

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação - FEEC
Departamento de Engenharia de Computação e Automação Industrial - DCA

Otimização em ambientes dinâmicos com variáveis contínuas empregando algoritmos de estimação de distribuição

André Ricardo Gonçalves

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientador: Fernando José Von Zuben

Campinas, SP, Brasil
Abril de 2011

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

G586o Gonçalves, André Ricardo
 Otimização em ambientes dinâmicos com variáveis
 contínuas empregando algoritmos de estimação de
 distribuição / André Ricardo Gonçalves. – Campinas, SP:
 [s.n.], 2011.

 Orientador: Fernando José Von Zuben.
 Dissertação (mestrado) - Universidade Estadual de
 Campinas, Faculdade de Engenharia Elétrica e de
 Computação.

 1. Otimização. 2. Algoritmos Genéticos. 3. Misturas -
 Métodos Estatísticos. 4. Metaheurística. 5. Teoria dos
 sistemas dinâmicos. I. Von Zuben, Fernando José. II.
 Universidade Estadual de Campinas. Faculdade de
 Engenharia Elétrica e de Computação. III. Título

Título em Inglês: Real-parameter optimization in dynamic environments
 using estimation of distribution algorithms
Palavras-chave em Inglês: Optimization, Genetic algorithms, Mixture models,
 Metaheuristics, Dynamic systems
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca Examinadora: Leandro Nunes de Castro Silva, Renato Tinós
Data da defesa: 19/04/2011
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: André Ricardo Gonçalves

Data da Defesa: 19 de abril de 2011

Título da Tese: "Otimização em ambientes dinâmicos com variáveis contínuas empregando algoritmos de estimação de distribuição"

Prof. Dr. Fernando José Von Zuben (Presidente):

Fernando José Von Zuben

Prof. Dr. Leandro Nunes de Castro Silva:

Leandro Nunes de Castro Silva

Prof. Dr. Renato Tinós:

Renato Tinós

Resumo

O dinamismo do mundo moderno traz consigo grandes desafios científicos e tecnológicos, particularmente junto a problemas de otimização. Problemas antes tratados de forma estática estão sendo reformulados para incorporar esse dinamismo, exigindo com isso novas estratégias de solução. Meta-heurísticas populacionais para otimização surgem então como abordagens promissoras, visto que favorecem a exploração do espaço de busca e contribuem para a adaptação ao dinamismo do ambiente. Foram tratados aqui algoritmos de estimação de distribuição (AEDs), os quais empregam modelos probabilísticos para identificar regiões promissoras do espaço de busca. Pelo fato de serem raras e limitadas as propostas de AEDs para problemas dinâmicos, principalmente em espaços de busca contínuos, foram concebidos AEDs baseados em modelos de mistura gaussianos flexíveis, auto-controláveis e com baixo custo computacional, incluindo ainda operadores de manutenção de diversidade e de controle de convergência. Uma extensa comparação com métodos alternativos de otimização para ambientes dinâmicos foi realizada e, em várias situações, a proposta deste trabalho superou o desempenho de métodos considerados estado-da-arte.

Palavras-chave: Otimização em ambientes dinâmicos, algoritmos de estimação de distribuição, modelos de mistura.

Abstract

The dynamism of the modern world gives rise to huge scientific and technological challenges. Problems until recently being treated as static are now being reformulated to incorporate that dynamism, thus requiring novel solution strategies. Population-based metaheuristics devoted to optimization emerge as promising approaches, given that they promote an effective exploration of the search space and contribute to the adaptation to the dynamism of the environment. Estimation of distribution algorithms (EDAs) were considered here, which make use of probabilistic models to identify promising regions of the search space. Due to the fact that the proposals of EDAs for dynamic problems are rare and limited, mainly in real-parameter search spaces, EDAs were conceived based on flexible Gaussian mixture models, self-controlable and computationally inexpensive steps, including diversity maintenance and convergence control mechanisms. An extensive comparison with alternative optimization methods for dynamic environments was accomplished and, in many situations, the proposed technique overcame the performance produced by state-of-the-art methods.

Keywords: Optimization in dynamic environments, estimation of distribution algorithms, mixture models.

Agradecimentos

Primeiramente agradeço a Deus por me dar o dom da vida.

À minha família, principalmente aos meus pais, Vera Lúcia e Lourival Gonçalves, pelo total apoio, servindo como uma base sólida, onde eu posso me sustentar.

Agradeço ao meu orientador, Prof. Fernando José Von Zuben, pela orientação e ensinamentos, tanto técnicos quanto pessoais, que contribuíram para minha formação acadêmica e que levo para toda minha vida.

Agradeço à minha namorada, Vânia, por sua companhia, carinho e incentivo durante esta jornada.

Agradeço aos colegas do Laboratório de Bioinformática e Computação Bioinspirada (LBiC) pelas discussões saudáveis e agregadoras.

Aos vários amigos conquistados nestes dois anos. Pessoas de diferentes lugares e experiências de vida, com as quais tive o prazer de conviver.

Agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro, que foi de extrema importância para a realização deste trabalho.

À minha família.

Sumário

Lista de Figuras	xv
Lista de Tabelas	xvii
Lista de Siglas	xix
1 Introdução	1
1.1 Objetivos da pesquisa	3
1.2 Organização do texto	3
2 Otimização em ambientes dinâmicos	5
2.1 Considerações iniciais	5
2.2 Otimização matemática	5
2.2.1 Caracterização da função-objetivo	8
2.3 Métodos de otimização	9
2.3.1 Métodos tradicionais para espaços discretos	10
2.3.2 Métodos tradicionais para espaços contínuos	11
2.3.3 Métodos heurísticos	14
2.4 Otimização em espaços variantes no tempo	15
2.4.1 Métodos tradicionais para otimização de ambientes dinâmicos	16
2.4.1.1 Métodos variacionais	17
2.4.1.2 Métodos de discretização	17
2.5 Algoritmos evolutivos em ambientes dinâmicos	19
2.5.1 Detecção de alterações no ambiente	20
2.6 Considerações finais	20
3 Estimação de densidade	23
3.1 Considerações iniciais	23

3.2	Métodos de estimação de densidade	24
3.2.1	Histograma	24
3.2.2	Estimação por distribuição de probabilidade paramétrica	25
3.2.3	Métodos de kernel	26
3.2.4	Modelos de mistura	28
3.2.4.1	Estimação dos parâmetros	29
3.2.4.2	Algoritmo EM	31
3.2.4.3	Versão Online do algoritmo EM	33
3.3	Considerações finais	35
4	Algoritmos Evolutivos e Algoritmos de Estimação de Distribuição	37
4.1	Considerações iniciais	37
4.2	Computação Evolutiva	38
4.2.1	Algoritmo Genético	39
4.2.2	Modelagem teórica dos algoritmos genéticos	41
4.3	Computação evolutiva baseada em modelagem probabilística	43
4.3.1	Modelos sem dependências	46
4.3.2	Modelos com dependência aos pares	47
4.3.3	Modelos com múltiplas dependências	48
4.3.4	Modelos de mistura	49
4.4	Considerações finais	50
5	Algoritmos de estimação de distribuição aplicados à otimização em ambientes dinâmicos	53
5.1	Considerações iniciais	53
5.2	Propostas de AEDs existentes na literatura	54
5.3	Novas propostas de AEDs para problemas em ambientes dinâmicos	55
5.3.1	Algoritmo AED _{MMG}	56
5.3.2	Controle de convergência	59
5.3.3	Algoritmo AED _{MMG} com versão <i>online</i> do algoritmo EM	61
5.4	Considerações finais	63
6	Experimentos	65
6.1	Considerações iniciais	65
6.2	Gerador de ambientes dinâmicos	65
6.2.1	Benchmark de Picos Móveis	66
6.2.1.1	Picos móveis com rotação e novas formas de alteração	68

6.3	CrITÉrios de avaliaÇo	72
6.4	Resultados experimentais	74
6.4.1	Anlise de sensibilidade dos parmetros do AED _{MMGO}	74
6.4.2	Treinoamento ao longo do tempo ou treinoamento completo	77
6.4.3	ComparaÇo com outras propostas	81
6.4.3.1	Anlise sobre o BPM: variando a severidade, nmero de picos e dimenso do espaÇo de busca	81
6.4.3.2	Anlise sobre o BPM com rotaÇo: variando o nmero de picos e frequncia das alteraÇes	85
6.5	ConsideraÇes finais	92
7	Concluses	95
	Trabalhos publicados pelo autor	99
	Referncias bibliogrficas	100

Lista de Figuras

2.1	Árvore de classes de problemas de otimização. Baseado em Biegler & Grossmann (2004).	7
2.2	Mínimos locais da função. Ponto A é ótimo global e B é ótimo local. A região destacada em preto denota a bacia de atração do mínimo global A.	9
2.3	Iterações iniciais do algoritmo <i>branch and bound</i>	11
2.4	Dinâmica temporal do ambiente.	16
3.1	Histograma da média anual de precipitação (em polegadas) de 70 cidades norte-americanas.	25
3.2	Estimação da variável <i>precipitação</i> por uma função densidade de probabilidade gaussiana.	26
3.3	Estimação por estimador <i>kernel</i> gaussiano com h igual a 3,8.	27
3.4	Estimação da variável <i>precipitação</i> por um modelo de mistura gaussiano.	33
3.5	Pontuações BIC para diversos modelos analisados.	34
4.1	Modelo gráfico com variáveis independentes (ausência de arestas indica ausência de dependência).	46
4.2	Modelos gráficos com dependência aos pares. Arestas direcionadas indicam que há dependência entre as variáveis.	48
4.3	Modelos gráficos com múltiplas dependências.	49
4.4	Modelo de mistura gaussiano.	50
5.1	Ilustração temporal da sobreposição de componentes do modelo de mistura. Inicialmente, os componentes representam regiões distintas do espaço de busca, mas, no decorrer do tempo, o algoritmo identifica que estas duas regiões passam a apresentar um alto grau de sobreposição.	59
5.2	Fluxograma do algoritmo AED_{MMG} com remoção de componentes sobrepostos.	60
5.3	Fluxograma do algoritmo AED_{MMG} com controle de convergência.	61

5.4	Passos E e M da variante <i>online</i> do algoritmo EM proposto por Nowlan (1991).	62
5.5	Taxa de convergência da versão <i>online</i> do algoritmo EM. A curva tracejada apresenta a execução utilizando $\gamma=0,99$ e a curva pontilhada, $\gamma=0,95$. A linha sólida representa uma versão incremental do algoritmo EM. Fonte: Neal & Hinton (1998).	63
6.1	Espaço de busca contínuo com 50 picos gerado pelo BPM.	67
6.2	Movimentação do máximo global em um espaço bidimensional, sujeito a 100 alterações, com diferentes valores de s e tomando $\lambda=0,5$	68
6.3	Espaço de busca contínuo com 50 picos gerado pelo BPM com rotação.	69
6.4	Movimentação do máximo global em um espaço bidimensional, sujeito a 100 alterações, para os seis tipos de alterações (exceto para a alteração com variação na dimensão) definido no BPM com rotação.	71
6.5	Comportamento do algoritmo utilizando diferentes valores do parâmetro η	76
6.6	Comportamento do algoritmo em relação à variação do parâmetro γ	76
6.7	Análise da média da norma da matriz de covariância dos componentes do modelo de mistura, quando empregado treinamento completo e ao longo do tempo. Quanto maior este valor, mais abertos são os componentes e, conseqüentemente, eles cobrem uma maior área do espaço de busca.	80
6.8	Performance relativa dos algoritmos para o Cenário 2 com alterações T_7	88
6.9	Variação do número de componentes do modelo de mistura utilizado no AED_{MMGO} (eixo y) no decorrer do tempo (número de avaliações de função - eixo x) para o Cenário 1 (10 picos). A variação em cada um dos sete tipos de alterações, descritas no BPM com rotação, estão ilustradas.	90
6.10	Variação do número de componentes do modelo de mistura utilizado no AED_{MMGO} (eixo y) no decorrer do tempo (número de avaliações de função - eixo x) para o Cenário 2 (50 picos). A variação em cada um dos sete tipos de alterações, descritas no BPM com rotação, estão ilustradas.	91

Lista de Tabelas

6.1	Configuração do <i>Scenario2</i> definido para o <i>benchmark</i> de picos móveis.	75
6.2	Ganho obtido e resultados da aplicação do teste <i>t</i> de <i>Student</i> ou Wilcoxon-Mann-Whitney (fundo destacado) para comparação entre treinamento ao longo do tempo e treinamento completo a cada geração.	78
6.3	<i>Speedup</i> do algoritmo com treinamento ao longo do tempo em relação ao treinamento completo.	79
6.4	Performance dos algoritmos em relação ao <i>Scenario2</i> definido para o BPM.	82
6.5	Performance dos algoritmos para ambientes com diferentes níveis de severidade nas alterações.	83
6.6	Performance dos algoritmos para espaços de busca com diferentes quantidades de picos e $s = 1$	83
6.7	Performance dos algoritmos para espaços de busca com diferentes quantidade de picos e $s = 5$	84
6.8	Performance dos algoritmos em problemas com média e alta dimensionalidades, tomando $s = 5$	84
6.9	Configuração do BPM com rotação utilizado para comparação entre as propostas de algoritmo.	85
6.10	Configuração dos ambientes dinâmicos.	86
6.11	Configuração de parâmetros dos algoritmos analisados.	86
6.12	Média do <i>offline error</i> e erro padrão para os cenários 1 e 2.	87
6.13	Média do <i>offline error</i> e erro padrão para os cenários 3 e 4.	87
6.14	Medidas de marcação dos algoritmos para o cenário com 10 picos.	92
6.15	Medidas de marcação dos algoritmos para o cenário com 50 picos.	92

Lista de Siglas

AE	-	Algoritmo evolutivo
AED	-	Algoritmo de estimação de distribuição
AED _{MMG}	-	Algoritmo de estimação de distribuição com modelo de mistura gaussiano
AED _{MMGO}	-	Algoritmo de estimação de distribuição com modelo de mistura gaussiano <i>online</i>
AG	-	Algoritmo genético
AGS	-	Algoritmo genético simples
AIC	-	<i>Akaike Information Criterion</i>
BB	-	<i>Branch-and-Bound</i>
BIC	-	<i>Bayesian Information Criterion</i>
BPM	-	<i>Benchmark</i> de Picos Móveis
EM	-	<i>Expectation-Maximization</i>
FDP	-	Função densidade de probabilidade
IUMDA	-	<i>Improved Univariate Marginal Distribution Algorithm</i>
PL	-	Programação linear
PNL	-	Programação não-linear
POVT	-	Problema de otimização variante no tempo
UMDA	-	<i>Univariate Marginal Distribution Algorithm</i>
OEP	-	Otimização por enxame de partículas
DynDE	-	<i>Dynamic Differential Evolution</i>
mCPSO	-	<i>multiple Clustering Particle Swarm Optimization</i>
SPSO	-	<i>Speciation-based Particle Swarm Optimizer</i>
CESO	-	<i>Collaborative Evolutionary-Swarm Optimization</i>
MMEO	-	<i>Multi-phase Multi-individual Evolutionary Optimisation</i>
ESCA	-	<i>Evolutionary Swarm Cooperative Algorithm</i>
MS _{FGBF}	-	<i>Multiple Swarm with Fast Global Best Formation</i>
DASA	-	<i>Differential Ant-Stigmergy Algorithm</i>
CPSO	-	<i>Clustering Particle Swarm Optimizer</i>
dopt-aiNet	-	<i>artificial immune Network for dynamic optimization</i>
UEP	-	<i>Unbiased Evolutionary Programming</i>
jDE	-	<i>j-Differential Evolution</i>

Capítulo 1

Introdução

O tratamento de problemas de otimização encontrados no mundo real geralmente se dá por meio da construção de um modelo físico-matemático, sendo que espera-se que este seja capaz de representar todos os aspectos relevantes vinculados ao problema de otimização de forma coerente e confiável. Uma vez alcançada uma solução para o modelo, confia-se que esta também seja a solução para o problema real.

O modelo de mundo é comumente representado através de uma função matemática, chamada de *função-objetivo*, junto à qual atuará o algoritmo de otimização, buscando encontrar pontos de máximo ou mínimo, satisfazendo algumas restrições impostas.

Ao estabelecer uma função-objetivo como representação do mundo real, considera-se que este permanece estático por toda a execução do método de otimização. No entanto, em diversas situações esta suposição não é verificada, e uma solução encontrada pode não ser mais válida, dado que as configurações internas do problema se alteraram e, conseqüentemente, o espaço de busca pode não ser mais o mesmo. Neste contexto, o tempo é uma variável-chave e que deve ser levado em consideração pelo algoritmo de otimização.

Em um problema de agendamento (*scheduling*) de um conjunto de eventos, é comum o surgimento de novos eventos e o cancelamento de outros. Assim, a agenda ótima varia no decorrer do tempo. O caminho mais rápido entre um ponto de partida e o destino é passível de mudança no meio do percurso, devido a alterações nas condições de tráfego. No controle e otimização de um processo químico, além de controlar variáveis como pressão e temperatura, existem objetivos adicionais como minimização do consumo de energia e maximização da conversão de um produto ou vários (Duraiski), sendo que a importância relativa desses múltiplos objetivos pode variar ao longo da operação da planta industrial.

Esta pesquisa destina-se a tratar de problemas dinâmicos de natureza discreta, onde a função-objetivo é modificada em uma forma não-infinitesimal e o método de otimização possui tempo hábil

para realizar um número fixo de avaliações de função entre modificações consecutivas da função-objetivo.

Em ambientes dinâmicos, o algoritmo de otimização deve ser capaz de perseguir a solução ótima no decorrer do tempo, reagindo de forma adequada a cada alteração do ambiente.

Considerando a adaptabilidade do algoritmo a uma nova função-objetivo, as meta-heurísticas populacionais, como os algoritmos evolutivos, surgem como abordagens promissoras para a tarefa de otimização em ambientes dinâmicos, desde que mantenham diversidade na população e sejam providos de mecanismos de adaptação capazes de promover reações adequadas às variações da função-objetivo. Considerando esta população, é possível atuar sobre o novo cenário e criar soluções mais bem adaptadas, através do uso de operadores específicos. O fato de que um conjunto de soluções é mantido, e não somente a melhor solução, tende a propiciar uma adaptação mais efetiva às novas condições possíveis (Rocha et al., 2005).

No contexto de algoritmos evolutivos, por muito tempo o estado-da-arte era representado por meta-heurísticas populacionais dotadas de operadores “cegos” para a perturbação das soluções correntes, acompanhados, sempre que possível, de algum mecanismo de busca local para compensar a falta de conhecimento sobre o espaço de busca, para alimentar a tomada de decisão. Três limitações estavam atreladas a essas meta-heurísticas: (i) a existência de vários parâmetros a determinar, os quais tinham influência direta no desempenho da busca; (ii) a necessidade prática de mecanismos de busca local eficientes; (iii) a incapacidade de utilizar o conhecimento já existente sobre o espaço de busca, a partir da experiência adquirida ao longo da execução da busca. Já nos últimos anos, essas propostas de meta-heurísticas têm dado lugar a metodologias que procuram gerar modelos de densidade de probabilidade no espaço de busca, chamados Algoritmos de Estimação de Distribuição (AEDs) (do inglês *Estimation of Distribution Algorithms*) ou PMBAs (do inglês *Probabilistic Model-Building Algorithms*). A intenção é que regiões promissoras do espaço de busca tenham associadas a si uma maior probabilidade de gerar soluções candidatas, o que implica que a exploração do espaço de busca tenderá a se concentrar nessas regiões. Assim, continua havendo um viés estocástico na busca, mas esta fica polarizada pelo modelo de distribuição, o qual, por sua vez e de forma genérica, deriva da experiência passada da própria busca. A tomada de decisão sobre como explorar o espaço de busca fica condicionada, então, ao desempenho obtido para as amostras já avaliadas durante a busca e à disponibilidade de algum conhecimento a priori sobre a natureza do espaço de busca (caso este conhecimento esteja disponível).

Embora os AEDs tenham surgido como meta-heurísticas interessantes para tratar de problemas de otimização em espaços contínuos, poucas propostas de algoritmos para atuar em ambientes dinâmicos estão disponíveis na literatura. Destacam-se apenas dois algoritmos simples que fazem suposições sobre o problema, como a independência entre as variáveis, e não apresentam

flexibilidade suficiente, por exemplo, para lidar com a multimodalidade do espaço de busca, geralmente presente em problemas de otimização de interesse prático.

Face a este cenário, este estudo tem como um dos objetivos investigar a aplicabilidade de AEDs guiados por modelos de mistura gaussianos, os quais são aptos a representar a dependência das variáveis e, ao mesmo tempo, a multimodalidade do espaço de busca.

1.1 Objetivos da pesquisa

Este trabalho possui como tema central, a investigação sobre a aplicabilidade de algoritmos de estimação de distribuição dotados de modelos probabilísticos flexíveis e parcimoniosos, no contexto de otimização em espaços contínuos e com função-objetivo variante no tempo.

