8: copie υ_{i_g} do respectivo pai
9: aplique mutação com probabilidade γ_{i_g} (Eq. 6.1) em cada gene $g = 1,\ldots,l$ de cada descendente
10: avalie o fitness dos descendentes
11: compute os novos valores de υ_{i_g} para cada descendente e cada pai (equações 6.2-6.3)
12: **fim enquanto**
13: $t \leftarrow t+1$
14: **fim enquanto**

É interessante notar que, inicialmente, o GAGDM demorará mais para convergir do que o GA padrão. Isso ocorre porque a busca através do operador de mutação será mais concentrada naqueles indivíduos que já sofreram mutação (e, muitas vezes, convergiram para o valor desejado). Para evitar este efeito, propõe-se que o parâmetro β seja iniciado com valor zero e convirja para um valor β_{final} através dos ciclos de mudanças. Assim, as probabilidades de mutação são iguais no ciclo de mudança inicial, i.e., antes da primeira mudança, e se modificam gradualmente a partir daí. Aqui, a seguinte função para ajuste de β é proposta:

$$\beta = \beta_v(1 - \exp^{\hat{e}/\sigma}) \tag{6.4}$$

na qual $\hat{e} = 0, 1, \ldots$ é uma estimativa do índice do ciclo de mudança e σ controla a variação de β. O índice do ciclo de mudança (e) não está diretamente disponível para o algoritmo. No entanto, ele pode ser estimado monitorando a mudança no fitness dos melhores indivíduos da população.

Os parâmetros α e β devem ser escolhidos para que a variação de υ_{i_g} seja positiva dentro de um ciclo de mudança (ou seja, que o acréscimo promovido pela mutação benéfica seja maior que o decréscimo produzido por mutações deletérias). Os parâmetros α e β também devem ser escolhidos de modo a garantir que a probabilidade dos genes que geralmente não devem ser mutados não seja drasticamente baixa. Assim, garante-se que a busca será realizada em todo o espaço das soluções. A seguir, um método para determinação dos parâmetros α e β é apresentado.

6.2 Determinação dos Parâmetros

Através da Eq. 6.1, é possível calcular as probabilidades máxima e mínima do gene g de um indivíduo i ser escolhido para mutação entre os l genes deste indivíduo em uma época qualquer. A probabilidade mínima ocorre quando $\upsilon_{i_g} = \upsilon_{min}$ e $\upsilon_{i_j} = \upsilon_{max}$ para todo $j = 1, \ldots, l; j \neq g$. Ou seja, usando a Eq. 6.1, a probabilidade mínima do gene g de um indivíduo i ser escolhido para mutação é dada por

$$\gamma_{min} = l p_m \frac{\upsilon_{min}}{\upsilon_{min} + (l-1)\upsilon_{max}} \tag{6.5}$$

Utilizando a Eq. 6.5, υ_{max} pode ser computado como uma função de γ_{min} e υ_{min}.

A probabilidade máxima ocorre quando $\upsilon_{i_g} = \upsilon_{max}$ e $\upsilon_{i_j} = \upsilon_{min}$ para todo $j = 1, \ldots, l; j \neq g$. Ou seja,

$$\gamma_{max} = l p_m \frac{\upsilon_{max}}{\upsilon_{max} + (l-1)\upsilon_{min}} \quad . \tag{6.6}$$

que resulta em $\gamma_{max} > p_m$. Deste modo, as equações 6.5 e 6.6 podem ser empregadas para definir os parâmetros υ_{max} e γ_{max} dados os parâmetros υ_{min} e γ_{min}.

No GAGDM, os parâmetros α e β_v são escolhidos de modo que a média de $\Delta\upsilon_{i_g}$ seja igual ou maior que 0 quando pelo menos uma mutação benéfica ocorre no gene i_g ao longo de um ciclo de mudança. Quando somente uma mutação benéfica ocorre no gene i_g em um ciclo de mudança $e \gg 0$, então, pela Eq. 6.2, a soma dos ajustes causados pelas mutações benéficas é dada por:

$$\Delta\upsilon_{i_g+}(e) = +\beta_v\alpha 1 = +\beta_v\alpha \tag{6.7}$$

Quando mutações deletérias ocorrem no gene i_g com máxima probabilidade por $(d(e)-1)$ gerações (o número de gerações no ciclo de mudança e menos 1), o número esperado de ajustes causados pelas mutações deletérias (Eq. 6.2) é dado por:

$$\Delta\upsilon_{i_g-}(e) = -\alpha\gamma_{max}(\hat{d}(e)-1) \tag{6.8}$$

sendo $\hat{d}(e)$ uma estimativa do número de gerações no ciclo de mudança e, o qual pode ser estimado do mesmo modo que $\hat{e}$. Como a média da soma de ajustes deve ser igual ou maior que zero durante um ciclo de mudanças, então, pelas equações 6.7 e 6.8, a seguinte equação pode ser obtida:

$$\beta_v \geq \gamma_{max}(\hat{d}(e) - 1) \tag{6.9}$$

O parâmetro α é escolhido de modo a fazer $\upsilon_{i_g} = \upsilon_{min}$ depois de k_p ciclos de mudança quando o gene i_g não experimenta nenhuma mutação benéfica e a probabilidade de mutação é máxima. O número esperado da soma da variações negativas (Eq. 6.8) em k_p ciclos é dado por:

$$\Delta\upsilon_{i_g k_{p-}} = -k_p \gamma_{max} \hat{d}(e)\alpha = \upsilon_{i_g} - \upsilon_{min} \tag{6.10}$$

e, então, α pode ser calculado, assumindo $\upsilon_{i_g} = \upsilon_{max}$, por:

$$\alpha = \frac{\upsilon_{max} - \upsilon_{min}}{k_p \gamma_{max} \hat{d}(e)} \tag{6.11}$$

6.3 Análise

Quando aplicar o GAGDM é uma importante questão que é discutida nesta seção. A análise da modificação do número de templates representando possíveis soluções ao longo do processo evolutivo pode fornecer importantes informações sobre o processo de otimização realizado pelo GA. Estes templates são conhecidos como esquemas, tendo sido utilizados por Holland [48], [17] numa tentativa de explicar o funcionamento do GA. O número esperado de cópias que um esquema H recebe na próxima geração $t+1$ em um GA padrão com seleção proporcional, crossover de um ponto e mutação é dado por:

$$m(H,t+1) \geq m(H,t)\frac{f(H,t)}{\bar{f}(t)}\left(1 - p_c\frac{\delta(H)}{l-1}\right)(1-p_m)^{o(H)} \tag{6.12}$$

na qual $f(H,t)$ é o fitness médio dos indivíduos representando o esquema H na geração t, $\bar{f}(t)$ é o fitness médio da população na geração t, $\delta(H)$ é o comprimento definidor do esquema H (distância entre o primeiro e o último bit fixo do esquema), e $o(H)$ é a ordem do esquema (número de bits fixos do esquema).

Se agora consideramos que os genes têm probabilidades de mutação diferentes (GAGDM), o número esperado de cópias do esquema H fica:

$$m(H,t+1) \geq m(H,t)\frac{f(H,t)}{\bar{f}(t)}\left(1 - p_c\frac{\delta(H)}{l-1}\right)\left(\prod_{j=1}^{o(H)}(1-\gamma_{H_j}(t))\right) \tag{6.13}$$

sendo $\gamma_{H_j}(t)$ a média das probabilidades de mutação dos genes nas posições fixas j dos indivíduos representando o esquema H na geração t.

Considere agora uma classe de DOPs nos quais, para se adaptar, o GA precisa alterar apenas alguns genes da solução corrente depois da mudança no problema. Neste caso, podemos considerar que os genes que devem ficar fixos após a mudança no problema formam um esquema H. Quando este esquema H está presente na população, genes de H que não estão nas posições fixas devem ser modificados mais frequentemente que os genes que estão nas posições fixas após as mudanças no problema. Se as mutações dos genes que não estão nas posições fixas aumentam o fitness dos indivíduos que representam o esquema H, as probabilidades de mutação destes genes devem ser maiores que aquelas dos genes nas posições fixas de H conforme o número de ciclos de mudança aumenta.

Deste forma, as probabilidades de mutação dos genes na posições fixas de H serão menores que as probabilidades de mutação do GA padrão em um ciclo $e \gg 0$, se a mesma taxa p_m é utilizada. Assim, a seguinte equação pode ser escrita para $e \gg 0$:

$$\prod_{j=1}^{o(H)} (1-\gamma_{H_j}) > (1-p_m)^{o(H)} \quad . \tag{6.14}$$

Como resultado das equações 6.12-6.14, o número esperado de cópias que o esquema H recebe na próxima geração $t+1$ para o GAGDM é maior ou igual que aquele para o GA padrão para mesma taxa de mutação p_m. Se o esquema H recebe mais cópias, o GAGDM pode consequentemente atingir o ótimo mais rapidamente, se este é claro baseado no esquema H. Contudo, este fato não é suficiente para que o GAGDM possa achar o ótimo mais rapidamente que o GA padrão para esta classe de DOPs já que existe um limite para o número esperado de cópias que o esquema H pode receber na população.

Vamos agora investigar a probabilidade de mudar apenas um gene que não está entre as posições fixas de H. Para a classe de problemas estudado, mudar estes genes é importante para se achar o ótimo baseado no esquema H. Queremos aqui verificar se a probabilidade de mudar somente um gene que não está em uma posição fixa de H é maior para o GAGDM.

Para o GAGDM, a seguinte equação pode ser escrita para a probabilidade γ_m de mudar somente um gene g que não está nas posições fixas do indivíduo i representando o esquema H:

$$\gamma_m = \gamma_{i_g} \prod_{j=1, j\neq g}^{l} (1-\gamma_{i_j}) > \gamma_{i_g}(1-\gamma_{max})^{l-1} \quad . \tag{6.15}$$

Para o GA padrão, a probabilidade de mudar somente um gene é dada por $\gamma_m = p_m(1-$

$p_m)^{l-1}$. Como $\gamma_{max} > p_m$ (Seção 6.2), a probabilidade de mutação γ_{i_g} para o caso em que a probabilidade de mudar somente um gene g é maior para o GAGDM pode ser calculada por:

$$\gamma_{i_g} > \frac{p_m(1-p_m)^{l-1}}{(1-\gamma_{max})^{l-1}} \quad . \tag{6.16}$$

Para valores de probabilidade de mutação γ_{i_g} dados pela Eq. 6.16 para os genes que não estão nas posições fixas de H, o seguinte pode ser afirmado:

- as probabilidades de mudar somente um dos genes que não estão nas posições fixas de H através de mutação são maiores para o GAGDM, e;
- as probabilidades de mudar os genes que estão nas posições fixas de H através de mutação são menores para o GAGDM.

Na Eq. 6.16, os valores de γ_{i_g} aumentam com γ_{max}. Para valores maiores de γ_{max}, maiores valores de γ_{i_g} são necessários. Contudo, existe um limite para o valores de γ_{i_g}, o qual é diretamente relacionado com a ordem do esquema H. Para valores menores de $o(H)$, a probabilidade de que dois ou mais genes sejam mutados em uma geração é maior, o que pode tornar mais difícil a convergência do algoritmo. Isso ocorre porque as probabilidades de mutação para todos os genes que não estão nas posições fixas de H são altas. Na sequência, a ordem $o_*(H)$ do esquema H nos problemas no qual o GAGDM pode ser interessante é investigada.

Considere que $\upsilon_{i_g} = \upsilon_{min}$ para os $o_*(H)$ genes que não estão nas posições fixas de H para um indivíduo i representando o esquema H, e que $\upsilon_{i_g} = \upsilon_{max}$ para os $(l - o_*(H))$ genes restantes para $e \gg 0$. Então, pela Eq. 6.1, a probabilidade de mutação para o gene que não está na posição fixa de H é dada por:

$$\gamma_* = l p_m \frac{\upsilon_{max}}{o_*(H)\upsilon_{min} + (l - o_*(H))\upsilon_{max}} \quad . \tag{6.17}$$

Através da Eq. 6.17, a ordem $o_*(H)$ a qual pode tornar o GAGDM interessante pode ser calculada como:

$$o_*(H) \geq \frac{l\upsilon_{max}(\gamma_* - p_m)}{\gamma_*(\upsilon_{max} - \upsilon_{min})} \tag{6.18}$$

na qual, pela Eq. 6.16,

$$\gamma_* = \frac{p_m(1-p_m)^{l-1}}{(1-\gamma_{max})^{l-1}}$$

A ordem de H restringe a classe de DOPs nos quais o GAGDM pode ser interessante. Para DOPs nos quais as soluções podem ser representadadas pelo mesmo esquema H depois das mudanças no problema (i.e., alguns genes da solução corrente devem ser modificados enquanto outros devem permanecer fixos após as mudanças no problema), a classe de DOPs nos quais

a probabilidades de mudar somente um dos genes que não estão nas posições fixas de H são maiores para o GAGDM pode ser definida pela ordem mínima de H dada pela Eq. 6.18. O desempenho do GAGDM pode ser ruim nos casos em que muitos genes (i.e., esquemas com ordem que não é restrita pela Eq. 6.18) devem ser modificados. Nestes casos, a probabilidade de mudar somente um gene torna-se menor.

O GAGDM é mais interessante que o GA padrão para a classe de DOPs nos quais somente alguns genes devem ser modificados depois da mudança no problema (i.e., esquemas cuja ordem não é restrita pela Eq. 6.18). Nese caso, a busca no espaço de soluções pelo operador de mutação é concentrada em regiões associadas com os genes que não estão nas posições fixas de H. Como resultado, a velocidade de adaptação esperada é, em geral, maior para o GAGDM para a classe de DOPs considerada. A busca, contudo, não é restrita somente a estas regiões já que todas as probabilidades de mutação são maiores que zero.

6.4 Resultados

Nesta seção são apresentados os resultados de dois experimentos simples que ajudam a entender o funcionamento do GAGDM. Estes experimentos foram apresentados em [49], sendo que experimentos com DOPs mais complexos e comparações com outros algoritmos são apresentadas em [50] (que apresenta resultados para DOPs multimodais) e [7] (que apresenta resultados para DOPs envolvendo o aprendizado de Redes Neurais Artificiais). Em todos os experimentos apresentados nesta seção, a população é inicialmente aleatória. Os operadores de crossover de um ponto, mutação e seleção por elitismo e proporcional são empregados. Os parâmetros do GAGDM são determinados através do método apresentado na Seção 6.2, sendo que seus valores e aqueles do GA padrão, podem ser encontrados em [50]. Cada GAGDM e GA padrão foi executado 10 vezes. A seguir, os resultados médios obtidos para cada exemplo são apresentados.

6.4.1 Seguimento de Padrões Binários

Este problema, utilizado em [51] entre outros, não representa nenhum DOP particular e é usado aqui porque a análise de seus resultados é fácil e direta. O GA deve procurar por uma cadeia de bits igual a um vetor desejado de bits aleatoriamente gerados no início de cada execução. A função de $fitness$ é dada pela diferença entre o $fitness$ máximo e a distância Hamming entre o indivíduo e o vetor desejado.

O DOP é caracterizado pela mudança de 5 componentes (correspondentes aos genes de 11 a 15) do vetor desejado no início de cada ciclo de mudanças, sendo que cada ciclo é composto por $d_e = 200$ gerações. Os resultados obtidos são apresentados nas figuras 6.1 e 6.2. Observe que, após poucos ciclos de mudança, o GAGDM alcança a solução mais rapidamente que o GA padrão (Figura 6.1). Observe também as probabilidades de mutação (γ_{i_g}) dos genes do melhor indivíduo correspondentes aos padrões que se alteram aumentam com o número de ciclos de mudança (Figura 6.2), o que explica os melhores resultados alcançados pelo GAGDM.

6.4.2 Otimização de Função Unimodal

Neste problema, o GA deve procurar pelo mínimo global de uma função unimodal. A função de *fitness* é dada por

$$f(\mathbf{x}_i) = \begin{cases} \frac{1}{\|\mathbf{x}_i - \mathbf{d}\|^2} & \text{se } \mathbf{x}_i \neq \mathbf{d} \\ 1,1 & \text{caso contrário} \end{cases} \tag{6.19}$$

na qual o máximo da função ocorre quando $\mathbf{x}_i = \mathbf{d}$ e as componentes do vetor desejado $\mathbf{d}$ são números inteiros aleatoriamente gerados entre x_{min} e x_{max} no início de cada execução. No início de cada ciclo de mudança (na qual $d_e = 2000$ gerações), 5 componentes (correspondentes aos genes de 11 a 15) de $\mathbf{d}$ são substituídos por valores aleatoriamente gerados.

Os resultados obtidos são apresentados nas figuras 6.3 e 6.4. Novamente, observe que, após poucos ciclos de mudança, o GAGDM alcança a solução mais rapidamente que o GA padrão (Figura 6.3). Isso ocorre devido ao maiores valores de probabilidade de mutação dos genes 11 a 15 (correspondentes aos componentes alterados na função dada pela Eq. 6.19) alcançados, o que pode ser verificado na Figura 6.4.

6.5 Comentários Finais

No GAGDM, o conhecimento obtido durante o processo evolutivo é utilizado para modificar taxas de probabilidade associadas com cada gene do indivíduo. O conhecimento obtido durante o processo evolutivo é utilizado para mudar tais taxas de mutação, o que caracteriza a técnica como da classe que usa o conhecimento obtido durante o processo evolutivo (Seção 5.2).

Se a modificação de um conjunto de genes melhora a adaptação frente às mudanças do ambiente, as taxas de mutação destes genes são aumentadas. É importante observar que o

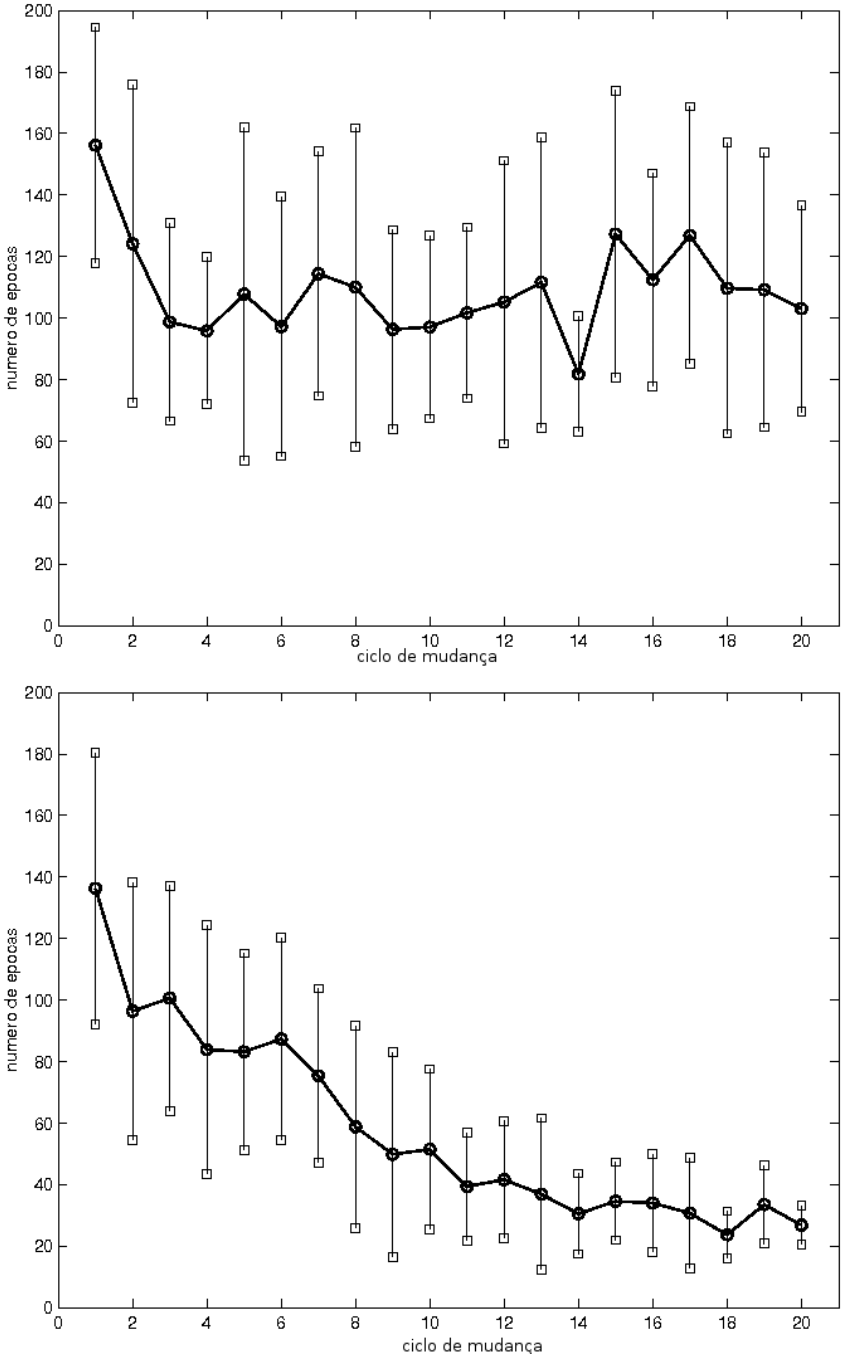

Figura 6.1: Número médio de épocas após o início do ciclo de mudança em que o melhor indivíduo alcança o *fitness* máximo e respectivo desvio padrão para o exemplo do seguimento de padrões binários. Acima: GA padrão. Abaixo: GAGDM.

GAGDM geralmente melhora a adapatação, ao longo das gerações, frente a tais mudanças.

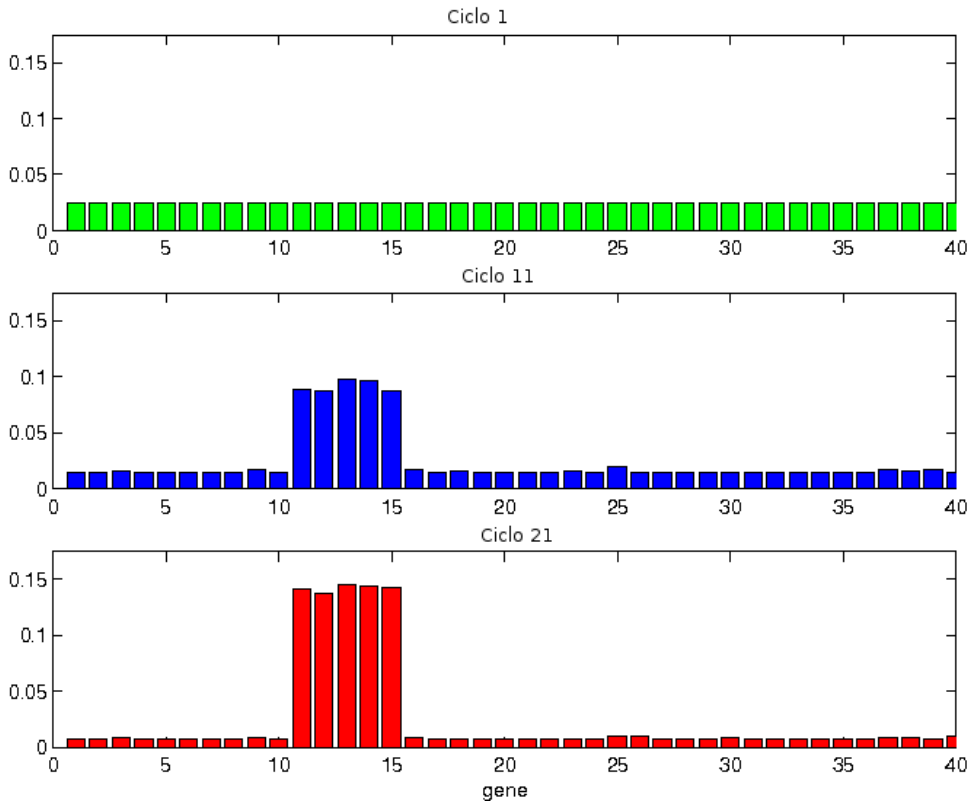

Figura 6.2: Valores médios das probabilidades de mutação dos genes do melhor indivíduo no fim de três ciclos de mudança distintos no exemplo do seguimento de padrões binários.

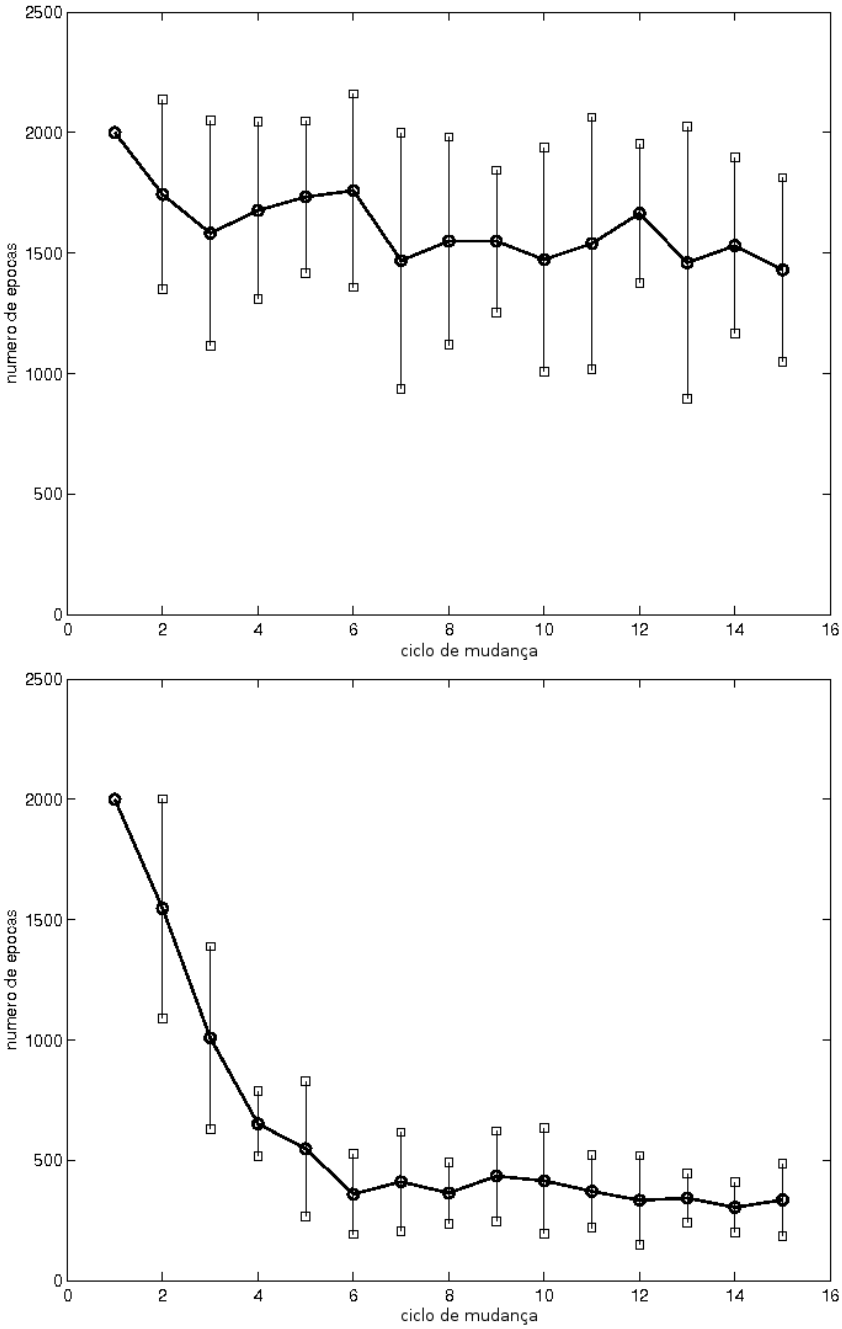

Figura 6.3: Número médio de épocas após o início do ciclo de mudança em que o melhor indivíduo alcança o *fitness* máximo e respectivo desvio padrão para o exemplo da otimização de função unimodal. Acima: GA padrão. Abaixo: GAGDM.