Algoritmos de estimação de distribuição (AEDs) fazem uso de modelos probabilísticos para estimar as regiões promissoras, ou seja, regiões nas quais foram encontradas as melhores soluções até o momento. Posteriormente, novas soluções são geradas a partir do modelo e o processo se repete.

Considerando que a função-objetivo é constantemente alterada, e esta alteração geralmente não é previsível, o que tende a promover variações significativas e não antecipáveis acerca da localização das regiões promissoras, o modelo probabilístico deve ser flexível e capaz de se adaptar a este novo cenário.

Para que o modelo seja capaz de rapidamente se ajustar ao novo cenário criado pela variação temporal da função-objetivo, duas abordagens serão verificadas: o treinamento do modelo de mistura ao longo do tempo, ou seja, a convergência do algoritmo de treinamento do modelo de mistura (o algoritmo *Expectation-Maximization* (EM) (Dempster et al., 1977) é aqui utilizado) é adquirida no decorrer da execução e não em cada geração; e uma versão *online* do algoritmo EM, que é capaz de manter informações do histórico da busca e, ao mesmo tempo, rapidamente se adequar ao novo cenário, uma vez alterado.

A partir disto, essa pesquisa busca desenvolver ferramentas competitivas de otimização em espaços contínuos e com função-objetivo dinâmica, além de analisar o desempenho destas ferramentas, comparado-as com outras abordagens existentes na literatura, em problemas com diferentes níveis de complexidade.

1.2 Organização do texto

Na sequência desta introdução, esta dissertação é composta por 6 capítulos, os quais versam sobre os seguintes conteúdos.

No Capítulo 2, são discutidos os diversos tipos de problemas de otimização e as metodologias para resolvê-los, com ênfase em problemas de otimização em ambientes dinâmicos.

Os conceitos e principais métodos de estimação de densidade disponíveis na literatura, além das motivações que levaram ao emprego de modelos de mistura nesta pesquisa, são discutidos no Capítulo 3.

No Capítulo 4, são introduzidos os algoritmos de estimação de distribuição, caracterizando-os quanto ao nível de complexidade dos modelos probabilísticos utilizados. Além disso, os principais conceitos de computação evolutiva são apresentados, dando maior ênfase aos algoritmos genéticos e seus operadores.

Algumas propostas de AEDs aplicados a problemas de otimização em ambientes dinâmicos, encontradas na literatura, serão apresentadas no Capítulo 5. Além disso, serão desenvolvidas novas metodologias de AEDs, as quais estão entre as principais contribuições desta pesquisa.

A experimentação dos algoritmos propostos e comparação com resultados obtidos por outros métodos de otimização populacionais, como Otimização por Enxame de Partículas, Evolução Diferencial, Sistemas Imunológicos Artificiais, Otimização por Colônia de Formigas, além das duas propostas de AEDs encontradas na literatura, são apresentadas no Capítulo 6.

Por fim, as conclusões desta pesquisa são relatadas no Capítulo 7, destacando suas principais contribuições e apontando novas direções e melhorias que podem ser implementadas no futuro.

Capítulo 2

Otimização em ambientes dinâmicos

2.1 Considerações iniciais

Otimizar é o ato de escolher a melhor forma de realizar uma tarefa sem violar certas restrições (Boyd & Vandenberghe, 2004). Otimização é uma atividade cotidiana na vida das pessoas. Em tarefas simples realizadas diariamente, sempre busca-se executá-las da melhor forma possível. Por exemplo, muitos problemas práticos estão associados com a escolha do menor caminho ou do caminho mais rápido entre um ponto de partida e um de destino. Na indústria, este cenário não é diferente. Executivos buscam constantemente soluções para otimizar seus processos.

Tarefas complexas que possuem muitas variáveis e grande quantidade de restrições requerem maneiras mais sistemáticas para sua otimização. A área de estudo que desenvolve ferramentas matemáticas e computacionais para tratar deste tipo de problema é chamada de *otimização matemática* (ou programação matemática) (Bazaraa & Jarvis, 1977; Boyd & Vandenberghe, 2004; Fletcher, 1987).

A maioria dos métodos de otimização são aplicáveis a ambientes estáticos, onde a função-objetivo permanece inalterada em toda a execução. No entanto, em muitos problemas reais a função-objetivo é dinâmica, sendo constantemente alterada. Neste tipo de ambiente, os algoritmos tradicionais tendem a obter baixa performance, dado o baixo poder de reação ao novo cenário. Assim, algoritmos dedicados devem ser desenvolvidos para tratar destes problemas.

Neste capítulo, serão discutidos os diversos tipos de problemas de otimização e as metodologias para resolvê-los, com ênfase em problemas de otimização em ambientes dinâmicos.

2.2 Otimização matemática

A otimização envolve a determinação da “melhor” solução para certos problemas matematicamente definidos, que são, na maioria das vezes, modelos da realidade física (Fletcher, 1987).

A melhor solução pode ser expressa no sentido de máximo ou mínimo, dependendo do problema tratado. Vale ressaltar que as tarefas de maximização e minimização são trivialmente relacionadas, pois uma função f a ser maximizada pode ser representada como a minimização de $(-f)$.

Para que seja possível a resolução de um problema de otimização real, inicialmente é necessária a criação de um modelo matemático que represente tal problema, sendo esta etapa chamada de *modelagem*. O modelo é geralmente expresso como uma função matemática, chamada de *função-objetivo*.

A etapa de modelagem é de vital importância, pois, caso o modelo não expresse adequadamente o ambiente real, a solução obtida pelo método de otimização não será a melhor solução para o problema real.

Um problema de otimização matemática é constituído de três componentes: uma *função-objetivo*, a qual se deseja otimizar; um *conjunto de variáveis*, que define o espaço de busca pela solução ótima; e um *conjunto de restrições* a serem atendidas. A equação 2.1 mostra a definição matemática de um problema de otimização em espaços contínuos.

$$\begin{aligned} & \underset{\mathbf{x} \in \mathbf{B}}{\text{otimizar}} && f(\mathbf{x}) \\ & \text{sujeito a} && f_i(\mathbf{x}) \leq b_i, \quad i = 1, \dots, m. \end{aligned} \tag{2.1}$$

Aqui, \mathbf{B} é o espaço de busca, o vetor $\mathbf{x} \in \mathbb{R}^n$ contém o conjunto de variáveis a serem otimizadas, a função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é a função-objetivo, as funções $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$, são as funções que permitem definir as restrições do problema, e as constantes b_1, \dots, b_m são os limitantes para as restrições. O vetor \mathbf{x}^* é chamado de *ótimo* ou solução do problema (2.1), se \mathbf{x}^* tem o valor extremo da função-objetivo dentre todos os vetores \mathbf{x} que satisfazem as restrições, como mostra a equação 2.2:

$$\begin{cases} f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbf{B} \text{ e } \mathbf{x} \text{ factível,} & \text{problemas de minimização,} \\ f(\mathbf{x}^*) \geq f(\mathbf{x}), \forall \mathbf{x} \in \mathbf{B} \text{ e } \mathbf{x} \text{ factível,} & \text{problemas de maximização.} \end{cases} \tag{2.2}$$

Considera-se aqui que \mathbf{B} pode conter tanto soluções factíveis como soluções infactíveis.

As características matemáticas da função-objetivo, das funções de restrição e das variáveis a serem otimizadas determinam o grau de dificuldade para a resolução do problema, quais métodos podem

ser aplicados e o grau de confiança na otimalidade da solução obtida.

A figura 2.1 mostra uma classificação possível para os problemas de otimização, considerando algumas de suas principais características.

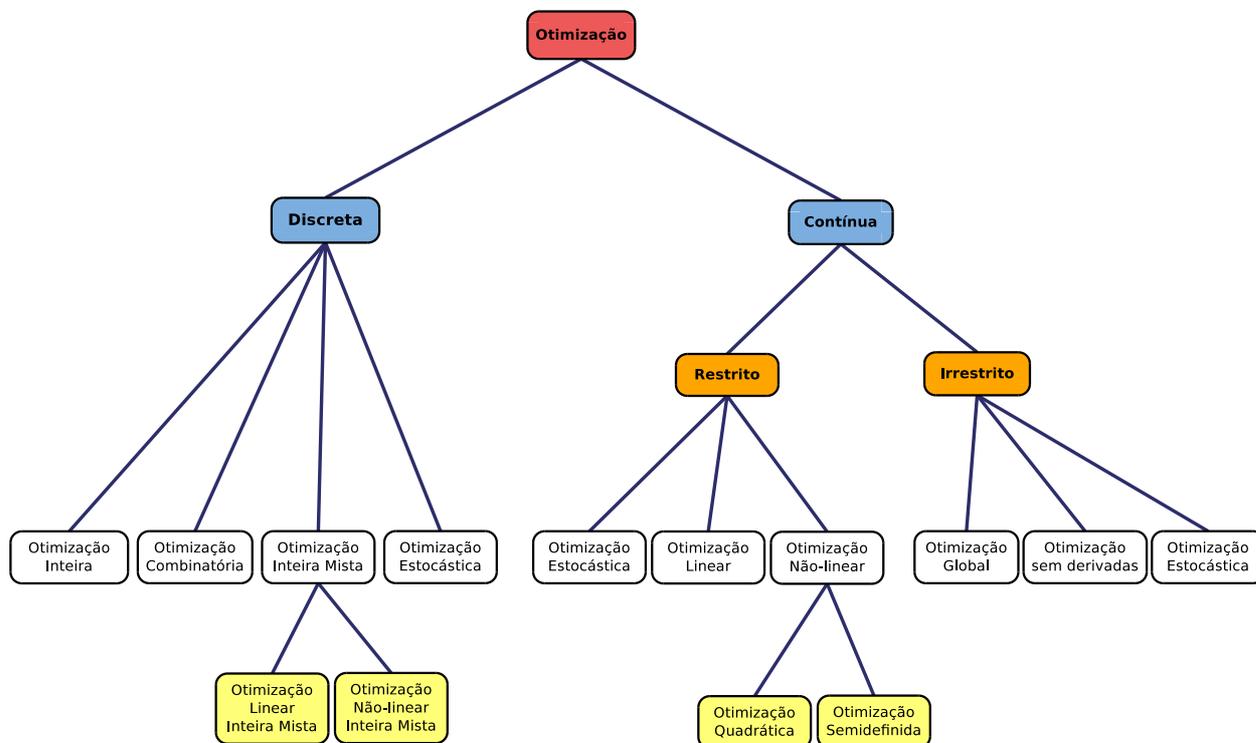


Fig. 2.1: Árvore de classes de problemas de otimização. Baseado em Biegler & Grossmann (2004).

Inicialmente, os problemas de otimização podem ser separados em termos de variáveis contínuas e discretas. A maioria dos problemas de otimização contínua são classificados quanto à linearidade da função-objetivo e das funções de restrição, distinguindo-os entre problemas de *programação linear* (PL) e de *programação não-linear* (PNL). Um problema de PL envolve a maximização ou minimização de uma função-objetivo linear na presença de restrições lineares de desigualdade e/ou igualdade (Bazaraa & Jarvis, 1977). Caso contrário, é dito ser não-linear.

O conceito de *convexidade* da função-objetivo em um problema de PNL é importante para a caracterização do problema, podendo ela ser convexa ou não-convexa. A convexidade informará a quantidade de ótimos locais presentes no espaço de busca. Esta característica do espaço será melhor discutida na Seção 2.2.1.

Outra distinção realizada é em relação à diferenciabilidade da função-objetivo. A maioria dos métodos tradicionais de otimização podem somente ser aplicados quando a função é diferenciável e convexa.

No caso de problemas com variáveis discretas e contínuas, inicialmente eles são classificados em

programação linear inteira mista (PLIM) e *programação não-linear inteira mista* (PNLIM). Quando todas as variáveis são discretas recai em um problema de *programação inteira* (PI) ou combinatória.

Outra extensão importante inclui problemas que envolvem incertezas, os quais dão origem a problemas de *programação estocástica*.

2.2.1 Caracterização da função-objetivo

O extremo (máximo ou mínimo) da função-objetivo pode ser classificado como *global* ou *local*. Ótimo global é o maior (ou menor) valor da função-objetivo em todo o espaço de busca, enquanto que o ótimo local é o maior (ou menor) valor da função-objetivo em uma determinada vizinhança (um subespaço do espaço de busca). A vizinhança é definida por alguma métrica de distância, como por exemplo a distância euclidiana para o caso contínuo ou de Hamming para o caso binário, limitada por uma constante $\epsilon > 0$. Define-se uma vizinhança de $\mathbf{x} \in \mathbb{R}^n$ por $V(\mathbf{x})$, como mostra a equação 2.3:

$$V(\mathbf{x}) = \{\mathbf{y} \in \mathbf{B} | d(\mathbf{x}, \mathbf{y}) \leq \epsilon\} \quad (2.3)$$

onde $\mathbf{y} \in \mathbb{R}^n$, \mathbf{B} é o espaço de busca e $d(\cdot, \cdot)$ é a função de distância utilizada.

A partir disto, pode-se definir formalmente ótimos locais e globais. Uma solução \mathbf{x} é dita ser ótimo local, se e somente se

$$\begin{cases} f(\mathbf{x}) \leq f(\mathbf{y}), \forall \mathbf{y} \in V(\mathbf{x}), & \text{para minimização,} \\ f(\mathbf{x}) \geq f(\mathbf{y}), \forall \mathbf{y} \in V(\mathbf{x}), & \text{para maximização,} \end{cases} \quad (2.4)$$

para algum $\epsilon > 0$, e ótimo global, se e somente se

$$\begin{cases} f(\mathbf{x}) \leq f(\mathbf{y}), \forall \mathbf{y} \in \mathbf{B}, & \text{para minimização,} \\ f(\mathbf{x}) \geq f(\mathbf{y}), \forall \mathbf{y} \in \mathbf{B}, & \text{para maximização.} \end{cases} \quad (2.5)$$

A figura 2.2 ilustra um espaço de busca unidimensional de um problema de minimização de uma função-objetivo contínua com ótimos locais (pontos A e B) e global (ponto A). Também é apresentada a *bacia de atração* do ponto A, conceito este que interpreta a função-objetivo como uma superfície de energia e os algoritmos de otimização como aqueles que tendem a localizar pontos de energia mínima.

Em problemas com muitos ótimos locais (multimodal), a tarefa de buscar o extremo, em geral, torna-se mais complexa. Métodos tradicionais de otimização, em sua maioria, são destinados a

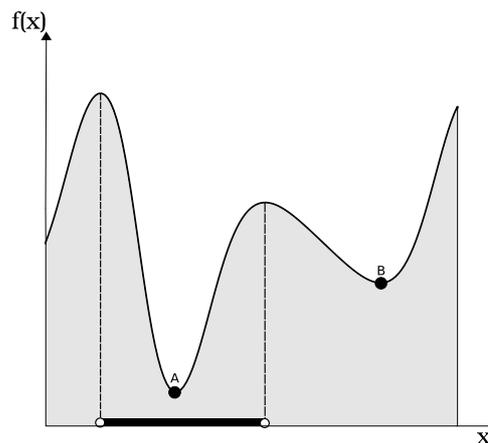


Fig. 2.2: Mínimos locais da função. Ponto A é ótimo global e B é ótimo local. A região destacada em preto denota a bacia de atração do mínimo global A.

problemas cuja função-objetivo é unimodal, ou seja, com apenas um ótimo (global) e diferenciável.

Duas formas para tentar contornar este problema são (Press et al., 2007): (i) execute métodos de busca local diversas vezes, com diferentes valores iniciais, e escolha a melhor solução dentre todas as execuções; ou (ii) utilize técnicas de busca estocástica que fazem perturbações nas soluções, de forma a explorar o espaço, em busca de melhores soluções. Essas técnicas têm obtido grande sucesso em uma variedade de problemas multimodais (Press et al., 2007).

A área de pesquisa que busca por soluções globais em problemas multimodais é conhecida como *otimização global*. Esta pesquisa de mestrado trata deste tipo de problemas, em um contexto em que a função-objetivo é também variante no tempo.

2.3 Métodos de otimização

Os métodos para resolução de problemas de otimização são, de forma geral, específicos para cada tipo de problema. Para escolher qual método utilizar, é preciso identificar o tipo de problema que se deseja otimizar. Esta decisão, basicamente, é tomada com base na identificação das seguintes características do problema:

- *tipo das variáveis*: se discretas, contínuas ou ambas;
- *presença de restrições*;
- *linearidade da função-objetivo e das funções de restrição*;
- *diferenciabilidade da função-objetivo*;

- *convexidade da função-objetivo*: se unimodal, multimodal ou com presença de platôs;
- *presença de incertezas na função-objetivo*.

A seguir, serão apresentados alguns dos principais métodos de otimização, tanto para espaços discretos quanto para espaços contínuos.

2.3.1 Métodos tradicionais para espaços discretos

Quando as variáveis do problema de otimização são discretas, sejam elas inteiras ou binárias, o conjunto de soluções factíveis é contável. Assim, uma abordagem direta e exata seria calcular os valores da função-objetivo para todas as soluções factíveis e escolher aquela com melhor valor.

No entanto, em problemas reais, o número de variáveis e o conjunto de possíveis valores tendem a ser muito grandes, tornando assim computacionalmente proibitivo avaliar todos os casos. Portanto, procedimentos para restringir o conjunto de soluções factíveis, direcionando a busca e desprezando regiões que seguramente não possuem a solução ótima, são abordagens altamente indicadas. Este é o princípio dos bem conhecidos métodos de *branch-and-bound* (Land & Doig, 1960).

Métodos de Branch-and-Bound

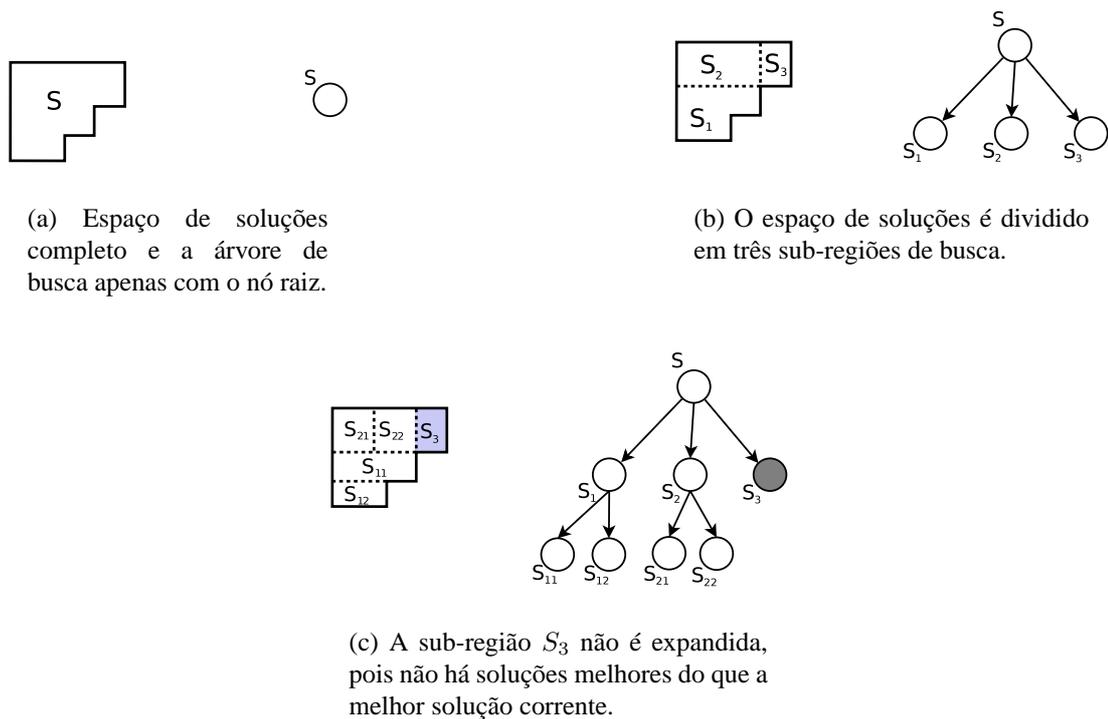
Métodos de *branch-and-bound* (BB) realizam uma busca controlada dentro do espaço de soluções factíveis. O uso de limitantes para a função a ser otimizada, combinado com o valor da melhor solução encontrada até o momento, capacita o algoritmo a buscar apenas em regiões promissoras do espaço de busca.

Partindo do conjunto completo de potenciais soluções, o algoritmo cria uma árvore de busca no decorrer do processo, expandindo apenas os nós que satisfizerem as restrições de limite. Uma iteração do algoritmo é composta por três etapas principais (Clausen, 1997): (i) seleciona um nó para o processamento; (ii) calcula os limitantes (superior e inferior) e (iii) expande os nós.

As figuras 2.3(a), 2.3(b) e 2.3(c) mostram três iterações iniciais do algoritmo *branch-and-bound*, partindo do conjunto de soluções completo. Do lado esquerdo, é mostrado como o espaço de busca está sendo explorado e, ao lado direito, a árvore de busca corrente.

A busca termina quando não há regiões inexploradas do espaço de soluções e a melhor solução corrente é dita ser a solução ótima.

A eficiência do método depende fortemente do procedimento de estimação dos limitantes, superior e inferior, além do procedimento para expansão dos nós (Fletcher, 1987). Muitos problemas de otimização não admitem essas estimativas e, portanto, não podem ser tratados por métodos de

Fig. 2.3: Iterações iniciais do algoritmo *branch and bound*.

branch-and-bound.

Além do método *branch-and-bound*, outros métodos para problemas discretos foram propostos na literatura. Uma compilação dos algoritmos mais conhecidos para problemas discretos pode ser encontrada em Biegler & Grossmann (2004).

2.3.2 Métodos tradicionais para espaços contínuos

Métodos tradicionais de otimização em espaços contínuos são, em sua maioria, aplicados a problemas cuja função-objetivo é diferenciável e convexa. Estes métodos partem de um ponto inicial qualquer e realizam uma busca direcionada do ótimo local, na sua vizinhança. Em problemas convexos, o ótimo local é o ótimo global, dado que esses problemas são unimodais. No caso de problemas não-convexos, geralmente o que se obtém é o ótimo local cuja bacia de atração contém a condição inicial.

Para problemas contínuos não-lineares, destacam-se algoritmos que utilizam derivadas da função-objetivo para determinar a direção da busca. Dentre eles, destaca-se o *método do gradiente descendente* e o *método de Newton*.

Método do gradiente descendente

O método do gradiente descendente parte de um ponto inicial e obtém a direção que produz o maior decréscimo (problema de minimização) da função-objetivo, indicada pela informação contida no seu vetor gradiente. Como o gradiente corresponde à direção de maior crescimento da função, busca-se então caminhar na direção contrária.

Dado o tamanho de passo α , pré-determinado pelo usuário ou obtido por algum método de busca linear, o próximo ponto é então calculado e o processo se repete até que o algoritmo não obtenha ganhos significativos. O Algoritmo 1 apresenta o pseudo-código do método do gradiente descendente para o problema de minimização (Boyd & Vandenberghe, 2004), onde $\mathbf{x} \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$, $\nabla f(\mathbf{x}^{(k)})$ representa o vetor gradiente avaliado no ponto $\mathbf{x}^{(k)}$ e $\|\cdot\|_2$ a norma euclidiana.

Algoritmo 1: Método do Gradiente Descendente (versão para minimização).

```

1 início
2   Escolha  $\mathbf{x}^{(k=0)}$  como ponto inicial;
3   enquanto  $\|\nabla f(\mathbf{x}^{(k)})\|_2 > \epsilon$  faça
4      $\Delta \mathbf{x}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$ ;
5     Utilize busca linear para encontrar o tamanho do passo  $\alpha$ ;
6      $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \Delta \mathbf{x}^{(k)}$ ;
7   fim
8 fim
```

A busca linear é um subproblema de otimização que deve ser resolvido para que se melhore o desempenho do método do gradiente descendente. Vários algoritmos foram propostos na literatura, dentre eles a busca linear exata, o método de Fibonnaci e o método da seção áurea (Fletcher, 1987).

A principal vantagem do método do gradiente é sua simplicidade. E a principal desvantagem é que a taxa de convergência é fortemente dependente do problema, mais especificamente da condição da matriz hessiana (Boyd & Vandenberghe, 2004).

Método de Newton

O método de Newton realiza sucessivas minimizações de aproximações quadráticas da função-objetivo, convergindo em geral mais rapidamente, quando comparado ao método do gradiente. A aproximação quadrática, $\hat{f}(\mathbf{x})$, é feita por uma expansão em série de Taylor até segunda ordem, na

forma:

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \quad (2.6)$$

onde $\nabla^2 f(\mathbf{x}_0)$ é a matriz hessiana, que é uma matriz quadrada das derivadas parciais de segunda ordem da função $f(\mathbf{x})$ no ponto \mathbf{x}_0 , $\nabla f(\mathbf{x}_0)$ é o vetor gradiente no ponto \mathbf{x}_0 e $\mathbf{x} \in \mathbb{R}^n$.

Para funções quadráticas, o método de Newton encontra o ótimo global em apenas uma iteração e, para funções com características similares à quadrática, o método converge rapidamente.

Assim como o método do gradiente, o método de Newton converge, geralmente, para o mínimo local cuja bacia de atração contém o ponto inicial. A sequência de passos do método de Newton para um problema de minimização é mostrada no Algoritmo 2.

Algoritmo 2: Método de Newton (versão para minimização).

```

1 início
2   Escolha  $\mathbf{x}^{(k=0)}$  como ponto inicial;
3   enquanto  $\|\nabla f(\mathbf{x}^{(k)})\|_2 > \epsilon$  faça
4      $\Delta \mathbf{x}^{(k)} = -(\nabla^2 f(\mathbf{x}^{(k)}))^{-1} \cdot \nabla f(\mathbf{x}^{(k)})$ ;
5      $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta \mathbf{x}^{(k)}$ ;
6   fim
7 fim

```

O método de Newton possui uma rápida taxa de convergência, tipicamente quadrática. Após encontrada a vizinhança da solução, o método converge para a solução ótima em poucas iterações.

O principal problema com o método de Newton é a necessidade do cálculo e da inversão da matriz hessiana, $\nabla^2 f(\mathbf{x})$, que pode ser altamente custosa e sujeita a problemas numéricos (Nocedal & Wright, 1999).

Os chamados métodos *quase-Newton* buscam contornar este problema, realizando uma aproximação da inversa da matriz hessiana, que é menos custosa, e esta é atualizada a cada passo, levando em conta o conhecimento obtido nessa iteração.

Embora estes algoritmos convirjam para um ótimo local da função, eles requerem que a função seja contínua e diferenciável (duas vezes para o método de Newton). Além disso, se a função-objetivo não for convexa, não há garantia de convergência, pois a matriz hessiana pode deixar de ser definida positiva (no caso de minimização). Existem adaptações que buscam positivar a matriz hessiana e introduzir um passo α equivalente ao definido pelo método do gradiente, dando origem ao método de Newton modificado (Goldfeld et al., 1966).

Além dos métodos apresentados, vários outros métodos foram propostos na literatura. Boas

referências são encontradas em Bazaraa & Jarvis (1977); Nocedal & Wright (1999) e Boyd & Vandenberghe (2004).

Quando a função-objetivo e as funções de restrição são lineares (problema de programação linear) o espaço de busca é um polítopo convexo e, uma vez encontrado o ótimo local, este também é ótimo global.

Dentre os métodos mais conhecidos de PL, destacam-se o *método simplex* (Dantzig, 1998) e os *métodos de pontos interiores* (Karmarkar, 1984). Mais detalhes sobre métodos de programação linear podem ser encontrado em Bazaraa & Jarvis (1977) e Fletcher (1987).

Branch-and-Bound para espaços contínuos

O conceito central de expansão controlada, empregada pelos algoritmos de *Branch-and-Bound*, pode ser estendido para métodos de otimização em espaços contínuos, uma vez estabelecidas formas de obter os limitantes inferiores e superiores. Um exemplo desta extensão é o algoritmo α -BB (do inglês, *α -based Branch-and-Bound*) (Maranas & Floudas, 1994a,b).

O α -BB é um método determinístico que oferece garantia de convergência para um ponto arbitrariamente próximo ao mínimo global, em problemas não-lineares com função-objetivo e funções de restrição duas vezes diferenciáveis. O algoritmo constrói uma sequência convergente de limitantes superiores e inferiores para mínimo global através da relaxação convexa do problema original (Adjiman et al., 1998).

Cada iteração do algoritmo consiste em um passo de *branch* seguido por um passo de *bound*. Várias estratégias de *branching* podem ser utilizadas, sendo comumente adotada a bissecção de todas, ou um conjunto, das variáveis do problema (Adjiman et al., 1998).

No passo de *bound*, o limitante inferior é obtido pela construção de subestimadores convexos para as funções do problema, e então resolve-se o problema não-linear convexo resultante. Para o limitante superior, uma abordagem possível é resolver o problema original localmente para cada um dos subdomínios do espaço de soluções.

A obtenção do ótimo global é dependente da validade da estimação dos limitantes inferiores, bem como da construção de limitantes inferiores cada vez mais estreitos para sucessivas partições do espaço de busca. Tais propriedades levam à geração de uma sequência não-decrescente de limitantes inferiores que progridem no sentido da solução ótima.

Uma desvantagem do método é que tanto a função-objetivo quanto as funções de restrição devem ser duas vezes diferenciáveis, o que nem sempre é satisfeito.

2.3.3 Métodos heurísticos

Devido à complexidade de muitos problemas de otimização, especialmente aqueles com grande quantidade de variáveis, encontrados em contextos mais práticos, os algoritmos exatos (aqueles que garantem encontrar a solução exata para o problema) muitas vezes apresentam baixo desempenho ou nem mesmo podem ser aplicados. A partir disto, *métodos heurísticos*, que encontram boas soluções em tempo satisfatório, são preferidos em aplicações práticas.

Heurísticas são técnicas (consistindo de uma regra ou um conjunto de regras) que buscam por boas soluções em um tempo computacional aceitável (Voß, 2001). É possível dizer que uma heurística é um método delineado especificamente para um determinado tipo de problema, uma vez que utiliza conhecimento sobre características deste problema para definir as regras de busca. As heurísticas são identificadas como métodos aproximados, no sentido que provêem uma boa solução com relativamente pouco esforço computacional, mas não possuem garantias de convergência e de otimalidade.

Existe uma classe de heurísticas mais generalizadas, as chamadas *meta-heurísticas*, que são abordagens que buscam tratar de diversos tipos de problemas sem a necessidade de realizar grandes alterações na estrutura do algoritmo. Esta metodologia inclui a manipulação de uma ou várias outras heurísticas.

Segundo Voß et al. (1999) uma metaheurística é um processo mestre iterativo que guia e modifica as operações de heurísticas subordinadas para eficientemente produzir soluções de alta qualidade.

Uma meta-heurística pode manipular uma única solução, parcial ou completa, ou ainda uma coleção de soluções a cada iteração. A heurística subordinada pode ser um procedimento de alto (ou baixo) nível, uma busca local ou um método construtivo. A família de meta-heurísticas inclui, mas não está restrito a, busca tabu (Glover & Marti, 2006), algoritmos de colônia de formigas (Dorigo et al., 2006), algoritmos de enxame de partículas (Kennedy & Eberhart, 2001), GRASP (do inglês, *Greedy Randomized Adaptive Search Procedure*) (Feo & Resende, 1995), algoritmos evolutivos (Bäck & Schwefel, 1993; Goldberg, 1989), *simulated annealing* (Kirkpatrick et al., 1983), e suas hibridizações.

2.4 Otimização em espaços variantes no tempo

Otimização de ambientes dinâmicos consiste em uma subárea da otimização que trata de sistemas dinâmicos. Neste tipo de problema, além dos três componentes citados na seção anterior (função-objetivo, variáveis e restrições), a variável *tempo* é adicionada à equação 2.1. Logo, define-se um problema de otimização variante no tempo (POVT) como mostra a equação 2.7:

$$\begin{aligned} & \underset{\mathbf{x} \in \mathbf{B}(t)}{\text{otimizar}} && f(\mathbf{x}, t) \\ & \text{sujeito a} && f_i(\mathbf{x}, t) \leq b_i(t), \quad i = 1, \dots, m. \end{aligned} \tag{2.7}$$

Neste cenário, o método de otimização deve encontrar valores ótimos para a variável \mathbf{x} em cada período de tempo em que o ambiente permanece em uma determinada configuração, considerando *tempo discreto*; ou a cada ponto em um dado intervalo de tempo, para o caso de *tempo contínuo* (Chiang, 1999), como é ilustrado na figuras 2.4.

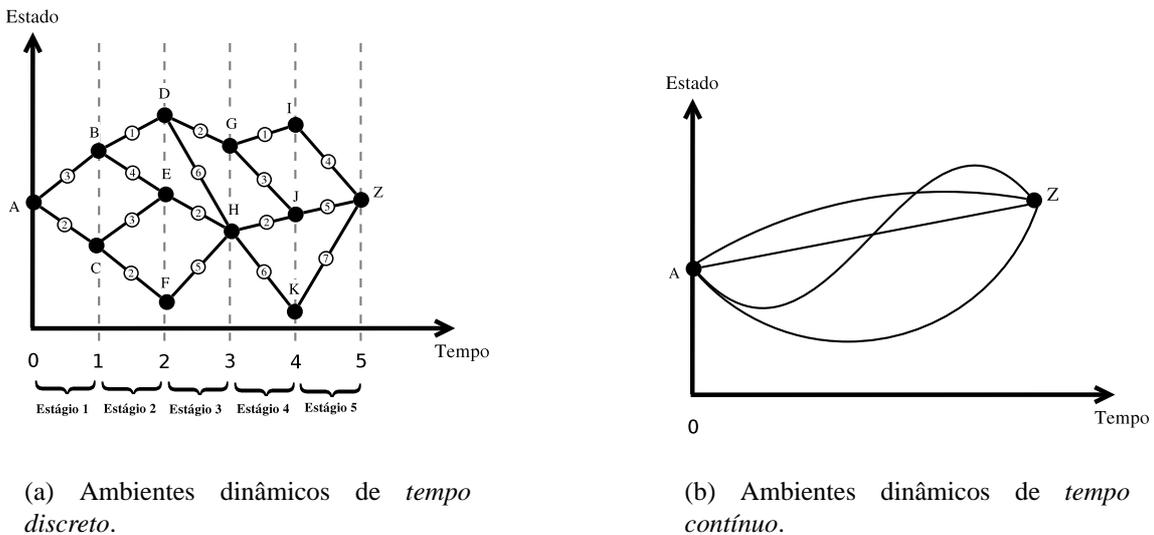


Fig. 2.4: Dinâmica temporal do ambiente.

As variáveis a serem otimizadas no instante de tempo t são chamadas de *variáveis de estado*, em analogia ao conceito de espaço de estados em sistemas dinâmicos. A configuração total do ambiente, ou seja, o conjunto de variáveis, de restrições e a função-objetivo, é chamada de *estado* do ambiente.

A característica da variável tempo e das variáveis a serem otimizadas, quanto ao tipo, discreto ou contínuo, são fundamentais para a resolução do POVT via métodos de otimização, como será visto nas próximas seções.

Formas de classificar problemas de otimização em ambientes dinâmicos, para nortear a escolha de qual método de otimização utilizar, foram sugeridas por diversos autores. Branke (2001b) sugere os seguintes critérios para caracterizar ambientes dinâmicos:

1. **Frequência das alterações:** Mede a frequência de ocorrência de alterações no ambiente. No contexto de otimização, este parâmetro limitará o número de avaliações da função-objetivo ou de sua derivada, realizadas pelos algoritmos de otimização.

2. **Severidade da alteração:** Esse critério é usualmente caracterizado pela distância entre o antigo e o novo ótimo.
3. **Previsibilidade da alteração:** O quão previsíveis são as alterações, se são puramente aleatórias ou seguem algum padrão que pode ser aprendido.
4. **Tamanho e acurácia do ciclo:** Em ambientes cíclicos, esse critério define quanto tempo leva para o ambiente retornar a um estado anterior e quão preciso é o retorno.

Na próxima seção, são discutidos alguns dos métodos tradicionais para a resolução de problemas de otimização em ambientes dinâmicos.

2.4.1 Métodos tradicionais para otimização de ambientes dinâmicos

Assim como os métodos de resolução de problemas de otimização estática, em otimização de ambientes dinâmicos, os métodos tradicionais são aqueles que utilizam as derivadas da função-objetivo para a tomada de decisão. Além disso, como nesta classe de problemas deseja-se modelar a dinâmica do sistema, os *métodos variacionais* fazem uso de equações diferenciais para modelagem (Van Breusegem & Bastin, 1990). Estes métodos são também chamados de *métodos indiretos* (Biegler & Grossmann, 2004).

Outra abordagem comum é a discretização do espaço e/ou tempo, convertendo o problema original contínuo em um problema discreto, possibilitando, portanto, a aplicação de métodos numéricos bem estabelecidos em otimização discreta. Dependendo do nível de discretização do problema, estes são classificados entre métodos de *discretização parcial* ou *completa*. Basicamente, problemas parcialmente discretizados podem ser resolvidos por *programação dinâmica* ou programação não-linear e, para os completamente discretizados, também podem ser aplicados métodos de programação dinâmica, juntamente com equações diferenciais (Biegler & Grossmann, 2004).

A seguir, são brevemente apresentados os métodos variacionais e métodos que discretizam o espaço de busca, como a programação dinâmica.

2.4.1.1 Métodos variacionais

Problemas que podem ser expressos na forma de equações diferenciais podem ser resolvidos pela satisfação das condições necessárias de primeira ordem, que são obtidas através do princípio de máximo de Pontryagin (Pontryagin et al., 1962). Aqui, o problema de otimização de uma função-objetivo é reformulado como a minimização de uma função hamiltoniana (Srinivasan et al., 2003) e as condições de otimalidade são formuladas por um conjunto de equações algébrico-diferenciais.

De acordo com Cervantes & Biegler (2001), o principal problema em obter soluções destas equações são as condições de contorno. Embora existam métodos padrões para resolução deste problema, como por exemplo, *single shooting* e *invariant embedding*, estes sofrem de diversos inconvenientes, como instabilidade, alta não-linearidade e conversão para outros problemas com alta dimensão (Biegler & Grossmann, 2004; Srinivasan et al., 2003).

Além disso, para todos os métodos citados, quando as restrições são limitadas, o sistema resultante recai em um problema combinatório que é proibitivamente custoso, exceto para problemas pequenos.

Além das dificuldades enunciadas, outra grande limitação das abordagens analíticas é que envolvem computação simbólica, que se mostra inviável para problemas de grande porte (Biegler & Grossmann, 2004; Roubos et al., 1999).

2.4.1.2 Métodos de discretização

Em muitos problemas reais, é possível deparar-se com situações em que há dezenas ou centenas de variáveis a serem otimizadas. Neste cenário, as abordagens analíticas se tornam proibitivas, devido a seu alto custo computacional. Assim, as abordagens numéricas são as mais indicadas.

Diferentemente de métodos analíticos de otimização em ambientes dinâmicos, os métodos numéricos discretizam o tempo e as variáveis do sistema em um conjunto de valores pré-estabelecidos. Duas estratégias são usualmente consideradas, uma baseada em programação dinâmica e outra em programação não-linear. Métodos de otimização não-linear foram apresentados na Seção 2.3.

Programação dinâmica

Programação dinâmica é um poderoso paradigma no qual o problema é solucionado pela quebra do problema original em uma coleção de subproblemas mais simples. Procedendo a resolução serial destes subproblemas, iniciando pelos menores, utilizam-se as soluções destes para a resolução dos maiores, até que todos estejam resolvidos. É assim que a solução do problema original é encontrada. A ideia por trás desta abordagem é o princípio da otimalidade, isto é, “parte de uma trajetória ótima também é ótima” (Bellman, 1957).

Aqui, o tempo $[0, t]$ é discretizado em P estágios, com $[t_p, t_{p+1}]$ sendo o intervalo de tempo correspondente ao $(p + 1)$ -ésimo estágio. As variáveis são discretizadas em forma de grade, cuja granularidade deve ser determinada. Se o grânulo é grande, ou seja, o número de valores possíveis é baixo, o valor ótimo encontrado pode estar longe do valor ótimo real. Com o decréscimo do tamanho da grade, as soluções são mais próximas, porém o tempo computacional cresce explosivamente (Roubos et al., 1999).

O problema de otimização dinâmica é encontrar o estágio (dentre os P estágios) em que um conjunto de variáveis de estado minimiza a função-objetivo. Iniciando do último estágio, o algoritmo parte de um conjunto de valores pré-definidos ($M = \{1, \dots, m\}$) para as variáveis de estado, obtém a melhor configuração destes valores para o estágio e o algoritmo segue para o estágio anterior. Uma vez todos os estágios considerados, o espaço de busca é reduzido a uma região em torno do melhor conjunto de valores obtidos e o processo se repete.

De acordo com Srinivasan et al. (2003), duas principais vantagens deste método são: (i) é um dos poucos métodos capazes de obter mínimos globais e (ii) o número de iterações, conseqüentemente o tempo necessário para a otimização, pode ser estimado *a priori*, pela especificação do número de valores possíveis para as variáveis de estado (M) e estágios utilizados (P).

A maior desvantagem da programação dinâmica é sua complexidade computacional. Embora problemas pequenos e na presença de restrições, onde o espaço de busca é limitado, possam ser solucionados eficientemente, em problemas de médio e grande porte esta abordagem torna-se computacionalmente inviável (Srinivasan et al., 2003).

Neste cenário, algoritmos evolutivos surgem como abordagens promissoras para a tarefa de otimização dinâmica, desde que mantenham uma população de soluções que podem ser facilmente adaptadas.