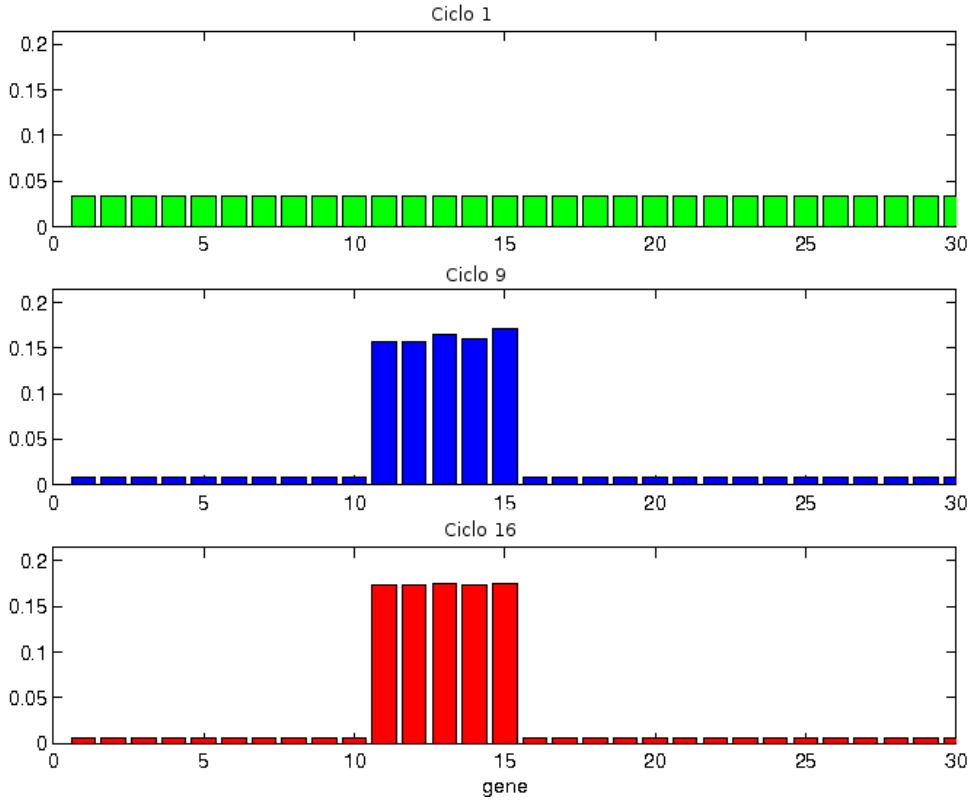

Figura 6.4: Valores médios das probabilidades de mutação dos genes do melhor indivíduo no fim de três ciclos de mudança distintos no exemplo da otimização de função unimodal.

7 *Algoritmo Genético com Imigrantes Aleatórios Auto-Organizado*

O GA com imigrantes aleatórios, discutido na Seção 5.1 e que é inspirado no fluxo de indivíduos que entram e saem de uma população entre duas gerações na Natureza [39], é bastante simples e interessante para DOPs. Em cada geração do processo de otimização, alguns indivíduos da população corrente são substituídos por indivíduos criados aleatoriamente. Uma estratégia de substituição, como, por exemplo, substituir indivíduos escolhidos aleatoriamente ou indivíduos com os menores valores de $fitness$, define quais indivíduos são substituídos. Através da introdução de novos indivíduos, o método dos imigrantes aleatórios tenta manter o nível de diversidade da população. A manutenção da diversidade da população pode ser bastante útil em problemas sujeitos a mudanças (ver Seção 5.1) para evitar que a população fique presa em ótimos locais representados pelas soluções adotadas antes da mudança no problema.

Contudo, em alguns casos, como quando o número de genes em um indivíduo é alto e/ou quando os ótimos locais em que a população encontra-se têm valores de $fitness$ muito maiores que os valores médios de todos os indivíduos possíveis do espaço de busca, a probabilidade que novos indivíduos sejam preservados é geralmente muito pequena. Isso ocorre porque o GA preserva, direta ou indiretamente, os melhores indivíduos da população corrente, e a probabilidade de que novos indivíduos tenham valores de $fitness$ próximos ou maiores que os valores dos indivíduos atuais é geralmente baixa. Neste capítulo, o GA com imigrantes aleatórios auto-organizado (*self-organizing random imigrants GA* - SORIGA), proposto por Tinós e Yang [41] para evitar tais problemas do GA com imigrantes aletórios, é apresentado. Antes de apresentarmos o algoritmo, o conceito de criticalidade auto-organizada é apresentado.

7.1 Criticalidade Auto-Organizada

Em [52], o autor sugere que sistemas constituídos de muitos componentes interagindo entre si podem apresentar um interessante tipo de comportamento auto-organizável [53]. Tal compor-

tamento é chamado em [52] de criticalidade auto-organizada (*self-organized criticality* - SOC), estando presente em diversos fenômenos, como pilhas de areia, terremotos, panes em grandes sistemas elétricos e incêndios em florestas. A característica mais fascinante em sistemas que exibem SOC é que eles se auto-organizam em um determinado estado crítico sem a necessidade de qualquer ação de controle externo. Tal estado crítico é caracterizado pela resposta do sistema a perturbações externas. Em um sistema que exibe comportamento não-crítico, a distribuição das respostas do sistema às perturbações em diferentes posições e instantes é estreita e bem descrita por um valor médio. Em um sistema que exibe comportamento crítico, não existe uma única resposta característica. Em outras palavras, o sistema exibe invariância de escala, sendo que uma pequena perturbação em um único componente pode gerar tanto um efeito pequeno apenas nos componentes da sua vizinhança, quanto uma reação em cadeia que afeta todos os componentes constituintes do sistema.

As distribuições estatísticas que descrevem a resposta de um sistema que exibe SOC são dadas por leis de potência nas formas:

$$P(s) \sim s^{-\tau} \tag{7.1}$$

e

$$P(d) \sim d^{-\alpha} \tag{7.2}$$

sendo s o número de componentes do sistema afetados pela perturbação, d a duração da reação em cadeia gerada pela perturbação, e τ e α constantes reais.

Como exemplo, considere o modelo da pilha de areia descrito em [52], no qual um grão de areia é adicionado em um posição aleatória em cada intervalo de tempo Δt. Com o intuito de caracterizar a resposta do sistema, pode-se aferir o número de grãos de areia envolvidos em uma avalanche induzida pela adição de um único grão (s), e a duração de cada avalanche (d). No estado crítico, as distribuições estatísticas descrevendo a resposta do modelo da pilha de areia para a adição de um único grão são dadas pelas leis de potência apresentadas nas equações 7.1 e 7.2, sendo que a adição de um grão em uma dada posição pode provocar desde um pequeno deslizamento que afeta apenas a sua vizinhança até um grande deslizamento que afeta toda a pilha de areia.

Pesquisadores sugeriram ainda que a evolução natural pode, também, apresentar SOC [52]. Uma evidência de tal afirmação seria o fato de que a evolução ocorre através de períodos curtos de atividade intensa intercalados por períodos calmos, ao invés de continuamente em passos constantes e lentos. Existem muito mais extinções pequenas (ou seja, que afetam poucas espécies) do que extinções grandes, como a grande extinção do Cretáceo que eliminou os di-

nossauros. Além disso, extinções ocorrem em uma grande variedade de escalas de tamanhos [54]. Estes fatos sugerem que as extinções propagam-se através dos ecossistemas, tal qual avalanches em uma pilha de areia, e perturbações de um mesmo tamanho podem desencadear extinções com uma grande variedade de tamanhos. Segundo alguns pesquisadores, isto ocorre porque as espécies co-evoluem em direção a um estado crítico [55].

Bak e Sneppen [52] propuseram um modelo de simulação muito simples para estudar a presença de SOC na evolução. Na versão unidimensional do modelo, os indivíduos (ou espécies na terminologia dos autores) são dispostos em um círculo, sendo que um valor aleatório de $fitness$ é dado a cada um deles. Em cada geração da simulação, o indivíduo com o valor de $fitness$ mais baixo e os indivíduos vizinhos mais próximos a sua direita e a sua esquerda têm seus valores de $fitness$ substituídos por novos valores aleatórios. Uma analogia desta conexão entre vizinhos neste modelo seria a interação entre as espécies na natureza. Se, por exemplo, uma espécie de predador tem seu $fitness$ alterado, o valores de $fitness$ das espécies que são suas presas irão mudar também. O modelo de evolução de Bak-Sneppen pode ser resumido pelo Algoritmo 4.

Algoritmo 4 Modelo de evolução de Bak e Sneppen

1: Ache o índice j do indivíduo com o menor valor de $fitness$
2: Substitua o $fitness$ dos indivíduos com índices j, $j-1$, e $j+1$ por valores aleatórios gerados com distribuição uniforme

Este modelo simples pode levar a um comportamento interessante. No começo da simulação, o $fitness$ médio da população é pequeno, mas, conforme o número de gerações aumenta, o $fitness$ médio aumenta também. Eventualmente, o $fitness$ médio para de crescer, indicando que o estado crítico foi alcançado. No estado crítico, os valores de $fitness$ dos vizinhos do pior indivíduo são frequentemente substituídos por valores menores. Então, o pior indivíduo na geração seguinte pode ser um destes vizinhos, o qual é substituído com seus próximos vizinhos, caracterizando uma reação em cadeia, aqui chamada de extinção, que pode afetar todos os indivíduos da população. As extinções exibem invariância de escala e suas distribuições estatísticas são dadas por leis de potência nas formas apresentadas nas equações 7.1 e 7.2.

É interessante notar que as grandes extinções geralmente ocorrem neste modelo quando grande parte dos indivíduos da população tem valores de $fitness$ altos e similares. Desta forma, SOC evita que as espécies fiquem presas em ótimos locais presentes na superfície de $fitness$ (espaço de soluções). A idéia é extremamente interessante e relativamente simples, e logo pesquisadores propuseram seu uso em processos de otimização [56], [57]. Em GAs, Krink e Thomsen [58] propuseram o uso do modelo de pilhas de areia discutido anteriormente para gerar leis de potência utilizadas para controlar o tamanho das zonas espaciais de extinção em um

modelo de difusão. Quando um indivíduo é extinto, uma versão mutada do melhor indivíduo da população é criada em seu lugar. É importante salientar que, no algoritmo proposto em [58], SOC aparece no modelo de pilhas de areia, e não como um resultado da auto-organização dos componentes do sistema (indivíduos do GA).

7.2 Algoritmo

No SORIGA, a substituição do indivíduo com o menor valor de $fitness$ (pior indivíduo) da população corrente e de seus vizinhos próximos, como no modelo de evolução de Bak e Sneppen, por novos indivíduos aleatórios em GAs com imigrantes aleatórios é investigada. Os vizinhos do pior indivíduo são determinados através do índice de cada indivíduo. Espera-se que, agindo desta forma, o sistema possa exibir um comportamento auto-organizável, de modo que o nível de diversidade da população seja controlado sem que precise ser explicitamente calculado, permitindo que os indivíduos escapem dos ótimos locais presentes na superfície de $fitness$ induzidos pelas mudanças inerentes ao problema não-estacionário.

Contudo, este procedimento simples não garante que o sistema exiba SOC, já que os novos indivíduos são geralmente substituídos por indivíduos melhores presentes na população. Propõe-se, então, a adoção de um segundo procedimento, no qual os novos indivíduos criados em uma extinção sejam confinados em uma subpopulação. O tamanho desta subpopulação não é definido pelo programador, mas sim pelos indivíduos aleatórios e seus descendentes. Os indivíduos que não pertencem a esta subpopulação são proibidos de substituir os indivíduos que estão presentes nela.

Existem duas modificações principais em relação ao GA padrão no algoritmo investigado. Na primeira modificação, a qual é apresentada no Algoritmo 5, o tamanho corrente de cada extinção, denotado por d, é computado. Os índices mínimo e máximo dos indivíduos substituídos ($i_{min} - 1$ e $i_{max} + 1$) são utilizados para calcular o número de indivíduos afetados pela extinção corrente, ou seja, presentes na subpopulação. Quando uma reação em cadeia é interrompida, isto é, o indivíduo com o valor de $fitness$ mais baixo não está dentro da subpopulação ou não é vizinho dos mínimos e máximos índices dos indivíduos que a definem, o tamanho da extinção assume o valor 1.

A segunda modificação, a qual é apresentada no Algoritmo 6, indica o mecanismo de seleção para cada indivíduo da nova população. Dois casos podem ser considerados na seleção. No primeiro, quando o índice do indivíduo é diferente dos índices dos indivíduos presentes na subpopulação, o indivíduo é selecionado de acordo com o método de seleção escolhido pelo

Algoritmo 5 Substituição dos indivíduos no SORIGA

1: Ache o índice j do indivíduo com o menor $fitness$ da população corrente
2: Substitua os indivíduos com índices j, $j-1$, e $j+1$ por indivíduos aleatórios
3: **se** $(j \geq i_{min} - 1)$ e $(j \leq i_{max} + 1)$ **então**
4: $\quad d \leftarrow d + 1$
5: $\quad$ **se** $(j = i_{min} - 1)$ **então**
6: $\quad\quad i_{min} \leftarrow j$
7: $\quad$ **fim se**
8: $\quad$ **se** $(j = i_{max} + 1)$ **então**
9: $\quad\quad i_{max} \leftarrow j$
10: $\quad$ **fim se**
11: **senão**
12: $\quad d \leftarrow 1$
13: $\quad i_{min} \leftarrow j$
14: $\quad i_{max} \leftarrow j$
15: **fim se**

projetista. Neste caso, o indivíduo é escolhido a partir dos indivíduos presentes em toda a população de acordo com um dado critério, por exemplo, o método da roleta [17]. Já quando o índice é igual ao índice de um indivíduo que está presente na subpopulação (ou seja, indivíduos com este índice sofreram substituição por imigrantes na extinção corrente), o indivíduo é selecionado a partir dos indivíduos presentes na subpopulação corrente (indivíduos com índices entre $i_{min} - 1$ e $i_{max} + 1$).

Algoritmo 6 Seleção dos indivíduos no SORIGA

Pré-Condição: índice i do indivíduo da nova população, sendo $1 \leq i \leq N$ e N o tamanho da população

1: **se** $(i < i_{min} - 1)$ ou $(i > i_{max} + 1)$ **então**
2: $\quad$ Selecione indivíduo de toda a população, ou seja, dos indivíduos com índices entre 1 e N
3: **senão**
4: $\quad$ Selecione indivíduo da subpopulação composta pelos indivíduos da população com índices entre $i_{min} - 1$ e $i_{max} + 1$
5: **fim se**

Um exemplo da ocorrência de avalanches no SORIGA é apresentado a seguir. Considere que o SORIGA seja aplicado na otimização da seguinte função de fitness:

$$f(\mathbf{x}) = \frac{u(\mathbf{x})}{l}, \tag{7.3}$$

sendo $u(\mathbf{x})$ a função unitária do vetor binário (individuo) $\mathbf{x}$ de tamanho l. No SORIGA aqui utilizado, a população inicial é aleatória e os critérios de seleção empregados são o elitismo e o método da roleta. Mutação binária com $p_m = 0,01$ e crossover de dois pontos com $p_c = 0,6$ são empregados. O tamanho da população é 12 e $l = 30$. A Figura 7.1 apresenta as três primeiras

gerações de uma extinção. A figura apresenta o fitness de todos os indivíduos da população corrente nas gerações 27, 28 e 29 respectivamente. Na geração 27, o indivíduo com índice 6 (variável j no Algoritmo 5) tem o menor fitness da população. Assim, este indivíduo, bem como os indivíduos com índices 5 e 7 são substituídos por indivíduos aleatórios. Na geração seguinte, o indivíduo com índice $j = 5$ tem agora o menor fitness, sendo que ele junto com os indíviduos 4 e 6 são então substituídos. Na geração 29, o indivíduo com índice $j = 4$ tem agora o menor fitness da população. Pode-se observar que a reação em cadeia foi propagada porque os indivíduos restantes tem fitness mais alto que os indivíduos da subpopulação, o que geralmente ocorre quando a diversidade da população é baixa.

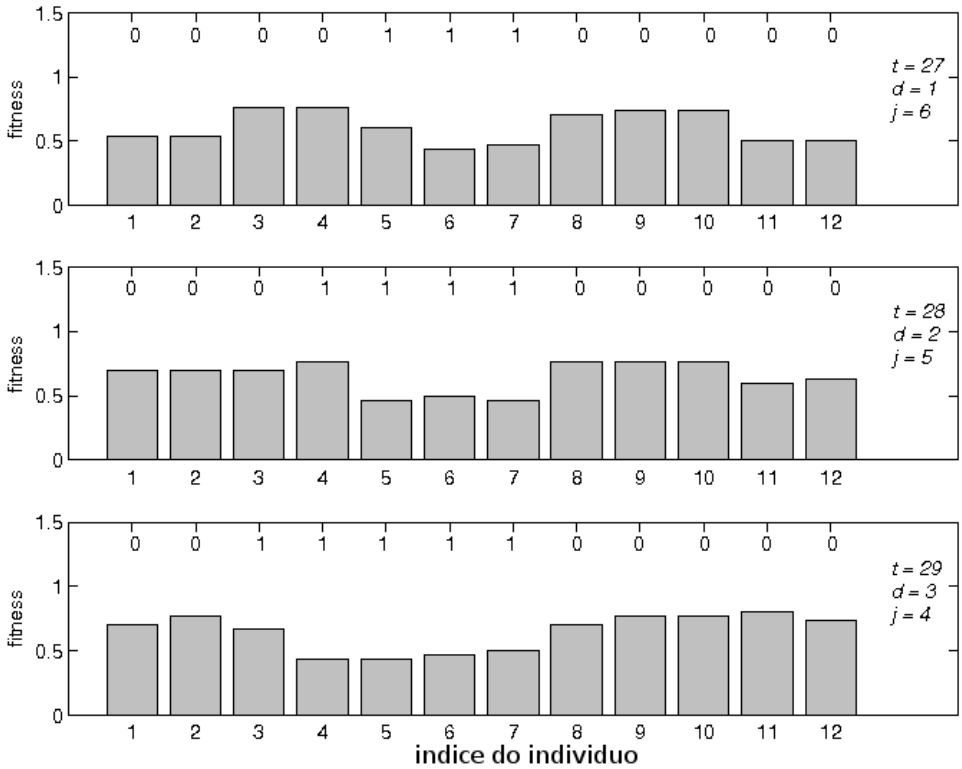

Figura 7.1: Fitness dos indivíduos da população corrente nas gerações 27, 28 e 29 em uma execução do SORIGA no exemplo dado. Os 0's e 1's no alto dos gráficos indicam os indivíduos que pertencem à subpopulação.

7.3 Resultados

Nesta seção são apresentados os resultados de experimentos com o SORIGA envolvendo a simulação de robôs evolutivos [8] em ambientes dinâmicos. Estes experimentos foram anteriormente apresentados em [59]. Com o objetivo de comparar o SORIGA com outros AEs desenvolvidos para otimização dinâmica, foram também apresentados experimentos com DOPs

gerados através do Gerador de DOPs XOR (Seção 3.3) a partir de 5 diferentes problemas estacionários em [41] e experimentos com 3 DOPs criados pelo método descrito por [60] em [61].

Aqui, dois conjuntos de simulações de robôs evolutivos em ambientes dinâmicos são apresentados. Em tais simulações, o SORIGA é comparado ao GA padrão, ao GAGDM (Capítulo 6) e a duas outras versões de GA com imigrantes aleatórios, que diferem na estratégia de substituição dos indivíduos. Na primeira versão, que será aqui chamada de GA com imigrantes aleatórios 1, três indivíduos aleatoriamente escolhidos são substituídos por novos indivíduos. Na segunda versão, que será aqui chamada de GA com imigrantes aleatórios 2, os três piores indivíduos da população corrente, isto é, os indivíduos com os valores mais baixos de $fitness$, é que são substituídos.

7.3.1 Descrição das Simulações

Simulações baseadas no experimento proposto em [62] foram utilizadas para testar o SORIGA. Nas simulações apresentadas aqui, uma versão modificada do simulador EVOROBOT desenvolvido por S. Nolfi [8] é utilizada. No simulador utilizado, Redes Neurais Artificiais (RNAs) cujos pesos são ajustados por um GA são utilizadas para controlar um robô simulado cujas características são semelhantes às de um robô Khepera com 8 sensores de distância infravermelhos, 2 sensores de luz ambiente (um de cada lado, colocados sobre o robô), e 1 sensor para detectar a luminosidade do chão colocado sob o robô.

Cada indivíduo do GA é composto por um vetor contendo todos os pesos da RNA. A RNA utilizada para controlar os robôs é do tipo $feedforward$, com uma única camada oculta composta por 5 neurônios com conexões recorrentes (Rede de Elman). A RNA possui duas saídas (saídas o0 e o1), cada uma ligada a um motor responsável pelo controle de uma das rodas do robô. As entradas da RNA são os sinais provenientes de:

- 8 sensores de distância infravermelhos colocados ao redor do robô (entradas I0 a I7). A Figura 7.2 mostra a disposição dos sensores de distância no robô. Com o objetivo de aumentar a robustez do sistema de controle a incertezas, os sinais provenientes dos sensores infravermelhos e de luminosidade são acrescidos de um ruído branco com limiar de $\pm$ 0,05.
- 2 sensores de luz ambiente colocados na parte de cima do robô (entradas I8 e I9).
- 1 sensor para detectar a luminosidade do chão colocado sob o robô (entrada I10).

- 1 sensor para detectar o nível de energia da bateria (entrada I11).

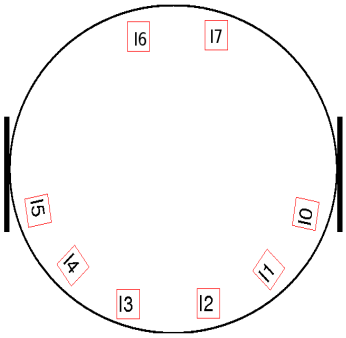

Figura 7.2: Disposição dos sensores de distância no robô simulado.

Nas simulações realizadas, o robô deve navegar por um ambiente desconhecido sem que ocorram choques com obstáculos. O robô possui uma quantidade limitada de energia, a qual é recarregada sempre que atravessa uma área de recarga de bateria. Esta área é caracterizada por possuir uma cor diferente no chão, o que altera as respostas do sensor colocado sob o robô, e por possuir uma torre com uma fonte de luz (ver Figura 7.5).

Desta forma, o $fitness$ do indivíduo é definido como a velocidade média de rotação das duas rodas acumulada durante a vida do robô, i.e. enquanto sua bateria tem energia ou até que ele se choque com um obstáculo, considerando um limite máximo de 1 minuto. Como a energia da bateria totalmente carregada dura 20 segundos, o robô tem que voltar pelo menos duas vezes para a área de recarga para que a duração de sua vida seja máxima. Vale ressaltar que o $fitness$ do robô não é acumulado quando ele se encontra na área de recarga de bateria. Apesar de a função de $fitness$ não especificar explicitamente que o robô deve retornar para a área de recarga de bateria quando esta estiver se exaurindo, os indivíduos que desenvolvem esta habilidade enquanto estiverem explorando o ambiente sem se chocarem com obstáculos apresentam valores maiores de $fitness$. Em cada simulação, a posição inicial do robô é fixa, sendo que sua orientação inicial varia aleatoriamente em 10 graus.

Dois conjuntos de simulações com 2000 gerações cada foram utilizados para testar o SORIGA. Os dois conjuntos de simulações foram propostos para testar o desempenho do SORIGA em tarefas em que o ambiente do robô se modifica e que envolvem tolerância a falhas.

No primeiro conjunto (simulação 1), o ambiente utilizado para a evolução dos robôs é alterado após 1000 gerações. Esta situação é frequente em problemas reais, quer devido a alterações do ambiente, quer devido às situações em que os robôs são primeiramente evoluídos em simulações, e, quando um desempenho satisfatório é alcançado, são transferidos para um

ambiente real. Nas simulações aqui apresentadas, os robôs são evoluídos em uma arena de 40 $\times$ 45 cm livre de obstáculos (salvo as paredes) nas primeiras 1000 gerações, e em uma arena de 60 $\times$ 35 cm com cinco objetos cilíndricos com posições fixas durante as últimas 1000 gerações (ver figuras 7.5 e 7.6).

No segundo conjunto de simulações (simulação 2), o robô é afetado por uma falha de funcionamento em seis sensores de distância após 1000 gerações. Para simular esta falha, os valores dos sinais provenientes dos sensores de distância do robô correspondentes às entradas I0, I1, I2, I5, I6 e I7 são igualados a zero (ver Figura 7.2). A simulação 2 foi proposta para que o problema de reconfiguração após falhas seja investigado. Os robôs evoluem durante todo o tempo em uma arena de 60 $\times$ 35 cm com três objetos cilíndricos fixos (ver figuras 7.7 e 7.8).

Para cada problema, os cinco GAs (GA padrão, GAGDM, GAs com imigrantes aleatórios 1 e 2 e SORIGA) foram simulados 20 vezes cada com diferentes sementes aleatórias. No início de cada simulação, os indivíduos foram gerados aleatoriamente. Em cada geração, os 20 melhores indivíduos da população corrente foram selecionados e cada um gerou 5 descendentes, considerando uma taxa de mutação igual a 0,01 (o operador *crossover* não foi utilizado). Para o GAGDM, os parâmetros são: $\upsilon_{max} = 20$; $\alpha = 0,010$ e $\beta = 10$.

7.3.2 Resultados

A Tabela 7.1 apresenta os resultados de adaptabilidade, que é definida como a diferença média entre o $fitness$ do melhor indivíduo corrente e o respectivo valor ótimo (definido aqui como 1) [63], o $fitness$ médio da população, e o $fitness$ do melhor indivíduo após 2000 gerações. Enquanto que os melhores resultados para a adaptabilidade são aqueles com os menores valores, os melhores resultados para o $fitness$ são aqueles com os maiores valores. Os resultados apresentados na Tabela 7.1 correspondem à média sobre as 20 simulações de cada algoritmo para as simulações 1 e 2.

Testes de hipótese t indicam que o $fitness$ do melhor indivíduo após 2000 gerações é maior para o SORIGA, quando comparado com o GA padrão, com níveis de significância iguais a 0,04 para a simulação 1 e 0,1 para a simulação 2. As Figuras 7.3 e 7.4 mostram, respectivamente, os valores médios (sobre as 20 simulações) de $fitness$ dos melhores indivíduos para as simulações 1 e 2.

As Figuras 7.5 e 7.6 mostram, respectivamente, simulações de indivíduos obtidos em uma geração anterior à mudança do ambiente e na geração final da sétima execução da simulação 1. Observe que na simulação apresentada na Figura 7.5, a estratégia encontrada pelo GA é a de

Tabela 7.1: Resultados do SORIGA no problema de falhas em robôs móveis

	GA	Simulação 1 (mudança do ambiente)	Simulação 2 (reconfiguração após falhas)
Adaptabilidade	Padrão	0,362 (36,14 %)	0,600 (60,02%)
	GAGDM	0,339 (33,88 %)	0,574 (57,41 %)
	Imigrantes aleatórios 1	0,351 (35,08 %)	0,520 (52,01 %)
	Imigrantes aleatórios 2	0,313 (31,29 %)	0,627 (62,68 %)
	SORIGA	0,302 (30,22 %)	0,519 (51,91 %)
Fitness Médio	Padrão	0,299	0,186
	GAGDM	0,300	0,183
	Imigrantes aleatórios 1	0,341	0,232
	Imigrantes aleatórios 2	0,354	0,184
	SORIGA	0,348	0,230
Fitness Final do Melhor Indivíduo	Padrão	0,638	0,284
	GAGDM	0,685	0,425
	Imigrantes aleatórios 1	0,669	0,388
	Imigrantes aleatórios 2	0,727	0,359
	SORIGA	0,755	0,441

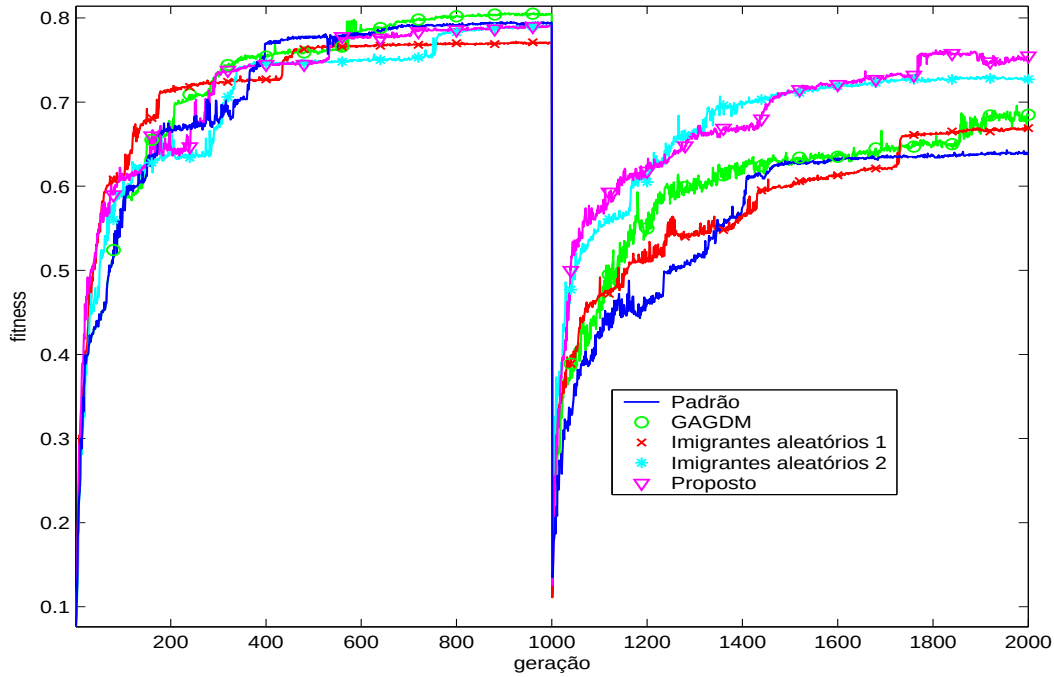

Figura 7.3: *Fitness* do melhor indivíduo da geração corrente para a simulação 1.

andar em linha reta e girar para a direita sempre que encontrar uma parede, detectada através dos sensores correspondentes às entradas I6 e I7. Quando a área de recarga é encontrada, que é detectada através do sensor de luminosidade correspondente à entrada I10 e cuja aproximação e afastamento são detectados respectivamente através dos sensores de luz ambiente (entradas

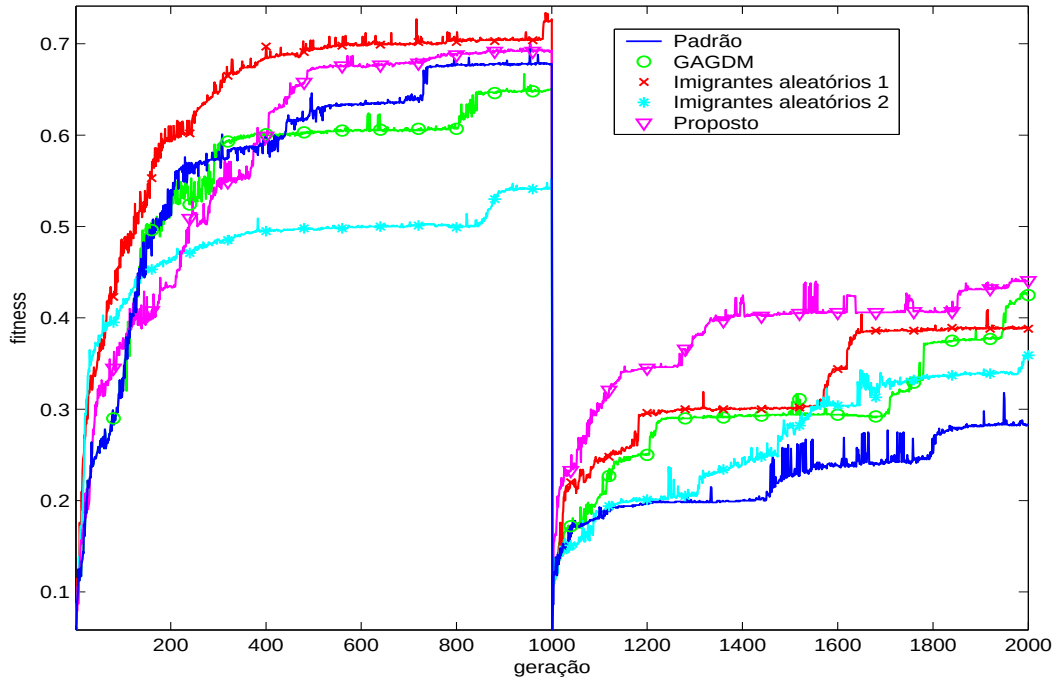

Figura 7.4: *Fitness* do melhor indivíduo da geração corrente para a simulação 2.

I8 e I9), o robô muda de direção. É interessante observar que o robô somente volta para a área de recarga quando o nível de energia da bateria, que é medido pelo sensor correspondente à entrada I11, está quase em zero. Esta estratégia, que permite que o *fitness* acumulado seja alto já que o robô navega o maior tempo possível fora da área de recarga, foi encontrada pelo GA sem que o projetista a especificasse.

Pode-se observar que a estratégia adotada na simulação apresentada na Figura 7.6 é diferente. Nesta estratégia, o robô se movimenta em uma área menor, já que obstáculos estão presentes no ambiente. Observe que após a recarga da bateria, o robô faz um círculo completo, uma curva para a direita e caminha até encontrar a parede, detectada através dos sensores correspondentes às entradas I6 e I7. É interessante notar que o círculo é feito para que o robô possa navegar o maior tempo possível fora da área de recarga.

Já nas simulações apresentadas nas Figuras 7.7 e 7.8 para a décima execução da simulação 2, o robô praticamente só anda em linha reta. A simulação apresentada na Figura 7.7 utiliza uma estratégia de controle obtida pelo SORIGA em uma geração anterior à introdução da falha. Observe que, ao encontrar uma parede, o robô faz pequenos movimentos em que gira para a direita sob seu eixo, até que fique com suas rodas paralelas à parede. Isto pode ser observado pelas ativações dos sensores correspondentes às entradas de I1 à I4. Então, o robô caminha de forma quase paralela à parede, utilizando o sensor lateral correspondente à entrada I0, até que a área de recarga ou outra parede seja encontrada. Uma vez encontrada a área de recarga, o robô

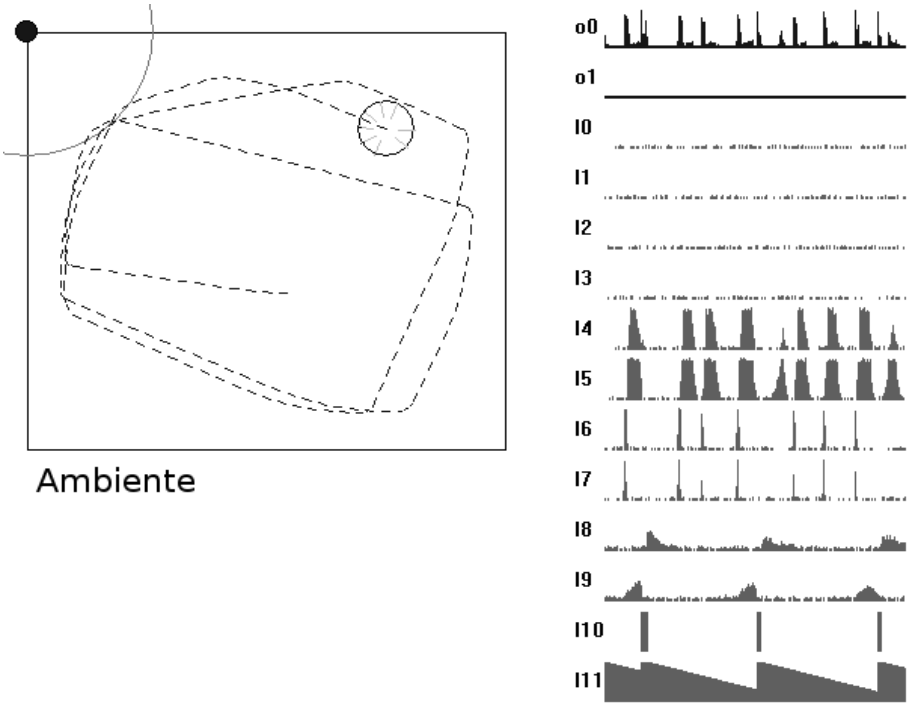

Figura 7.5: Simulação de um individuo obtido na geração anterior à mudança no ambiente na sétima execução da simulação 1 utilizando o SORIGA. O robô é representado através de um círculo de cor branca e sua trajetória através de uma linha tracejada. A área de recarga de bateria, representada por um semi-círculo no canto superior da arena, possui uma cor diferente no chão. Uma torre de iluminação, representada por um pequeno círculo hachurado, se encontra no canto superior da arena. Nos gráficos à direita, os sinais de saída para os motores (o0 e o1) e as entradas da rede provenientes dos sensores (I0 a I11) são representados nos eixos verticais e o tempo no eixo horizontal. Esta figura foi obtida através do simulador EVOROBOT.

gira para a direita e anda em linha reta até a parede oposta. Novamente observe que a bateria somente é recarregada quando o seu nível está quase em zero.

Uma estratégia diferente é adotada na simulação apresentada na Figura 7.8, que utiliza um indivíduo obtido pelo SORIGA na geração final da décima execução da simulação 2. Nesta simulação, o robô apresenta uma falha que o faz perder os sinais de todos os sensores de distância, salvo os sensores correspondentes às entradas I3 e I4. Pode-se observar nos gráficos apresentados, que somente estes dois sensores apresentam ativação diferente de zero. Na estratégia encontrada pelo SORIGA, o robô caminha em direção à área de recarga utilizando um dos sensores de luz ambiente (entrada I8). Antes de chegar na área de recarga, o robô desvia de um obstáculo utilizando os sensores de distância que ainda funcionam (entradas I3 e I4). Ao chegar na área de recarga, o robô passa a fazer pequenos movimentos lineares para frente e para trás, saindo e entrando na área de recarga (observe a entrada I10 correspondente ao sensor de luminosidade colocado sob o robô). Desta forma, sua bateria fica em um nível quase estável

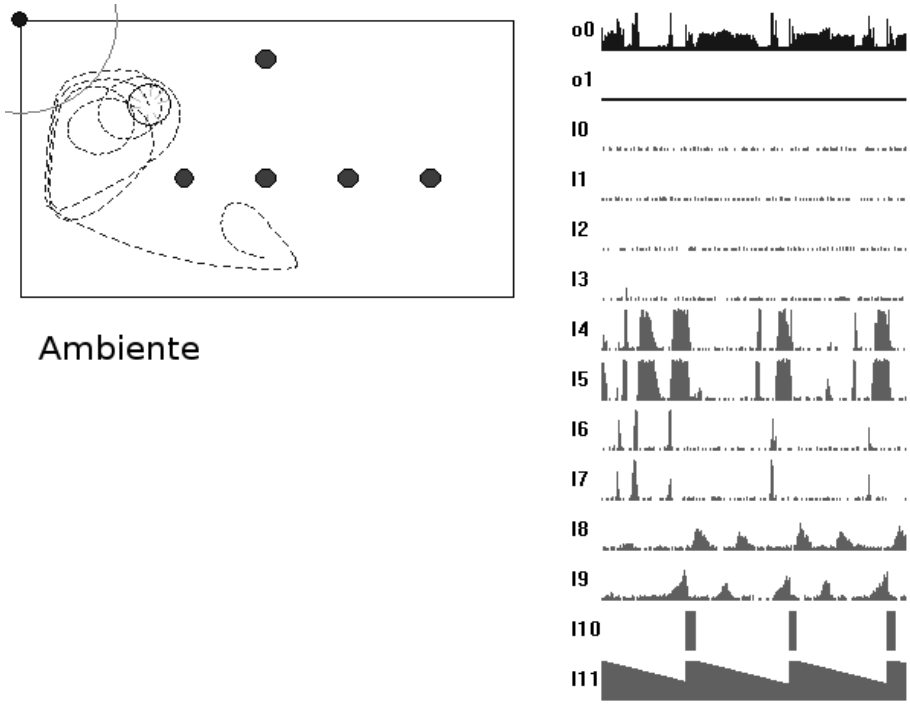

Figura 7.6: Simulação de um individuo obtido na geração final na sétima execução da simulação 1 utilizando o SORIGA. A arena simulada tem tamanho diferente daquela apresentada na Figura 7.5, além de possuir cinco obstáculos cilíndricos, representados na figura por círculos hachurados.

(entrada I11). Apesar de não acumular valores altos de *fitness* já que robô não navega pelo ambiente, esta estratégia faz com que o robô não se choque com os obstáculos do ambiente já que possui pouca capacidade sensorial.

7.3.3 Análise dos Resultados

Nos conjuntos de simulações apresentados na Seção 7.3.2, os GAs encontram, ao longo das gerações, pesos da RNA que definem a estratégia de navegação adotada. Para isso, mapas topológicos do ambiente são criados internamente e são utilizados para a navegação no ambiente de modo a aumentar ao máximo o *fitness* acumulado durante a vida do robô. A criação destes mapas topológicos internos nestas simulações é semelhante àquela encontrada nos resultados experimentais apresentados em [8].

Quando mudanças induzidas nos problemas são drásticas, como aquelas apresentadas na seção anterior, novas estratégias muito diferentes das antigas têm que ser encontradas. Desta forma, o robô não pode mais confiar nas representações internas do ambiente previamente encontradas. Na simulação 1, o ambiente mudou de tal forma, que o robô teve que abandonar a estratégia previamente encontrada e buscar uma outra (veja figuras 7.5 e 7.6). O mesmo vale

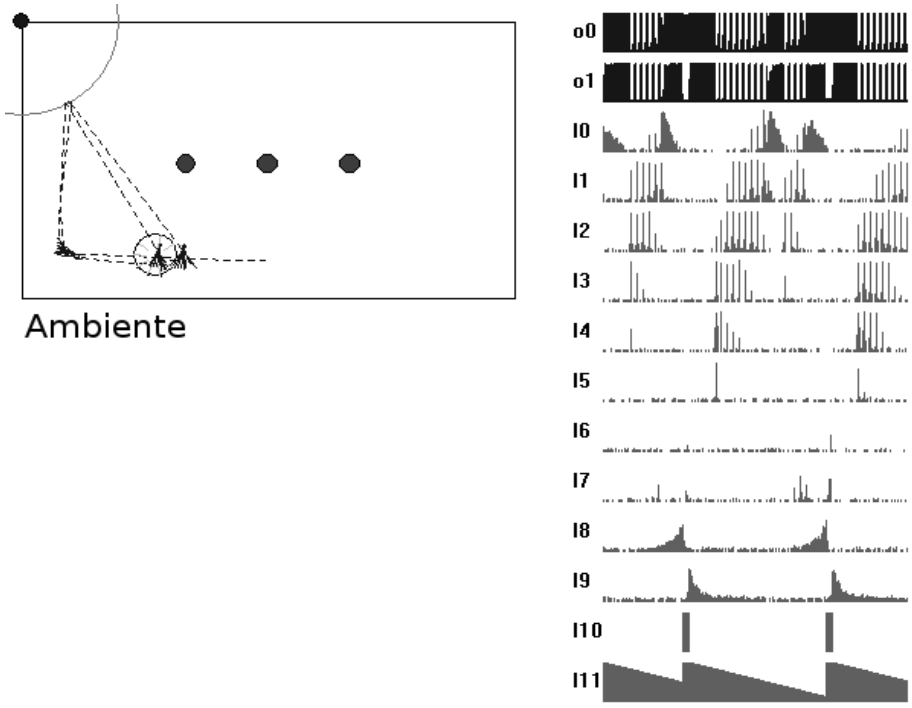

Figura 7.7: Simulação de um individuo sem falhas obtido na geração anterior à mudança no problema na décima execução da simulação 2 utilizando o SORIGA.

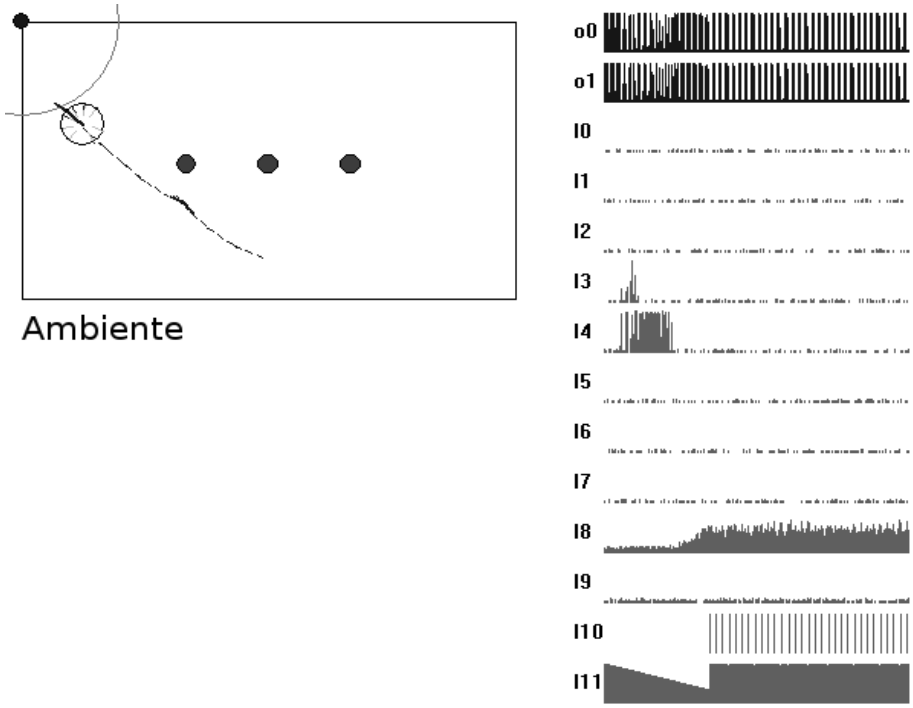

Figura 7.8: Simulação de um individuo com falhas obtido na geração final na décima execução da simulação 2 utilizando o SORIGA.

para a simulação 2, já que as falhas introduzidas mudam a forma que o robô enxerga o ambiente (veja figuras 7.7 e 7.8). Desta forma, o robô precisou desenvolver estratégias que não utilizassem os sinais provenientes dos sensores com falha. As alterações de estratégia provocadas pelas mudanças no problema foram produzidas por grandes alterações na representação da RNA responsável pelo controle do robô.

No GA padrão, e mesmo no GAGDM, uma nova estratégia para o problema é mais difícil de ser encontrada após a mudança no problema, já que o algoritmo muitas vezes fica preso em ótimos locais dados pela solução antiga. Este problema pode ser evitado quando novas soluções aleatórias são geradas, como nos GAs com imigrantes aleatórios. Os resultados dos GAs com imigrantes aleatórios e do SORIGA serão agora analisados. Primeiro, o modo com o SORIGA funciona será comentado.

Nas primeiras gerações das execuções do SORIGA, os indivíduos geralmente apresentam valores de $fitness$ baixos e um nível de diversidade alto. No SORIGA, os novos indivíduos que substituem o pior indivíduo e seus dois vizinhos próximos também têm valores de $fitness$ baixos. Como diversos indivíduos da população têm valores baixos de $fitness$, a probabilidade que o novo pior indivíduo seja um dos vizinhos do pior indivíduo anterior é baixa. Consequentemente, substituições de um único indivíduo dificilmente geram grandes reações em cadeia.

Conforme o $fitness$ médio da população aumenta, eventualmente um ótimo local é alcançado. Então, conforme aumenta o número de gerações, grande parte da população converge para o ótimo local, resultando em um baixo nível de diversidade. Nesta situação, diversos indivíduos da população têm valores altos de $fitness$ se comparados com o $fitness$ médio de todos os novos indivíduos possíveis. Quando o pior indivíduo da população é substituído com seus dois vizinhos próximos, a probabilidade que o pior indivíduo da população seguinte seja um destes vizinhos é alta já que eles geralmente apresentavam valores de $fitness$ altos. Então, uma grande reação em cadeia (extinção) pode ser formada e um grande número de indivíduos pode ser extinto, aumentando assim o nível de diversidade da população. De fato, o fator de correlação entre o tamanho das extinções e a variância do $fitness$ da população é positiva, indicando que a diversidade cresce com o tamanho das extinções. É interessante observar que este é um comportamento auto-organizável, que muda com o nível de diversidade da população.

Os melhores resultados do SORIGA quando comparado com os outros dois GAs com imigrantes aleatórios nas simulações apresentadas podem ser explicados por dois fatores principais. Primeiro, o número de índices diferentes de indivíduos que são substituídos em um período fixo de gerações geralmente é mais alto no SORIGA. No GA com imigrantes aleatórios em que os piores indivíduos são substituídos, é comum a substituição de indivíduos que acabaram de apa-

recer na população, ou seja, com os mesmos índices. Isto é explicado pelos valores baixos de $fitness$ dos novos indivíduos, gerando baixos níveis de diversidade. Já no SORIGA, as reações em cadeia geram um número maior de índices de indivíduos que são substituídos e mantidos em uma subpopulação. Este fator pode ser observado na análise do $fitness$ médio da população na Tabela 7.1 . Observe que o SORIGA apresenta valores de $fitness$ médio da população próximos ao dos outros GAs com imigrantes aleatórios, mesmo apresentando os maiores valores de $fitness$ do melhor indivíduo.

O segundo fator principal que explica os melhores resultados para o SORIGA é que a probabilidade de que um novo indivíduo seja preservado geralmente é maior no SORIGA. Isto ocorre porque, em um GA com imigrantes aleatórios, os valores de $fitness$ dos indivíduos da população corrente, que geralmente estão localizados próximos do ótimo local após algumas gerações, são em geral maiores que o $fitness$ médio do espaço de busca, isto é, de todos os novos indivíduos possíveis. Um imigrante geralmente sobrevive somente se seu $fitness$ for próximo de ou maior que o $fitness$ médio da população, o que é um evento raro quando a população está em um ótimo local muito maior que o $fitness$ médio do espaço de busca. Nas simulações apresentadas, pode-se observar que a solução antiga é geralmente muito melhor que a maioria das novas soluções geradas, já que na solução antiga, o robô geralmente conseguia navegar em linha reta pelo ambiente. Por outro lado, o SORIGA preserva uma nova solução potencial em uma subpopulação e permite que esta solução evolua enquanto a extinção corrente está em progresso. Quando a extinção termina, versões evoluídas de soluções potenciais dadas por imigrantes podem ser dispersas na população atual.

A Figura 7.9 apresenta o $fitness$ do melhor indivíduo e a duração das extinções para o SORIGA na décima execução da simulação 2. Observe que, após a introdução da falha (geração 1000), o $fitness$ do melhor indivíduo diminui, já que o robô não consegue navegar pelo ambiente com a estratégia previamente evoluída. No entanto, após algumas extinções, uma nova estratégia é encontrada (depois da geração 1800). Observe, ainda, que extinções com durações médias altas geralmente ocorrem quando o valor de $fitness$ é alto.

Nas simulações apresentadas nesta seção, assim como nos registros fósseis sobre tamanhos de extinções na natureza [54], existem mais extinções pequenas que grandes. Na Figura 7.10, o gráfico em uma escala log-log do número de extinções contra o tamanho de cada uma na décima execução da simulação 2 é apresentado.

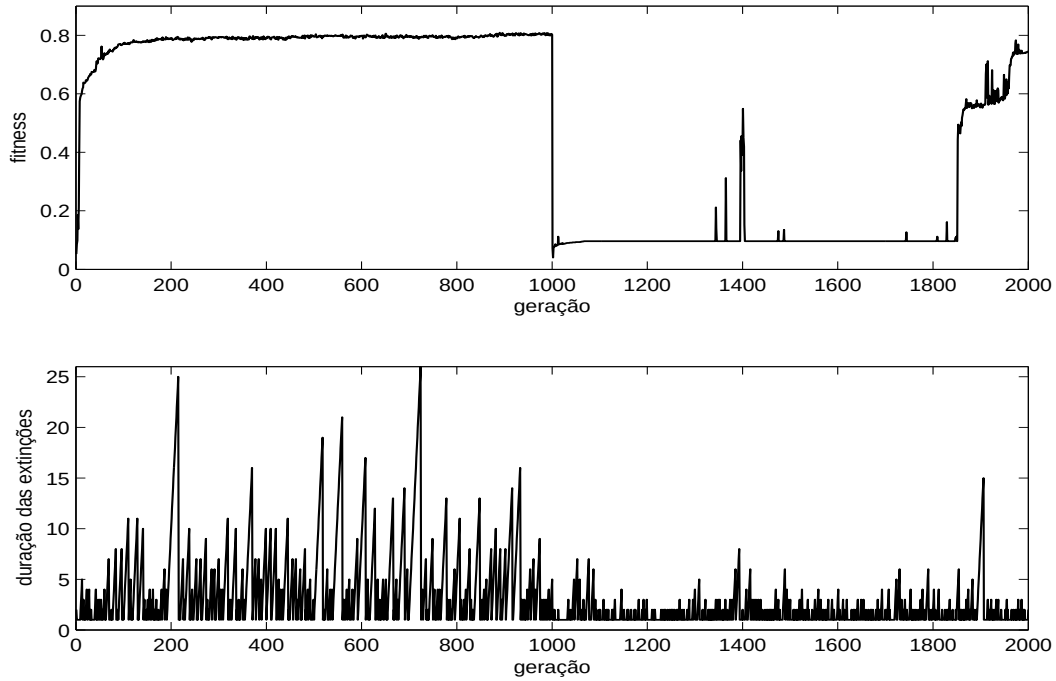

Figura 7.9: *Fitness* do melhor indivíduo e duração das extinções na décima execução da simulação 2 (reconfiguração após falhas).

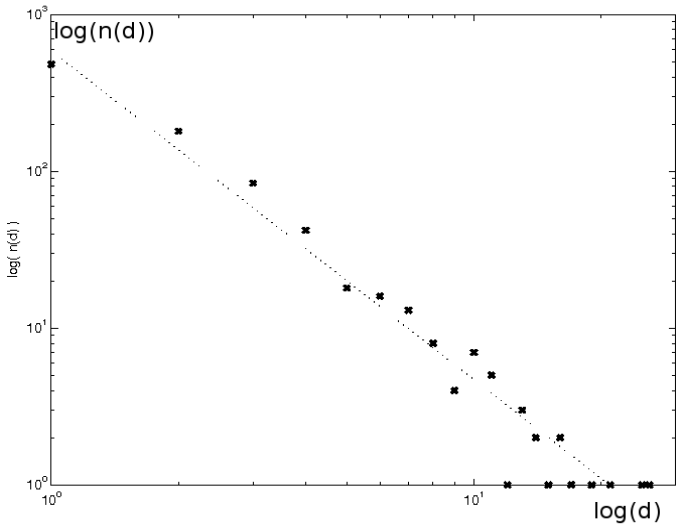

Figura 7.10: Número de ocorrências para cada tamanho das extinções na décima execução da simulação 2 (reconfiguração após falhas).

7.4 Comentários Finais

No SORIGA, ocorre a substituição do pior indivíduo e de seus dois vizinhos próximos por imigrantes aleatórios em cada geração. Os novos imigrantes são preservados em uma subpopulação, definida não pelo programador e sim pelo tamanho da extinção corrente. No algoritmo

proposto, a população se auto-organiza de modo a permitir a ocorrência de extinções com uma grande variedade de escalas de tamanhos. Grandes extinções geralmente ocorrem quando a maior parte dos indivíduos da população exibe valores altos de fitness, e quando o nível de diversidade da população é pequeno. Desta forma, a diversidade da população é aumentada nestes casos, o que faz com que o AG possa escapar de ótimos locais ocasionados pelas mudanças no ambiente.

Tal característica minimiza o problema encontrado na definição do número de substituições na estratégia padrão dos imigrantes aletórios. Se o número de imigrantes é grande, menos soluções são utilizadas para encontrar os ótimos locais. Se por outro lado, o número é pequeno, os imigrantes são diluídos na população, tornando o benefício de introduzí-los desprezível. O SORIGA minimiza este problema utilizando a auto-organização para definir o tamanho da subpopulação de imigrantes e de seus descendentes.

8 *Algoritmos Evolutivos com Mutação q-Gaussiana*

Quando AEs com codificação real são aplicados, o operador de mutação geralmente emprega distribuições Gaussianas isotrópicas para gerar novas soluções candidatas [64]. O uso de distribuições Gaussianas isotrópicas é interessante porque ela maximiza, sob certas condições, a Entropia de Boltzmann-Gibbs (e consequentemente a Entropia da Informação de Shannon para o caso contínuo) [65]. Como a distribuição Gaussiana isotrópica não privilegia nenhuma direção de busca, a geração de novas soluções candidatas não requer nenhum conhecimento sobre a geometria do espaço de busca.

Contudo, pesquisadores têm proposto recentemente o uso de distribuições com caudas mais longas e com segundo momento infinito em AEs, em oposição à distribuição Gaussiana isotrópica, que tem segundo momento finito. Como exemplos, podem ser citados o *Fast Evolutionary Programming* (FEP) [66], que utiliza a distribuição de Cauchy, e a Programação Evolutiva com distribuição de Lévy (LEP) [67]. O uso de mutações com distribuições com segundo momento infinito implica em saltos com tamanhos livres de escala, permitindo que regiões mais distantes sejam alcançadas no espaço de busca. Esta propriedade pode ser interessante em problemas multimodais ou de otimização dinâmica já que pode fazer com que as soluções candidatas escapem com mais facilidade dos ótimos locais. Contudo, alguns pesquisadores argumentam que, em geral, os algoritmos propostos utilizam distribuições anisotrópicas, ou seja, em que algumas direções do espaço de busca são privilegiadas [68], o que seria interessante somente para problemas com funções de aptidão separáveis. Além disso, em geral, é difícil atingir uma região interessante do espaço de busca com um salto longo partindo de um ótimo local [69], já que a aptidão da solução corrente é em geral maior que a aptidão média do espaço de busca.

Em [70], Tinós e Yang propõem o uso de AEs com auto-adaptação da distribuição de mutações. Desta forma, a decisão de escolher qual distribuição de mutações é mais indicada para o problema, ou para um determinado momento do processo evolutivo, fica a cargo do algoritmo, e não mais do projetista. Na metodologia proposta, distribuições do tipo q-Gaussiana, derivadas

do conceito de entropia generalizada de Tsallis [65], são utilizadas. Mudando-se o parâmetro real q é possivel mudar o formato da distribuição q-Gaussiana, reproduzindo distribuições com segundo momento finito, como a distribuição Gaussiana, e com segundo momento infinito, como a distribuição de Cauchy. Nos algoritmos, o parâmetro q que define a distribuição de mutações é codificado no cromossomo dos indivíduos e é submetido à evolução.