Considerando esta população, é possível atuar sobre o novo cenário e criar soluções melhor adaptadas, através do uso de operadores específicos. O fato de que um conjunto de soluções é mantido simultaneamente, e não somente a melhor solução, tende a propiciar uma rápida adaptação às novas condições do espaço de busca (Rocha et al., 2005).

2.5 Algoritmos evolutivos em ambientes dinâmicos

Algoritmos evolutivos (AE's) são inspirados em princípios da evolução natural, que é um processo estocástico e dinâmico, que manipula uma população de soluções simultaneamente de forma a obter melhorias contínuas em um problema de otimização. Estas características dos AE's os candidatam como uma alternativa promissora na aplicação em otimização de ambientes variantes no tempo (Branke, 1999). Algoritmos evolutivos são melhor discutidos no Capítulo 4.

Uma abordagem direta para tratar de problemas de otimização em ambientes dinâmicos é a simples reinicialização do algoritmo, a cada alteração no ambiente. No entanto, esta abordagem é muitas vezes impraticável, devido ao alto custo computacional para resolver um problema sem reutilizar as informações do passado (Branke, 1999).

Além disso, se for considerado que as alterações sofridas pelo ambiente são relativamente

pequenas, provavelmente o novo ótimo estará, de certa forma, relacionado com o ótimo antigo. Neste caso, um algoritmo capaz de transferir conhecimento da população anterior para a população atual tende a ser mais eficiente do que uma simples reinicialização (Branke, 2001a).

Em problemas multimodais, é desejável que o algoritmo encontre e mantenha diversos ótimos locais na população, pois estes podem vir a se tornar um ótimo global no novo ambiente.

AE's tradicionais, quando aplicados a problemas de otimização em ambientes dinâmicos, sofrem de falta de adaptação ao novo ambiente, uma vez que o algoritmo tenha convergido. Com isso, o algoritmo, provavelmente, convergirá para um ótimo local neste novo ambiente.

Com o objetivo de resolver este problema, diversas abordagens foram propostas na literatura. Jin & Branke (2005) caracterizam estas abordagens como pertencentes a quatro categorias:

1. **Aumentar a diversidade na população:** Explicitamente aumentar a diversidade da população a cada alteração sofrida pelo sistema. O problema desta abordagem é que o processo de gerar diversidade é equivalente a descartar informações passadas em troca de informações aleatórias (Jin & Branke, 2005). Além disso, é preciso detectar de forma consistente a ocorrência de alterações no ambiente.
2. **Manter a diversidade durante toda a execução:** Evitar a convergência do método, com o objetivo de manter a diversidade da população. A principal desvantagem é alocar uma menor quantidade de indivíduos para explorar as melhores soluções correntes.
3. **Utilização de memórias:** O emprego de formas de armazenar informações anteriores, mostra-se como uma abordagem satisfatória em situações em que o(s) ótimo(s) retorna(m) ao(s) valor(es) passado(s). Alguns autores identificaram que a memória é muito dependente da diversidade, sendo necessária a combinação com uma técnica de preservação da diversidade (Branke, 1999).
4. **Abordagens Multipopulacionais:** A população é dividida em várias subpopulações, permitindo seguir múltiplos picos na superfície de busca. As diferentes subpopulações podem manter informações sobre diversas regiões promissoras. O custo adicional de gerenciar as subpopulações pode ser considerado uma desvantagem desta abordagem.

A combinação destas heurísticas com algoritmos evolutivos cria modelos híbridos capazes de realizar uma busca global e ainda manter diversidade na população de soluções, para se adaptar a novos ambientes.

Além de algoritmos evolutivos, outras meta-heurísticas têm sido aplicadas em problemas de otimização em ambientes dinâmicos, como *Ant Colony Optimization* (ACO) (Guntsch et al., 2000,

2001), *Particle Swarm Optimization* (PSO) (Kennedy & Eberhart, 2001; Mendes et al., 2004), Sistemas Imunológicos Artificiais (SIA) (de França et al., 2005; Hart & Hoss, 1999), entre outros.

2.5.1 Detecção de alterações no ambiente

O aumento da diversidade na população após a ocorrência de uma alteração no ambiente, apontado por Jin & Branke (2005), pressupõe que é sempre possível identificar quando o ambiente sofreu uma alteração. Assim, é necessário definir um procedimento de detecção de alterações.

A abordagem direta é analisar o valor da função de *fitness* das soluções e comparar com o valor obtido na iteração anterior, se for diferente, é considerado que o ambiente foi alterado. A questão é: *Quais soluções devem ser analisadas?*

Branke (1999) e Yuan et al. (2008) empregaram um conjunto de soluções que compõem uma estrutura de memória e, a cada iteração do algoritmo, os valores da função de *fitness* são comparados com os valores da iteração anterior. Já Lung & Dumitrescu (2010), utilizam apenas o melhor indivíduo para detectar as alterações no ambiente. Além das estratégias citadas, outros procedimentos podem ser utilizados, como análise de estatísticas dos valores de *fitness* da população de soluções, como a média e variância.

Em determinados ambientes, nem sempre é possível identificar as alterações (Yang & Yao, 2005). Nestes casos, algoritmos que não requerem a detecção de alterações para disparar alguma ação de controle são mais indicados.

2.6 Considerações finais

Problemas de otimização podem ser encontrados praticamente em todas as áreas de conhecimento. Na indústria, estes problemas se tornam ainda mais críticos e soluções de qualidade devem ser buscadas.

Muitos destes sistemas possuem muitas variáveis e são altamente dinâmicos, o que demanda ferramentas computacionais capazes de encontrar soluções otimizadas em tempo real.

Pesquisas na formulação, solução e análise de ferramentas computacionais tiveram grande avanço e produziram várias metodologias para tratar destes problemas.

Neste capítulo, foram introduzidos alguns conceitos de otimização, tipos de problemas e métodos tradicionais de resolução. Também foi apresentada a classe de problemas de otimização que inclui a variável tempo, necessitando, assim, de ferramentas computacionais dedicadas.

Algumas das principais abordagens para lidar com estes problemas foram brevemente apresentadas, apontando suas vantagens e limitações. Foi dada especial atenção às abordagens

evolutivas para problemas de otimização dinâmicos, que vêm ganhando interesse nos últimos anos devido aos sucessos alcançados em aplicações reais.

Capítulo 3

Estimação de densidade

3.1 Considerações iniciais

Suponha que exista um conjunto de dados observados de um determinado processo e deseja-se conhecer as regras de produção desses dados, ou seja, busca-se criar um modelo que represente este processo para melhor estudá-lo ou gerar novos dados a partir deste modelo.

Uma forma de construir o modelo a partir de um conjunto de dados observados é por meio da análise da distribuição dos dados no espaço. A distribuição provê uma informação importante sobre a espacialidade, assimetria e multimodalidade do espaço amostrado.

Modelando o sistema por uma variável aleatória, busca-se então estimar a função de distribuição de probabilidades que melhor represente os dados amostrados, empregando alguma métrica de distância.

Uma variável aleatória é classificada de acordo com os valores que ela pode assumir, sendo identificada como *discreta* quando assume um conjunto de valores finito ou infinito enumerável (por exemplo, os inteiros), ou *contínua* quando assume um conjunto de valores incontáveis (por exemplo, os reais). A partir disso, a distribuição dos dados é caracterizada dependendo do tipo de variável aleatória.

No caso discreto, a distribuição é caracterizada pela *função massa de probabilidade*, que fornece a probabilidade da variável discreta X assumir o valor x , denotada por $p(x)$. Já para o caso contínuo, as probabilidades são expressas em relação a intervalos de valores. Distribuições contínuas são caracterizadas pela *função densidade de probabilidade* (fdp), denotada por $f(x)$.

Neste estudo, foca-se na análise de espaços contínuos. Logo, serão consideradas metodologias para tratar este tipo de variável.

O problema de estimar a função densidade de probabilidade, em variáveis aleatórias contínuas, é conhecido como *estimação de densidade*. Estimação de densidade envolve a construção de um

modelo para a função densidade de probabilidade a partir dos dados observados (Silverman, 1986). Esses modelos podem ser obtidos por meio de diversos métodos, os quais, basicamente, são divididos em duas classes:

Modelos de estimação paramétricos: Esta classe de métodos assume que os dados seguem uma distribuição paramétrica conhecida, como, por exemplo, a distribuição gaussiana com média μ e variância σ^2 . Busca-se então a estimativa dos seus parâmetros a partir dos dados, seguida da substituição dessas estimativas na fórmula da densidade. Exemplo deste tipo de modelo são os modelos de mistura (Bishop, 2007).

Modelos de estimação não-paramétricos: Estes modelos não fazem suposições *a priori* acerca da distribuição dos dados. Aqui, a estimativa da densidade é feita localmente por um pequeno número de dados vizinhos (Fukunaga, 1990). Exemplo: métodos de *kernel* e *k*-vizinhos mais próximos.

Nas seções seguintes, serão apresentados vários métodos de estimação de densidade, apontando suas vantagens e limitações.

3.2 Métodos de estimação de densidade

A estimação de densidade é uma área bem estabelecida na matemática e estatística, dispondo de diversos métodos para gerar modelos probabilísticos que representam um conjunto de dados. Alguns métodos serão apresentados nesta seção, com ênfase em modelos de estimação paramétricos, os quais foram empregados nesta pesquisa.

3.2.1 Histograma

Um método simples e amplamente utilizado para análise de densidade univariada é o *histograma*. Seja a variável a ser analisada $X \in [a, b]$, dados os limites a e b do intervalo e um número de barras m , pode-se definir o histograma como um conjunto de barras que representam a frequência de dados observados nos m intervalos $(B_i, i = 1, 2, \dots, m)$ de largura h , onde $h = \frac{(b-a)}{m}$. A equação 3.1 mostra os intervalos onde cada barra está definida.

$$B_1 = [a, a + h), B_2 = [a + h, a + 2h), \dots, B_m = [a + (m - 1)h, b] \quad (3.1)$$

Observa-se a necessidade de informar o número de barras m . Existem diversas formas de estimar o parâmetro m , como: regra de Scott (Scott, 1979), de Freedman-Diaconis (Freedman & Diaconis,

1981), de minimização da função risco (Shimazaki & Shinomoto, 2007), entre outras. No entanto, a maioria dessas heurísticas fazem fortes suposições sobre a forma da distribuição.

A figura 3.1 mostra um exemplo de um histograma de dados reais da média anual de precipitação (em polegadas) em 70 cidades norte-americanas no ano de 1975. Neste exemplo, o número de barras foi estimado utilizando a regra de Freedman-Diaconis.

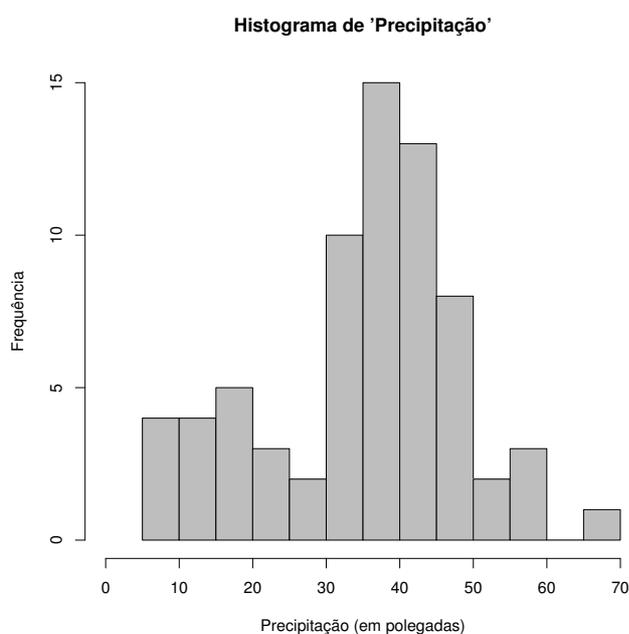


Fig. 3.1: Histograma da média anual de precipitação (em polegadas) de 70 cidades norte-americanas.

Embora o histograma seja útil para apresentação e análise da distribuição dos dados no espaço, ele possui várias restrições como (Silverman, 1986): (1) a descontinuidade da função inviabiliza sua utilização se a derivada da estimativa é necessária; (2) no caso de variáveis no espaço multidimensional, torna-se complicada uma análise espacial dos dados, além de que o número de amostras deve crescer muito rápido com o aumento da dimensão; (3) quando a estimativa de densidade é utilizada como passo intermediário de outros métodos.

3.2.2 Estimação por distribuição de probabilidade paramétrica

Uma forma simples de estimação de densidade ocorre quando se conhece *a priori* a distribuição de probabilidades que gerou o conjunto de dados, ou quando é possível avaliar, entre diversas distribuições, qual a que mais provavelmente os gerou. Estima-se, então, os parâmetros da distribuição e os substitui na expressão analítica. O método mais comum para obtenção dos parâmetros da distribuição é a maximização da verossimilhança dos dados frente ao modelo, o que

pode ser geralmente feito de modo analítico. A figura 3.2 mostra a estimação da variável *precipitação* por uma distribuição gaussiana.

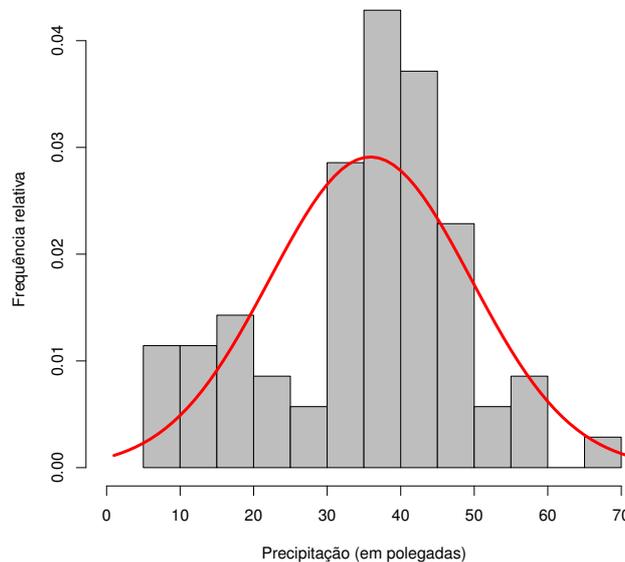


Fig. 3.2: Estimação da variável *precipitação* por uma função densidade de probabilidade gaussiana.

Apesar de ser computacionalmente eficiente, raramente se conhece a distribuição de probabilidade do processo gerador dos dados. A abordagem comumente utilizada é aplicar diversas distribuições conhecidas e avaliar a que melhor se ajusta aos dados. O melhor ajuste pode ser calculado pela maximização da log-verossimilhança dos dados à distribuição (veja mais detalhes na Seção 3.2.4.1).

É possível observar, através da figura 3.2, que essa estimativa não é capaz de representar múltiplos picos em variáveis multimodais. Esta característica ocorre com frequência em aplicações práticas. Neste contexto, outros métodos mais gerais e flexíveis são necessários, os quais são apresentados nas seções a seguir.

3.2.3 Métodos de kernel

Dentro da família de modelos de estimação não-paramétricos, destaca-se o *estimador kernel* (do inglês *kernel density estimator*) (Parzen, 1962; Rosenblatt, 1956), que emprega funções *kernel* para a estimativa da fdp. Aqui, cada dado será representado por uma função *kernel* de abertura h (*bandwidth*). O estimador, então, é a soma de todas as funções *kernel* utilizadas. Formalmente, define-se um estimador *kernel* pela equação 3.2:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right), \quad (3.2)$$

onde n é o número de dados, X_i é o i -ésimo dado do conjunto amostral e $K(\cdot)$ é a função *kernel*.

Algumas das funções *kernel* mais utilizadas são: gaussiana, Epanechnikov, triangular e cosseno. A escolha da função *kernel* não é tão sensível quanto a escolha de h , o qual altera drasticamente a função densidade $\hat{f}(\cdot)$. Diversos métodos para estimar h foram propostos na literatura, dentre eles destacam-se o método de Scott (Scott, 1992), o método de Silverman (Silverman, 1986) e o método por validação cruzada (Rudemo, 1982).

A figura 3.3 mostra o resultado de uma estimação utilizando o estimador *kernel* gaussiano e $h=3,8$, o qual foi definido pelo método de Silverman (Silverman, 1986) (pg. 48), sendo h obtido por:

$$h = 0,9 \cdot \hat{\sigma} \cdot n^{-1/5}, \quad (3.3)$$

onde $\hat{\sigma}$ é estimado por $\hat{\sigma} = \min\{\sigma, IQR/1,34\}$, σ é o desvio padrão do conjunto de dados e IQR é o intervalo interquartil ($Q_3 - Q_1$).

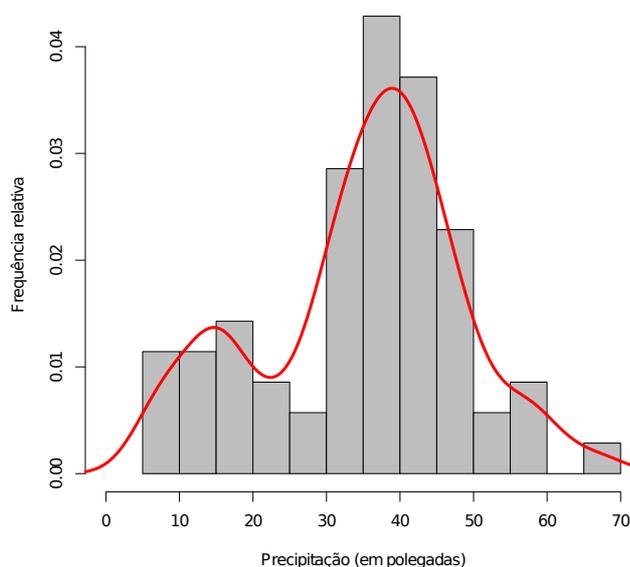


Fig. 3.3: Estimação por estimador *kernel* gaussiano com h igual a 3,8.

Assim como as regras para definir o número de barras para o histograma, as heurísticas citadas para definição de h fazem fortes suposições sobre a distribuição dos dados. Além disso, não há uma heurística melhor que as outras para todos os casos, pois a escolha de h está intrinsecamente ligada à distribuição dos dados (Silverman, 1986).

O estimador *kernel* possui alguns inconvenientes quando aplicado junto a dados provindos de distribuições com cauda longa. Como a abertura do *kernel* (*bandwidth*) é fixa para todos os dados, há uma tendência de ruídos espúrios na cauda da estimativa \hat{f} . Este problema se agrava junto a dados de alta dimensionalidade, onde a cauda exerce uma grande importância no modelo.

Para contornar este problema, vários métodos adaptativos têm sido propostos. Essas abordagens permitem que a abertura dos *kernels* varie de um dado para outro. No entanto, demandam um maior custo computacional. Mais detalhes sobre estimadores *kernel* adaptativos podem ser encontradas em Terrell & Scott (1992).

3.2.4 Modelos de mistura

Na classe de modelos de estimação paramétricos, considera-se que os dados seguem alguma função densidade de probabilidade conhecida, e o problema então consiste em estimar os parâmetros desta distribuição. Uma abordagem mais generalizada é considerar diversas fdps, distintas ou não, e fazer uma “mistura” de forma a melhor representar o conjunto de dados. Esta técnica é conhecida por *modelo de mistura*.

Nesta pesquisa, serão propostos algoritmos que utilizam a estimação de densidade como forma de identificar regiões promissoras no espaço de busca em um problema de otimização em ambientes dinâmicos, empregando algoritmos populacionais. Para a estimação de densidade será feito uso de modelos de estimação paramétricos, mais especificamente um modelo de mistura gaussiano.

Optou-se por utilizar este tipo de estimador pelo fato dele ter mostrado ser computacionalmente eficiente e resultando em um bom desempenho quando empregado em técnicas de otimização (Bosman & Thierens, 1999, 2000a). O estimador *kernel*, apresentado na seção anterior, tem se mostrado um método promissor na sua aplicação em algoritmos de otimização, entretanto, demanda um alto custo computacional (Bosman & Thierens, 2000b), razão pela qual este não foi considerado.

Modelos de mistura se mostram como uma abordagem interessante, devido ao balanço entre flexibilidade e custo computacional. Além disso, seus parâmetros podem ser estimados usando o algoritmo EM, que elimina a necessidade de ajustar os parâmetros manualmente. Este tipo de estimador será devidamente formalizado a seguir.

Considere que dispõe-se de um conjunto de N dados, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, o qual é dividido em K subconjuntos. Cada subconjunto será modelado por uma função densidade de probabilidade (fdp), as quais são denotadas por f_k , $k = \{1, \dots, K\}$, e então seus parâmetros (θ_k) são estimados a partir dos dados. Dependendo da quantidade de dados em cada subconjunto, uma fdp pode representar mais dados do que outras, sendo esta de maior importância para o modelo total dos dados. Assim, para cada fdp é definida uma ponderação que expressa o quanto da totalidade dos dados podem ser explicados por esta fdp, esta informação é conhecida como *coeficiente de mistura* π_k , e satisfaz:

$$0 \leq \pi_k \leq 1 \quad (3.4a)$$

$$\sum_{k=1}^K \pi_k = 1. \quad (3.4b)$$

Portanto, constrói-se um modelo de probabilidades que combina subconjuntos modelados por fdps, denominado *modelo de mistura*, e cada subconjunto é conhecido como *componente*.

Esta técnica é também muito utilizada para análise de *cluster*, dado que ela identifica K agrupamentos de dados e os modela por fdps conhecidas (Bradley et al., 1998).

3.2.4.1 Estimação dos parâmetros

A tarefa de obtenção dos parâmetros da função densidade é conhecida como *estimação de parâmetros*, a qual pode ser resolvida com diversas abordagens, dentre elas a baseada em máxima verossimilhança (do inglês *maximum likelihood*), que é a abordagem padrão para estimação de parâmetros de modelos de mistura (Nurmi, 2008).

Para a estimativa $\hat{\theta}$ dos verdadeiros parâmetros θ , define-se a verossimilhança dos dados aos parâmetros do modelo como a probabilidade condicional $p(\mathbf{x}|\hat{\theta})$, a qual será generalizada como $f(\mathbf{x}; \hat{\theta})$ para denotar qualquer função densidade $f(\cdot)$ com vetor de parâmetros $\hat{\theta}$. Em outras palavras, a verossimilhança informa a probabilidade dos dados terem sido gerados de um processo eficientemente representado por um modelo com esta configuração parâmetros.

Considerando que os dados são independentes e identicamente distribuídos (iid), pode-se definir a verossimilhança conjunta dos dados como $\prod_j f(\mathbf{x}_j; \hat{\theta})$. Usando uma transformação logarítmica, a função de log-verossimilhança assume a forma (Nurmi, 2008):

$$L(\hat{\theta}) = \ln \prod_{j=1}^N f(\mathbf{x}_j; \hat{\theta}) = \sum_{j=1}^N \ln f(\mathbf{x}_j; \hat{\theta}). \quad (3.5)$$

O objetivo do estimador de máxima verossimilhança é encontrar o vetor de parâmetros $\hat{\theta}$ que maximiza a função de verossimilhança da equação 3.5, sendo uma condição necessária para a otimalidade de $\hat{\theta}$ a derivada ser igual a zero, isto é:

$$\frac{\partial L(\hat{\theta})}{\partial \hat{\theta}} = 0. \quad (3.6)$$

Este conceito serve de base para a derivação do algoritmo para aprendizagem dos parâmetros do modelo de mistura, conhecido como *expectation-maximization* (EM) (Dempster et al., 1977).

Quando a fdp gaussiana é usada para representar os subconjuntos, o estimador é chamado de modelo de mistura gaussiano, o qual será utilizado nesta pesquisa. Para uma fdp gaussiana multivariada, o vetor de parâmetros $\hat{\theta}$ corresponde a *média* $\boldsymbol{\mu}$ e *covariância* $\boldsymbol{\Sigma}$. Doravante, a análise será focada no modelo gaussiano.

Neste contexto, o problema de estimação consiste em inferir os $K - 1$ valores da variável latente $\boldsymbol{\pi}$, estimar os K valores das médias ($\boldsymbol{\mu}$) e as K matrizes de covariância ($\boldsymbol{\Sigma}$). A função densidade de probabilidade conjunta $p(\mathbf{x})$, para o modelo gaussiano é:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (3.7)$$

e pode-se definir a função de log-verossimilhança de um modelo de mistura gaussiano como (Bishop, 2007):

$$\ln p(\mathbf{x} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{j=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_j | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \quad (3.8)$$

Considere uma variável aleatória binária K -dimensional \mathbf{Z} , em que um elemento particular z_k é igual a 1 e o restante é igual a 0. Os valores de z_k satisfazem $z_k \in \{0, 1\}$ e $\sum_{k=1}^K z_k = 1$. A distribuição marginal sobre \mathbf{Z} é especificada em termos dos coeficientes de mistura $\{\pi_k\}$, como segue (Bishop, 2007):

$$p(z_k = 1) = \pi_k, \quad (3.9)$$

onde os parâmetros $\pi_k, k = \{1, \dots, K\}$ devem satisfazer 3.4a e 3.4b. Outra informação importante é a probabilidade condicional de \mathbf{Z} dado \mathbf{x} e usa-se $\gamma(z_k)$ para denotar $p(z_k = 1 | \mathbf{x})$, cujos valores podem ser obtidos usando o teorema de Bayes, como será visto mais adiante.

A partir disso, π_k pode ser interpretada como a probabilidade *a priori* de $z_k = 1$ e a informação $\gamma(z_k)$ como a probabilidade *a posteriori* correspondente, uma vez observada a variável \mathbf{x} . Além disso, $\gamma(z_k)$ pode ser vista como a responsabilidade atribuída ao k -ésimo componente do modelo na explicação das observações \mathbf{X} (Bishop, 2007).

Em particular, é possível representar o valor da responsabilidade do componente associado a cada dado \mathbf{x}_n , denotada por $\gamma(z_{nk})$, podendo ser interpretada como a probabilidade do dado \mathbf{x}_n ter sido gerado pelo k -ésimo componente do modelo de mistura. Com a convergência do algoritmo EM, estas probabilidades convergem para valores 0's ou 1's. A probabilidade $\gamma(z_{nk})$ é calculada usando o teorema de Bayes, como:

$$\begin{aligned} \gamma(z_{nk}) \equiv p(z_{nk} = 1 | \mathbf{x}_n) &= \frac{p(z_{nk} = 1)p(\mathbf{x}_n | z_{nk} = 1)}{\sum_{j=1}^K p(z_{nj} = 1)p(\mathbf{x}_n | z_{nj} = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{aligned}$$

Na literatura de modelos estatísticos, a variável \mathbf{Z} é conhecida por *variável latente*, sendo essa uma variável não-observada que deve ser inferida por algum procedimento matemático, a partir das variáveis observadas. O algoritmo EM para treinamento do modelo manipula eficientemente este tipo de variável, inferindo e utilizando-a no processo de estimação dos parâmetros do modelo de mistura.

A maximização da função de log-verossimilhança (equação 3.8) é dificultada pela presença do somatório sobre os K componentes dentro do logaritmo, de modo que o logaritmo não atua diretamente sobre a gaussiana (Bishop, 2007). Dentre as abordagens para atacar este problema, pode-se destacar técnicas de otimização baseadas em gradiente (Nocedal & Wright, 1999) e o bem conhecido algoritmo EM que tem sido largamente aplicado (Bishop, 2007).

3.2.4.2 Algoritmo EM

O EM é um poderoso método para encontrar soluções de máxima verossimilhança para modelos com variáveis latentes (Bishop, 2007). O Algoritmo 3 apresenta o pseudo-código do EM para o treinamento de um modelo de mistura gaussiano.

A cada atualização dos parâmetros resultantes do passo-E seguido por um passo-M, é garantido um aumento na função de log-verossimilhança, sendo que esta função geralmente possui vários máximos locais e o EM garante convergência para um máximo local, sendo dependente dos valores iniciais dos parâmetros a serem maximizados (Nurmi, 2008). Com isso, a geração de bons valores iniciais é importante para alcançar o máximo global, sendo métodos heurísticos indicados para esta tarefa, como a aplicação do método *K-means* (Lloyd, 1982) para obter as médias iniciais e as matrizes de covariâncias, as quais são calculadas a partir dos dados pertencentes a cada *cluster* gerado pelo *K-means*.

Outra dificuldade para execução do EM é a necessidade de definir o número adequado de componentes que melhor represente os dados. Uma alternativa geralmente utilizada é a aplicação de métricas de pontuação e, então, seleciona-se o modelo com melhor pontuação. Dentre estas métricas, destacam-se as fundamentadas em conceitos de teoria da informação, derivadas da divergência de Kullback-Leibler, conhecidas como *critério de informação* (do inglês *Information*

Algoritmo 3: Pseudocódigo do algoritmo EM.

Entrada: Dados \mathbf{X} e número de componentes K

Saída: Parâmetros $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ e $\boldsymbol{\pi}$

1. Faça $t = 0$ e inicialize os parâmetros das gaussianas, médias $\boldsymbol{\mu}_k^{(t)}$, covariâncias $\boldsymbol{\Sigma}_k^{(t)}$ e os coeficientes de mistura $\pi_k^{(t)}$ e calcule o valor inicial da função de log-verossimilhança.
2. **Passo-E.** Calcule as probabilidades *a posteriori* usando os valores dos parâmetros correntes:

$$\gamma(z_{nk}) = \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{i=1}^K \pi_i^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i^{(t)})} \quad (3.10)$$

3. **Passo-M.** Re-estime os parâmetros usando as probabilidades *a posteriori* correntes:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (3.11)$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^T \quad (3.12)$$

$$\pi_k^{(t+1)} = \frac{N_k}{N} \quad (3.13)$$

onde

$$N_k = \sum_{n=1}^N \gamma(z_{nk}) \quad (3.14)$$

4. Calcule a log-verossimilhança com os novos parâmetros:

$$\ln p(\mathbf{x} | \boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)}) = \sum_{j=1}^N \ln \left\{ \sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_j | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \right\}. \quad (3.15)$$

e verifique a convergência, seja dos parâmetros ou da função de log-verossimilhança. Caso o critério de convergência não seja satisfeito, faça $t = t + 1$ e retorne ao passo 2.

Criterion). Esses critérios ponderam a verossimilhança dos dados com o modelo, penalizando modelos complexos, quando um número grande de parâmetros deve ser ajustado. Dentre os mais utilizados, destacam-se o *Akaike Information Criterion* (AIC) e o *Bayesian Information Criterion* (BIC), os quais são descritos pelas equações 3.16 e 3.17, respectivamente:

$$AIC = -2 \ln L(\hat{\theta}) + 2d, \quad (3.16)$$

onde $\ln L(\hat{\theta})$ é o máximo da função de log-verossimilhança encontrado pelo método de treinamento do modelo, como o algoritmo EM; e d é o termo de complexidade do modelo, consistindo do número de parâmetros estimados.

$$BIC = -2 \ln L(\hat{\theta}) + d \ln N, \quad (3.17)$$

onde N é o número de dados disponíveis.

Quando a estimativa de máxima verossimilhança é aplicada no problema de estimação dos parâmetros do modelo, é possível aumentar a verossimilhança adicionando mais componentes e, conseqüentemente, mais parâmetros, podendo resultar em sobre-treinamento. Tanto o BIC quanto o AIC resolvem este problema empregando o termo de penalização para o número de parâmetros no modelo (McLachlan & Peel, 2000). BIC penaliza mais fortemente, do que o AIC, modelos mais complexos (Nurmi, 2008). O modelo escolhido é aquele que obtiver menor pontuação, seja utilizando AIC ou BIC.

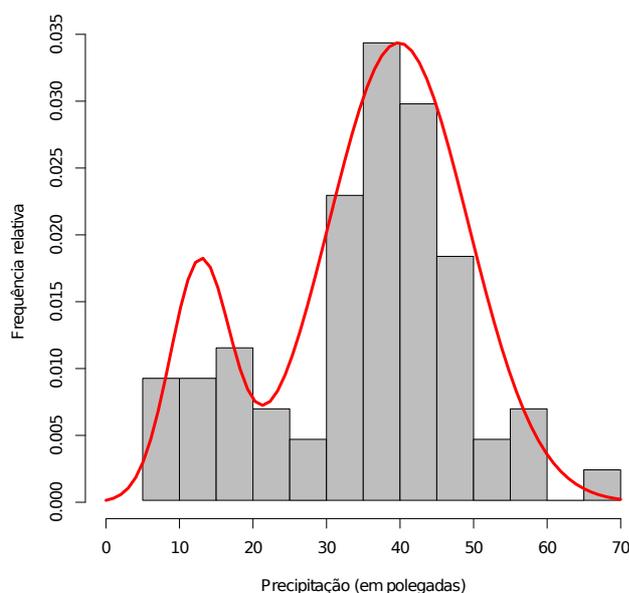


Fig. 3.4: Estimação da variável *precipitação* por um modelo de mistura gaussiano.

A figura 3.4 apresenta a estimação, utilizando modelo de mistura gaussiano com dois componentes, dos dados de precipitação citados anteriormente. Aqui, o número de componentes do modelo de mistura foi definido com base na análise do BIC de vários modelos com diferentes quantidades de componentes, escolhendo aquele com menor BIC, como mostra a figura 3.5. É possível observar que a menor pontuação ocorreu quando foram utilizados dois componentes.

A proposta de algoritmo para otimização em ambientes dinâmicos, a ser apresentada no Capítulo 5, utilizará o BIC como medida para identificar o número adequado de componentes no modelo de

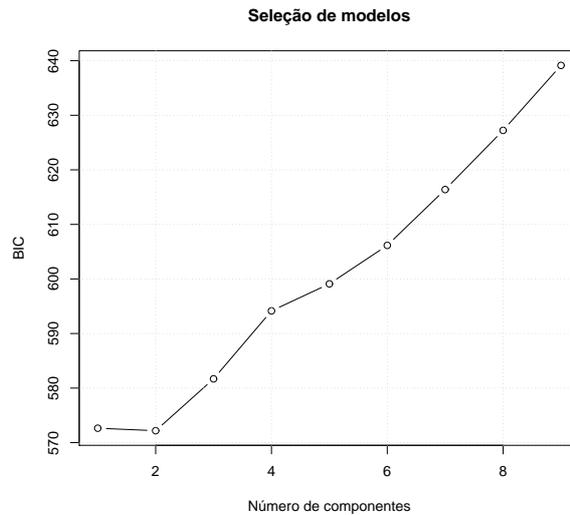


Fig. 3.5: Pontuações BIC para diversos modelos analisados.

mistura gaussiano.

3.2.4.3 Versão Online do algoritmo EM

O algoritmo EM, apresentado na seção anterior, pressupõe que uma certa quantidade de dados já está disponível. Porém, em alguns problemas, não há de antemão esse conjunto de dados, pois os dados chegam de forma contínua. Para este tipo de problema, uma versão *online* de treinamento é necessária. Diversas propostas de algoritmos *online* podem ser encontradas na literatura, dentre elas a versão de Neal & Hinton (1998), que será utilizada neste trabalho.

O conceito central desta proposta é armazenar um vetor de *estatísticas suficientes* dos dados e, assim, estimar os parâmetros do modelo de mistura a partir deste vetor. Estatísticas suficientes correspondem a uma função $s(\cdot)$ do conjunto de dados que contém toda a informação relevante e necessária para a estimação dos parâmetros (Duda et al., 2001).

Para exemplificar, em um modelo de mistura gaussiano com dois componentes, as estatísticas suficientes $s(\cdot)$ para o i -ésimo dado (\mathbf{x}_i) é descrito pela equação 3.18 (Neal & Hinton, 1998):

$$s(\mathbf{x}_i, z_i) = [(1 - \mathbf{x}_i), (1 - \mathbf{x}_i)z_i, (1 - \mathbf{x}_i)z_i^2, \mathbf{x}_i, \mathbf{x}_iz_i, \mathbf{x}_iz_i^2], \quad (3.18)$$

onde \mathbf{Z} a variável binária não-observada que indica de qual gaussiana, das duas possíveis, o dado foi gerado e o conjunto de parâmetros é dado por $\hat{\boldsymbol{\theta}} = (\boldsymbol{\pi}, \boldsymbol{\mu}_1, \boldsymbol{\sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\sigma}_2)$.

Dado $s(\mathbf{x}, z) = \sum_i s(\mathbf{x}_i, z_i) = (n_1, m_1, q_1, n_2, m_2, q_2)$, a estimativa dos parâmetros é dada pelas equações 3.19.

$$\pi = \frac{n_2}{n_1 + n_2} \quad (3.19a)$$

$$\mu_1 = \frac{m_1}{n_1} \quad (3.19b)$$

$$\sigma_1 = \frac{q_1}{n_1} - \left(\frac{m_1}{n_1}\right)^2 \quad (3.19c)$$

$$\mu_2 = \frac{m_2}{n_2} \quad (3.19d)$$

$$\sigma_2 = \frac{q_2}{n_2} - \left(\frac{m_2}{n_2}\right)^2 \quad (3.19e)$$

Partindo de um valor inicial $s^{(0)}$, podendo ser ou não consistente com os parâmetros iniciais $\hat{\theta}^{(0)}$, as iterações subsequentes do EM *online* procedem como mostra o Algoritmo 4.

Algoritmo 4: Pseudocódigo da versão *Online* do algoritmo EM.

Entrada: Dados X

Saída: Parâmetros $\hat{\theta}$

Passo E: Escolha um dado i para ser atualizado.

Faça $s_j^{(t)} = s_j^{(t-1)}$ para $j \neq 1$

Faça $s_i^{(t)} = E_{P_i}[s_i(\mathbf{x}_i, z_i)]$, para $P_i(\mathbf{x}_i) = P(\mathbf{x}_i|z_i, \hat{\theta}^{(t-1)})$

Faça $s^{(t)} = s^{(t-1)} - s_i^{(t-1)} + s_i^{(t)}$

Passo M: Atribua a $\hat{\theta}^{(t)}$ os parâmetros $\hat{\theta}$ com máxima verossimilhança, dado $s^{(t)}$.

Com isso o algoritmo EM *online* vai salvando as estatísticas de cada dado de forma a incorporá-los ao modelo de mistura, removendo dados antigos ao inserir os novos. Portanto, é possível evitar que o modelo, no decorrer da execução, se torne rígido e insensível às alterações do sistema modelado.

Outras estratégias que buscam privilegiar dados recentes foram abordadas por Neal & Hinton (1998), dentre elas o emprego de um decaimento exponencial na participação dos dados adicionados anteriormente ao modelo.

3.3 Considerações finais

Estimação de densidade é um importante tópico de pesquisa em matemática e estatística, que tem por objetivo capturar distribuições desconhecidas de um conjunto de dados gerados por uma variável contínua. Uma estimativa de densidade adequada pode ser explorada com o intuito de prover

informações sobre as principais características dos dados. Com isso, podem alimentar outros métodos que necessitam explorar o espaço da variável observada.

Neste capítulo, buscou-se discutir formas de estimar a densidade de um conjunto de dados, seja empregando métodos mais simples como o histograma, até métodos mais robustos como o estimador *kernel* e os modelos de mistura, sendo estes últimos melhor detalhados, por serem adotados nesta pesquisa.

Capítulo 4

Algoritmos Evolutivos e Algoritmos de Estimação de Distribuição

4.1 Considerações iniciais

A teoria da evolução das espécies, descrita por Darwin (1859), configura entre as teorias mais bem aceitas para explicar a existência de uma variedade (diversidade) tão grande de seres vivos na natureza. Nela é apresentada a “seleção natural” como o principal mecanismo para a manutenção das variações favoráveis à sobrevivência e reprodução de um organismo em seu ambiente. Darwin (1859) sugeriu que, para haver evolução, é necessário que uma população de indivíduos esteja sujeita a: *reprodução com herança, variação e seleção natural*.

A teoria evolucionista de Darwin foi levada ao computador inicialmente para a simulação de processos naturais e, posteriormente, foi estendida para a otimização de sistemas, sendo esta área de estudo denominada de *Computação Evolutiva*. Métodos presentes nesta linha de pesquisa utilizam um procedimento artificial inspirado na seleção natural, juntamente com a aplicação de operadores genéticos de recombinação e mutação.

Este capítulo visa introduzir os algoritmos de estimação de distribuição (AEDs), que são métodos evolutivos que utilizam técnicas de estimação de distribuição no espaço de busca, ao invés de operadores genéticos. Inicialmente, a computação evolutiva é apresentada, dando maior ênfase ao *algoritmo genético* e destacando a teoria dos *blocos construtivos*, que busca explicar o seu funcionamento. Os problemas que emergiram desta análise motivaram o desenvolvimento dos AEDs.

4.2 Computação Evolutiva

A evolução admite uma interpretação na forma de um processo de otimização, que procura por melhores soluções dentro de um espaço de busca. Sendo assim, este processo procura identificar regiões mais promissoras na superfície de adaptação, ou seja, regiões que contêm indivíduos com melhores índices de adaptação ou *fitness*. Sob a óptica matemática, a superfície de adaptação pode ser a função que se deseja maximizar (ou minimizar). Assim, soluções ótimas locais se encontram em extremos desta superfície.

Um algoritmo evolutivo pode ser definido como um procedimento iterativo de busca (otimização) inspirado nos mecanismos evolutivos biológicos. E a *Computação Evolutiva* se configura como a área de pesquisa, dentro da linha maior de *inteligência computacional*, que trata dos algoritmos evolutivos.

O algoritmo evolutivo padrão é constituído de quatro elementos básicos:

1. Uma população de indivíduos, onde cada indivíduo corresponde a uma solução candidata;
2. Uma maneira de criar novos indivíduos, a partir de indivíduos já existentes, e inserir variabilidade nos descendentes;
3. Uma forma de medir a qualidade de cada indivíduo;
4. Um método de selecionar os melhores indivíduos, aplicando assim o princípio da seleção natural.