O uso de mutações geradas a partir de distribuições q-Gaussianas em AEs não é novo [71], [72]. Contudo, nos AEs propostos nestas referências, similarmente ao que ocorre na maioria dos AEs que utilizam distibuições com cauda longa [66], [67], novas soluções candidatas são produzidas gerando-se desvios aleatórios independentes em cada coordenada da solução atual, o que implica em distribuições anisotrópicas nas quais grandes saltos ocorrem com muito mais frequência nas regiões vizinhas aos eixos das coordenadas. Além disso, o parâmetro q, o qual define a forma da distribuição, é mantido fixo durante todo o processo de otimização ou decresce durante o processo de busca, similarmente ao que é feito no Recozimento Simulado (i.e., o controle do parâmetro é determinista).

Controle das distribuições de probabilidade de mutação, por outro lado, também não é novo. Em [73], a função de densidade de probabilidades de mutação é gerada a partir de um histograma com 101 barras representando os valores de probabilidades entre os intervalos de interesse em um espaço unidimensional. Auto-adaptação é utilizada para controlar a altura das barras, permitindo a mudança na forma da distribuição de probabilidades de mutação durante a execução do AE [74]. Experimentos em [73] indicaram a formação de histogramas com um único pico central, confirmando que distribuições como a Gaussiana e a de Cauchy são boas candidatas para distribuições de probabilidade de mutação. Contudo, o uso de histogramas não é viável quando o número de dimensões do problema aumenta, já que o número de histogramas requerido aumenta exponencialmente. Como indicado em [74], o controle da função de densidade da probabilidade de mutação usando poucos parâmetros de controle é altamente desejável.

Em [75], quatro tipos de mutação ocorrem: do tipo Cauchy, do tipo Lévy e duas Gaussianas do tipo padrão nas quais as mudanças ocorrem separadamente em cada coordenada. Um vetor com quatro elementos contendo as probabilidades de se escolher cada tipo de mutação é adicionado ao indivíduo e é modificado ao longo das gerações de acordo com o desempenho de cada tipo de mutação.

A metologia apresentada em [70] para controlar o formato da distribuição de mutações, e que é aqui apresentada, emprega somente um parâmetro para cada indivíduo: o parâmetro q da distribuição q-Gaussiana. Diferentemente da estratégia usada em [75], o controle do parâmetro q permite mudanças suaves e contínuas no formato da distribuição, já que o parâmetro q é um

parâmetro real e uma pequena mudança em seu valor causa uma pequena mudança no formato da distribuição de mutações. Desta forma, as principais contribuições apresentadas em [70] são: 1) Mutações q-Gaussianas geradas a partir de distribuições anisotrópicas e isotrópicas são propostas e comparadas; 2) Auto-adaptação é utilizada para controlar o parâmetro q, o que permite mudar a forma da distribuição durante o processo de otimização; 3) Mutações Gaussiana, de Cauchy e q-Gaussianas geradas a partir de distribuições anisotrópicas e isotrópicas são comparadas em um conjunto de experimentos; 4) Uma ES com mutação q-Gaussiana é proposta e comparada com outros AEs para otimização contínua.

Na próxima seção, a distribuição q-Gaussiana é brevemente introduzida.

8.1 A Distribuição q-Gaussiana

Apesar de a distribuição Gaussiana ser um atrator para sistemas aleatórios com componentes independentes e com variância finita, ela não representa bem sistemas correlacionados com segundo momento infinito [65]. A distribuição q-Gaussiana pode ser vista como uma generalização da distribuição Gaussiana, aparecendo quando a entropia generalizada de Tsallis é maximizada. A entropia generalizada de Tsallis foi proposta como uma generalização da Entropia de Boltzmann-Gibbs, sendo dada por:

$$S_q = \frac{1 - \int_{-\infty}^{+\infty} p(x)^q dx}{q-1}, \tag{8.1}$$

na qual $q \in \mathbb{R}$. A Eq. 8.1 reproduz a equação da Entropia de Boltzmann-Gibbs para $q \to 1$. A distribuição q-Gaussiana aparece quando maximizamos a entropia generalizada dada pela Eq. 8.1.

Quando $-\infty < q < 3$, a função densidade da distribuição q-Gaussiana [65] é dada por:

$$p_{q(\bar{\mu}_q, \bar{\sigma}_q)}(x) = \frac{\sqrt{B_q}}{A_q}\, e_q^{-B_q(x-\bar{\mu}_q)^2}, \tag{8.2}$$

sendo que $\bar{\mu}_q$ e $\bar{\sigma}_q$ são, respectivamente, a q-média e a q-variância, A_q é o fator de normalização, B_q controla o espalhamento da distribuição q-Gaussiana e $e_q^{-u^2}$ (função q-Gaussiana de u) é definida como:

$$e_q^{-u^2} \equiv \begin{cases} \left(1+(q-1)u^2\right)^{-\frac{1}{q-1}}, & \text{se } 1+(q-1)u^2 \geq 0 \\ 0, & \text{caso contrário} \end{cases}. \tag{8.3}$$

Na Eq. 8.2, a q-média $\bar{\mu}_q$ e a q-variância $\bar{\sigma}_q$ [65] são dadas por:

$$\bar{\mu}_q \equiv \frac{\int x p(x)^q dx}{\int p(x)^q dx}, \tag{8.4}$$

$$\bar{\sigma}_q^2 \equiv \frac{\int (x-\bar{\mu}_q)^2 p(x)^q dx}{\int p(x)^q dx}, \tag{8.5}$$

as quais respectivamente reproduzem a média e a variância usuais quado $q \to 1$. Já o fator de normalização A_q é dado por $A_q = \int_{-\infty}^{+\infty} e_q^{-(x-\bar{\mu}_q)^2} dx$ [76] e B_q é dado por:

$$B_q = \left((3-q)\bar{\sigma}_q^2\right)^{-1}. \tag{8.6}$$

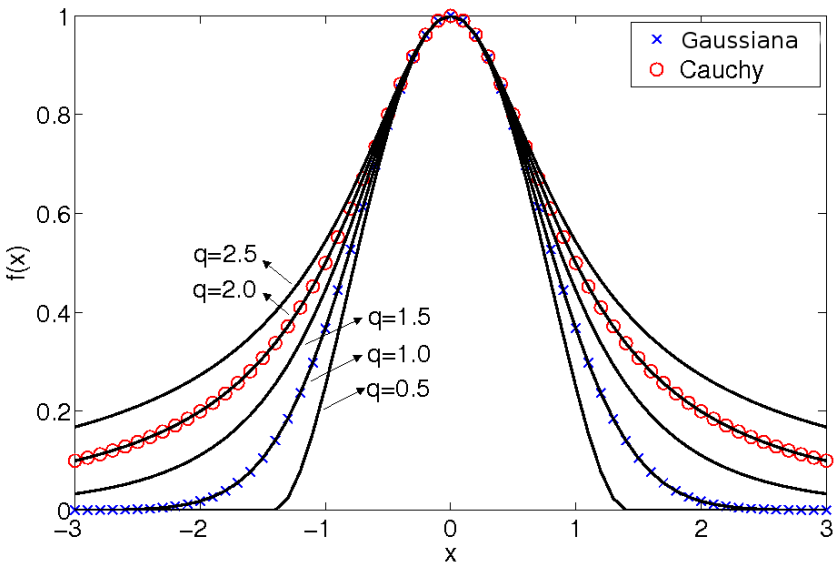

Figura 8.1: Função q-Gaussiana para diferentes valores de q. As funções Gaussiana e de Cauchy são também apresentadas.

A Figura 8.1 ilustra a forma da função q-Gaussiana para diferentes valores de q. Nesta função, o parâmetro real q altera a forma da função, o que permite controlar suavemente e continuamente a forma da distribuição q-Gaussiana. Para valores de q menores que 5/3, a distribuição apresenta segundo momento finito e para $q \to 1$ ela reproduz a distribuição Gaussiana (veja Figura 8.1). Quando $q < 1$, a distribuição q-Gaussiana apresenta uma forma compacta, e decai assintoticamente como uma lei de potência para $1 < q < 3$. Para $q = 2$, a distribuição reproduz a distribuição de Cauchy [77].

Uma variável aleatória x com distribuição q-Gaussiana com q-média $\bar{\mu}_q$ e q-variância $\bar{\sigma}_q^2$ é denotada aqui por $x \sim \mathcal{N}_q(\bar{\mu}_q, \bar{\sigma}_q)$. Em [70], o método de Box-Müller generalizado proposto em [65], o qual é muito simples (ver pseudo-código em [65]), é empregado para a geração de

variáveis aleatórias $x \sim \mathcal{N}_q(0,1)$ para $q < 3$.

8.2 Auto-adaptação da distribuição de mutações

Em um espaço de busca l-dimensional real, uma nova solução candidata é gerada pelo operador de mutação real de um AE, a partir de um indivíduo $\mathbf{x}_i$, da seguinte forma:

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{C}\,\mathbf{z}, \tag{8.7}$$

na qual $i = 1, \ldots, \mu$, $\mathbf{z}$ é um vetor aleatório l-dimensional gerado a partir de uma distribuição multivariável com média zero, $\mathbf{C}$ é a matriz que define a força de mutação em cada coordenada $j = 1, \ldots, l$. No caso mais simples,

$$\mathbf{C} = \sigma\mathbf{I}, \tag{8.8}$$

sendo $\mathbf{I}$ a matriz identidade e um único parâmetro, σ, define a força de mutação em todas as componentes de $\mathbf{x}_i$. Contudo, é interessante em muitos casos definir parâmetros $\sigma(j)$ diferentes para cada componente de $\mathbf{x}_i$, ou seja:

$$\mathbf{C} = \mathrm{diag}(\sigma^{\mathrm{T}}). \tag{8.9}$$

Neste caso, $\mathbf{C}$ é uma matriz diagonal cujos elementos na diagonal principal são dados pelo vetor $\sigma = [\sigma(1)\ \sigma(2) \ldots \sigma(l)]^{\mathrm{T}}$. No caso mais geral, e.g., na ES com adaptação da matriz de variância (CMA-ES) [78], $\mathbf{C}$ é uma matriz com os elementos fora da diagonal principal indicando a correlação entre os componentes de $\mathbf{z}$.

Em geral, quando auto-adaptação é usada, considerando-se a utilização da Eq. 8.8, o parâmetro da força de mutação para cada descendente $i = 1, \ldots, \mu$ da população é atualizado por:

$$\tilde{\sigma}_i = \sigma_i e^{\tau_b \mathcal{N}(0,1)}, \tag{8.10}$$

na qual τ_b denota o desvio padrão da distribuição Gaussiana usada para gerar a mudança em σ_i. Se cada elemento do vetor $\mathbf{x}_i$ tem um parâmetro de mutação individual (Eq. 8.9), $\sigma_i(j)$ é atualizado de acordo com:

$$\tilde{\sigma}_i(j) = \sigma_i(j) e^{\tau_b \mathcal{N}(0,1)_i + \tau_c \mathcal{N}(0,1)} \tag{8.11}$$

sendo que τ_b denota o desvio padrão da distribuição Gaussiana usada para gerar o desvio aletório $\mathcal{N}(0,1)_i$, o qual é comum para todos os elementos do vetor $\mathbf{x}_i$, e τ_c é o desvio padrão da distribuição Gaussiana usada para gerar os desvios aleatórios individuais $\mathcal{N}(0,1)$ para cada elemento $j = 1, \ldots, l$. Trabalhos teóricos e experimentais sugerem que os parâmetros τ_b e τ_c

sejam definidos por $\tau_b = \frac{k_b}{\sqrt{2l}}$ e $\tau_c = \frac{k_c}{\sqrt{2\sqrt{l}}}$, sendo k_b e k_c números reais positivos [64].

Em AEs, o uso da distribuição Gaussiana é geralmente empregada para gerar o vetor l-dimensional $\mathbf{z}$ [64]. Aqui, um vetor aleatório l-dimensional gerado a partir da distribuição Gaussiana é denotado $\mathbf{z} \sim \mathcal{N}^l$. Um vetor aleatório Gaussiano $\mathbf{z} \sim \mathcal{N}^l$ é gerado pela amostragem de l variáveis Gaussianas independentes $\mathcal{N}(0,1)$. É importante observar que quando o mesmo procedimento é adotado para gerar amostras aleatórias multivariadas com uma distribuição de cauda longa, algumas direções no espaço de busca são muito mais exploradas do que outras, ou seja, a distribuição é altamente anisotrópica.

Duas formas de gerar as distribuições q Gaussianas são utilizadas nos AEs desenvolvidos em [79] e [70], uma anisotrópica e outra isotrópica. Em ambas, um parâmetro q é associado com cada indivíduo. Aqui, apenas resultados da forma anisotrópica, na qual l variáveis aleatórias independentes são geradas para compor o vetor z, são apresentados. Baseado na forma de ajuste do parâmetro da força de mutação em ESs [64], o parâmetro q é atualizado para o indivíduo i em cada geração de acordo com:

$$\tilde{q}_i = q_i e^{\tau_q \mathcal{N}(0,1)} \tag{8.12}$$

sendo que τ_q denota o desvio padrão da distribuição Gaussiana utilizada. Aqui, τ_q é dado por:

$$\tau_q = \frac{k_q}{\sqrt{2l}}, \tag{8.13}$$

sendo k_q um número positivo real. Usando a Eq. 8.12, diferentes distribuições podem ser reproduzidas durante o processo evolutivo.

De modo a testar as idéias propostas, dois EPs com mutação q-Gaussiana foram propostos em [70] e uma ES com o mesmo tipo de mutação em [79]. A seguir, este último algoritmo é apresentado.

8.3 ES com Mutação q-Gaussiana

Em [79], a mutação q-Gaussiana com auto-adaptação é utilizada em uma (μ,λ)-ES aplicada em DOPs.

Nota-se que dois componentes podem ser mudados em cada indivíduo no esquema proposto: o vetor de parâmetros da força de mutação σ_i e o parâmetro da distribuição q-Gaussiana q_i. Como não é possível identificar a influência individual da mudança em σ_i ou em q_i na aptidão do indivíduo i se ambos os vetores de força mutação e parâmetro q sofrem mutação juntos no indivíduo (por exemplo, uma mutação benéfica em σ_i pode ser mascarada na aptidão

Algoritmo 7 (μ,λ)-ES com mutação q-Gaussiana (qGES)

1: Inicialize a população composta dos indivíduos ($\mathbf{x}_k$, σ_k, q_k) para $k = 1, \ldots, \mu$
2: Avalie os indivíduos ($\mathbf{x}_k$, σ_k, q_k) para $k = 1, \ldots, \mu$
3: **enquanto** (critério de parada não é satisfeito) **faça**
4: Use recombinação para gerar os indivíduos ($\tilde{\mathbf{x}}_i$, $\tilde{\sigma}_i$, $\tilde{q}_i$) para $i = 1, \ldots, \lambda$ a partir dos indivíduos ($\mathbf{x}_k$, σ_k, q_k) para $k = 1, \ldots, \mu$
5: **para** $i \leftarrow 1$ to λ **faça**
6: **se** $rand(0,1) \geq r_q$ **então**
7: Atualize vetor de parâmetros de força de mutação $\tilde{\sigma}_i$ de acordo com Eq. 8.11
8: **senão**
9: Atualize o parâmetro $\tilde{q}_i$ de acordo com Eq. 8.12
10: **fim se**
11: $\tilde{\mathbf{x}}_i \leftarrow \tilde{\mathbf{x}}_i + \mathbf{C}_i\,\mathbf{z}$, sendo $\mathbf{z}$ o vetor gerado pela distribuição q-Gaussiana com parâmetro $\tilde{q}_i$ e $\mathbf{C}_i = \mathrm{diag}(\tilde{\sigma}_i^{\mathrm{T}})$
12: **fim para**
13: Avalie os descendentes ($\tilde{\mathbf{x}}_i$, $\tilde{\sigma}_i$, $\tilde{q}_i$) para $i = 1, \ldots, \lambda$
14: Selecione os melhores μ indivíduos a partir dos descendentes ($\tilde{\mathbf{x}}_i$, $\tilde{\sigma}_i$, $\tilde{q}_i$), $i = 1, \ldots, \lambda$, para compor a nova população de indivíduos ($\mathbf{x}_k$, σ_k, q_k), $k = 1, \ldots, \mu$
15: **fim enquanto**

do indivíduo i se o parâmetro q_i é alterado para um valor ruim na mesma geração), apenas um dos dois componentes, i.e., σ_i ou q_i, podem sofrer mutação em cada geração. Assim, o vetor força de mutação σ_i é atualizado para o indivíduo i em cada geração se um número aleatório uniforme gerado no intervalo $[0,1]$ é igual ou maior do que um parâmetro real $r_q \in [0,1]$; caso contrário, o valor de q_i é que é atualizado.

A (μ,λ)-ES com mutação q-Gaussiana, chamada *qGES* em [79], é apresentado no Algoritmo 7. A principal diferença em relação à ES tradicional e à *Fast* ES [80] é que a mutação q-Gaussiana é empregada (passo 11) ao invés das mutações Gaussiana (ES tradicional) ou de Cauchy (*Fast* ES). A outra alteração é a adição de um procedimento para adaptar o parâmetro q (passos 6, 8, 9 e 10).

8.4 Análise da mutação q-Gaussiana

Nesta seção, o impacto de mudar o parâmetro da força de mutação σ e o parâmetro q da distribuição q-Gaussiana na probabilidade de gerar um salto σx, sendo $x \sim \mathcal{N}_q(0,1)$, na vizinhança de um ponto x^* é analisado. A análise apresentada aqui é similar àquela para as mutações Gaussiana e de Cauchy apresentada em [66].

Quando $\overline{\mu}_q = 0$ e $\overline{\sigma}_q^2 = 1$, a densidade da distribuição q-Gaussiana para $-\infty < q < 3$ (veja Eq. 8.2) é dada por:

$$p_q(x) = \frac{1}{\sqrt{3-q}A_q} \quad e_q^{\frac{-x^2}{3-q}}, \tag{8.14}$$

na qual considera-se $\left(1+x^2(q-1)/\sqrt{3-q}\right) \geq 0$ e que a q-exponencial é dada por:

$$e_q^{\frac{-x^2}{3-q}} = \left(1+\frac{q-1}{3-q} \quad x^2\right)^{\frac{1}{1-q}}. \tag{8.15}$$

Consideramos que a mutação q-Gaussiana é aplicada em um AE no caso unidimensional, ou seja, a mutação produz um salto σx, sendo σ o parâmetro da força de mutação e $x \sim \mathcal{N}_q(0,1)$. Para simplificar, vamos considerar $1 < q < 3$. A probabilidade de alcançar a vizinhança de um ponto x^* a partir de um salto σx, i.e., a probabilidade que $x^* - \varepsilon \leq \sigma x \leq x^* + \varepsilon$, na qual $\varepsilon > 0$ define o tamanho da vizinhança, é dado por:

$$P_q(|\sigma x - x^*| \leq \varepsilon) = \int_{\frac{x^*-\varepsilon}{\sigma}}^{\frac{x^*+\varepsilon}{\sigma}} p_q(x)dx. \tag{8.16}$$

O teorema do valor médio para integrais definidas afirma que existe um número δ ($0 < \delta < 2\varepsilon$) no qual o valor da integral dada por Eq. 8.16 é igual à diferença entre os limites da integral multiplicados por $p_q((x^* - \varepsilon + \delta)/\sigma)$. Assim, Eq. 8.16 pode ser escrita como:

$$P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sigma} p_q\left(\frac{x^* - \varepsilon + \delta}{\sigma}\right). \tag{8.17}$$

Substituindo as Eqs. 8.14 e 8.15 na Eq. 8.17, obtemos:

$$P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sigma\sqrt{3-q}A_q}\left(1+\frac{q-1}{3-q}\frac{c^2}{\sigma^2}\right)^{\frac{1}{1-q}}, \tag{8.18}$$

sendo $c = x^* - \varepsilon + \delta$.

8.4.1 O impacto de mudar σ

Derivando a Eq. 8.18 com respeito a σ, então:

$$\frac{\partial}{\partial\sigma}P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sqrt{3-q}A_q}\frac{\partial}{\partial\sigma}\left(\frac{1}{\sigma}\left(1+\frac{q-1}{3-q}\frac{c^2}{\sigma^2}\right)^{\frac{1}{1-q}}\right), \tag{8.19}$$

e, assim,

$$\frac{\partial}{\partial\sigma}P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sqrt{3-q}A_q}\left(\frac{2q_a c^2}{(q-1)\sigma^4}\left(1+q_a\frac{c^2}{\sigma^2}\right)^{\frac{q}{1-q}} - \frac{1}{\sigma^2}\left(1+q_a\frac{c^2}{\sigma^2}\right)^{\frac{1}{1-q}}\right), \tag{8.20}$$

sendo $q_a = (q-1)/(3-q)$. Depois de alguma manipulação, podemos obter:

$$\frac{\partial}{\partial\sigma}P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sqrt{3-q}A_q\sigma^2}\left(1+q_a\frac{c^2}{\sigma^2}\right)^{\frac{1}{1-q}}\frac{c^2-\sigma^2}{q_a c^2+\sigma^2}. \tag{8.21}$$

A partir da Eq. (8.21), podemos escrever para $1 < q < 3$:

$$\frac{\partial}{\partial\sigma}P_q(|\sigma x - x^*| \leq \varepsilon) \begin{cases} > 0 & \text{se } |c| > \sigma \\ < 0 & \text{se } |c| < \sigma \end{cases} . \tag{8.22}$$

Eq. 8.22 afirma que um aumento na força de mutação σ resulta em um aumento na probabilidade de atingir o ponto c, que está localizado na vizinhança do ponto x^*, a partir de um salto σx somente se $|c| < \sigma$. Em outras palavras, um aumento na força de mutação é benéfico para alcançar a vizinhança de um ponto x^* se este está distante ($|c| > \sigma$) da solução atual (antes da mutação). Um resultado semelhante foi encontrado para a mutações Gaussiana e de Cauchy em [66]. A análise acima também afirma que a taxa de mudança de probabilidade dada pela Eq. 8.21 depende do valor de q.

8.4.2 O impacto de mudar q

Derivando a Eq. (8.18) com respeito a q, podemos escrever que:

$$\frac{\partial}{\partial q}P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon}{\sigma}\frac{\partial}{\partial q}\left(\frac{1}{\sqrt{3-q}A_q}y\right), \tag{8.23}$$

na qual:

$$y = \left(1 + \frac{q-1}{3-q}\frac{c^2}{\sigma^2}\right)^{\frac{1}{1-q}} \tag{8.24}$$

é a q-exponencial dada pela Eq. 8.15 no ponto $x = c/\sigma$. Analisaremos agora a derivada da q-exponencial y dada pela Eq. 8.24. Aplicando o logaritmo natural em ambos os lados da Eq. 8.24 e derivando em relação a q, temos:

$$\frac{\partial y}{\partial q} = y\frac{\partial}{\partial q}\left(\frac{1}{1-q}\ln\left(1 + \frac{q-1}{3-q}\frac{c^2}{\sigma^2}\right)\right). \tag{8.25}$$

Depois de alguma manipulação, temos:

$$\frac{\partial y}{\partial q} = \frac{1}{q-1}p^{\frac{1}{1-q}}\left(\frac{\ln(p)}{q-1} - \frac{2}{(3-q)^2}\frac{c^2}{\sigma^2}p^{-1}\right), \tag{8.26}$$

sendo:

$$p = 1 + \frac{q-1}{3-q}\frac{c^2}{\sigma^2}, \tag{8.27}$$

i.e., $y = p^{\frac{1}{1-q}}$. Na Eq. 8.26, $p \geq 1$ para $1 < q < 3$. Assim, podemos escrever:

$$\frac{\partial y}{\partial q} \begin{cases} > 0 & \text{se } a > b \\ < 0 & \text{se } a < b \end{cases} , \tag{8.28}$$

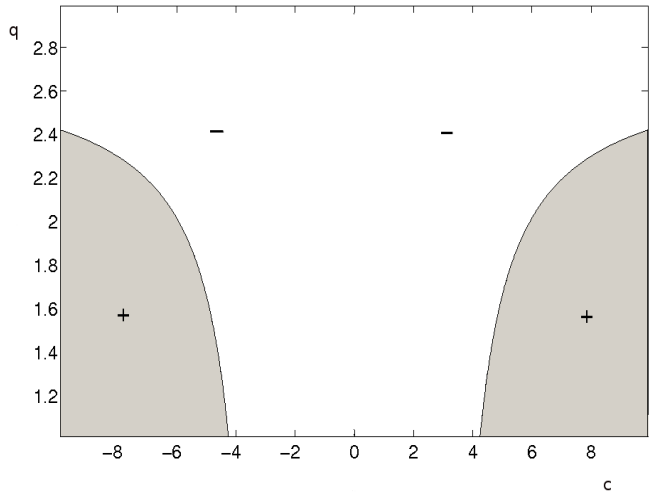

Figura 8.2: Regiões positivas (em cinza) e negativas (em branco) das derivadas da q-exponencial dada pela Eq. 8.24 para $\sigma = 3$.

sendo $a = \frac{\ln(p)}{q-1}$ e $b = \frac{2}{(3-q)^2}\frac{c^2}{\sigma^2}p^{-1}$. Enquanto $|a| > |b|$ para valores pequenos de $|c|$ (até o valor de c no qual $a = b$), $|a| < |b|$ para valores maiores de $|c|$ já que $\lim_{|c|\to\infty} a = +\infty$ e $\lim_{|c|\to\infty} b = \frac{2}{(3-q)(q-1)}$.

A Figura 8.2 mostra as regiões onde a derivada da q-exponencial dada pela Eq. 8.24 é positiva ou negativa para $\sigma = 3$ e $|c| \leq 10$. Quando a derivada é positiva, aumentando (ou diminuindo) q por um pequeno valor implica em aumentar (ou diminuir) o valor da q-exponencial no ponto c, enquanto o oposto ocorre para uma derivada negativa. Pode-se observar que a localização de c onde a derivada muda seu sinal se move de acordo com o valor de q. Quanto maior o valor de q, mais longe a posição de $|c|$ onde a derivada muda de sinal. A Figura 8.2 (e a Eq. 8.26) também sugere(m) que os valores de q pertos de 3 não são interessantes já que a localização onde a derivada muda de sinal é muito longe da solução atual.

Usando a derivada da q-exponencial dada pela Eq. 8.24, podemos agora escrever, depois de algumas manipulações, a derivada da Eq. 8.18 em relação a q (veja Eq. 8.23):

$$\frac{\partial}{\partial q}P_q(|\sigma x - x^*| \leq \varepsilon) = \frac{2\varepsilon p^{\frac{1}{1-q}}}{A_q\sigma\sqrt{3-q}(q-1)}\left(\frac{(q-1)(A_q-(3-q)A'_q)}{(3-q)A_q} + a - b\right). \tag{8.29}$$

Como o primeiro termo dentro dos parênteses não depende de c, a análise é semelhante ao que foi apresentado antes para a q-exponencial dada pela Eq. 8.26. A Eq. 8.29 indica onde uma pequena mudança no valor de q é benéfica para aumentar a probabilidade de alcançar a vizinhança de um ponto x^* a partir de um salto σx. Em outras palavras, um aumento no valor de q é benéfico para alcançar a vizinhança de um ponto x^* se este é distante (em um local onde

a derivada dada por Eq 8.29 é positiva) da solução atual (antes da mutação). Caso contrário, o valor de q deve ser diminuído.

8.5 Experimentos

De modo a avaliar o desempenho da mutação q-Gaussiana em AEs para otimização contínua em problemas estacionários, experimentos com 25 funções do *benchmark* proposto em [81] foram realizados em [70]. Os experimentos indicam o bom desempenho da mutação q-Gaussiana em problemas altamente multimodais e/ou ruidosos. Já em [79], o algoritmo apresentado na Seção 8.3 foi aplicado em dois conjuntos de experimentos com DOPs. No primeiro conjunto, que é apresentado aqui, o Gerador *Moving Peaks* (MP) (Seção 3.4) é utilizado para criar DOPs. No segundo conjunto de experimentos, robôs evolutivos são simulados em ambiente dinâmicos.

Em cada experimento com o Gerador MP aqui apresentado, 3 ESs são executadas 50 vezes cada (com diferentes sementes aleatórias). No primeiro algoritmo, qGES, o parâmetro q da mutação q-Gaussiana é permitido evoluir ao longo do processo evolucionário, sendo $r_q = 0,8$ no Algoritmo 7. Nos outros dois algoritmos, $r_q = 0,0$ no Algoritmo 7 e, consequentemente, o parâmetro q é fixo, ou seja, começa com um determinado valor e não é modificado durante a evolução. No algoritmo GES, $q = 1$, ou seja, a distribuição q-Gaussiana reproduz a distribuição Gaussiana. No algoritmo CES, $q = 2$, ou seja, a distribuição de Cauchy é reproduzida pela distribuição q-Gaussiana. Assim, os três tipos de mutação, q-Gaussiana, Gaussiana e de Cauchy são comparados nos experimentos.