Na população, cada indivíduo é codificado em uma estrutura de dados, geralmente um vetor numérico que representa de forma não-ambígua, uma possível solução do problema tratado. Novos indivíduos vão sendo gerados, por meio da reprodução, produzindo descendentes que herdam características dos pais. Para a inserção de variabilidade genética, utiliza-se alguma forma de mutação nos filhos gerados, o que promove uma maior exploração na superfície de adaptação. O nível de adaptação dos descendentes é calculado e a nova população (geração seguinte) será escolhida de tal forma a privilegiar indivíduos mais adaptados, que correspondem a soluções melhores do problema em questão. Na literatura de computação evolutiva, as soluções codificadas são comumente chamadas de *cromossomos*, fazendo referência à função dos cromossomos biológicos.

Embora não tenham sido mencionadas ainda, existem aplicações de algoritmos evolutivos em que a solução corresponde a toda a população de indivíduos a cada geração, e não a um indivíduo em particular. Este enfoque não será adotado neste trabalho.

Existem várias propostas de algoritmos evolutivos na literatura, divergindo entre si pelo tipo de codificação, operadores genéticos utilizados, forma de selecionar indivíduos que comporão a nova população e sequência de aplicação de operadores genéticos e do processo de seleção.

Com suas raízes no trabalho de Holland (1975), o *Algoritmo Genético* (AG) foi desenvolvido para a simulação de sistemas naturais e teve uma grande expansão no campo da otimização combinatória com o trabalho de Goldberg (1989). A partir de então, suas aplicações logo se estenderam à otimização numérica. Na próxima subseção, os algoritmos genéticos serão abordados em mais detalhes. No que segue, serão mencionados outros algoritmos evolutivos de interesse histórico e prático.

As Estratégias Evolutivas foram desenvolvidas por Rechenberg (1973) e Schwefel (1981) no contexto de otimização em espaços contínuos. Já a Programação Evolutiva (Fogel et al., 1966), proposta inicialmente para evolução de máquinas de estado finito, teve também suas aplicações no campo de otimização numérica. A Programação Genética (Koza, 1992) foi originalmente proposta para evoluir programas de computador através da manipulação de suas árvores sintáticas.

Esses algoritmos evolutivos, suas variantes e extensões, assim como novas propostas de algoritmos evolutivos como a Evolução Diferencial (Storn & Price, 1997), ajudaram a consolidar a Computação Evolutiva como uma área de pesquisa ativa e voltada para a resolução de problemas de interesse prático.

4.2.1 Algoritmo Genético

Algoritmo Genético é uma meta-heurística de otimização populacional baseada no mecanismo de seleção natural proposto por Darwin e em operadores genéticos de recombinação e mutação.

O algoritmo genético simples (AGS) proposto por Holland (1975) manipula um conjunto (população) de soluções codificadas (indivíduos) em cadeias binárias de tamanho fixo. Esses indivíduos, ao longo do processo evolutivo, tendem a povoar melhores regiões do espaço de busca (as quais são caracterizadas por conterem indivíduos com melhores índices de adaptação), por meio de um processo estocástico de seleção, recombinação e mutação dos indivíduos.

No AGS as soluções são codificadas na forma de cadeias binárias de tamanho fixo d , isto é, $c = \{0, 1\}^d$. Sendo assim, na aplicação da evolução de variáveis contínuas ou inteiras, uma forma de convertê-las em cadeias binárias se faz necessária.

Formalmente, pode-se definir um AGS como uma heptupla da forma:

$$\text{AGS} = \{C, \mathcal{P}^n, S, \mathcal{R}, \mathcal{M}, p_r, p_m\}$$

onde,

C : método para codificação dos indivíduos;

\mathcal{P}^n : população de n indivíduos codificados;

S : operador estocástico de seleção;

\mathcal{R} : operador estocástico de recombinação;

\mathcal{M} : operador estocástico de mutação;

p_r : probabilidade de ocorrência de recombinação por indivíduo;

p_m : probabilidade de ocorrência de mutação por *bit*.

A recombinação (ou *crossover*) combina importantes segmentos de diferentes indivíduos, dada uma probabilidade de ocorrência (p_r) por indivíduo. Valores típicos para p_r geralmente se encontram no intervalo $[0,6;1,0]$ (Bäck & Schwefel, 1993). O operador de recombinação $\mathcal{R}_{\{p_r\}} : \mathcal{P}^n \rightarrow \mathcal{P}^m$, recebe uma população \mathcal{P}^n e troca segmentos entre os indivíduos selecionados (via operadores de seleção), gerando uma nova população \mathcal{P}^m , podendo gerar $m \neq n$ indivíduos. As trocas são feitas dois a dois. Sejam $u = (u_1, \dots, u_d)$ e $v = (v_1, \dots, v_d)$ dois indivíduos pais, selecionados aleatoriamente da população, o operador de recombinação de *um-ponto*, gera dois novos indivíduos (descendentes), u' e v' , de acordo com:

$$\begin{aligned} u' &= (u_1, \dots, u_{\chi-1}, u_{\chi}, v_{\chi+1}, \dots, v_d) \\ v' &= (v_1, \dots, v_{\chi-1}, v_{\chi}, u_{\chi+1}, \dots, u_d) \end{aligned} \quad (4.1)$$

onde χ é uma variável aleatória com distribuição uniforme no espaço amostral $\{1, \dots, d-1\}$, e um dos descendentes é escolhido aleatoriamente como resultado da recombinação. O método de *um-ponto* pode ser naturalmente estendido a uma recombinação generalizada de *múltiplos pontos*.

A mutação é feita complementando o valor de posições da cadeia, seguindo uma probabilidade (p_m) de mutação por posição, normalmente com valores muito pequenos, da ordem de 10^{-2} ou 10^{-3} . Sendo c uma cadeia binária qualquer, pode-se representar o operador de mutação $\mathcal{M}_{\{p_m\}} : \mathcal{P}^m \rightarrow \mathcal{P}^m$, de tal forma que $\forall i \in (1, \dots, d)$:

$$c'_i = \begin{cases} c_i & , \chi_i > p_m \\ 1 - c_i & , \chi_i \leq p_m \end{cases} \quad (4.2)$$

onde χ_i é uma variável aleatória com distribuição uniforme no intervalo contínuo $[0,1]$. O operador de mutação no AGS foi tido como um operador secundário, havendo maior ênfase na recombinação.

O método de seleção utilizado pelo AGS enfatiza uma regra de sobrevivência probabilística proporcional ao *fitness*, denominado posteriormente de *método da roleta*. Holland identificou a necessidade de utilizar uma seleção proporcional, de forma a balancear a exploração global do espaço de busca e a exploração de regiões promissoras. Para o operador de seleção $\mathcal{S} : \mathcal{P}^m \rightarrow \mathcal{P}^n$, a probabilidade (p_i) de um indivíduo c_i ser selecionado para compor a nova população, dada a função de *fitness* $\Phi(\cdot)$, é obtida de acordo com:

$$p_i(c_i) = \frac{\Phi(c_i)}{\sum_{j=1}^m \Phi(c_j)}. \quad (4.3)$$

A partir das probabilidades calculadas, n indivíduos são amostrados para comporem a nova população. O pseudocódigo do AGS é mostrado no Algoritmo 5.

Algoritmo 5: Pseudocódigo do Algoritmo Genético Simples (AGS).

```

1 início
2    $t \leftarrow 0$ ;
3   Inicialize (aleatoriamente):  $\mathcal{P}(t) = \{c^i, \dots, c^n\}, \forall i, c^i \in \{0, 1\}^d$ ;
4   Avaliação:  $\Phi(\mathcal{P}(t))$ ;
5   enquanto condição de parada não for satisfeita faça
6     Recombinação:  $\mathcal{R}_{\{p_c\}}(\mathcal{P}(t))$ ;
7     Mutação:  $\mathcal{M}_{\{p_m\}}(\mathcal{P}(t))$ ;
8     Avaliação:  $\Phi(\mathcal{P}(t))$ ;
9     Seleção:  $\mathcal{P}(t+1) \leftarrow \mathcal{S}(\mathcal{P}(t))$ ;
10     $t \leftarrow t + 1$ ;
11  fim
12 fim

```

O AGS recebe um conjunto de elementos da heptupla e o algoritmo executa esta sequência de operações até que o critério de parada seja satisfeito.

Holland (1975) ainda utilizou outro operador genético, denominado inversão. Neste operador, um indivíduo é escolhido, dada uma probabilidade de seleção por indivíduo, e são selecionadas aleatoriamente duas posições da cadeia e seus conteúdos são trocados. Segundo Holland (1975), este operador visa alterar a informação de ligação entre as variáveis, corroborando com o fato de que tentativas foram feitas para encontrar dependências entre as variáveis. Este operador não é comumente empregado nos algoritmos atuais.

4.2.2 Modelagem teórica dos algoritmos genéticos

Muitos foram os esforços empregados no estudo teórico dos algoritmos genéticos, a fim de formalizar e explicar o seu funcionamento. O resultado mais comumente utilizado é a *Teoria dos Esquemas*, proposta por Holland (1975), que foi elaborada sobre o AGS.

Holland (1975) considera que, em cada população, o algoritmo genético amostra subespaços do espaço de busca e estes são definidos por meio de um *esquema*. Um esquema é um modelo (*template*)

de um cromossomo que representa um conjunto de cromossomos.

Definição 4.1 Em representação binária, um esquema $\xi = (\xi_1, \xi_2, \dots, \xi_d)$ é uma cadeia pertencente a $\{0, 1, *\}^d$ que representa um subespaço de $\mathbb{B}^d = \{0, 1\}^d$, tal que uma cadeia $\mathbf{x} \in \mathbb{B}^d$ pertence ao esquema se satisfaz a condição:

$$x_i \neq \xi_i \Leftrightarrow \xi_i = *, \forall i = 1, 2, \dots, d.$$

Em outras palavras, um esquema representa um conjunto de cromossomos que podem ser representados por ele, onde o símbolo $*$ (*don't care*) indica que aquela posição pode receber qualquer valor pertencente ao alfabeto $\{0, 1\}$.

Por exemplo, o esquema $(01*1)$ simboliza os cromossomos (0101) e (0111) . Fica claro que o esquema (0001) representa um único cromossomo, e $(****)$ representa todos os cromossomos de tamanho igual a 4, ou seja, 16 cromossomos. Em termos gerais, um esquema pode representar 2^r cromossomos, onde r é o número de símbolos *don't care* presentes no esquema.

Outros dois conceitos são necessários para introduzir o teorema dos esquemas de Holland (1975), que são: a *ordem* e o *comprimento* do esquema.

Definição 4.2 Dado um esquema ξ , é definida a ordem de ξ , $o(\xi)$, como o número de posições fixas em ξ , isto é, o número de posições com valores 0 ou 1. O comprimento de ξ , $l(\xi)$, é a diferença entre a posição mais à direita e a mais à esquerda que contém símbolos fixos.

Por exemplo, o esquema $\xi = (011*1*1*)$ possui comprimento $l(\xi) = 7 - 1 = 6$ e sua ordem é igual a $o(\xi) = 5$. A ordem define a especialidade do esquema, enquanto que o comprimento, o seu nível de compactação de informação.

A seguir, é introduzida a equação do crescimento reprodutivo do esquema (*reproductive schema growth equation*), que define o número esperado de indivíduos pertencentes ao esquema ξ para a próxima geração.

Seja $N_\xi(t)$ o número de indivíduos que pertencem ao esquema ξ no tempo t ; $\hat{\mu}_\xi(t)$ representa a média do valor de *fitness* dos indivíduos do esquema ξ no tempo t ; $\hat{\mu}(t)$ é a média dos valores de *fitness* da população no tempo t ; n é o número de posições do esquema; p_r é a probabilidade de recombinação e p_m , a probabilidade de mutação.

Teorema 4.1 Considere um algoritmo genético simples: seleção por roleta, recombinação de um ponto e mutação por inversão. O número esperado de descendentes pertencentes a algum esquema ξ na próxima geração de um algoritmo genético, $E[N_\xi(t+1)]$, é tal que:

$$E[N_\xi(t+1)] \geq N_\xi(t) \cdot \frac{\hat{\mu}_\xi(t)}{\hat{\mu}(t)} \left(1 - p_r \cdot \frac{l(\xi)}{n-1} \right) (1 - p_m)^{o(\xi)} \quad (4.4)$$

A equação 4.4 estabelece que o número de indivíduos pertencentes ao esquemas ξ crescerá exponencialmente na próxima geração, isso se a média dos valores de *fitness* destes indivíduos for maior do que a média dos valores de *fitness* de toda a população.

O resultado final da equação de crescimento de um esquema é o Teorema dos Esquemas, proposto por Holland (1975):

Teorema 4.2 (*Teorema dos Esquemas*) *Esquemas curtos, com baixa ordem e com fitness acima da média da população aumentam exponencialmente sua participação em gerações subsequentes.*

A partir deste teorema, Goldberg (1989) propôs a hipótese dos blocos construtivos:

Hipótese 4.1 (*Hipótese dos Blocos Construtivos*) *Um algoritmo genético apresenta um desempenho quase-ótimo através da justaposição de esquemas curtos, de baixa ordem e de alto desempenho, chamados de blocos construtivos.*

Embora tentativas tenham sido feitas para provar esta hipótese (Bethke, 1981), para a maioria dos problemas a hipótese era sustentada pelos resultados empíricos do algoritmo. Porém, para alguns problemas, esta hipótese é facilmente violada. Nestes casos, dois blocos construtivos combinados resultam em um de menor qualidade. Este fenômeno é conhecido como decepção (*deception*): alguns blocos construtivos podem iludir o algoritmo genético e causar convergência para soluções ótimas locais (Michalewicz, 1996).

Este tipo de fenômeno ocorre com cromossomos com alta *epistasia*. Em um AG, entende-se por epistasia o nível de interação dos genes de um cromossomo, ou seja, quanto o valor de um gene influencia no valor de outros genes.

A hipótese dos blocos construtivos não fornece uma explicação do motivo pelo qual o AG funciona, apenas indica os motivos pelos quais ele funciona para uma determinada classe de problemas (Iyoda, 2000).

Na próxima seção, será apresentada uma nova classe de algoritmos, os Algoritmos de Estimação de Distribuição, que, além da vantagem de não romperem os blocos construtivos, eliminam a dependência de vários parâmetros associados ao algoritmo genético.

4.3 Computação evolutiva baseada em modelagem probabilística

Um algoritmo genético simples (AGS), como visto nas seções anteriores, é um algoritmo populacional que guia a exploração do espaço de busca por meio da aplicação de seleção e de

operadores genéticos, particularmente recombinação e mutação. Um AGS é usualmente aplicado a problemas em que as soluções são representadas, ou podem ser mapeadas, em cadeias de comprimentos fixos, sobre um alfabeto finito (Pelikan et al., 2002).

A teoria dos esquemas e a hipótese dos blocos construtivos, que figuram entre os conceitos mais aceitos para explicar o funcionamento do AGS, afirmam que o algoritmo manipula implicitamente um grande número de blocos construtivos, através dos mecanismos de seleção e recombinação, reproduzindo e mesclando-os (Pelikan et al., 2000a). Entretanto, para alguns tipos de problemas, os operadores rompem os blocos construtivos, fazendo com que o algoritmo não consiga manipulá-los de maneira eficiente. Como visto na Seção 4.2.2, isso ocorre devido à interação das variáveis do problema - alta epistasia. O problema de rompimento de blocos construtivos é muitas vezes referenciado como *problema de ligação (linkage problem)*.

Holland (1975) já havia reconhecido que, se a interação das variáveis fosse utilizada, certamente traria benefícios ao AGS. Esta fonte de informação, até então inexplorada, foi chamada de *informação de ligação (linkage information)*.

Partindo deste contexto, foram propostos algoritmos em que, ao invés de utilizar operadores de recombinação e mutação, novos indivíduos são gerados a partir de informações extraídas do conjunto de soluções promissoras presentes na população daquela geração. Uma maneira de utilizar a informação global sobre um conjunto de soluções promissoras é estimar sua distribuição e usar esta estimativa para gerar novos indivíduos. Os algoritmos que fazem uso deste princípio são chamados de Algoritmos de Estimação de Distribuição - AED (do inglês *Estimation of Distribution Algorithm*) (Mühlenbein & Paaß, 1996). Essas distribuições contemplam diretamente as interações das variáveis que compõem o espaço de busca.

Formalmente, pode-se definir um AED como uma quádrupla da forma:

$$\text{AED} = \{\mathcal{C}, \mathcal{P}^n, \mathcal{S}, \mathcal{E}\}$$

onde,

\mathcal{C} : método para codificação dos indivíduos;

\mathcal{P}^n : população de n indivíduos codificados;

\mathcal{S} : operador estocástico (ou determinístico) para seleção;

\mathcal{E} : Estimador da distribuição das melhores soluções (na iteração corrente).

Nota-se a eliminação dos componentes de recombinação, de mutação e suas probabilidades, em relação ao AGS. No entanto, a tarefa de estimação deverá ser feita levando em conta a complexidade do modelo utilizado, como será visto nas seções seguintes. Dependendo do modelo de estimação utilizado, alguns parâmetros adicionais deverão ser informados.

Diferentemente do AGS, em que as inter-relações (blocos construtivos) das variáveis representando os indivíduos são mantidas implicitamente, nos AEDs essas relações são expressadas explicitamente através da distribuição de probabilidade conjunta associada aos indivíduos selecionados em cada iteração.

A sequência estrutural do AED é bastante similar ao AGS, como mostra o Algoritmo 6, diferindo na maneira como são gerados novos indivíduos. A seleção das soluções promissoras pode ser realizada de forma determinística ou estocástica, privilegiando indivíduos com melhores valores de *fitness*. A população pode ser amostrada integralmente a partir da distribuição de probabilidade; ou, visando maior exploração do ambiente, estipular uma porcentagem para adição de novos indivíduos gerados aleatoriamente; ou ainda substituindo alguns indivíduos da população anterior pelos novos indivíduos amostrados pela distribuição.

Algoritmo 6: Pseudocódigo do Algoritmo de Estimação de Distribuição

```
1 início
2    $t \leftarrow 0$ ;
3   Gera população inicial  $\mathcal{P}(t)$  aleatoriamente;
4   enquanto condição de parada não for satisfeita faça
5     Avaliação:  $\Phi(\mathcal{P}(t))$ ;
6     Seleção:  $\mathcal{S}(\mathcal{P}(t))$ ;
7     Estimação da distribuição de probabilidades,  $\mathcal{E}$ , do conjunto selecionado;
8     Amostre a nova população  $\mathcal{P}(t + 1)$  a partir de  $\mathcal{E}$ ;
9      $t \leftarrow t + 1$ ;
10  fim
11 fim
```

Exemplos de critérios de parada são: um número fixo de iterações, um número fixo de diferentes avaliações de *fitness* dos indivíduos e ausência de melhorias do melhor indivíduo em um certo número de iterações subsequentes.

O passo fundamental nesta classe de algoritmos é como estimar a distribuição das soluções promissoras. Na realidade, a estimação da distribuição de probabilidade conjunta associada às variáveis, a partir de soluções promissoras selecionadas, constitui o gargalo destes algoritmos. Será necessário então um balanço entre a precisão desta estimação e seu custo computacional.

Os algoritmos presentes nesta classe diferenciam-se basicamente pela forma como a estimação é realizada, sendo classificados de acordo com a complexidade do modelo probabilístico utilizado: *variáveis sem dependência*; *dependência aos pares*; *dependência multivariada* e *modelos de mistura* (Larrañaga, 2001).

A seguir, serão apresentadas algumas abordagens de AEDs, tanto para problemas de otimização em espaços discretos, quanto em espaços contínuos. Esta revisão será organizada considerando os quatro níveis de complexidade do modelo probabilístico, citados acima e utilizados para extrair as interdependências das variáveis que formam o espaço de busca.

4.3.1 Modelos sem dependências

A maneira mais simples de estimativa é supor que as variáveis do problema são independentes, ou seja, não há qualquer tipo de relação entre elas (veja figura 4.1). Formalmente, a distribuição de probabilidade conjunta é fatorizada como o produto de n distribuições de probabilidades independentes e univariadas:

$$p(x) = \prod_{i=1}^n p(x_i) \quad (4.5)$$

onde x é o conjunto de soluções promissoras, em um espaço de busca n -dimensional, selecionadas para a estimação.

O modelo de estimação utilizado para gerar novos indivíduos contém o conjunto de frequências de valores das variáveis que representam a solução no conjunto selecionado. Assim, estas frequências são usadas para guiar a busca, gerando novos indivíduos, variável por variável de acordo com os valores de frequência. Dessa forma, os blocos construtivos de primeira ordem são reproduzidos e mesclados eficientemente (Pelikan et al., 2002). Algoritmos baseados neste mecanismo são indicados para problemas em que as variáveis não possuem interação mútua (Harik et al., 1997; Mühlenbein, 1997).

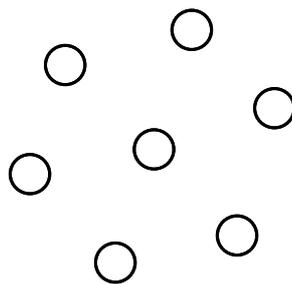


Fig. 4.1: Modelo gráfico com variáveis independentes (ausência de arestas indica ausência de dependência).

Exemplos de algoritmos que fazem uso deste modelo probabilístico, no espaço de variáveis discretas, são: o UMDA (*Univariate Marginal Distribution Algorithm*) (Mühlenbein, 1997), onde

cada distribuição marginal univariada é estimada pelas respectivas frequências marginais; o PBIL (*Population Based Incremental Learning*) (Baluja, 1994), que utiliza a regra de aprendizado hebbiano para a atualização do vetor de probabilidades, a partir do conjunto de soluções selecionadas; e o CGA (*Compact Genetic Algorithm*) (Harik et al., 1998), que atualiza o vetor de probabilidades $p(x)$ a partir de uma competição entre dois indivíduos amostrados com base em $p(x)$, sendo $p(x)$ deslocado em direção ao indivíduo vencedor.

Para o caso contínuo, supõe-se que a função densidade de probabilidade conjunta segue uma distribuição normal n -dimensional, que é fatorizada pelo produto de n densidades normais unidimensionais e independentes (Larrañaga, 2001). O problema então consiste na estimação dos parâmetros da distribuição. Formalmente, para o caso de uma distribuição normal, a função densidade de probabilidades é descrita por:

$$f(x; \mu, \Sigma) = \prod_{i=1}^n f_{\mathcal{N}}(x_i; \mu_i, \sigma_i^2) \quad (4.6)$$

onde μ_i denota a média da i -ésima variável e σ_i^2 sua variância. O algoritmo UMDA foi estendido ao espaço de variáveis contínuas (UMDA_c) (Larrañaga et al., 1999). Aqui, a cada iteração o algoritmo busca por uma função densidade que melhor se ajuste à variável, obtendo seus parâmetros pela estimativa da máxima verossimilhança. Já o algoritmo SHCLVND (*Stochastic Hill Climbing with Learning by Vectors of Normal Distributions*) (Rudlof & Köppen, 1996) atualiza os parâmetros da distribuição normal, média e variância, pela regra de Hebb para a média e uma política de redução para a variância.

4.3.2 Modelos com dependência aos pares

Embora a suposição de independência entre as variáveis seja satisfeita para alguns problemas, nem sempre esta hipótese é verificada, sendo então necessário levar em consideração algumas dependências. Quando a dependência aos pares é considerada, estabelece-se um compromisso entre a precisão e o custo computacional. A distribuição de probabilidades conjunta é fatorizada como:

$$p(x) = \prod_{i=1}^n p(x_i | x_{j(i)}) \quad (4.7)$$

onde $x_{j(i)}$ é a variável da qual x_i é dependente.

Enquanto os algoritmos que consideram variáveis independentes estimam apenas os parâmetros do modelo, na dependência aos pares a aprendizagem paramétrica é estendida para a aprendizagem da estrutura do modelo de dependência (um grafo direcionado).

No contexto discreto, o MIMIC (*Mutual Information Maximizing Input Clustering*) (De Bonet

et al., 1997) usa uma cadeia simples de distribuição (figura 4.2(a)) que maximiza a informação mútua das variáveis vizinhas (posições na cadeia), através de uma abordagem gulosa, que, embora eficiente, não garante a otimalidade global. A utilização de árvores de dependência foi formalizada por Baluja & Davies (1997, 1998) no COMIT (*Combining Optimizers with Mutual Information Trees*), que utiliza uma rede bayesiana com estrutura em árvore que é construída pelo algoritmo proposto por Chow & Liu (1968). O algoritmo BMDA (*Bivariate Marginal Distribution Algorithm*) (Pelikan & Mühlenbein, 1999) implementa uma floresta (um conjunto de árvores de dependência mutuamente independentes) como modelo probabilístico. Uma floresta pode ser interpretada como uma generalização do conceito de árvores de dependência. O algoritmo usa o teste Chi-quadrado de Pearson para determinar quais variáveis serão conectadas.

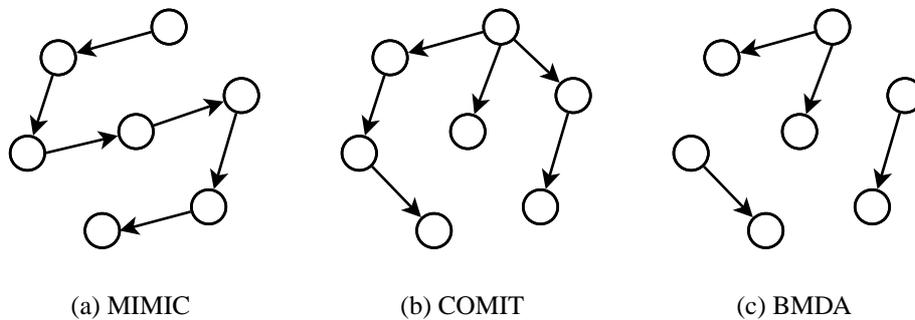


Fig. 4.2: Modelos gráficos com dependência aos pares. Arestas direcionadas indicam que há dependência entre as variáveis.

No caso contínuo, o MIMIC foi estendido para problemas com variáveis contínuas, supondo que o modelo de probabilidades é uma distribuição normal bivariada que utiliza um resultado de Whittaker (1990) para calcular e usar a entropia como medida para determinar dependência entre duas variáveis.

Algoritmos desta família reproduzem e mesclam blocos construtivos de segunda ordem eficientemente, obtendo sucesso na aplicação junto a problemas lineares e quadráticos (De Bonet et al., 1997; Mühlenbein, 1997).

4.3.3 Modelos com múltiplas dependências

Apesar de modelos que consideram dependência aos pares serem eficientes em certos cenários, em problemas multivariados ou com alta sobreposição de blocos construtivos, este tipo de modelo se mostra insuficiente para representar o espaço das boas soluções. Esta classe de problemas requer estimativas mais completas, mesmo à custa de uma maior demanda computacional.

No algoritmo ECGA (*Extended Compact Genetic Algorithm*) (Harik, 1999), no domínio discreto, as variáveis são divididas em grupos, cujas variáveis de um mesmo grupo (bloco construtivo) são

independentes entre si, como mostra a figura 4.3(a). A divisão dos grupos é feita por um procedimento guloso que utiliza a métrica *minimum description length (MDL)* (Rissanen, 1978). A distribuição de probabilidades conjunta foi codificada por uma rede bayesiana no algoritmo EBNA (*Estimation of Bayesian Networks Algorithm*) (Etxeberria & Larrañaga, 1999). Neste método, a estrutura da rede é construída por uma estratégia gulosa de busca local utilizando alguma métrica para medir a qualidade da rede, como BIC (*Bayesian Information Criterion*), K2 com penalidades ($K2+pen$) ou teste de (in)dependência condicional.

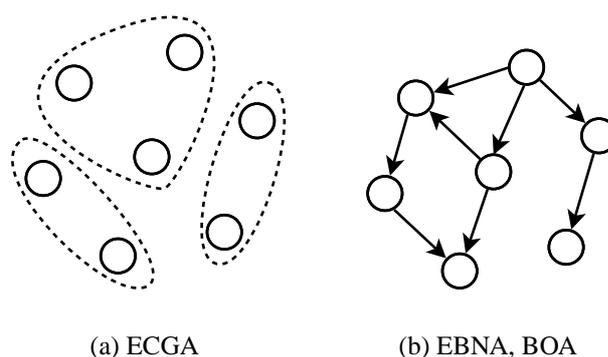


Fig. 4.3: Modelos gráficos com múltiplas dependências.

A métrica de pontuação *Bayesian-Dirichlet (BD)* foi utilizada no algoritmo BOA (*Bayesian Optimization Algorithm*) (Pelikan et al., 2000a), que, de forma incremental, vai adicionando novas arestas no grafo até que não haja melhorias na qualidade do modelo. Com o objetivo de reduzir a cardinalidade do espaço de busca, os autores propuseram restringir, a um número fixo, a quantidade de pais para cada nó da rede bayesiana.

No contexto de variáveis contínuas, o $EMNA_{global}$ (*Estimation Multivariate Normal Algorithm*) (Larrañaga et al., 2001) é baseado na estimação de uma função densidade normal multivariada em cada geração. Apesar do grande número de parâmetros a serem estimados, os cálculos envolvidos são muito simples (Larrañaga, 2001). Uma abordagem utilizando redes gaussianas, denominada EGNA (*Estimation Gaussian Network Algorithm*), foi proposta por Larrañaga et al. (1999), em que a cada geração a rede é construída e novos indivíduos são amostrados.

Os algoritmos desta classe se distinguem das outras abordagens pela capacidade de representar as interações multivariadas presentes no problema. Apesar destes algoritmos demandarem alto custo computacional, devido ao aprendizado de modelos mais complexos, o número de avaliações de *fitness* tende a ser reduzido significativamente (Pelikan et al., 2000a,b; Schwarz & Očenášek, 1999). Por isso, a complexidade computacional total tende a ser reduzida para problemas mais complexos, em que o cálculo da função-objetivo é computacionalmente oneroso.

4.3.4 Modelos de mistura

No caso de problemas multimodais, outros modelos probabilísticos mais flexíveis podem ser empregados. Modelos de mistura fazem uso do procedimento de agrupamento das soluções promissoras e posicionam distribuições de probabilidade junto a cada agrupamento. O modelo pode ser descrito como segue:

$$p(x) = \sum_{i=1}^K \pi_i p_i(x) \quad (4.8)$$

onde π_i representa o peso do i -ésimo componente da mistura e $p_i(x)$ é a distribuição de probabilidade do i -ésimo grupo criado pelo método de agrupamento de dados. O peso de cada componente juntamente com os parâmetros das distribuições de probabilidades são obtidos por algum procedimento dedicado, como o algoritmo EM (*Expectation-Maximization*) (Dempster et al., 1977). A figura 4.4 apresenta um modelo de mistura gaussiano arbitrário composto de quatro componentes.

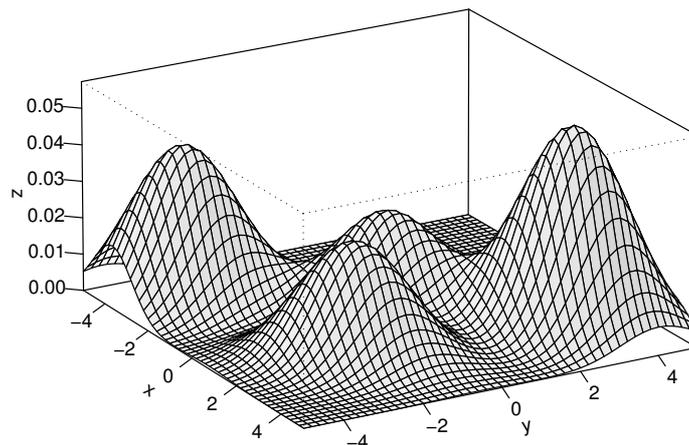


Fig. 4.4: Modelo de mistura gaussiano.

Um dos principais obstáculos desses métodos é a necessidade de conhecer, de antemão, o número de componentes, onde cada grupo busca representar um pico da função a ser otimizada. De fato, o número de componentes do modelo dependerá de quão multimodal o problema é. No entanto, existem formas de estimar este parâmetro, como foi visto no Capítulo 3.

No caso discreto, é possível criar um modelo de mistura em que cada componente é uma rede bayesiana, como o EMDA (*Estimation of Mixtures of Distributions Algorithm*) descrito por Pena et al. (2001), embora este tipo de estimador seja mais aplicado para o caso contínuo.

Para o espaço de variáveis contínuas, Gallagher et al. (1999) propuseram o algoritmo AMix

(*Adaptive Gaussian Mixture*), que utiliza um modelo de mistura adaptativo proposto por (Priebe, 1994). O número de componentes do modelo de mistura pode variar durante a execução do algoritmo. Bosman & Thierens (2000a) criam o modelo de mistura posicionando distribuições normais multivariadas, guiadas pelos algoritmos *leader* e *k-means*.

4.4 Considerações finais

Neste capítulo, foi apresentada uma classe de algoritmos que utilizam modelos de estimação da distribuição das soluções promissoras para guiar um algoritmo evolutivo, denominados de Algoritmos de Estimação de Distribuição (AED). Além disso, foram introduzidos os algoritmos evolutivos, dando ênfase no algoritmo genético descrito por Holland (1975), apontando as principais deficiências que motivaram a criação dos AEDs.

Os principais algoritmos de estimação de distribuição propostos na literatura foram levantados, classificando-os pelo nível de complexidade dos modelos probabilísticos usados na estimação.

Esta pesquisa empregará modelos de mistura gaussianos flexíveis e com baixo custo computacional como guia para um algoritmo evolutivo, na otimização de funções-objetivo contínuas e variantes no tempo.

Capítulo 5

Algoritmos de estimação de distribuição aplicados à otimização em ambientes dinâmicos

5.1 Considerações iniciais

Algoritmos de estimação de distribuição foram bastante explorados no contexto de otimização em ambientes estáticos. Para o caso discreto, pode-se destacar: Baluja & Davies (1997), Pelikan & Mühlenbein (1999), Harik et al. (2006), Shakya & McCall (2007), Gámez et al. (2007), Santana et al. (2010) e Li et al. (2011). Já para o caso contínuo, pode-se citar: Sebag & Ducoulombier (1998), Gallagher et al. (1999), Lu & Yao (2005), Dong & Yao (2007), Wierstra et al. (2008), Dong & Yao (2008). Uma revisão sobre algoritmos de estimação de distribuição e uma compilação das propostas de algoritmos desenvolvidos até então podem ser encontradas em Pelikan et al. (2002), Lozano (2006) e Armañanzas et al. (2008).

Em problemas não-estacionários, a esses algoritmos são acoplados operadores que possibilitam uma adaptação aos novos cenários que se apresentam. Dentre estes operadores, pode-se destacar: inserção de diversidade na população, reação às alterações, emprego de memória para armazenar e resgatar informações úteis do passado e/ou a utilização de múltiplas populações (Jin & Branke, 2005). Assim, AEDs tradicionais devem ser adaptados e/ou hibridizados para tratar de problemas de otimização em ambientes dinâmicos.

Neste capítulo, serão discutidas algumas propostas de AEDs desenvolvidas para aplicação em problemas de otimização, cuja função-objetivo é variante no tempo. Algumas ferramentas para otimização em ambientes dinâmicos a tempo discreto, ou seja, em que a variação ocorre em intervalos discretos de tempo, são citadas, destacando a família de algoritmos PBIL (do inglês,

Population-based incremental learning).

Esta pesquisa concentra-se em metodologias de otimização de problemas dinâmicos com variáveis contínuas. Para este cenário, algumas propostas já existentes serão discutidas e novas metodologias de AEDs, baseadas em modelos de mistura gaussianos flexíveis e pouco custosos computacionalmente, aliados a técnicas de manutenção de diversidade na população e controle de convergência, são apresentadas, sendo esta uma das contribuições desta pesquisa.

5.2 Propostas de AEDs existentes na literatura

No espaço de variáveis discretas, uma grande quantidade de metodologias que incorporam AEDs foi desenvolvida, principalmente no caso binário. Esta gama de abordagens é motivada pela possibilidade de armazenar as probabilidades em um único vetor fixo n -dimensional, sendo n o número de dimensões do problema tratado. Este é o conceito central da família de algoritmos PBIL (*Population-based incremental learning*) (Baluja, 1994).

Algoritmos PBIL foram largamente aplicados a problemas de otimização em ambientes dinâmicos, introduzindo conceitos de memória associativa (Yang & Yao, 2008), múltiplas populações e dualidade/complementariedade (Yang & Yao, 2005) para melhorar a performance destes algoritmos em ambientes dinâmicos.

Já para o caso de variáveis contínuas, sendo este o cenário tratado por esta pesquisa, modelos gaussianos são os mais utilizados, devido, principalmente, à sua eficiência computacional (Larrañaga et al., 1999).

Embora eficientes, modelos gaussianos vêm sendo pouco empregados em problemas de otimização em ambientes dinâmicos e, conseqüentemente, poucos trabalhos são encontrados na literatura.

Uma possível dificuldade em trabalhar com modelos gaussianos, está em como manipulá-lo de forma eficiente e com baixo custo computacional em problemas multimodais (os quais são comuns em situações reais), o que é altamente relevante em problemas dinâmicos.

O emprego de um único componente gaussiano, para representar as regiões promissoras do espaço de busca, pode muitas vezes alcançar resultados insatisfatórios, dada sua incapacidade de tratar a multimodalidade. Tentativas de contornar este problema foram propostas, como o IUMDA (Liu et al., 2008) e o Tri-EDA_G (Yuan et al., 2008), os quais são melhor discutidos a seguir.

O *Improved Univariate Marginal Distribution Algorithm* (IUMDA) (Liu et al., 2008) supõe que as variáveis são mutuamente independentes e as modela por uma função densidade de probabilidade (fdp) gaussiana. O treinamento do modelo consiste, então, em obter os parâmetros (μ e Σ) para cada gaussiana. Além disso, implementa o conceito de *repulsor* para aumentar a capacidade de exploração

do espaço de busca.

O IUMDA é um algoritmo que possui baixo custo computacional, além de simples implementação. Em problemas cujas variáveis são pouco ou não correlacionadas, provavelmente produzirá bons resultados.

O Tri-EDA_G (Yuan et al., 2008) desempenha uma exploração global e emprega a manutenção de diversidade na população, por meio de três componentes probabilísticos, mais especificamente: (i) uma fdp gaussiana multivariada para exploração local; (ii) uma variável uniforme para exploração global, e (iii) uma fdp gaussiana multivariada, chamada de *modelo opcional*. Inicialmente inativo, o modelo opcional é disparado somente quando a variável uniforme do componente (ii) encontra uma solução melhor do que as melhores soluções obtidas pelo componente (i). Assim, o componente (iii) é posicionado sobre esta solução, com uma matriz de covariância pré-definida.

Embora o Tri-EDA_G utilize a abordagem de inserção de *random immigrants* na população, o algoritmo é capaz de manipular, no máximo, duas regiões promissoras simultaneamente, uma de forma contínua e a outra opcional. Em ambientes com múltiplos ótimos locais, esta restrição pode interferir fortemente na performance do algoritmo, como poderá ser visto nas simulações realizadas (ver seção 6.4).

Em ambas as propostas, a estimação do modelo probabilístico é ótima, no sentido da maximização da verossimilhança entre os dados e o modelo. No entanto, a relação das variáveis é negligenciada no IUMDA e a multimodalidade é tratada de uma maneira superficial no Tri-EDA_G.

5.3 Novas propostas de AEDs para problemas em ambientes dinâmicos

Com base em análises acerca dos algoritmos existentes, em direcionamentos e resultados encontrados na literatura de otimização em ambientes dinâmicos, em um estudo sobre metodologias para estimação de densidade de probabilidades e em observações retiradas de inúmeras simulações, propõe-se uma nova ferramenta para tratar de problemas de otimização com variáveis contínuas em ambientes dinâmicos a tempo discreto.

Como um dos resultados obtidos por esta pesquisa, nesta seção será descrito um AED híbrido baseado em modelos de mistura gaussianos e que incorporam diversos conceitos abordados ao longo da pesquisa, o qual será denominado de AED_{MMG}. Neste algoritmo buscou-se incluir as seguintes características: flexibilidade para tratar de problemas multimodais, baixo custo computacional, rápida convergência, capacidade de corrigir a direção da busca e capacidade de exploração global. Os procedimentos empregados com o objetivo de alcançar estas características são destacados nas próximas seções.

Em problemas multimodais, é necessário utilizar múltiplas funções densidade de probabilidade, cada uma responsável por representar uma região promissora do espaço de busca. Isso permite modelar mais adequadamente o espaço de busca, produzindo assim uma mistura de componentes probabilísticos, podendo ser modelado como um modelo de mistura.

Como foi discutido na Seção 3.2.4, o modelo de mistura tem se mostrado computacionalmente eficiente, quando empregado em técnicas de otimização. Já foi mencionado que, em ambientes dinâmicos, onde o espaço se altera frequentemente, algoritmos computacionalmente baratos são desejáveis.

Uma característica interessante e possível de ser explorada por algoritmos de otimização é que, no treinamento de modelos de mistura, há um aumento significativo da verossimilhança entre os dados e o modelo estimado, já nas primeiras iterações do algoritmo EM.

Esta peculiaridade pode ser utilizada para reduzir o tempo computacional dos AEDs, pelo emprego de modelos de mistura menos acurados (com truncamento no processo de treinamento) e que, ainda assim, fazem uma boa aproximação da distribuição de soluções promissoras no espaço de busca. Este é um conceito importante nas propostas de AEDs para otimização em ambientes dinâmicos desenvolvidas nesta pesquisa.

5.3.1 Algoritmo AED_{MMG}

A estimação do modelo de mistura no algoritmo AED_{MMG}, apresentado a seguir, é realizada a partir da aplicação de uma única sequência de passos E e M. Dado que o conjunto de parâmetros do modelo de mistura estimado é armazenado e utilizado na próxima geração do algoritmo evolutivo, o modelo tende a ser mais acurado no decorrer das gerações, pois a convergência do algoritmo EM é obtida ao longo das gerações e não a cada geração. No Capítulo 6, serão apresentados alguns experimentos que corroboram estas suposições.