Para cada execução de um algoritmo, os indivíduos da população inicial são escolhidos aleatoriamente. Em todos os experimentos do algoritmo qGES, o parâmetro q inicial é igual a 1,0 (valor no qual a distribuição Gaussiana é reproduzida) e os valores mínimo e máximo do parâmetro q são, respectivamente, iguais a 0,8 e e 2,2, ou seja, valores respectivamente menores e maiores que os valores de q nos quais as mutações Gaussiana e de Cauchy são reproduzidas. Os parâmetros k_b e k_c, respectivamente utilizados para calcular τ_b e τ_c, são iguais a 1,0. O mesmo valor é adotado para k_q, que é utilizado na Eq. 8.13 para o cálculo de τ_q.

Para o Gerador MP (ver Eq. 3.5), os parâmetros $h_i(e)$ e $w_i(e)$, para $e > 1$, são respectivamente alterados em cada mudança de acordo com:

$$h_i(e+1) = h_i(e) + \rho\ \mathcal{N}(0,1), \tag{8.30}$$

e:

$$w_i(e+1) = w_i(e) + \rho\ \mathcal{N}(0,1), \tag{8.31}$$

enquanto que o vetor definindo a posição do centro de cada pico é alterado por rotação, multiplicando a posição do vetor em cada ciclo de mudança e por uma matriz de rotação composta por sucessivas rotações simples em planos aleatórios, da mesma maneira que é feito no gerador de problemas contínuos apresentado no Capítulo 9. Os ângulos da matriz de rotação, em graus, são dadas por 180ρ. Enquanto o parâmetro ρ controla a severidade das mudanças, o parâmetro τ controla a velocidade das mudanças.

Nesta seção, experimentos com 5 valores fixos de ρ, ou seja, nos quais ρ não muda durante a execução, são apresentados. Experimentos nos quais o valor de ρ muda durante a execução de acordo com uma dinâmica caótica, utilizando a função logística discreta com parâmetro 3,67, também são apresentados (tais experimentos são identificados como $\rho = c$ nas tabelas e figuras). Experimentos com $\tau = 10$, 50, e 300 são apresentados. Desta forma, 18 experimentos com diferentes valores de ρ e τ são gerados, ou seja, os algoritmos são testados com diferentes combinações de severidade e velocidade das mudanças. Para todos os experimentos, o número de ciclos de mudança (n_c) durante o processo evolutivo é 50, $l = 10$ e o número de picos é $n_p = 10$. Para os algoritmos, o número de pais (μ) é definido como 15, enquanto o número de filhos (λ) é 100. Recombinação intermediária com dois pais para cada filho é usada.

De forma a avaliar o desempenho dos algoritmos, o erro da média do melhor fitness ($\bar{e}_j$) em cada ciclo de mudança j é calculado. Este valor é calculado utilizando o erro do fitness ao invés do fitness na Eq. 4.3. Quando menor o valor de $\bar{e}_j$, melhor o desempenho do algoritmo.

8.5.0.1 Resultados

A Figura 8.3 apresenta a mediana do erro da média do melhor fitness em cada ciclo de mudança para cada algoritmo considerando diferentes valores de τ e ρ. A média da norma Euclidiana do vetor de parâmetros de força de mutação força e do parâmetro de distribuição q do melhor indivíduo nos últimos 10 ciclos de mudança do experimento com $\tau = 300$ e $\rho = 0,1$ ou $\rho = 0,2$ são mostrados na Figura 8.4.

Na Tabela 8.1, a comparação estatística dos algoritmos em relação ao erro da média do melhor fitness em cada ciclo de mudança é feita utilizando o teste de Wilcoxon [82]. A Tabela 8.1 mostra o p-valor do teste de Wilcoxon para cada experimento, o que indica a significância de testar a hipótese nula de que a diferença entre as amostras pareadas dos resultados referentes aos Alg. X e Alg. Y vêm de uma mesma distribuição com média igual a zero. Para cada

Tabela 8.1: Comparação estatística em relação ao erro da média do melhor fitness em cada ciclo de mudança.

τ	ρ	qGES - GES	qGES - CES
10	0,1	5,02E-01 ($-$)	2,18E-03 ($s+$)
10	0,2	3,04E-01 ($-$)	6,40E-01 ($+$)
10	0,3	6,40E-01 ($+$)	2,31E-04 ($s+$)
10	0,4	8,36E-01 ($+$)	4,43E-06 ($s+$)
10	0,5	1,81E-01 ($+$)	1,60E-02 ($s-$)
10	c	6,89E-01 ($-$)	1,64E-04 ($s-$)
50	0,1	1,97E-01 ($+$)	2,74E-02 ($s-$)
50	0,2	6,59E-02 ($+$)	9,04E-01 ($+$)
50	0,3	1,44E-02 ($s+$)	1,28E-03 ($s+$)
50	0,4	9,59E-02 ($+$)	7,83E-01 ($+$)
50	0,5	2,48E-02 ($s+$)	2,36E-02 ($s+$)
50	c	9,02E-03 ($s+$)	3,04E-01 ($+$)
300	0,1	8,31E-07 ($s+$)	2,19E-02 ($s+$)
300	0,2	1,05E-03 ($s+$)	7,98E-01 ($+$)
300	0,3	2,24E-02 ($s+$)	9,88E-01 ($+$)
300	0,4	2,36E-02 ($s+$)	3,27E-01 ($+$)
300	0,5	6,12E-01 ($+$)	1,38E-01 ($+$)
300	c	8,36E-01 ($+$)	5,78E-02 ($+$)

experimento, o resultado em relação à comparação Alg. X-Alg. Y é mostrado, entre parênteses, como "$=$" quando os valores das medianas dos Alg. X e Alg. Y são iguais. Quando os valores da mediana são diferentes, mas o p-valor é maior do que 0,05, ou seja, o teste indica que a hipótese de que a mediana da diferença entre os resultados é igual a zero não pode ser rejeitada ao nível de 5%, o resultado é mostrado como, respectivamente, "$+$" quando a mediana do Alg. X é menor que a mediana do Alg. Y e "$-$" quando a mediana do Alg. X é maior do que a mediana do Alg. Y. Caso contrário, quando o resultado é estatisticamente significativo, o resultado é mostrado como '$s+$" ou "$s-$" quando a mediana do Alg. X é, respectivamente, menor ou maior do que a mediana do Alg. Y.

8.5.0.2 Análise dos Resultados

Pode ser observado na Figura 8.4 que para os três algoritmos (com mutações Gaussiana, de Cauchy, e q-Gaussianas), o parâmetro da força mutação aumenta quando o ambiente muda e depois converge para valores pequenos até a próxima mudança. Quando a mudança é mais severa (para $\rho = 0,2$), o valor máximo atingido pela norma do parâmetro força de mutação é maior. Quando os parâmetros de força de mutação aumentam depois de uma mudança no ambiente, saltos maiores ocorrem com mais freqüência, o que pode ajudar a população a atingir novos ótimos mais rapidamente. Então, os parâmetros de menor força de mutação são selecionados, a

fim de aumentar a busca na vizinhança local da melhor solução atual.

Além disso, na ES com mutação q-Gaussiana, o parâmetro q aumenta depois da mudança no ambiente, o que resulta em distribuições com caudas mais longas. Então, menores valores de q são selecionados, resultando em distribuições mais compactas, aumentando consequentemente a busca na vizinhança local da melhor solução atual. Pode-se observar que, à semelhança dos parâmetros de força de mutação, o valor máximo atingido pelo parâmetro q para o algoritmo qGES na Figura 8.4 é maior quando a mudança é mais grave (para $\rho = 0,2$). Tais resultados confirmam a hipótese de que a auto-adaptação nas ESs é útil para DOPs porque fornece um mecanismo intrínseco de adaptação às alterações do problema.

Vamos agora analisar os resultados dos algoritmos com mutações Gaussianas e de Cauchy. Pode ser observado na Figura 8.3 que a mutação Gaussiana apresenta resultados melhores do que a mutação de Cauchy para $\tau = 10$ (mudança no ambiente rápida) quando $\rho < 0,5$. Quando $\tau = 10$, os algoritmos não têm tempo suficiente para explorar novos picos depois de uma mudança, e a maioria das mutações vantajosas ocorre devido a pequenos saltos no espaço de busca (processo *hill-climbing*). Como a distribuição Gaussiana produz mais saltos pequenos do que a mutação de Cauchy, a mutação Gaussiana gera um melhor desempenho quando comparado com a mutação Cauchy.

Para valores maiores τ, os algoritmos têm mais tempo para explorar outros picos e, como consequência, saltos maiores podem, eventualmente, permitir que as soluções candidatas saltem do pico corrente para um novo e promissor pico. Como a distribuição de Cauchy gera mais saltos grandes do que a distribuição Gaussiana, isso pode permitir que a população escape de ótimos locais mais rapidamente que a mutação Gaussiana, principalmente nas fases tardias da evolução nas quais os parâmetros de força de mutação convergiram para valores pequenos. Pode ser observado na Figura 8.3 que o CES apresenta melhor desempenho do que o GES para $\tau > 10$, principalmente para $\tau = 300$.

Na Figura 8.3, pode também ser observado que, para $\tau = 50$ e $\tau = 300$, diferentemente dos resultados para $\tau = 10$, o desempenho dos algoritmos são melhores para ρ na faixa $0 < \rho < 0,4$, e pior para ρ alto ($\rho \geq 0,4$), ou seja, a curva de desempenho tem a forma de "U"para $\tau = 50$ e $\tau = 300$. Na Gerador MP empregado aqui, maiores valores de ρ implicam em uma maior diferença média entre a aptidão da melhor solução corrente depois da mudança e a altura dos picos mais altos e, como conseqüência, um maior volume da região do espaço de busca com aptidão maior do que a aptidão da melhor solução atual. Assim, a probabilidade de saltar do pico corrente para um pico promissor aumenta com ρ devido às mudanças maiores na altura dos picos. Isso explica a diminuição do erro de fitness na faixa de $0 < \rho < 0,4$ para $\tau = 50$ e

$\tau = 300$, quando os algoritmos têm mais tempo para explorar novas regiões do espaço de busca através de saltos maiores (pode-se observar que os resultados para $\tau = 300$ são melhores que os resultados para $\tau = 50$). No entanto, maiores valores de ρ implicam também em mudanças maiores nas posições do centros dos picos, e, consequentemente, mais tempo para chegar ao centro do pico corrente. Isso explica a forma da curva para $\tau = 10$, quando os algoritmos não têm tempo suficiente para explorar o pico de corrente, e para a faixa de $\rho \geq 0,4$, quando as mudanças nas posições dos picos são muito grandes.

Nos experimentos com a mutação q-Gaussiana, as distribuições com caudas mais longas (isto é, valores mais elevados de q) são usadas pelo algoritmo logo após as mudanças no ambiente e distribuições compactas são usadas nos estágios mais tardios após as mudanças. Pode-se observar que o valor médio de q na Figura 8.4 atinge valores próximos ou superiores a 2,0 (distribuição de Cauchy) e diminui para valores menores que 1,0 (distribuição Gaussiana) após as mudanças no ambiente. Desta forma, passos maiores ocorrem com mais freqüência nas gerações logo após as mudanças, o que ajuda a população a fugir do ótimo local ou convergir mais rapidamente para o novo ótimo. Nas gerações mais tardias, as distribuições com valores baixosde q melhoram a busca local, o que ajuda o algoritmo a alcançar os melhores ótimos locais. No algoritmo proposto (qGES), valores mais altos do parâmetro q são empregados em algumas ocasiões de forma a permitir saltos mais longos entre as vizinhanças de dois picos.

A ES com mutação q-Gaussiana mutação apresenta um desempenho melhor ou similar ao algoritmo GES quando a mutação Gaussiana é melhor do que a mutação de Cauchy, e melhor ou semelhante ao algoritmo CES quando a mutação de Cauchy é melhor do que a mutação Gaussiana. Este comportamento é alcançado pelo auto-adaptação do parâmetro q da distribuição de mutação q-Gaussianas, sendo extremamente interessante em DOPs.

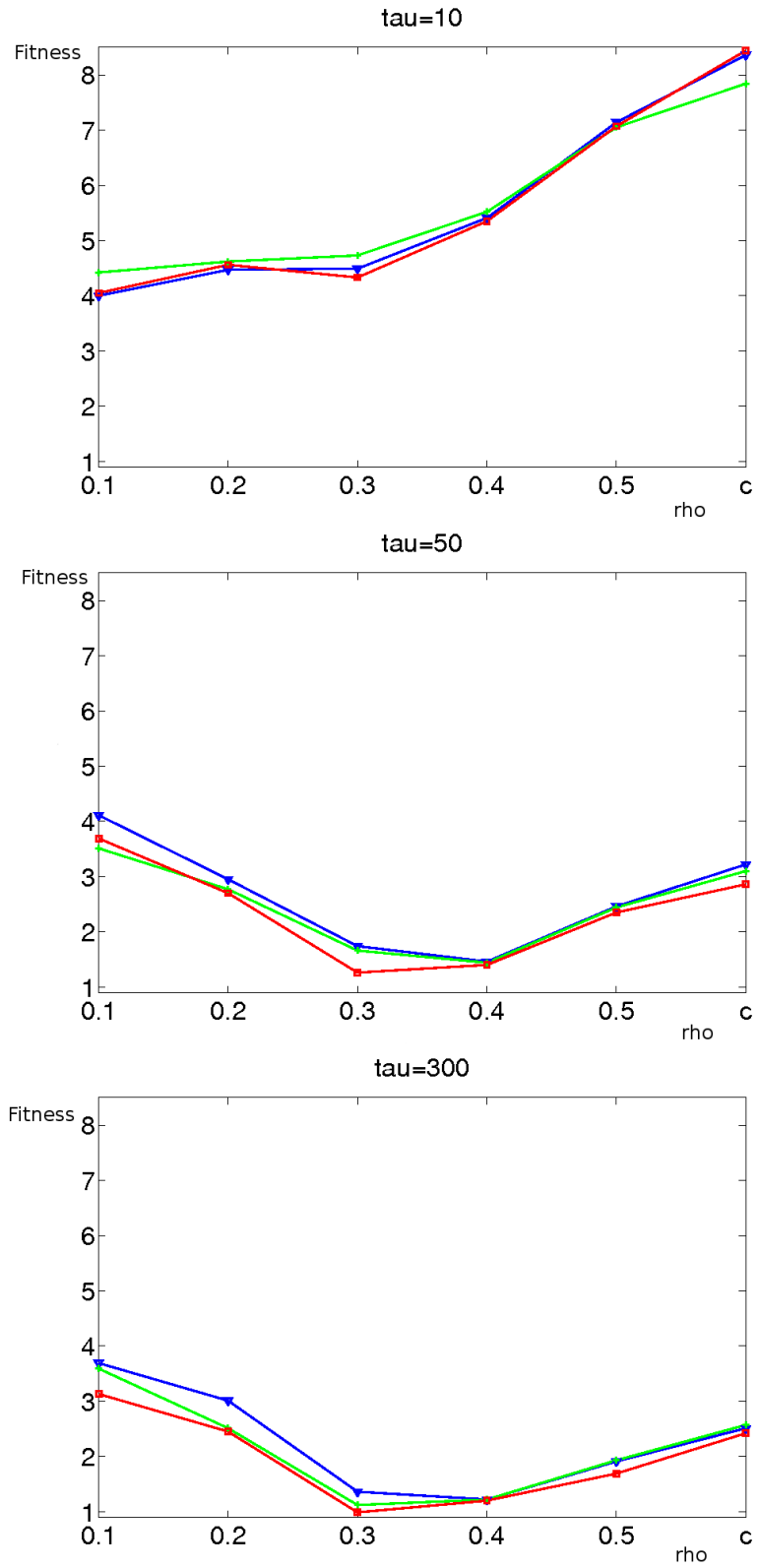

Figura 8.3: Mediana do erro da média do melhor fitness em cada ciclo de mudança para diferentes valores de ρ (GES: triângulo; CES: "+"; qGES: quadrado).

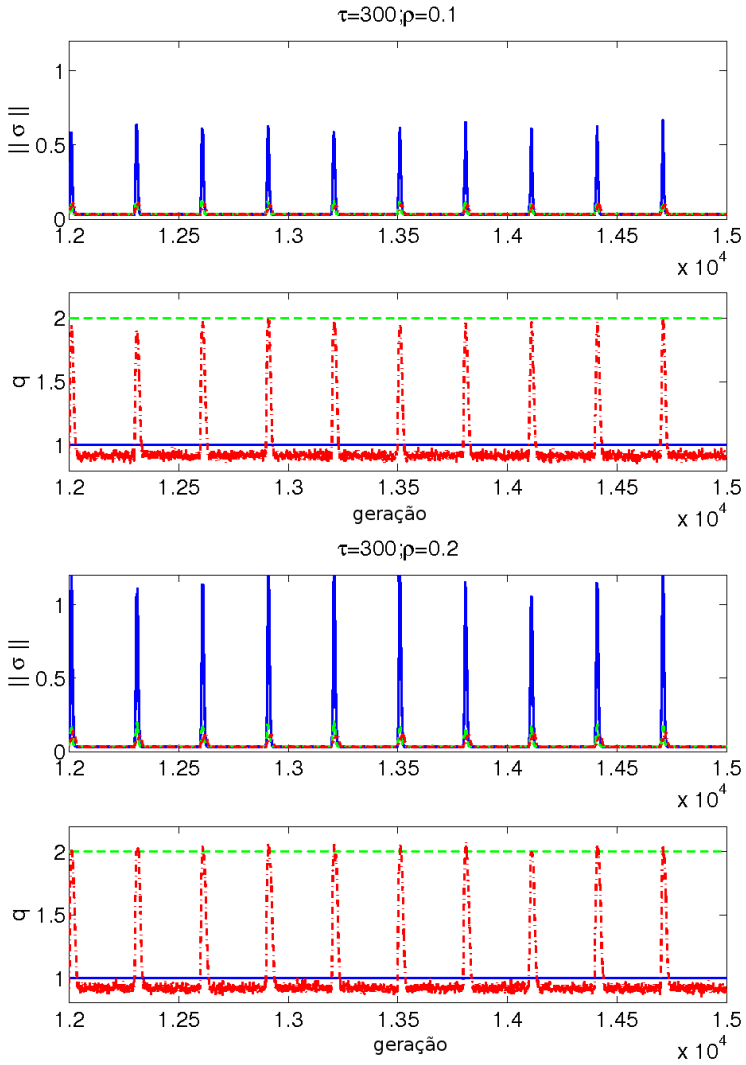

Figura 8.4: Média da norma Euclidiana do vetor de parâmetros de força de mutação e do parâmetro de distribuição q do melhor indivíduo nos últimos 10 ciclos de mudança do experimento com $\tau = 300$ e $\rho = 0,1$ ou $\rho = 0,2$ (GES: linha sólida azul; CES: linha tracejada verde; qGES: linha traço-ponto vermelha).

Parte III

Aspectos Teóricos

9 *Gerador de Problemas Dinâmicos Contínuos Através de Rotação*

Geradores de problemas dinâmicos permitem testar e comparar diferentes AEs em ambientes em que as propriedades, como severidade das mudanças, podem ser diretamente controladas. Desta forma, geradores de problemas dinâmicos têm sido cada vez mais usados. Para criar problemas dinâmicos binários, o Gerador de DOPs XOR, apresentado na Seção 3.3, é com certeza o mais utilizado. A razão disso é que o Gerador XOR permite criar DOPs a partir de qualquer problema de otimização binário estacionário, o que faz com que os AEs investigados possam ser testados em problemas dinâmicos com as mesmas propriedades presentes no problema estacionário. Em [83], Tinós e Yang apresentam dois geradores para problemas contínuos que possuem esta mesma característica. Um destes geradores é aqui apresentado. Mas antes, uma análise dos DOPs produzidos pelo Gerador XOR é apresentada.

9.1 Análise do Gerador de DOPs Binários XOR

A principal característica do Gerador de DOPs XOR, apresentado na Seção 3.3, é que cada indivíduo $\mathbf{x}$ da população corrente é movido para uma nova posição antes de ser avaliado no ciclo de mudança e (ver Eq. 3.3). A nova posição é calculada de acordo com a regra:

$$\mathbf{z}(e) = \mathbf{x} \oplus \mathbf{m}(e) \tag{9.1}$$

na qual $\oplus$ é o operador XOR e $\mathbf{m}(e)$ é a máscara binária para o ciclo de mudança e, a qual é incrementada de acordo com a Eq. 3.4. Se o vetor $\mathbf{x} \in [0,1]^l$ é normalizado como $\mathbf{x^n} \in [-1,1]^l$, a transformação representada pela Eq. 9.1 pode ser escrita como:

$$\mathbf{z^n}(e) = \mathbf{A}(e)\mathbf{x^n}, \tag{9.2}$$

sendo $\mathbf{z^n}(e) \in [-1,1]^l$ o vetor normalizado de $\mathbf{z}(e) \in [0,1]^l$, e a transformação linear $\mathbf{A}(e)$ dada por:

$$\mathbf{A}(e) = \begin{bmatrix} A_1(e) & 0 & \dots & 0 \\ 0 & A_2(e) & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & A_l(e) \end{bmatrix}, \tag{9.3}$$

na qual:

$$A_i(e) = \begin{cases} 1 & \text{se } m_i(e) = 0 \\ -1 & \text{se } m_i(e) = 1 \end{cases} \tag{9.4}$$

para $i = 1, \dots, l$, sendo $m_i(e)$ o i-ésimo elemento do vetor $\mathbf{m}(e)$.

Pode ser observado que a matriz $\mathbf{A}(k)$ é ortogonal, i.e., $\mathbf{A}(e)^{\mathrm{T}}\mathbf{A}(e) = \mathbf{A}(e\mathbf{A}(e)^{\mathrm{T}} = \mathbf{I}$, sendo que $\mathbf{A}(e)^{\mathrm{T}}$ denota a transposta de $\mathbf{A}(e)$ e $\mathbf{I}$ a matriz identidade. A transformação gerada pela matriz ortogonal $\mathbf{A}(e)$ equivale a uma rotação do vetor (indivíduo) no espaço.

Desta forma, podemos observar as seguintes propriedades no Gerador XOR analisando-se as Eqs. 9.1 and 9.2:

- Os indivíduos da população são rotacionados de modo que o i-ésimo gene de cada indivíduo seja alterado quando $m_i(e) = 1$. A rotação é equivalente a mudar a direção do i-ésimo eixo do espaço de fitness;
- As magnitudes dos vetores (indivíduos) são preservadas após as mudanças;
- As distâncias entres os indivíduos da população corrente permanecem inalteradas após a mudança (transformação) imposta pela Eq. 9.1, preservando assim as propriedades da população antes da mudança;
- As propriedades da superfície de fitness não são alteradas após a mudança ambiental.

Baseado nesta interpretação geométrica, fica fácil desenvolver um gerador para problemas dinâmicos contínuos baseados nas propriedades do Gerador XOR.

9.2 Um Gerador de DOPs Contínuos com Controle de Rotação dos Indivíduos

Dado um problema estacionário contínuo com função de fitness $f(\mathbf{x})$ e com cada elemento do vetor $\mathbf{x}$ sendo $x_i \in \{x_i^{min}, x_i^{max}\}$ para $i = 1, \dots, l$, a nova função de fitness $f(\mathbf{x}, e)$ do ambiente

que é alterado a cada τ gerações é dada por:

$$f(\mathbf{x}, e) = f(\mathbf{z}(e)), \tag{9.5}$$

na qual o vetor $\mathbf{z}(e)$ é obtido aplicando a operação contrária à normalização do vetor:

$$\mathbf{z^n}(e) = \mathbf{A}(e)\mathbf{x^n}, \tag{9.6}$$

sendo $\mathbf{x^n} \in [-1,1]^l$ dado pela normalização do vetor $\mathbf{x}$ e $\mathbf{z^n}(e) \in [-1,1]^l$. Pode-se observar que diferentes matrizes $\mathbf{A}(e)$ podem ser utilizadas na Eq. 9.6. De modo a manter a propriedade da invariância das distâncias entre os indivíduos após a mudança do espaço, a matriz $\mathbf{A}(e)$ deve ser ortogonal. Assim, uma escolha natural é uma matriz de rotação composta por múltiplas matrizes simples de rotação. É importante observar que a idéia de se utilizar matrizes de rotação para gerar problemas dinâmicos não é nova [84].

Uma rotação simples definida pela matriz $\mathbf{R}_{ij}$ é obtida pela rotação da projeção de $\mathbf{x}$ no plano $i-j$ por um ângulo θ a partir do i-ésimo eixo para o j-ésimo eixo. Depois da rotação, os elementos do vetor $\mathbf{x}$ permanecem fixos, com exceção dos i-ésimo e j-ésimo elementos. A matriz $\mathbf{R}_{ij}$ é obtida por:

- troque o elemento (i,i) da matriz de identidade $\mathbf{I}$ por cos(θ);
- troque o elemento (i,j) da matriz de identidade $\mathbf{I}$ por -sin(θ);
- troque o elemento (j,i) da matriz de identidade $\mathbf{I}$ por sin(θ).
- troque o elemento (j,j) da matriz de identidade $\mathbf{I}$ por cos(θ).

Propõe-se em [83] que a matriz de transformação $\mathbf{A}(e)$ seja dada por (por simplicidade, l é considerado par):

$$\mathbf{A}(e) = \mathbf{R}_{r_1 r_2}(e)\mathbf{R}_{r_3 r_4}(e)\ldots\mathbf{R}_{r_{l-1} r_l}(e) \tag{9.7}$$

sendo o vetor $\mathbf{r}^{\mathrm{T}} = [r_1\ r_2 \ldots r_l]$ obtido por uma permutação aleatória do vetor $\mathbf{r}^{\mathrm{T}} = [1\ 2 \ldots l]$ na primeira geração de cada ciclo de mudança e. Em outras palavras, a matriz $\mathbf{A}(e)$ é obtida por sucessivas rotações em planos diferentes.

Definindo $\rho = \theta/180$, sendo θ dado em graus, o parâmetro ρ pode ser utilizado para calcul-car o grau de severidade da mudança do DOP, similarmente ao que acontece no Gerador XOR. Se $\rho = 0,0$, o problema permanece estacionário, enquanto que a mudanças são máximas para $\rho = 1,0$.

9.3 Teste do Gerador de DOPs Contínuos Através de Rotação dos Indivíduos

De modo a testar o gerador proposto, este foi utilizado em [83] para criar DOPs a partir das seguintes funções de fitness estacionárias:

$$f(\mathbf{x}) = \frac{1}{1+h(\mathbf{x})}, \tag{9.8}$$

sendo que $h(\mathbf{x})$ é dado pela função esfera:

$$h(\mathbf{x}) = \sum_{i=1}^{l} (x_i - c)^2 \tag{9.9}$$

ou pela função de Griewank generalizada:

$$h(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{l} (x_i - c)^2 - \prod_{i=1}^{l} \cos\left(\frac{x_i - c}{\sqrt{i}}\right) \tag{9.10}$$

nas quais c é uma costante inteira. Na função multimodal dada pela Eq. 9.10, o número de ótimos locais aumenta exponencialmente com a dimensão do problema. Para esta função, os parâmetros $l = 30$ e $c = 300$ foram utilizados. Já para a função esfera, foram utilizados $l = 30$ e $c = 30$. Os resultados que aparecem em [83] para o gerador de DOPs contínuos através de rotação dos indivíduos são aqui apresentados e discutidos.

A figuras 9.1 e 9.2 mostram os resuldados obtidos por três algoritmos para DOPs produzidos pelo gerador para dois diferentes valores de τ: 50 (ambiente mudando rapidamente) e 1000 (ambiente mudando lentamente). Também variou-se a severidade das mudanças através de três valores de ρ produzindo: mudanças suaves ($\rho = 0,1$), mudanças com variação média ($\rho = 0,5$), e mudanças severas ($\rho = 1,0$). Cada algoritmo foi executado 30 vezes para cada uma das combinações de τ e ρ. Os algoritmos utilizados são o GA Geracional Padrão (*Standard Generational GA* - SGA), o GA com Imigrantes Aleatórios (*Random Immigrants GA* - RIGA) e o GA com Hipermutação (*Hypermutation GA* - HGA), sendo estes dois últimos algoritmos projetados especialmente para ambientes dinâmicos (ver Seção 5.1). Os parâmetros utilizados para cada algoritmo podem ser vistos em [83].

A partir dos resultados obtidos, pode-se observar que os valores médios de fitness geralmente diminuem quando o valor de ρ é aumentado e quando o valor de τ é reduzido, como era de se esperar. Este resultado é mais claro nos experimentos com a função esfera, por esta possuir apenas um ótimo no espaço de fitness. Observa-se também que o RIGA alcança melhores resultados quando as mudanças são severas ($\rho = 1,0$). Este resultado concorda com aqueles

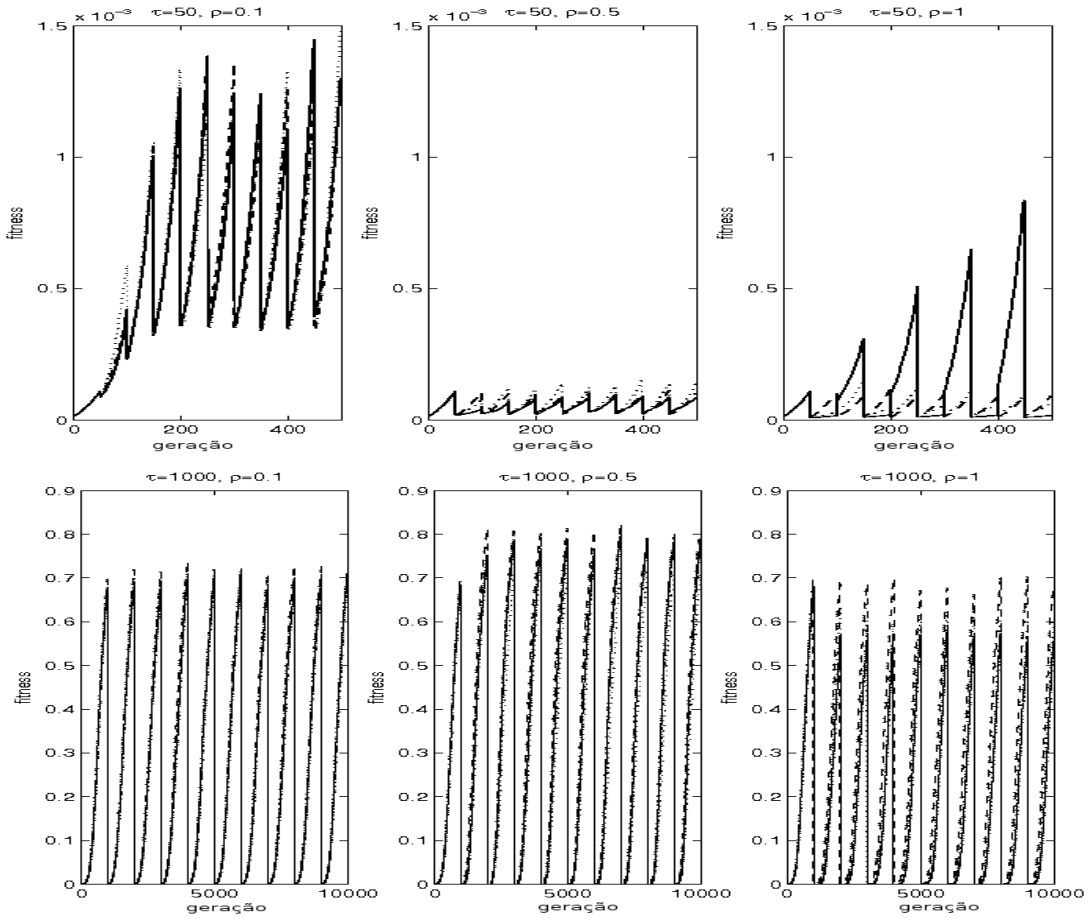

Figura 9.1: Fitness médio (sobre 30 execuções) do melhor indivíduo da população no DOP criado a partir da função esfera (SGA: linha sólida; RIGA: linha tracejada; HGA: linha pontilhada).

apresentados em [39] para outros DOPs. Tal resultado pode ser explicado pelo fato de o RIGA preparar a população para mudanças catastróficas através da manutenção do nível de diversidade através da inserção de imigrantes aleatórios. Observa-se que os melhores resultados são alcançados para valores maiores de τ, já que um maior tempo entre as mudanças permite que o algoritmo tenha mais tempo para explorar as boas soluções inseridas por alguns imigrantes. Contudo, o desempenho do RIGA é pior quando as mudanças são leves, já que menos soluções são geradas na vizinhança da atual melhor solução neste algoritmo. Ainda, observa-se que o HGA é melhor que o SGA em vários problemas, o que atesta o benefício de aumentar a taxa de mutação após a mudança no problema. Entretanto, para mudanças suaves ($\rho = 0,1$), esta estratégia leva a população a afastar-se da melhor solução atual devido à alta taxa de mutação, o que resulta em um pior desempenho para o HGA.

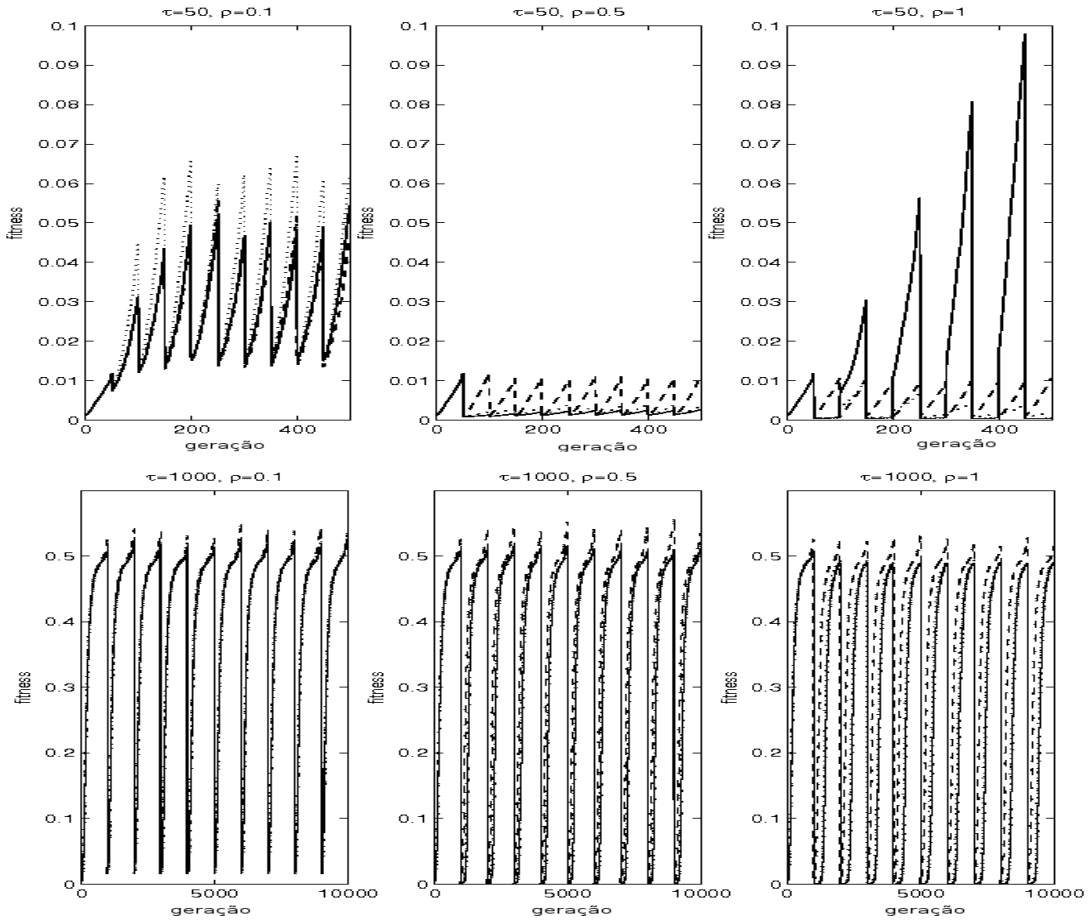

Figura 9.2: Fitness médio (sobre 30 execuções) do melhor indivíduo da população no DOP criado a partir da função de Griewank generalizada (SGA: linha sólida; RIGA: linha tracejada; HGA: linha pontilhada).

9.4 Comentários Finais

No gerador aqui apresentado, semelhantemente ao que ocorre no gerador XOR, os indivíduos são avaliados em posições diferentes daquelas que ocupam no espaço de busca. Esta estratégia possibilita tornar dinâmico diferentes problemas estacionários contínuos, o que permite explorar diferentes características presentes nos problemas. No gerador apresentado, o parâmetro ρ é utilizado para o controle da severidade da mudança, ao passo que o parâmetro τ controla a velocidade em que ocorrem as mudanças.

10 Análise por Sistemas Dinâmicos

A modelagem de GAs como sistemas dinâmicos, desenvolvida principalmente por Vose [14] e cujo modelo é conhecido como *modelo exato*, é uma técnica bastante interessante por permitir uma completa descrição da dinâmica populacional do GA [31]. Em [85], Tinós e Yang propuseram o uso do modelo exato para estudar o GA em ambientes dinâmicos criados pelo Gerador de Problemas Binários Dinâmicos XOR (Seção 3.3). Posteriormente, esta análise foi estendida para problemas dinâmicos criados pelo método descrito em [60] e para o problema da mochila 0-1 dinâmico em [86], e para DOPs criados a partir da simulação de um problema simples de navegação em robótica em [87].

O modelo exato do GA, desenvolvido para problemas estacionários, é apresentada na próxima seção. Na Seção 10.2, define-se formalmente DOPs e classes de problemas dinâmicos tendo como base o modelo exato do GA. Na sequência, dois tipos de DOPs são analisados à luz desta técnica: DOPs produzidos pelo gerador XOR na Seção 10.3 [86]; e DOPs em um problema simples de navegação envolvendo robôs móveis na Seção 10.4 [87].

10.1 Modelo Exato do GA

Em [14], Vose propõe o modelo exato do GA simples no qual este algoritmo é descrito e analisado como um sistema dinâmico discreto. No GA simples, a codificação binária é utilizada, sendo que cada indivíduo da população representa uma solução $\mathbf{x} \in \{0,1\}^l$ do problema. No modelo exato, todas as possíveis soluções candidatas do problema de otimização são representadas em um espaço discreto com $n = 2^l$ dimensões. Assim, a população atual do GA pode ser descrita como um vetor n-dimensional no qual cada elemento define a proporção de cada possível solução na população, ou seja, $\mathbf{p} = \mathbf{v}/N$, sendo que o k-ésimo elemento de $\mathbf{v}$ indica o número de cópias da k-ésima possível solução candidata na população com tamanho N. Como a soma dos elementos de $\mathbf{p}$ é igual a 1, o vetor de população pode ser descrito como membro de

um simplex Λ, i.e.,

$$\Lambda = \left\{ \mathbf{p} \in \mathbb{R}^n : p_k \geq 0, \text{ para } k = 0, 1, \ldots, n-1 \text{ e } \sum_{k=0}^{n-1} p_k = 1 \right\}, \tag{10.1}$$

sendo p_k o k-ésimo elemento do vetor de população $\mathbf{p}$. No simplex Λ, os vértices representam populações com cópias de uma única solução. No modelo exato, a evolução da população de soluções ao longo da execução do GA é descrita como uma trajetória no simplex Λ, sendo que os vetores de população são usados para descrever a distribuição de probabilidades dos indivíduos no espaço de busca. Assim, o operador geracional $\mathcal{G} : \Lambda \to \Lambda$ é definido por $\mathcal{G}(\mathbf{p}) = \mathbf{p}'$, sendo $\mathbf{p}'$ a distribuição de probabilidades amostrada para gerar a população posterior a $\mathbf{p}$, i.e., $\mathcal{G}(\mathbf{p})$ é a próxima população esperada [14]. O vetor $\mathcal{G}(\mathbf{p})$ descreve a média sobre todas as possíveis populações da próxima geração com variância inversamente proporcional ao tamanho da população N, sendo que no limite $N \to \infty$, conhecido como caso de população infinita (modelo exato), a variância se aproxima de zero e, como consequência, a trajetória da população no simplex Λ pode ser deterministicamente descrita. Assim, podemos definir o GA da sequinte forma:

Definição 1 (GA) *O GA é um sistema dinâmico discreto definido pela aplicação sucessiva da regra:*

$$\mathbf{p}(t) = \mathcal{G}(\mathbf{p}(t-1), t), \tag{10.2}$$

sendo $\mathcal{G}(.,t) : \Lambda \to \Lambda$ o operador geracional (mapa) na geração $t \geq 1$, e $\mathbf{p}(t)$ a população esperada em t.

Para o caso estacionário, $\mathcal{G}(.,t) = \mathcal{G}(.)$ para todo $t \geq 1$, fazendo com que a trajetória da população seja dada por $\mathbf{p}(0)$, $\mathcal{G}(\mathbf{p}(0))$, $\mathcal{G}^2(\mathbf{p}(0))$, ... Assim, para o caso de população infinita:

$$\mathbf{p}(t) = \mathcal{G}^t(\mathbf{p}(0)). \tag{10.3}$$

Para o GA simples com mutação e seleção proporcional, o operator geracional é computado por:

$$\mathcal{G} = \mathcal{U} \circ \mathcal{F}, \tag{10.4}$$

sendo o operador de seleção proporcional igual a:

$$\mathcal{F}(\mathbf{p}) = \frac{F\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}} \tag{10.5}$$

na qual $\mathbf{f}$ é o vetor com o fitness de cada solução $\mathbf{x}_i$ pertencente ao espaço χ, e $F = \mathrm{diag}(\mathbf{f})$ é uma matriz diagonal gerada a partir de $\mathbf{f}$. O vetor de fitness $\mathbf{f}$ fornece informação sobre a estrutura

do espaço de busca. Já o operador de mutação é dado por:

$$\mathcal{U}(\mathbf{p}) = U\mathbf{p}, \tag{10.6}$$

sendo U a matriz de mutação. Nesta matriz, cada elemento U_{ij} indica a probabilidade de gerar o i-ésimo elemento de χ (i-ésima solução candidata possível) a partir do j-ésimo elemento deste mesmo espaço. Através das equações 10.4-10.6, é possível escrever o operador geracional como:

$$\mathcal{G}(\mathbf{p}) = \frac{UF\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}. \tag{10.7}$$

A análise da Eq. 10.7 pode fornecer informações importantes para entendermos o comportamento do GA simples. Os pontos fixos de $\mathcal{G}$, i.e., pontos onde $\mathcal{G}(\mathbf{y}) = \mathbf{y}$, são dados pelos autovetores de UF. Para cada autovetor $\mathbf{y}$, um autovalor $\mathbf{f}^{\mathrm{T}}\mathbf{y}$, correspondente ao fitness médio de $\mathbf{y}$, pode ser computado. Como UF tem somente valores positivos, existe um único autovetor dentro de Λ, associado ao autovalor com o maior valor absoluto [31]. Como consequência, todas as trajetórias em Λ convergem para este ponto fixo, i.e., o sistema dinâmico é assintoticamente estável [88, 14]. Os autovetores restantes não são propriamente pontos fixos, já que, por exemplo, podem estar localizados fora do simplex. Contudo, eles representam um papel importante para o processo evolutivo já que podem mudar a trajetória da população no simplex ao criar estados metaestáveis, podendo até aprisionar populações finitas por várias gerações [89].

Uma análise semelhante pode ser feita quando o crossover padrão é adicionado ao operador geracional. Entretanto, neste caso, os estados metaestáveis devem ser obtidos através de um processo mais complicado que envolve a linearização das equações [14]. Por simplicidade, iremos nos ater aqui ao caso do GA simples com mutação e seleção proporcional, i.e., sem crossover.

10.2 DOPs sob o Ponto de Vista do Modelo Exato

A seguir, mudanças e problemas de otimização dinâmica são definidos [86] no contexto do modelo exato do GA.

Definição 2 (Mudança em um GA) *Considere um GA (Def. 1) no qual o operador geracional $\mathcal{G}(.,t)$ na geração t é hiperbólico para todo $t \geq 1$ e possui n_f pontos fixos (estados metaestáveis), i.e., $\mathbf{y}_i(t) = \mathcal{G}(\mathbf{y}_i(t),t)$ para $i = 1,\ldots,n_f$, sendo $\mathbf{y}_i(t)$ o i-ésimo ponto fixo (estado metaestável) de $\mathcal{G}(.,t)$. Uma mudança em um GA ocorre na geração t quando $\mathbf{y}_i(t) \neq \mathcal{G}(\mathbf{y}_i(t-1),t)$ para pelo menos um $\mathbf{y}_i(t)$ para $i = 1,\ldots,n_f$, i.e., pelo menos um ponto fixo de $\mathcal{G}(.,t)$ não é*

preservado.

Pode-se observar que nem todas as modificações no operador geracional $\mathcal{G}$ (Eq. 10.7 para o GA simples sem crossover) podem ser definidas como mudanças de acordo com a Def. 2. Outra importante observação é que nem toda mudança causa necessariamente uma alteração na trajetória das populações. Observa-se que, aqui, consideramos que mudanças somente ocorrem entre duas aplicações consecutivas do operador geracional. Desta forma, DOPs, no contexto de GAs, pode ser definido de acordo com o enfoque por sistemas dinâmicos.

Definição 3 (Problema de Otimização Dinâmica - DOP) *Um Problema de Otimização Dinâmica no contexto de GAs (Def. 1) é um problema de otimização onde pelo menos uma mudança (Def. 2) ocorre durante o processo evolutivo.*

Já que o operador geracional é modificado após uma mudança, a Eq. 10.3 não é mais válida para toda geração t em um DOP. Antes de apresentarmos a nova equação para a dinâmica do GA em um DOP, vamos definir ciclo de mudança.

Definição 4 (Ciclo de mudança) *Ciclo de mudança é uma série de aplicações de um mesmo operador geracional entre duas mudanças consecutivas.*

A duração d_e de um ciclo de mudanças e é o número de gerações consecutivas deste ciclo. Se o ciclo de mudanças e começa na geração t, então

$$\mathcal{G}(.,t) = \mathcal{G}(,t+1) = \mathcal{G}(.,t+2) = \ldots = \mathcal{G}(.,t+d_e-1), \tag{10.8}$$

sendo $d_e > 0$. Em abuso de notação, definimos agora $\mathcal{G}(.,e)$ como o operador geracional no ciclo de mudanças e. Desta forma, para o caso de população infinita, a população na geração t é agora dada por:

$$\mathbf{p}(t) = \mathcal{G}^{(t-\sum_{i=1}^{e-1} d_i)}(.,e) \circ \mathcal{G}^{d_{e-1}}(.,e-1) \ldots \mathcal{G}^{d_2}(.,2) \circ \mathcal{G}^{d_1}(\mathbf{p}(0),1), \tag{10.9}$$

sendo $e > 0$. Nota-se que o DOP pode ser visto como uma sequência de processos estacionários definidos pelos ciclos de mudança, nos quais a população inicial no i-ésimo ciclo de mudança é a população final gerada no ciclo de mudança $i-1$. O valor mínimo de d_i é uma geração, que é o caso em que as mudanças são contínuas, ao passo que o máximo valor de d_i é igual ao índice da geração atual, que é o caso em que o problema é estacionário (até a geração atual) e, como consequência, a Eq. 10.9 reproduz a Eq. 10.3. No GA simples aqui tratado, uma mudança altera pelo menos um dos termos do operador geracional definido pela Eq. 10.4. Geralmente,

mudanças na matriz de mutação U são relacionadas a mudanças no algoritmo, e.g., quando a taxa de mutação é aumentada durante o processo evolutivo. Entretanto, algumas mudanças podem também modificar os operadores de reprodução, e.g., quando existem mudanças nas restrições do problema e algumas soluções não são mais permitidas. Contudo, a comunidade que investiga DOPs no contexto de AEs em geral não está interessada em problemas deste tipo, sendo os DOPs que interessam aqueles em que há mudança na superfície de fitness e que são a seguir definidos.

Definição 5 (DOP com mudanças na superfície de fitness) *Um DOP com mudanças na superfície de fitness é um DOP (Def. 3) no qual a superfície de fitness do problema é alterada por uma mudança (Def. 2) pelo menos uma vez durante o processo evolutivo, i.e.,* $\mathbf{f}(e) \neq \mathbf{f}(e-1)$, *em pelo menos um ciclo de mudança* $e > 1$, *sendo* $\mathbf{f}(e)$ *o vetor de fitness no ciclo de mudança* e.

A Def. 5 é bastante geral, sendo que nem todos os DOPs com mudanças na superfície de fitness têm atraído a atenção da comunidade de AEs. Se a nova superfície de fitness não mantém pelo menos um mínimo de similaridade em relação a superfícies antigas, o melhor procedimento é reiniciar o processo de busca. Entretanto, se a nova superfície de fitness após a mudança pode ser relacionada com superfícies antigas, então o conhecimento adquirido durante o processo de busca das soluções antigas pode de alguma forma ser utilizado na busca pelas novas soluções [34].

Em geral, pesquisadores da área de AE investigam DOPs no qual a função de fitness muda de acordo com regras específicas, estocásticas ou deterministicas. Por exemplo, no gerador *Moving Peaks* apresentado na Seção 3.4, a dinâmica da mudanças da superfície de fitness é governada pela alteração periódica de um conjunto de parâmetros da superfície, como por exemplo a localização e largura de cada pico. Os parâmetros que definem a superfície são agregados aqui em um vetor $\phi(e)$ chamado de vetor de controle do sistema. No gerador *Moving Peaks*, o vetor de controle do sistema $\phi(e)$ é computado em cada ciclo de mudança e através de uma regra específica, tendo como base seus valores passados Por exemplo, o vetor $\phi(e)$ pode ser obtido adicionando-se ao vetor $\phi(e-1)$ um vetor de desvios aleatórios gerados a partir de distribuição Gaussiana. Deste modo, podemos classificar um DOP de acordo com a regra que modifica sua superfície de fitness. Um caso especial de DOPs com mudanças na superfície de fitness que tem atraído a atenção de pesquisadores é o de DOPs com dependência de um único ambiente.

Definição 6 (DOP com dependência de um único ambiente) *Um DOP com dependência de um único ambiente é um DOP com mudanças na superfície de fitness (Def. 5) no qual a super-*

fíce de fitness no ciclo e depende apenas da superfíce no ciclo $e-g$, i.e.,

$$\mathbf{f}(e) = h(\mathbf{f}(e-g), \phi(e)),$$

sendo $g \in \mathbb{N}^+$, $e-g \geq 1$, $h(.)$ uma função real, e $\phi(e)$ o vetor de controle do sistema. As mudanças na superfície de fitness são governadas pela definição da regra de mudança em $\phi(e)$.

É importante observar que esta definição é diferente daquela apresentada para o tipo de problemas apresentado em [27], que aqui é chamado de DOPs com mudanças regulares. Mudando a função $h(.,e)$ na Def. 6, podemos definir outras classes de problemas, como os DOPs periódicos.

Definição 7 (DOP periódico) *Considere DOPs com dependência de um único ambiente (Def. 6) nos quais a superfície de fitness no ciclo e é igual a superfície de fitness no ciclo $e-g$, i.e.,*

$$\mathbf{f}(e) = \mathbf{f}(e-g),$$

sendo $g \in \mathbb{N}^+$ e $e-g \geq 1$. Tal DOP é chamado de DOP periódico.

Em DOPs periódicos (Def. 7), as mudanças são determinísticas e previsíveis. Como consequência, enfoques baseados em memória de soluções anteriores (Seção 5.2) podem ser empregados com sucesso. Outro caso de DOPs com dependência de um único ambiente (Def. 6) é quando $g = 1$.

Definição 8 (DOP com dependência do último ambiente) *Considere um DOP com mudanças na superfície de fitness (Def. 5) no qual a superfície de fitness no ciclo de mudança e seja dependente somente da superfície de fitness no ciclo $e-1$, i.e.,*

$$\mathbf{f}(e) = h(\mathbf{f}(e-1), \phi(e)),$$

sendo $e > 1$, $h(.)$ uma função real, e $\phi(e)$ o vetor de controle do sistema. Tal DOP é chamado DOP com dependência do último ambiente (ou DOP de primeira ordem).

Vários tipos de DOPs com dependência do último ambiente podem ser definidos, entre eles, os DOPs lineares homogêneos.

Definição 9 (DOP linear homogêneo) *Um DOP linear homogêneo é um DOP com dependência do último ambiente (Def. 8) no qual a superfície de fitness no ciclo de mudança $e-1$ é*

modificada de acordo com uma transformação linear homogênea, i.e.,

$$\mathbf{f}(e) = \mathbf{A}(\phi(e))\mathbf{f}(e-1),$$

na qual $\mathbf{A}(.) \in \mathbb{R}^{n \times n}$.

Se o sistema linear é homogêneo e a matriz $A(\phi(e))$ é uma matriz de permutação, outro caso especial de DOP pode ser definido.

Definição 10 (DOP com permutação) *Um DOP com permutação é um DOP linear homogêneo (Def. 9) no qual a superfície de fitness no ciclo de mudança* $e-1$ *é modificada de acordo com uma matriz de permutação, i.e.,*

$$\mathbf{f}(e) = \sigma(\phi(e))\mathbf{f}(e-1),$$

sendo $\sigma(\phi(e))$ *uma matriz de permutação definida pelo vetor de controle do sistema* $\phi(.)$ *no ciclo e. A matriz de permutação mapeia os elementos do vetor* $\mathbf{f}(e-1)$ *para os elementos dos vetor* $\mathbf{f}(e)$.

Em um DOP com permutação (Def. 10), os valores de fitness são preservados no espaço de busca, i.e., o vetor de fitness é apenas reordenado.

10.3 DOPs Produzidos pelo Gerador XOR

Em [85], os autores apresentam um estudo sobre DOPs criados através do gerador de problemas binários dinâmicos XOR [13], que permite tornar dinâmico qualquer problema estacionário de otimização binária e foi apresentado aqui na Seção 3.3. Este gerador tem sido bastante utilizado para comparar o desempenho de diferentes AEs em DOPs, de modo que a descrição das propriedades dos ambientes produzidos pelo gerador XOR é de fundamental importância para entendermos os resultados obtidos pelos algoritmos.

Como apresentado na análise feita na Seção 9.1, nos ambientes criados pelo gerador XOR, cada indivíduo da população atual é movido para uma nova posição do espaço de fitness antes de ser avaliado. Assim, ao invés de avaliarmos o indivíduo na posição $\mathbf{x}$, seu fitness é avaliado na posição $\mathbf{x} \oplus \mathbf{m}(e)$. Pode-se constatar [85] que o gerador XOR produz um tipo especial DOP com permutação (Def. 10), no qual as permutações no espaço de fitness são regidas pela seguinte regra:

$$\mathbf{f}(e) = \sigma_{\mathbf{r}(e)}\mathbf{f}(e-1), \tag{10.10}$$

sendo $\sigma_{\mathbf{r}(e)}$ a matriz de permutação no ciclo e, o qual mapeia o elemento na posição $\mathbf{i}$ do vetor $\mathbf{f}(e-1)$ para o elemento na posição $\mathbf{i} \oplus \mathbf{r}(e)$ do vetor $\mathbf{f}(e)$. O vetor $\mathbf{i} \in \{0,1\}^l$ indica a posição do elemento no vetor de fitness. Já o vetor $\mathbf{r}(e) \in \{0,1\}^l$ controla a permutação dos elementos do vetor de fitness (veja Eq. 3.4), i.e., $\mathbf{r}(e)$ é o vetor de controle do sistema ($\phi(e)$) na Def. 10. Matrizes de permutação $\sigma_{\mathbf{r}(e)}$ têm as seguintes propriedades:

i) $\sigma_{\mathbf{r}(e)}$ são matrizes simétricas;

ii) $\sigma_{\mathbf{r}(i)}\sigma_{\mathbf{r}(j)} = \sigma_{\mathbf{r}(i)\oplus\mathbf{r}(j)}$;

iii) $\sigma_{\mathbf{r}(e)}$ comutam e são iguas a respectiva inversa.

Como consequência, o i-ésimo autovetor $\mathbf{y}_i(e)$ de $U(e)F(e)$ no ciclo de mudança e, o qual define um estado metaestável, pode ser obtido pela permutação do respectivo autovetor para o ambiente no ciclo de mudança $e-1$. Além disso, os autovalores de $U(e)F(e)$ para dois ambientes definidos nos ciclos $e-1$ e e são iguais, o que implica que o fitness médio no estado mestaestável permanece inalterado após as mudanças no problema.

Ainda, verifica-se que o sistema dinâmico do GA com mutação e seleção proporcional em um DOP com permutação (Def. 10) governado pela Eq. 10.10, como naqueles produzidos pelo gerador XOR, é similar ao sistema dinâmico do mesmo GA em um ambiente estacionário no qual a população é alterada de acordo com as mesmas matrizes de permutação usadas no DOP com permutação em cada d_e gerações, sendo d_e a duração do ciclo e no DOP com permutação [85]. Como consequência, o gerador XOR pode ser simplificado: ao invés de computarmos o fitness de cada indivíduo da população em cada nova posição $\mathbf{x} \oplus \mathbf{m}_e$ em cada geração, cada indivíduo da população inicial de cada ciclo de mudança e deve ser movido para a posição $\mathbf{x} = \mathbf{x} \oplus \mathbf{r}_e$, i.e., a população é movida apenas uma vez e o fitness continua a ser avaliado em $f(\mathbf{x})$, como no problema estacionário, o que reduz significativamente a complexidade computacional.

Além desta análise, a dinâmica do GA no DOP pode ser melhor entendida quando analisamos o comportamento do vetor de população, especialmente em relação aos estados metaestáveis do sistema dinâmico do GA. Apresentamos aqui simulações para a evolução da população um DOP criado pelo gerador XOR a partir de um problema estacionário com dois ótimos definido por:

$$f(\mathbf{x}) = \begin{cases} l, & \text{se } u(\mathbf{x}) = l \\ (l-1) - u(\mathbf{x}), & \text{caso contrário,} \end{cases} \tag{10.11}$$

sendo $u(\mathbf{x})$ a função unitária do vetor binário $\mathbf{x}$ de tamanho l, a qual computa o número de valores iguais a 1 no vetor. Nas simulações apresentadas, sete valores de τ (de $\tau = 10$ até $\tau = 70$)

e sete valores de ρ (de $\rho = 0,125$ até $\rho = 0,875$) são considerados. Assim, 49 simulações do sistema dinâmico do GA foram geradas, uma para cada par de τ e ρ, considerando-se 30 ciclos de mudança.

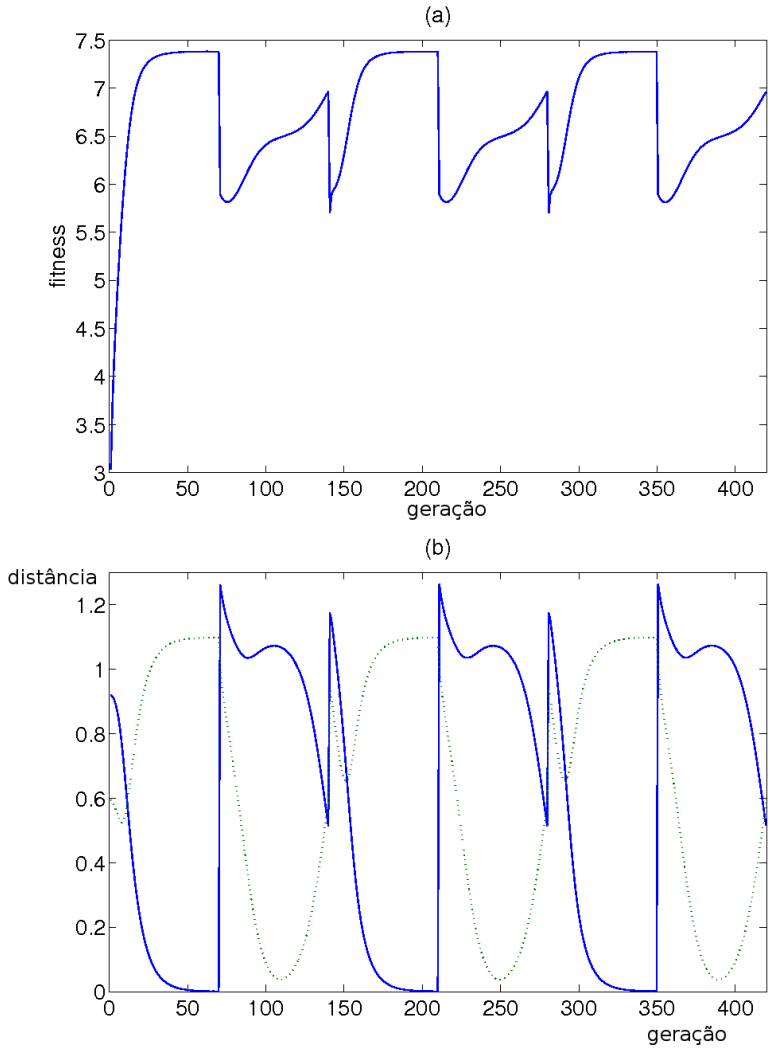

Figura 10.1: Trajetória da população durante 6 ciclos de mutança para $\tau = 70$ e $rho = 0,875$ no DOP criado pelo gerador XOR: (a) fitness médio da população; (b) distância para os atuais estados metaestáveis principal (linha sólida) e secundário (linha pontilhada).

A Figura 10.1 mostra a simulação com $\rho = 0,875$ e $\tau = 70$, na qual o fitness médio da população durante a evolução e a distância Euclidiana entre o o vetor de população na geração corrente e os dois autovetores associados com os maiores autovalores são apresentadas. O primeiro autovetor é relacionado com o principal estado mestaestável corrente (neste estado, o número de indivíduos da população no ótimo global é maior que o número de indivíduos em qualquer outra posição), enquanto que o segundo autovetor é relacionado com o estado metaestável com o segundo maior autovalor (no qual o número de indivíduos no ótimo local é maior que o número de indivíduos em qualquer outra posição). Pode-se observar que, em

alguns ciclos de mudança, a população sai da vizinhança do segundo estado mestaestável e, depois de algumas gerações, alcança a vizinhança do estado metaestável principal.

A Figura 10.2 apresenta os resultados de todas as simulações. A partir desta figura, algumas observações podem ser feitas. Quando τ é próximo de 70 gerações, i.e., em ambientes que mudam lentamente, a população sempre alcança o estado metaestável principal depois de mudanças com baixa severidade (ρ). Quando τ é grande, existe tempo suficiente para que a população vá da vizinhança do estado mestaestável principal (no qual a maioria da população está no ótimo global) no ciclo de mudança $e-1$ para a vizinhança do principal estado metaestável principal no ciclo e. Como consequência, o fitness no final de cada ciclo de mudança é maior quando τ é grande e ρ é pequeno. No gerador XOR, o parâmetro ρ controla a porcentagem de bits mudados do template $\mathbf{m}_{e-1}$ para o template $\mathbf{m}_e$. Assim, a distância Hamming entre $\mathbf{m}_{e-1}$ e $\mathbf{m}_e$ é $h(\mathbf{m}_e, \mathbf{m}_{e-1}) = \lfloor \rho \times l \rfloor$, e maiores valores de ρ implicam maiores distâncias Hamming entre os ótimos em dois ciclos de mudanças consecutivos, resultando em trajetórias mais longas do vetor de população no simplex e, consequentemente, mais tempo para alcançar a vizinhança do estado metaestável principal.

Contudo, pode-se observar que valores maiores de severidade da mudança na superfície de fitness (valores maiores de ρ) não implicam necessariamente em desempenho pior do GA em um DOP com valores médios e pequenos de τ. Observa-se que nestes casos, as simulações com $\rho = 0,375$ apresentam desempenho pior que para valores maiores de ρ. Este comportamento pode ser encontrado em experimentos com o gerador XOR para diferentes algoritmos (como exemplo, veja [90]). O desempenho do GA é relacionado com trajetórias da população no simplex, e estas trajetórias são relacionadas com o vetor de fitness e os operadores de transformação e seleção do GA. Em ambientes que mudam com velocidade média ou rápida, geralmente, quando a população alcança a vizinhança do estado metaestável principal no ciclo de mudança $e-2$, a população depois da mudança está mais perto do segundo estado metaestável principal no ciclo de mudança seguinte quando ρ é grande. Neste caso, a população não tem tempo suficiente para estar mais perto da vizinhança do novo estado metaestável principal no ciclo $e-1$ que da vizinhança do antigo estado metaestável principal. Contudo, quando o problema muda novamente, a população está mais perto da vizinhança do estado mestaestável principal no ciclo de mudança e para ρ perto de 1. A distância Hamming média do template $\mathbf{m}_e$ entre dois ciclos de mudança, a qual é dada por $\overline{h}(\mathbf{m}_e, \mathbf{m}_{e-2}) = 2l(\rho - \rho^2)$, explica o comportamento do GA neste caso. Pode ser observado que o fitness médio geralmente alterna entre dois valores diferentes para valores maiores de ρ e para valores pequenos e médios de τ. Pode-se observar que os valores de distância na Figura 10.2 são maiores para ρ próximos de um que para ρ próximo de zero, já que em parte dos ciclos de mudança, a população permanece na

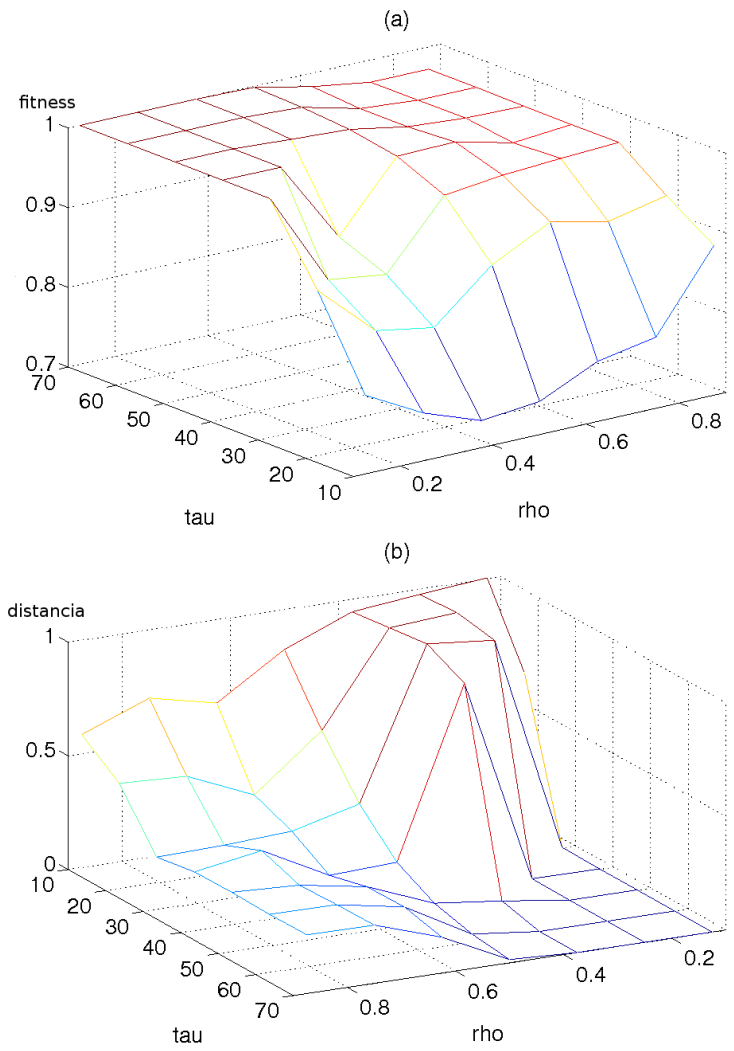

Figura 10.2: Fitness normalizado (a) e distância para o estado metaestável principal (b) no DOP criado pelo Gerador XOR para diferentes valores de τ e ρ. Os valores são relativos à média (sobre 30 ciclos de mudança) obtidos pelo vetor de população na geração antes da mudança.

vizinhança do segundo estado metaestável principal para maiores valores de ρ.

Duas observações podem ser feitas a partir da análise prévia. Primeiro, um grau maior de modificação da superfície (ρ) não implica necesariamente em um pior desempenho do GA. Este resultado pode ser observado em diversos experimentos com o gerador XOR. O desempenho do GA é relacionado com trajetórias da população no simplex, o que torna mais complexa a análise do desempenho do algoritmo.

Segundo, as métricas utilizadas para comparar os algoritmos em DOPs podem não ser adequadas em alguns problemas. Por exemplo, no problema investigado aqui, um algoritmo que mantém a população perto do segundo mestaestável principal para valores grandes de severidade das mudanças em ambientes que mudam rapidamente pode ter um valor de fitness médio maior que em um algoritmo que permite a população escapar do ótimo local, mas não tem

tempo suficiente para alcançar a vizinhança do estado metaestável principal.

10.4 DOPs em um Problema Simples Envolvendo Robôs Móveis

Um possível questionamento acerca do gerador XOR é sobre a relação dos ambientes gerados com DOPs reais. Para isso, deve-se questionar se DOPs reais podem ser do tipo DOP com permutação, como definido anteriormente. Em [87], o enfoque por sistema dinâmicos é utilizado para investigar o comportamento populacional de um GA aplicado em um DOP real: o controle de um robô móvel simples em um ambiente dinâmico. A escolha de uma aplicação simples envolvendo robôs móveis é justificada pela necessidade de uma descrição completa da dinâmica populacional do GA. Salienta-se que este é um trabalho teórico, sendo que, ao invés de executarmos o GA, este é simulado através de seu modelo exato. Isso permite estudarmos o comportamento dinâmico da população do GA, ajudando na análise do desempenho obtido na prática.

10.4.1 Robô Móvel Simulado

Aqui, um robô móvel com um único sensor frontal é simulado em um ambiente com 4 paredes. Considera-se que o robô possa ocupar 9 posições (quadrados) deste ambiente. O sensor frontal produz um sinal igual a $I = 1$ caso o robô esteja de frente para uma parede, e $I = 0$ caso contrário. O objetivo é encontrar através do GA uma lei de controle que permita o robô navegar pelo ambiente sem que se choque com as paredes durante o período de 10 iterações. O robô é controlado por uma máquina de estado finito (autômato) com l bits que determinam o que o robô deve fazer para cada possível condição de estado interno e entrada proveniente do sensor. Dois modelos são aqui estudados, um com $l = 4$ bits e outro com $l = 8$ bits.

No primeiro modelo, o robô pode executar apenas duas ações: seguir em frente (0), ou seja mover-se para próxima posição localizada em sua frente; ou girar no sentido horário (1), ou seja, mudar a direção sem que a posição seja alterada. Neste caso, além do sinal do sensor (I), o robô possui um bit de estado (S), ou memória interna, que indica se o último movimento foi uma rotação ($S = 0$) ou não ($S = 1$). Portanto, o autômato tem 4 bits que definem a ação para cada possível combinação de estado/posição, ou seja, (I, S) = { 0,0 ; 0,1 ; 1,0 ; 1,1 }. Por exemplo, se o autômato é dado por $\mathbf{x} = [0,0,0,1]^{\mathrm{T}}$, o robô irá girar apenas quando ele seguiu em frente na última iteração e encontrou uma parede em frente da posição atual. Assim, o espaço

de busca é composto por apenas $n = 16$ possíveis soluções. Já no segundo modelo, 4 ações são permitidas: ficar parado (00), girar no sentido horário (01), girar no sentido anti-horário (10) e seguir em frente (11). Agora, o autômato tem 8 bits, sendo dois para cada um das 4 combinações de posição/estado. Portanto, o espaço de busca é composto por $n = 256$ soluções possíveis.

Em ambos os modelos, o fitness é dado pelo número de posições ocupadas pelo robô (ou seja, o número de vezes que ele segue em frente mais 1) até que este se choque com uma parede ou, em caso contrário, até um limite de 10 iterações. Como o robô sempre começa em uma mesma posição e orientação (na primeira posição e voltado para a direita), o máximo valor de fitness que pode ser alcançado é 8, pois ele deve girar (sem sair da posição) pelo menos 3 vezes. O objetivo deste estudo é investigar o sistema dinâmico do GA simples (sem crossover) aplicado para buscar os autômatos que geram o maior valor de fitness. Como estamos interessados em DOPs, mudanças no problema foram introduzidas através da simulação de 3 tipos de falhas nas leituras do sensor.

Na primeira falha (falha 1), o sensor do robô tem os sinais zerados, ou seja, independentemente de haver um obstáculo a sua frente ou não, o sensor fornecerá sinal igual a 0. Este tipo de falha pode ocorrer, por entre outros motivos, pelo mal-funcionamento do sensor ou por um rompimento dos cabos que ligam o sensor ao micro-controlador responsável pelo controle do robô. No segundo tipo de falha (falha 2), o inverso ocorre, ou seja, o sensor sempre fornecerá sinal igual a 1, o que pode ocorrer em caso de um curto-circuito. Já no terceiro tipo de falha (falha 3), o sinal proveniente do sensor é invertido, ou seja, quando existe um obstáculo em sua frente, o sinal é igual a $I = 0$, sendo enviado um sinal $I = 1$ caso contrário. Este tipo de falha pode ocorrer devido a mal-funcionamento do sensor ou das portas de entrada do micro-controlador.

10.4.2 Análise do Sistema Dinâmico do GA

Para entendermos como as mudanças provocadas pelas falhas afetam o comportamento dinâmico do GA, devemos investigar como a superfície de fitness é alterada na transição dos ciclos de mudança. No caso da falha 1, o sinal de entrada será sempre $I = 0$, o que resulta no fato de as combinações de estado/posição serem reduzidas a (I,S) = { 0,0 ; 0,1 }. Como consequência, as ações dadas pelo terceiro e quarto elementos do vetor **x** serão respectivamente iguais às ações dadas pelo primeiro e segundo elementos deste mesmo vetor. Efeito similar ocorre no caso da falha 2, na qual o sinal do sensor é sempre $I = 1$. Neste caso, (I,S) = { 1,0 ; 1,1 } e, como consequência, as ações dadas pelo primeiro e segundo elementos do vetor **x** é que serão respectivamente iguais às ações dadas pelo terceiro e quarto elementos deste vetor.

Já no caso da falha 3, na qual o sinal de entrada é invertido, existirá uma permutação entre os elementos do vetor $\mathbf{x}$, sendo que as trocas serão entre o primeiro e terceiro elementos e entre o segundo e quarto elementos. Assim, podemos escrever que, quando ocorre a falha no ciclo de mudança e, supondo que o robô não apresenta falhas no ciclo $e-1$, o vetor $\mathbf{x}(e)$ fica:

$$\mathbf{x}(e) = \mathbf{B}(e)\mathbf{x}(e-1) \tag{10.12}$$

na qual, para o modelo 1 (com $l = 4$), $\mathbf{B}(e)$ é definido respectivamente para as falhas 1, 2 e 3 como:

$$\mathbf{B}_{f1}(e) = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}, \mathbf{B}_{f2}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}, \mathbf{B}_{f3}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}.$$

sendo $\mathbf{0}_2$ uma matriz 2×2 composta apenas por zeros e $\mathbf{I}_2$ uma matriz identidade 2×2. Como consequência da Eq. 10.12, o vetor de fitness no ciclo de mudanças e (com falhas), poderá ser calculado através do vetor de fitness no ciclo de mudanças $e-1$ (sem falhas). Para o caso da falha 1, os valores de fitness para as soluções do espaço de busca nas quais as ações (elementos do vetor $\mathbf{x}$) são iguais para $I = 0$ e $I = 1$ no ciclo $e-1$, substituirão no ciclo e os valores de fitness para as soluções cujas respectivas ações para $I = 0$ são iguais (ou seja, que apresentam a mesma primeira metade do vetor $\mathbf{x}$). O mesmo ocorre para a falha 2, mas agora substituindo as soluções cujas respectivas ações para $I = 1$ são iguais (ou seja, que apresentam a mesma segunda metade do vetor $\mathbf{x}$). Finalmente, para a falha 3, haverá uma permutação entre os elementos do vetor de fitness correspondentes às soluções que apresentam ações (elementos do vetor $\mathbf{x}$) diferentes para $I = 0$ e $I = 1$. Para o restante dos elementos do vetor de fitness (ou seja, elementos que apresentam a primeira e segunda metades do vetor $\mathbf{x}$ iguais), os valores permaneceram inalterados. Assim:

$$\mathbf{f}(e) = \mathbf{A}(e)\mathbf{f}(e-1), \tag{10.13}$$

na qual, para o modelo 1 (com $l = 4$), $\mathbf{A}(e)$ é definido respectivamente para as falhas 1, 2 e 3 como:

$$\mathbf{A}_{f1}(e) = \begin{bmatrix} \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \end{bmatrix}, \mathbf{A}_{f2}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix},$$

$$\mathbf{A}_{f3}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 \\ \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} \\ \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix}.$$

sendo: $\mathbf{M}_a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_b = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_c = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{M}_d = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \mathbf{M}_e = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$

Estas transformações são condizentes com o espaço de fitness do robô para cada uma das condições (ver Figura 10.3, na qual o vetor de $\mathbf{f}$ é apresentado para o modelo 1). Pode-se observar que as melhores soluções para o robô sem falhas são as soluções 1 ($\mathbf{x} = [0,0,0,1]^{\mathrm{T}}$) e 3 ($\mathbf{x} = [0,0,1,1]^{\mathrm{T}}$), ou seja, quando o robô irá girar apenas se ele seguiu em frente na última iteração e encontrou uma parede em frente da posição atual (solução 1), ou quando ele encontrou uma parede em frente da posição atual independente do seu estado (solução 3). Em ambos os casos, o valor de fitness é igual ao valor de fitness máximo (8), sendo que a estratégia adotada pelo robô é navegar em sentido horário nas posições ao lado das paredes. As soluções que apresentam melhor fitness na sequência (soluções 5 e 7), apresentam a estratégia de girar sempre após um movimento para frente. Desta forma, apresentam um fitness igual a 6, já que nesta estratégia o robô gira 5 vezes ao passo que nas estratégias que apresentam fitness máximo, o robô gira por 3 vezes. Nota-se, entretanto, que esta estratégia apresenta fitness máximo quando o robô está com as falhas 1 e 2 pelo fato de não usar o sensor para navegar. Assim, pode-se observar que algumas soluções (as quais a falha no sensor não afeta a estatégia desenvolvida), apresentam o mesmo fitness após as falhas 1 ou 2 no robô, ao passo que outras soluções têm seu fitness alterado de acordo com o exposto anteriormente. Já para o caso da falha 3, observa-se que os valores de fitness são mantidos, sendo apenas reordenados de acordo com a inversão do sinal do sensor (agora, as melhores soluções são as de índice 4 e 12).

De acordo com a Eq. 10.13, pode-se observar que as falhas geram DOPs lineares homogêneos (Def. 9). Além disso, como a matriz $\mathbf{A}_{f3}(e)$ é uma matriz de permutação, a falha 3 gera um DOP com permutação (Def. 10), que apresenta características similares àquelas que ocorrem em DOPs produzidos pelo gerador de DOPs XOR, o que responde a questão sobre DOPs com permutação de fato ocorrerem em problemas do mundo real. No entanto, a matriz de permutação neste caso é gerada de forma diferente. Enquanto que, nos DOPs produzidos pelo gerador XOR, os elementos do vetor $\mathbf{f}$ são reordenados de acordo com a regra $\mathbf{i} \oplus \mathbf{r}(e)$ (Seção 10.3), o que produz uma permutação uniforme, a falha 3 ocasiona uma permutação não-

uniforme de acordo com a matriz $\mathbf{A}_{f3}(e)$, sendo alguns dos elementos reordenados, enquanto outros permanecem fixos. De qualquer forma os valores de fitness são mantidos quando ocorre a falha 3, sendo apenas reordenados.

Como analisado em [85], em um DOP com permutação como o que ocorre no caso do robô com falha 3, o i-ésimo autovetor $\mathbf{y}_i(e)$ de $U(e)F(e)$ no ciclo de mudança e (Eq. 10.7), o qual define um estado metaestável, pode ser obtido pela permutação do respectivo autovetor no ciclo de mudança $e-1$. Assim, qualquer estado metaestável no ciclo de mudança e (robô com falha 3), pode ser obtido através da permutação de um estado metaestável no ciclo de mudança $e-1$ (robô sem falhas), e vice-versa. Além disso, os autovalores de $U(e)F(e)$ para os dois ambientes (robô sem falhas e com falha 3) são iguais, o que implica em valores iguais de fitness médio nos estados metaestáveis. Já nos casos das falhas 1 e 2, a transformação linear do vetor de fitness ocasiona o desaparecimento de alguns valores de fitness antes presentes. Desta forma, os estados metaestáveis são diferentes em cada ciclo de mudança, assim como os seus valores de fitness médio. Estes resultados podem ser observados nas simulações dos modelos do robô apresentadas na seção a seguir.

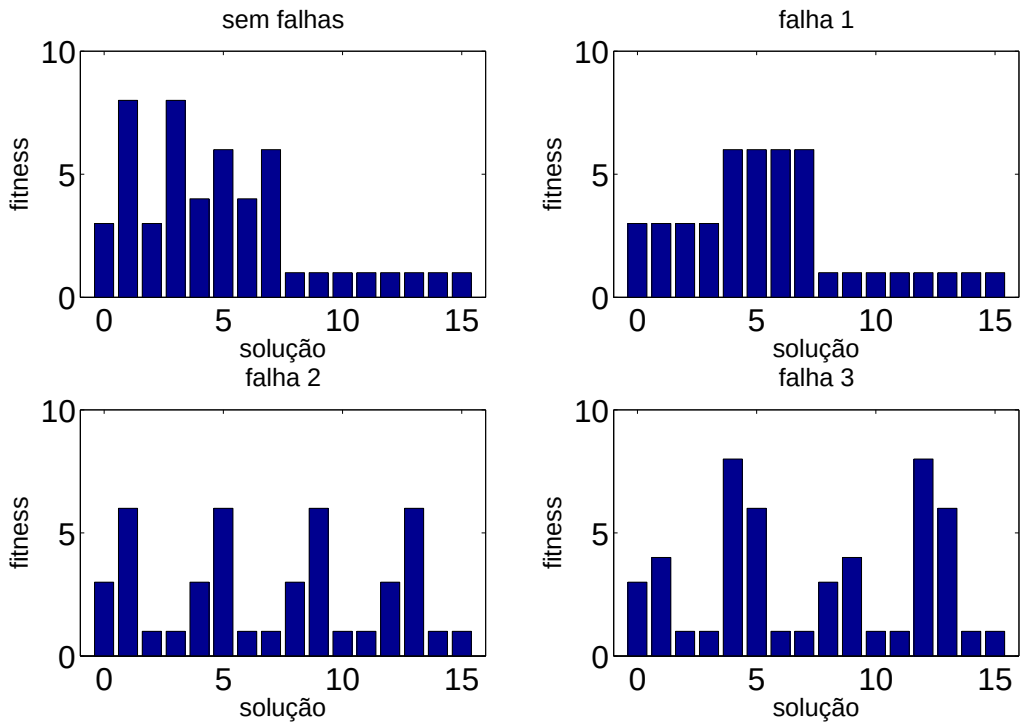

Figura 10.3: Espaço de fitness (vetor **f**) para o modelo 1 ($l = 4$). As soluções são apresentadas de acordo com o valor inteiro de **x**.

10.4.3 Simulações

Nesta seção, as trajetórias das populações do GA com mutação e seleção proporcional quando aplicado no sistema descrito nesta seção são analisadas. Nas simulações aqui apresentadas, ao invés de executarmos o GA, o seu sistema dinâmico é simulado, ou seja, acompanha-se a evolução do vetor populacional dado pela Eq. 10.9. Considera-se que a população inicial é uniformemente distribuída e que a taxa de mutação é igual a $0,01$. Nestas simulações, o sistema dinâmico correspondente ao GA aplicado para controle do robô sem falhas é simulado por $\tau = 50$ (ou seja, a duração do ciclo de mudanças é 50 gerações), ocorrendo um dos três tipos de falhas na sequência. Estas mudanças são equivalentes a alterações no espaço (vetor) de fitness, como considerado na seção anterior. Desejamos observar como os principais estados metaestáveis da população se comportam quando ocorre a mudança no processo evolutivo.

Apresentamos aqui apenas os resultados para as simulações do modelo 2 ($l = 8$). Entretanto, observa-se comportamento similar nas simulações do modelo 1 ($l = 4$). A Figura 10.4 mostra simulações do sistema dinâmico do GA para o modelo 2 ($l = 8$). Nesta figura, o fitness médio da população e a distância euclidiana entre o vetor de população na geração atual e os autovetores que apresentam o maior autovalor em um dado momento da simulação são apresentadas. O primeiro autovetor (o que apresentada maior autovalor) corresponde sempre ao principal estado metaestável do sistema, ou seja, àquele em que o número de indivíduos da população nos ótimos globais é maior do que o número de indivíduos em qualquer outro lugar do espaço. Os outros autovetores correspondem a outros estados metaestáveis que apresentam importância para o entendimento da dinâmica da população.

As simulações apresentadas ajudam a entender o funcionamento do GA. Note que no primeiro ciclo de mudança, quando o robô não apresenta falhas, o vetor da população rapidamente converge para o estado metaestável principal, sendo que, neste caso, grande parte da população encontra-se distribuída entre os dois ótimos globais da superfície de fitness e, como consequência, o fitness médio da população é próximo do fitness máximo permitido. Quando ocorre uma falha, a população encontra-se localizada no principal estado metaestável correspondente ao primeiro ciclo de mudanças. Esta posição é diferente daquela correspondente à do principal estado metaestável no segundo ciclo de mudanças (ou seja, quando o robô apresenta falhas), sendo que o vetor de população deve migrar para esta nova posição. Neste caso, dependendo do tipo de falha, duas situações ocorrem. Na primeira situação, mesmo com a mudança do principal estado metaestável, este ainda é o estado mais perto da posição atual da população (ou seja, da posição do antigo estado mestaestável principal). Desta forma, o vetor da população não tem dificuldades em alcançar o novo estado metaestável principal. Este é o caso da falha 2, na qual

o estados metaestáveis principais antes e depois da falhas encontram-se próximos devido ao fato de uma das soluções que apresenta fitness máximo antes da falha também apresenta fitness máximo depois da falha. Note que, para a falha 2, o fitness médio rapidamente converge para o valor próximo ao fitness máximo (valor igual a 6).

Repare, entretanto, que isso não ocorre para o caso das falhas 1 e 3. Nestes casos, quando ocorre a falha, existem estados metaestáveis mais próximos da posição atual do vetor de população (correspondente à posição do estado metaestável para o ciclo de mudanças 1, ou seja, antes da falha) do que o estado metaestável principal após a falha. Note que, nestes casos, a população primeiro aproxima-se destes estados, ficando um tempo em sua vizinhança, antes de convergir para o estado metaestável principal. Como consequência, nota-se que a população demora mais para convergir para o novo ponto ótimo, fato que pode ser observado nos gráficos do fitness médio da população. Desta forma, as falhas 1 e 3 representam mudanças mais severas do que a falha 1 para este problema de otimização dinâmica. Outro fator importante, que pode ser observado em problemas de otimização dinâmica, é a duração dos ciclos de mudança (d_e). Repare que, para as falhas 1 e 3, no caso das ciclos de mudanças menores (ou seja, nas quais as mudanças são mais frequentes, como no caso de falhas intermitentes que ocorrem com curta duração), o desempenho será mais afetado devido ao que foi explicitado no parágrafo anterior.

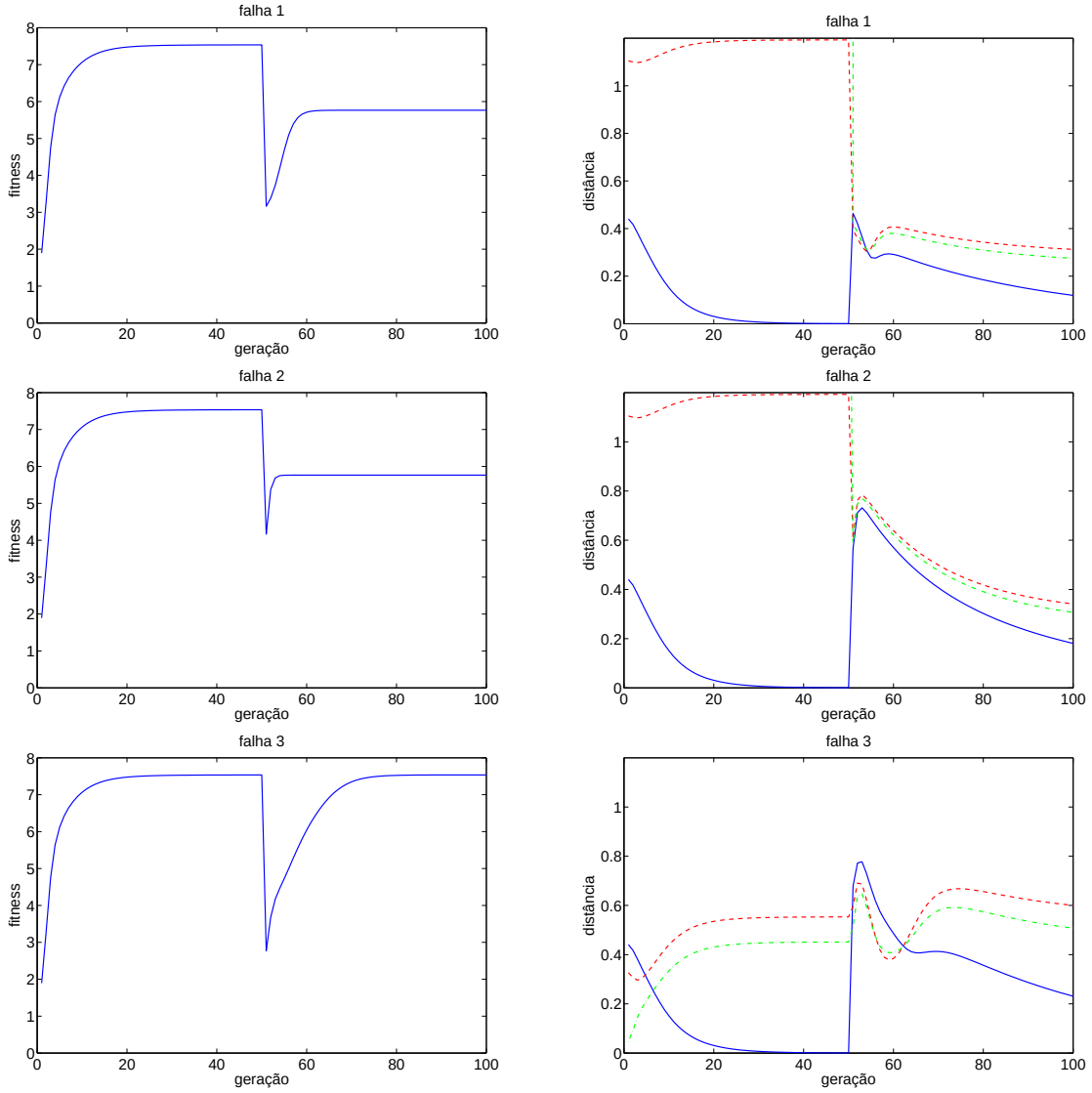

Figura 10.4: Média do fitness da população e distância para três estados metaestáveis em simulações para os três tipos de falhas. A distância para o atual estado metaestável principal é apresentada pela linha sólida.

Parte IV

Conclusão

Diversos estudos têm sido feitos considerando-se AEs em ambientes dinâmicos. Como exemplo do crescente interesse por esta área de pesquisa, pode-se notar a criação de eventos dedicados a esta tema [1]. Esta tese teve como foco principal esta (relativamente) nova área de pesquisa dentro da Computação Evolutiva. O problema de aplicação de AEs em ambientes dinâmicos foi aqui discutido, sendo apresentadas classes de algoritmos, métricas e técnicas teóricas desenvolvidas especialmente para otimização evolutiva dinâmica (Parte I). Vale ressaltar que o número de AEs, e outras meta-heurísticas, desenvolvidos nos últimos para tais problemas é bastante grande, o que demonstra o potencial da área. Aqui foram apresentados com detalhes três AEs para ambientes dinâmicos desenvolvidos pelo autor em trabalhos em que o mesmo é o primeiro autor (Parte II).

O primeiro algoritmo apresentado foi o GAGDM (Capítulo 6), no qual a taxa de mutação é associada individualmente a cada gene. É interessante notar que este algoritmo guarda semelhanças com as Estratégias Evolutivas nas quais cada gene do indivíduo tem associado um parâmetro que controla a força de mutação (quando a mutação Gaussiana é utilizada, este parâmetro equivale ao desvio padrão da distribuição de probabilidades Gaussiana associada ao gene). No entanto, diferentemente destas, no GAGDM as taxas de mutação são ajustadas considerando-se o histórico de mudanças que ocorrem no ambiente. Desta forma, em problemas dinâmicos em que alguns genes mudam mais frequentemente que outros, este algoritmo apresenta um desempenho interessante.

Já no algoritmo SORIGA (Capítulo 7), uma estratégia de inserção de indivíduos aleatórios baseada na interação entre os indivíduos da população é apresentada. Devido a forma com que a interação entre os indivíduos ocorre, a população se auto-organiza, sendo que o tamanho da subpopulação de imigrantes e seus descendentes geralmente é pequeno quando a diversidade da população é alta, ao passo que aumenta a probabilidade de subpopulações maiores quando a diversidade da população é baixa. Esta característica é interessante em problemas de otimização dinâmica de forma a preparar a população para eventuais mudanças, especialmente em problemas altamente multimodais.

O último algoritmo apresentado nesta tese é o AE com mutação q-Gaussiana (Capítulo 8), no qual, assim como no GAGDM, o operador de mutação é modificado. Este tipo de mutação gera saltos aleatórios, nas soluções, que ocorrem com distribuição de probabilidades q-Gaussiana. Nesta distribuição, um parâmetro (q) controla de modo suave e contínuo a forma

[1] Como exemplos: *The European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments* (EvoSTOC), parte integrante dos Workshops EvoStar (www.evostar.org), e *The Special Session on Evolutionary Computation in Dynamic and Uncertain Environments* (ECiDUE), evento integrante do *IEEE Congress on Evolutionary Computation*, ambos já na décima edição.

da distribuição, podendo reproduzir distribuições com caudas mais curtas, como a distribuição Gaussiana usualmente utilizada em AEs, ou com caudas mais longas, como a distribuição de Cauchy. Na mutação q-Gaussiana, o parâmetro q é auto-adaptativo, mudando de acordo com o processo evolutivo. Em ambientes dinâmicos, esta propriedade é interessante já que possibilita aumentar a probabilidade de saltos maiores nas soluções quando o problema muda, o que é útil para fugir de ótimos locais ocasionados pela alteração no espaço de busca. Esta mutação é utilizada em problemas de otimização dinâmica contínua, sendo interessante em problemas multimodais e ruidosos.

Também foram apresentadas nesta tese técnicas teóricas para o estudo de otimização evolutiva dinâmica (Parte III), também desenvolvidas em trabalhos cujo primeiro autor é o autor desta tese. No Capítulo 9, um gerador de problemas de otimização dinâmica contínuos baseado em transformações lineares da população foi proposto. Este gerador é baseado no gerador de problemas binários dinâmicos XOR (Seção 3.3), desenvolvido para problemas binários. A transformação aplicada provoca uma rotação dos indivíduos da população, podendo ser controladas tanto a severidade quanto a frequência das mudanças. Este tipo de alteração nas soluções foi posteriormente empregado no gerador de problemas dinâmicos contínuos GDBG (Seção 3.5), utilizado em competições entre algoritmos evolutivos desenvolvidos para problemas de otimização dinâmica [25, 26].

Já no Capítulo 10, apresenta-se a análise de GAs pelo enfoque dos sistemas dinâmicos, desenvolvida para problemas estacionários por Vose [14], estendida para ambientes dinâmicos. Está análise foi aplicada para estudar o gerador XOR (Seção 3.3), gerando importantes observações acerca dos DOPs produzidos. A análise foi também aplicada em problemas dinâmicos criados pelo método descrito em [60], para o problema da mochila 0-1 dinâmico e para DOPs criados a partir da simulação de um problema simples de navegação em robótica. Recentemente, esta análise foi também aplicada em um outro problema real relacionado com a investigação do comportamento de ratos em um labirinto em cruz elevado [91].

A técnica descrita no Capítulo 10 permite demonstrar que nenhum dos problemas de otimização dinâmica reais analisados é do mesmo tipo daqueles gerados pelo gerador XOR, apesar de alguns deles serem também provocados por permutações das soluções do espaço, que ocorrem, entretanto, de modo não-uniforme, diferentemente daquelas causadas pelo gerador. Desta forma, esta análise leva a conclusão de que um novo gerador de problemas binários, que produz problemas mais semelhantes aos reais, se faz necessário, tema este para pesquisas futuras.

A análise por sistemas dinâmicos também será útil para a investigação do desempenho de algoritmos especialmente desenvolvidos para ambientes dinâmicos. Está é um importante traba-

lho futuro, pois a análise do desempenho de tais algoritmos tem sido feita apenas utilizando-se métodos experimentais. Em um primeiro momento, algoritmos já bastante utilizados, como o GA com hipermutação e a o GA com imigrantes aleatórios (Capítulo 5), devem ser analisados.

Está técnica também poderá ser empregada na análise de algoritmos que apresentam auto-organização como paradigma, e.g., o algoritmo SORIGA. Apesar de não ter sido ainda descrita formalmente para tal, a auto-organização, ao lado do controle determinístico, adaptativo e auto-adaptativo, pode ser vista como um método de controle de parâmetros [92]. Está é uma área em aberto, sendo que novos algoritmos e técnicas de auto-organização podem ser investigadas no futuro.

Uma outra área que demonstra grande potencial é a aplicação de conceitos de otimização em ambientes dinâmicos para a otimização multimodal e/ou multiobjetivos. Nas dissertações de mestrado [93] e [94], orientadas pelo autor, a técnica dos imigrantes aletórios (Seção 5.1) foi aplicada no problema de determinação da estrutura tridimensional de proteínas, que é um problema altamente multimodal. Enquanto que em [93], o algoritmo SORIGA foi utilizado para controlar a diversidade da população, em [94], imigrantes gerados através de similaridade com estruturas já conhecidas foram inseridos na população. Para este mesmo problema, funções de fitness tornadas dinâmicas de forma artificial foram utilizadas [95]. Tornar o problema dinâmico pode ser interessante já que ótimos locais são caracterizados por regiões da superfície de fitness com gradiente igual a zero, que podem tornar-se regiões com gradiente diferente de zero quando o problema muda. Já em outro mestrado [96], a técnica dos imigrantes aleatórios auto-organizados (Capítulo 7) foi incorporada a Sistemas Imunológicos Artificiais aplicados à outro complexo problema da Bioinformática: o *docking* entre proteínas.

De maneira geral, entre as áreas relacionadas ao uso de AEs em ambientes dinâmicos que necessitam de uma maior investigação, podem ser citadas: o estudo dos aspectos teóricos, o desenvolvimento de novas métricas de desempenho, o desenvolvimento de algoritmos utilizado técnicas que relacionem os diferentes indivíduos da população e seus genes, como aquelas utilizadas em Algoritmos de Estimação de Distribuição, a classificação de problemas de otimização dinâmica encontrados em aplicações do mundo real e o estudo de otimização evolutiva dinâmica multi-objetivo.

Referências

[1] S. Haykin, *Redes neurais: princípios e prática*. Bookman, 2nd ed., 2001.

[2] M. Dorigo and T. Stützle, *Ant colony optimization*. Bradford Books, 2004.

[3] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann Publishers, 2001.

[4] L. N. Castro and J. Timmis, *Artificial immune systems: a new computational intelligence paradigm*. Springer-Verlag New York, Inc., 2002.

[5] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer Verlag, 2003.

[6] R. Tinós and A. C. P. L. F. Carvalho, "Evolutionary computation in brazil: a review of the literature in two databases," *Revista de Informática Aplicada*, vol. 4, no. 2, pp. 34 – 69, 2008.

[7] R. Tinós and A. C. P. L. F. Carvalho, "Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems," *Neurocomputing*, vol. 70, no. 1-3, pp. 44–54, 2006.

[8] S. Nolfi and D. Floreano, *Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines*. MIT Press/Bradford Books: Cambridge, USA, 2000.

[9] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

[10] K. Weicker, *Evolutionary algorithms and dynamic optimization problems*. Der Andere Verlag, 2003.

[11] R. W. Morrison, *Designing evolutionary algorithms for dynamic environments*. Springer-Verlag New York Inc, 2004.

[12] S. Yang, Y. S. Ong, and Y. Jin, *Evolutionary computation in dynamic and uncertain environments*. Springer, 2007.

[13] S. Yang and X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Computing*, vol. 9, no. 11, pp. 815–834, 2005.

[14] M. D. Vose, *The simple genetic algorithm: foundations and theory*. The MIT Press, 1999.

[15] E. K. P. Chong and S. H. Zak, *An introduction to optimization*. John Wiley and Sons, Inc., 2a ed., 2001.

[16] Z. Michalewicz and D. B. Fogel, *How to solve it: modern heuristics*. Springer, 2004.

[17] D. A. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc., 1989.

[18] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*. Springer, 3rd ed., 1996.

[19] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. e1, pp. 3 – 17, 1997.

[20] M. Mitchell, *An introduction to genetic algorithms*. MIT Press, 1996.

[21] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[22] H.-G. Beyer, *The theory of evolution strategies*. Springer-Verlag, 2001.

[23] D. B. Fogel, *Evolutionary eomputation: toward a new philosophy of machine intelligence*. IEEE Press, 2nd ed., 1999.

[24] J. E. Smith and F. Vavak, "Replacement strategies in steady state genetic algorithms: dynamic environments," *Journal of Computing and Information Technology*, vol. 7, no. 1, pp. 49–59, 1999.

[25] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for CEC2009 competition on dynamic optimization," tech. rep., University of Leicester, University of Birmingham, Honda Research Institute Europe, Vorarlberg University of Applied Sciences, Nanyang Technological University, 2008.

[26] C. Li, S. Yang, and D. A. Pelta, "Benchmark generator for the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimization problems," tech. rep., China University of Geoscienses, Brunel University, University of Granada, 2011.

[27] C. Ronnewinkel, C. O. Wilke, and T. Martinetz, *Genetic algorithms in time-dependent environments*, pp. 263–288. Springer, 2001.

[28] S. A. Stanhope and J. M. Daida, "Genetic algorithm fitness dynamics in a changing environment," in *Proc. of the 1999 IEEE Cong. on Evolutionary Computation*, vol. 3, pp. 1851–1858, 1999.

[29] P. Rohlfshagen, P. K. Lehre, and X. Yao, "Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change," in *Proc. of the 11th Annual Conf. on Genetic and Evolutionary Computation*, pp. 1713–1720, 2009.

[30] D. Arnold and H. G. Beyer, "Random dynamics optimum tracking with evolution strategies," *Parallel Problem Solving from Nature VII, Lecture Notes in Computer Science*, vol. 2439, pp. 3–12, 2002.

[31] C. R. Reeves and J. E. Rowe, *Genetic algorithms - principles and perspectives: a guide to GA theory*. Kluwer Academic Publishers, 2003.

[32] P. J. Angeline, D. B. Fogel, and L. J. Fogel, "A comparison of self-adaptation methods for finite state machines in dynamic environments," in *Proc. of the 5th Annual Conf. on Evolutionary Programming*, pp. 441–449, 1996.

[33] F. Raman and F. B. Talbot, "The job shop tardiness problem: A decomposition approach," *European Journal of Operational Research*, vol. 69, no. 2, pp. 187–199, 1993.

[34] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments - a survey," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[35] University of Málaga, "Metaheuristics in dynamic environments (http://neo.lcc.uma.es/dynamic/index.html/)," july 2010.

[36] D. Angus and T. Hendtlass, "Dynamic ant colony optimisation," *Applied Intelligence*, vol. 23, no. 1, pp. 33–38, 2005.

[37] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proc. of the 8th Annual Conf. on Genetic and Evolutionary Computation*, pp. 51–58, 2006.

[38] F. O. França and F. J. Von Zuben, "A dynamic artificial immune algorithm applied to challenging benchmarking problems," in *Proc. of the IEEE 11th Cong. on Evolutionary Computation*, pp. 423–430, 2009.

[39] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proc. of the 5th Int. Conf. on Genetic Algorithms* (S. Forrest, ed.), pp. 523–530, 1993.

[40] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the thermodynamical genetic algorithm," *Parallel Problem Solving from Nature IV, Lecture Notes in Computer Science*, vol. 1141, pp. 513–522, 1996.

[41] R. Tinós and S. Yang, "Self-organizing random immigrants genetic algorithm for dynamic optimization problems," *Genetic Programming and Evolvable Machines*, vol. 8, no. 3, pp. 255–286, 2007.

[42] M. M. Gouvêa Júnior, *Algoritmo Evolucionário Adaptativo em Problemas Dinâmicos Multimodais*. PhD thesis, Universidade Federal de Pernambuco, Brasil, 2009.

[43] P. D. Stroud, "Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations," *IEEE Trans. on Evolutionary Computation*, vol. 5, no. 1, pp. 66–77, 2001.

[44] C. L. Ramsey and J. J. Grefenstette, "Case-based initialization of genetic algorithms," in *Proc. of the 5th Int. Conf. on Genetic Algorithms*, pp. 84–91, 1993.

[45] D. E. Goldberg and R. E. Smith, "Nonstationary function optimization using genetic algorithms with dominance and diploidy," in *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, pp. 59–68, 1987.

[46] B. S. Hadad and C. F. Eick, "Supporting polyploidy in genetic algorithms using dominance vectors," vol. 1213, pp. 223–234, Springer-Verlag, 1997.

[47] T. Maeshiro, "The importance of the robustness and changeability in evolutionary systems," pp. 2342–2347, 1998.

[48] J. H. Holland, *Adaptation in natural and artificial systems*. The University of Michigan Press, USA, 1975.

[49] R. Tinós and A. C. P. L. F. Carvalho, "Alteração da probabilidade de mutação do gene em algoritmos genéticos aplicados a problemas não-estacionários," in *Anais do IV Encontro Nacional de Inteligência Artificial*, pp. 1977–1986, 2003.

[50] R. Tinós and A. C. P. L. F. Carvalho, "A genetic algorithm with gene dependent mutation probability for non-stationary optimization problems," in *Proc. of the 2004 IEEE Cong. on Evolutionary Computation*, vol. 2, pp. 1278–1285, 2004.

[51] F. Vavak and T. C. Fogarty, "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments," vol. 1143, pp. 297–304, Springer, 1996.

[52] P. Bak, *How nature works: the science of self-organized criticality*. Oxford University Press, 1997.

[53] H. J. Jensen, *Self-organized criticality: emergent complex behavior in physical and biological systems*. Cambridge University Press, 1998.

[54] D. M. Raup, "Biological extinction in earth history," *Science*, vol. 231, pp. 1528–1533, 1986.

[55] S. A. Kauffman, *The origins of order: self-organization and selection in evolution*. Oxford University Press, 1993.

[56] S. Boettcher and A. G. Percus, "Optimization with extremal dynamics," *Complexity*, vol. 8, no. 2, pp. 57–62, 2003.

[57] M. Løvbjerg and T. Krink, "Extending particle swarm optimisers with self-organized criticality," in *Proc. of the 2002 IEEE Cong. on Evolutionary Computation*, vol. 2, pp. 1588–1593, 2002.

[58] T. Krink and R. Thomsen, "Self-organized criticality and mass extinction in evolutionary algorithms," in *Proc. of the 2001 IEEE Cong. on Evolutionary Computation*, vol. 2, pp. 1155–1161, 2001.

[59] R. Tinós, "Comportamento auto-organizável em algoritmos genéticos aplicados a problemas não-estacionários associados a robôs móveis," *Revista SBA Controle e Automação*, vol. 18, no. 1, pp. 13–23, 2007.

[60] S. Yang, "Constructing dynamic test environments for genetic algorithms based on problem difficulty," in *Proc. of the 2004 Cong. on Evolutionary Computation*, vol. 2, pp. 1262–1269, 2004.

[61] R. Tinós and S. Yang, "Genetic algorithms with self-organizing behaviour in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments* (S. Yang, Y.-S. Ong, and Y. Jin, eds.), pp. 105–127, Springer-Verlag Berlin Heidelberg New York, 2007.

[62] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 26, no. 3, pp. 396–407, 1996.

[63] K. Trojanowski and Z. Michalewicz, "Evolutionary algorithms for non-stationary environments," in *Proc. of the 8th Int. Workshop on Intelligent Information Systems* (M. A. Klopotek and M. Michalewicz, eds.), pp. 229–240, 1999.

[64] H.-G. Beyer and H. S. Schwefel, "Evolution strategies: a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[65] W. Thistleton, J. A. Marsh, K. Nelson, and C. Tsallis, "Generalized Box-Muller method for generating q-Gaussian random deviates," *IEEE Trans. on Information Theory*, vol. 53, no. 12, pp. 4805–4810, 2007.

[66] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 2, pp. 82 – 102, 1999.

[67] C. Y. Lee and X. Yao, "Evolutionary programming using mutations based on the levy probability distribution," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1 – 13, 2004.

[68] A. Obuchowicz, "Multidimensional mutations in evolutionary algorithms based on real-valued representation," *Int. Journal of Systems Science*, vol. 34, no. 7, pp. 469 – 483, 2003.

[69] N. Hansen, F. Gemperle, A. Auger, and P. Koumoutsakos, "When do heavy-tail distributions help?," *9th Int. Conf. on Parallel Problem Solving from Nature (PPSN IX), Lecture Notes in Computer Science*, vol. 4193 LNCS, pp. 62 – 71, 2006.

[70] R. Tinós and S. Yang, "Use of the q-gaussian mutation in evolutionary algorithms," *Soft Computing*, vol. 15, no. 8, pp. 155–168, 2011.

[71] M. Iwamatsu, "Generalized evolutionary programming with levy-type mutation," *Computer Physics Communications*, vol. 147, no. 1-2, pp. 729 – 732, 2002.

[72] M. A. Moret, P. G. Pascutti, P. M. Bisch, M. S. P. Mundim, and K. C. Mundim, "Classical and quantum conformational analysis using generalized genetic algorithm," *Physica A: Statistical Mechanics and its Applications*, vol. 363, no. 2, pp. 260–268, 2006.

[73] M. W. Davis, "The natural formation of gaussian mutation strategies in evolutionary programming," in *Proc. of the 3rd Annual Conf. on Evolutionary Programming*, World Scientific, 1994.

[74] T. Bäck, "Self-adaptation," in *Evolutionary Computation 2: advanced algorithms and operators* (T. Bäck, D. B. Fogel, and Z. Michalewicz, eds.), Institute of Physicis Publishing, 2000.

[75] H. Dong, J. He, H. Huang, and W. Hou, "Evolutionary programming using a mixed mutation strategy," *Information Sciences*, vol. 177, no. 1, pp. 312 – 327, 2007.

[76] S. Umarov, C. Tsallis, and S. Steinberg, "On a q-central limit theorem consistent with nonextensive statistical mechanics," *Milan Journal of Mathematics*, vol. 76, no. 1, pp. 307–328, 2008.

[77] A. M. C. Souza and C. Tsallis, "Student's t- and r-distributions: Unified derivation from an entropic variational principle," *Physica A: Statistical Mechanics and its Applications*, vol. 236, no. 1-2, pp. 52 – 57, 1997.

[78] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[79] R. Tinós and S. Yang, "Self adaptation of mutation distribution in evolution strategies for dynamic optimization problems," *The Int. Journal of Hybrid Intelligent Systems*, vol. 8, no. 3, pp. 1523–1549, 2011.

[80] X. Yao and Y. Liu, "Fast evolution strategies," *Control and Cybernetics*, vol. 26, no. 3, pp. 467–496, 1997.

[81] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real parameter optimization," tech. rep., Nanyang Technological University, 2005.

[82] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, pp. 617–644, 2009.

[83] R. Tinós and S. Yang, "Continuous dynamic problem generators for evolutionary algorithms," in *Proc. of the 2007 IEEE Cong. on Evolutionary Computation*, pp. 236–243, 2007.

[84] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *Proc. of the 2000 Cong. on Evolutionary Computation*, pp. 1125–1131, 2000.

[85] R. Tinós and S. Yang, "An analysis of the xor dynamic problem generator based on the dynamical system," in *Parallel Problem Solving from Nature - PPSN XI* (R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, eds.), vol. 6238 of *Lecture Notes in Computer Science*, pp. 274–283, Springer Berlin / Heidelberg, 2011.

[86] R. Tinós and S. Yang, "Analyzing evolutionary algorithms for dynamic optimization problems based on the dynamical system," in *To appear in the book Evolutionary Computation for Dynamic Optimization Problems* (S. Yang and X. Yao, eds.), pp. 1–25, Springer, 2012.

[87] R. Tinós and S. Yang, "Análise de algoritmos genéticos aplicados a robôs em ambientes dinâmicos via modelo exato," in *Anais do X Cong. Brasileiro de Inteligência Computacional*, 2011.

[88] C. Hayes and T. Gedeon, "Hyperbolicity of the fixed point set for the simple genetic algorithm," *Theoretical Computer Science*, vol. 411, no. 25, pp. 2368–2383, 2010.

[89] E. V. Nimwegen, J. P. Crutchfield, and M. Mitchell, "Finite populations induce metastability in evolutionary search," *Physics Letters A*, vol. 229, no. 3, pp. 144–150, 1997.

[90] S. Yang and R. Tinós, "A hybrid immigrants scheme for genetic algorithms in dynamic environments," *Int. Journal of Automation and Computing*, vol. 4, no. 3, pp. 243–254, 2007.

[91] R. Tinós, "Analysing fitness landscape changes in evolutionary robots," in *In the 1st Understanding Problems Workshop (GECCO-UP), To appear in the Proc. of the Genetic and Evolutionary Computacion Conf. (GECCO)*, 2012.

[92] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[93] V. T. do Ó, "Técnicas de controle da diversidade de populações em algoritmos genéticos para determinação de estruturas de proteínas," Master's thesis, University of São Paulo, Brasil, 2009.

[94] L. L. Oliveira, "Uso de estratégias baseadas em conhecimento para algoritmos genéticos aplicados a predição de estruturas tridimensionais de proteínas," Master's thesis, University of São Paulo, Brasil, 2011.

[95] L. H. U. Ishivatari, L. L. Oliveira, F. L. B. Silva, and R. Tinós, "Algoritmos genéticos com função de avaliação dinâmica para o problema de predição de estruturas de proteínas," in *Anais do X Cong. Brasileiro de Inteligência Computacional*, 2011.

[96] H. K. Shimo and R. Tinós, "Auto-organização da população em sistemas imunológicos artificiais para otimização multimodal contínua," in *Anais do X Cong. Brasileiro de Inteligência Computacional*, 2011.