O Algoritmo 7 apresenta o pseudocódigo do algoritmo AED_{MMG}, onde N é o número de indivíduos na população, η é a proporção de indivíduos da população utilizada para estimar as regiões promissoras no espaço de busca e K é o número de componentes no modelo de mistura. Neste algoritmo, a função $\lceil \cdot \rceil$ aproxima para o inteiro superior e $\lfloor \cdot \rfloor$ aproxima para o inteiro inferior.

Como apontado anteriormente, modelos de mistura requerem a definição *a priori* do número de componentes a serem utilizados. No algoritmo AED_{MMG}, uma regra simples é responsável por controlar a quantidade de componentes no modelo de mistura, a qual é baseada no *Bayesian Information Criterion* (BIC), que é uma métrica para a seleção de modelos, apresentada no Capítulo 3. Com isso, é possível estimar automaticamente um número adequado de componentes.

A diversidade da população tem sido o foco de trabalhos recentes para melhorar a adaptabilidade dos métodos evolutivos junto a problemas de otimização em ambientes dinâmicos. A diversidade

Algoritmo 7: Estrutura do algoritmo AED_{MMG}.

1. Defina $K \leftarrow 1$;
2. Gere uma população inicial de forma aleatória e inicialize os parâmetros da gaussiana: média ($\mu_k^{t=0}$) e matriz de covariância ($\Sigma_k^{t=0}$);
3. Avalie os indivíduos da população;
4. Selecione os $\lceil N \cdot \eta \rceil$ melhores indivíduos da população, usando algum método de seleção estocástico, como o método de *seleção por torneio*;
5. Faça $K \leftarrow K + 1$ e estime a distribuição dos dados selecionados, aplicando uma única iteração da sequência de passos E e M;
6. Calcule o BIC para o novo modelo estimado. Caso este BIC seja menor ou igual ao BIC do melhor modelo encontrado até agora, faça $K \leftarrow K - 1$;
7. Verifique se há componentes sobrepostos; caso houver, remova o de menor coeficiente de mistura;
8. Amostre $\lceil N \cdot \eta \rceil$ novos indivíduos a partir do modelo de mistura estimado;
9. Preserve, deterministicamente, $\lfloor \frac{(1-\eta)N}{2} \rfloor$ das melhores soluções selecionadas pelo método de seleção;
10. Amostre aleatoriamente $\lfloor \frac{(1-\eta)N}{2} \rfloor$ indivíduos;
11. Faça a nova população ser a união destas três sub-populações: amostrada pelo modelo, preservada e amostrada aleatoriamente. Retorne ao passo 3.

pode ser controlada, basicamente, de duas formas: incrementando a diversidade quando uma variação é detectada no ambiente, refletida na avaliação das soluções-candidatas (Cobb, 1990) ou realizando um controle da diversidade por toda a execução (Tinós & Yang, 2007).

Aqui foi utilizada uma versão simples da abordagem *random immigrants* (Grefenstette, 1992), a qual insere novos indivíduos, gerados aleatoriamente, na população corrente a cada geração (veja passo 10).

Como em todo AED, o modelo de mistura é obtido através de um sub-conjunto das melhores soluções da geração atual (veja passo 4). No AED_{MMG}, os indivíduos considerados na estimação são obtidos por um procedimento de seleção proporcional ao *fitness*, como o *método de seleção por torneio*. Este tipo de seleção privilegia os indivíduos mais bem adaptados, mas permite que indivíduos pouco adaptados também possam ser selecionados. Com isso, contribui-se para a manutenção da

diversidade na população.

Ademais, uma estratégia de elitismo é também aplicada para preservar e manter, na nova população, uma parcela das melhores soluções encontradas até o momento (veja passo 9).

A estimação do número de componentes no modelo de mistura é feita de uma maneira construtiva. Iniciando com um único componente, o algoritmo continuamente tenta adicionar mais componentes, baseando-se nos valores de BIC dos modelos. Assim, o número de componentes tende a refletir a multimodalidade do espaço de busca.

O parâmetro η definirá a proporção de indivíduos que serão utilizados na estimação da distribuição. Se η é alto, o algoritmo tende a convergir rapidamente para um ótimo local, dado que não há um número suficiente de indivíduos para realizar exploração global. Contudo, para valores baixos de η , o AED_{MMG} possivelmente terá uma convergência lenta, o que melhora a exploração global, mas não é adequado para aplicações em ambientes dinâmicos. Uma análise sobre a sensibilidade dos parâmetros é realizada no Capítulo 6.

No passo 7, o algoritmo procura por componentes que estão representando a mesma região promissora. Logo, pode haver uma redundância no modelo de mistura e esta deve ser tratada. A seção a seguir discute esta etapa do algoritmo AED_{MMG}.

Remoção de componentes sobrepostos

A proposta de algoritmo apresentada possui um caráter construtivo incremental: adiciona componentes ao modelo até que o mesmo se estabilize em um determinado número de componentes.

No entanto, em uma análise mais detalhada, observou-se que alguns componentes convergiam para uma mesma região promissora, ou seja, o algoritmo estava utilizando mais componentes do que o necessário para representar uma região promissora. Este inconveniente ocorre devido ao fato do modelo ser obtido a partir de uma pequena quantidade de amostras (soluções) do espaço de busca.

A figura 5.1 ilustra a situação na qual, inicialmente, dois componentes representam regiões distintas, porém, no decorrer do tempo, convergem para a mesma região. A distância ϵ entre os dois componentes tende a zero no decorrer da execução.

Este inconveniente pode ser contornado pela simples remoção de um dos componentes sobrepostos, necessitando então, de um procedimento de identificação destes componentes. Uma possível abordagem é a comparação entre os centros dos componentes. Se a distância for menor do que um limiar ϵ , remove-se aquele componente com menor coeficiente de mistura, ou seja, aquele que detém a menor participação, entre os dois componentes, na representação do espaço de busca.

Alternativamente, a comparação pode ser realizada via análise dos desvios-padrão. A probabilidade de pontos serem amostrados por uma distribuição normal decresce conforme os

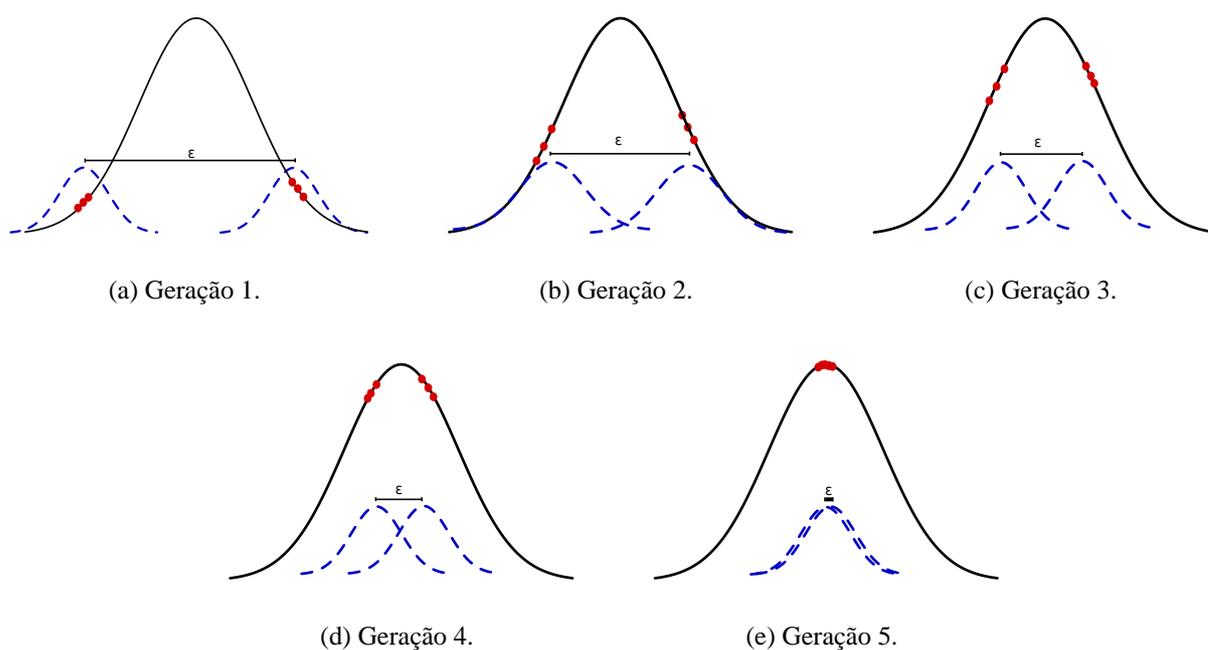


Fig. 5.1: Ilustração temporal da sobreposição de componentes do modelo de mistura. Inicialmente, os componentes representam regiões distintas do espaço de busca, mas, no decorrer do tempo, o algoritmo identifica que estas duas regiões passam a apresentar um alto grau de sobreposição.

pontos se afastam da média. É conhecido que 65% das observações amostradas estão distantes da média por, no máximo, um desvio-padrão; aproximadamente 95% por dois e 99,7% por, no máximo, três desvios-padrão (Moore et al., 2009). Logo, pode-se considerar que dois componentes estão sobrepostos se o centro de um componente está situado dentro do raio de n desvios-padrão do primeiro componente.

A abordagem de comparação por desvios-padrão será empregada no AED_{MMG}, na qual será considerado um desvio-padrão como limiar de sobreposição.

A rotina de remoção, passo 7 do algoritmo AED_{MMG} (Algoritmo 7), pode ser disparada em toda geração ou a cada m gerações, reduzindo assim o custo computacional do algoritmo. A figura 5.2 ilustra o fluxograma do AED_{MMG}.

O procedimento de remoção contribui para a redução do custo computacional do algoritmo, pois, apesar do aumento no tempo assintótico do algoritmo - $O(n^2)$, sendo n o número de componentes do modelo - haverá uma redução no tempo de execução, devido ao fato do algoritmo tender a considerar o número mínimo de componentes necessários.

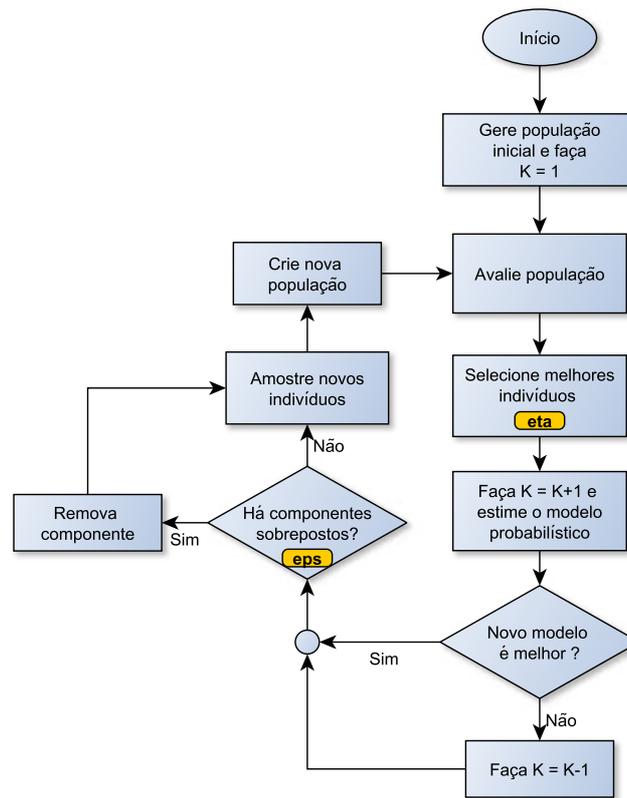


Fig. 5.2: Fluxograma do algoritmo AED_{MMG} com remoção de componentes sobrepostos.

5.3.2 Controle de convergência

Embora uma certa diversidade é inserida na população por meio da abordagem de *random immigrants*, em determinadas situações, estes indivíduos não alcançam êxito na tarefa de encontrar melhores soluções do que as obtidas até o momento, seja pela a severidade das alterações, pela baixa frequência das alterações no espaço de busca ou pela alta dimensionalidade. Assim, uma ação mais enérgica para promover a exploração do espaço de busca pode trazer benefícios ao algoritmo.

Um procedimento passível de adoção para evitar a convergência do algoritmo AED_{MMG} é empregar um controle contínuo, composto por duas etapas: *identificação* e *ação*.

A identificação da convergência pode ser realizada tanto em uma análise no espaço de busca, como investigar a distância entre os indivíduos da população, quanto no espaço dos valores de *fitness*, como, por exemplo, examinar a variância ou desvio-padrão destes valores. A análise de diversidade no espaço de busca é mais precisa, embora seja mais custosa computacionalmente.

Pode-se definir um limiar δ , a partir do qual uma rotina de “espalhamento” dos indivíduos é ativada. Esta rotina de exploração pode ser implementada por algum procedimento de alteração nos indivíduos da população, como, por exemplo, substituir indivíduos muito próximos entre si no espaço

de busca e adotar uma reinicialização completa da população de indivíduos. Aqui, o procedimento utilizado é a reinicialização aleatória de todos os indivíduos da população.

Uma constante δ é definida como limiar de ativação deste procedimento. Se o desvio-padrão (ou variância) dos valores de *fitness* das soluções geradas a partir do modelo de mistura for menor do que δ , a população é reinicializada. Uma vez que o algoritmo é dotado de mecanismos de preservação das melhores soluções, estas são mantidas e a reinicialização é responsável por procurar novas regiões promissoras, tão boas ou melhores do que as regiões encontradas até o momento.

A figura 5.3 apresenta o fluxograma do AED_{MMG}, com a inclusão do procedimento de controle de convergência.

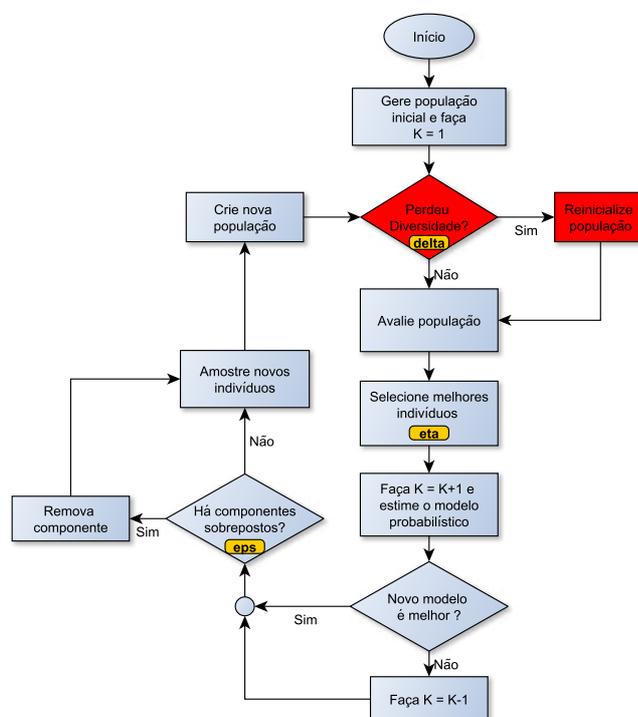


Fig. 5.3: Fluxograma do algoritmo AED_{MMG} com controle de convergência.

Apesar da inclusão de um novo parâmetro livre no algoritmo, os experimentos mostram que, em determinados cenários, principalmente em espaço de busca de alta dimensão e em ambientes com baixa frequência de alterações, este mecanismo provê uma melhora significativa na performance do método de otimização.

5.3.3 Algoritmo AED_{MMG} com versão *online* do algoritmo EM

Uma versão *online* do algoritmo EM é motivada pela necessidade de processar dados em fluxo contínuo, onde os dados chegam de forma contínua e sequencial. Neste contexto, o algoritmo EM tradicional é impraticável, devido à necessidade de que todos os dados estejam disponíveis em todas as iterações do algoritmo.

A variante *online* do algoritmo EM, utilizada nesta pesquisa, é baseada no trabalho de Nowlan (1991). Este algoritmo sumariza todos os dados em um vetor de estatísticas suficientes que pode ser incrementalmente atualizado. Posteriormente, este vetor é utilizado para calcular os parâmetros do modelo de mistura.

Estatísticas suficientes corresponde a uma função $s(\cdot)$ das amostras que contém toda a informação relevante para estimar determinados parâmetros (Bishop, 2007).

Os passos E (*Expectation*) e M (*Maximization*) da variante *online* descrita em Neal & Hinton (1998), são apresentados na figura 5.4, onde Z é a variável não-observada, X são os dados observados, $\tilde{s}^{(t)}$ é o vetor de estatísticas suficientes no tempo t , $E_{\tilde{P}_i}$ denota a esperança em relação à distribuição sobre a variável não-observada Z e γ é uma constante de decaimento.

Passo E: Selecione o próximo dado i , para atualização.

Faça $\tilde{s}_i^{(t)} = E_{\tilde{P}_i}[s_i(z_i, x_i)]$, para $\tilde{P}_i(z_i) = P(z_i|x_i, \theta^{(t-1)})$.

Faça $\tilde{s}^{(t)} = \gamma\tilde{s}^{(t-1)} + \tilde{s}_i^{(t)}$.

Passo M: Faça $\theta^{(t)}$ receber θ com máxima verossimilhança, dado $\tilde{s}^{(t)}$.

Fig. 5.4: Passos E e M da variante *online* do algoritmo EM proposto por Nowlan (1991).

O algoritmo utiliza as estatísticas computadas como uma média com decaimento exponencial dos dados recentemente visitados. De acordo com Neal & Hinton (1998), esta variante *online* do algoritmo EM não convergirá para um solução exata, mas é capaz de convergir para a vizinhança da solução mais rapidamente do que o algoritmo EM padrão, uma vez escolhido um valor apropriado para o parâmetro γ . No contexto de otimização, este infortúnio pode ser amenizado pela capacidade de exploração do espaço de busca provida pelo algoritmo evolutivo.

A taxa de convergência de duas execuções da versão *online* do algoritmo EM apresentada, com $\gamma=0,99$ e $0,95$, são apresentadas na figura 5.5. Quando utilizado $\gamma=0,99$, o algoritmo convergiu para uma boa (mas não ótima) solução mais rapidamente do que quando se emprega uma abordagem incremental (linha sólida), apresentada no trabalho de Neal & Hinton (1998). Esta, por sua vez, possui uma convergência mais rápida do que o EM padrão. A execução com $\gamma=0,95$ convergiu rapidamente para uma solução, no entanto, para uma solução pior. Assim, deve ser escolhido um γ que combina

convergência rápida com garantia de estabilidade, e que alcance uma boa solução (Neal & Hinton, 1998).

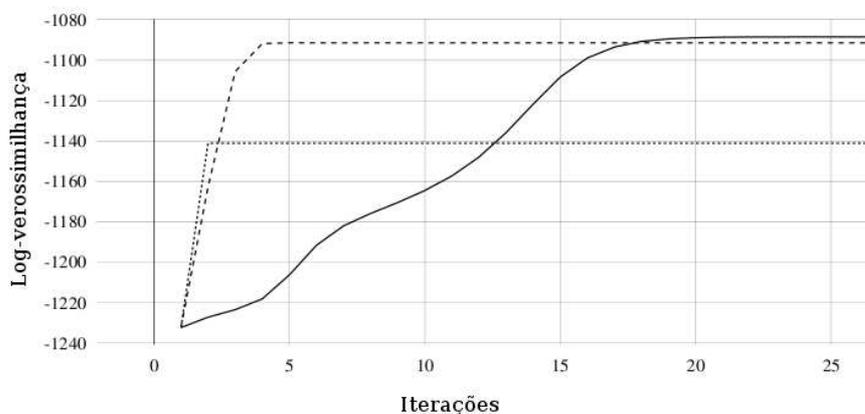


Fig. 5.5: Taxa de convergência da versão *online* do algoritmo EM. A curva tracejada apresenta a execução utilizando $\gamma=0,99$ e a curva pontilhada, $\gamma=0,95$. A linha sólida representa uma versão incremental do algoritmo EM. Fonte: Neal & Hinton (1998).

No contexto de otimização baseada em algoritmos populacionais, ao invés do decaimento ser aplicado a cada dado, aqui o decaimento será aplicado ao longo das gerações. Assim, em cada geração g , o vetor de estatísticas suficiente $\tilde{s}^{(g)}$ é obtido como mostra a equação 5.1:

$$\tilde{s}^{(g)} = \gamma \tilde{s}^{(g-1)} + \tilde{s}^{(g)}. \quad (5.1)$$

Logo, caso não se queira manter informações sobre os dados das gerações anteriores, faz-se $\gamma = 0$. Atribuindo outros valores ao parâmetro γ , restrito ao intervalo $[0;1)$, é possível controlar a participação de dados anteriores no modelo de mistura atual.

Com o emprego do vetor de estatísticas suficientes, não há mais a necessidade de armazenar os parâmetros de cada componente, sendo eles calculados a cada geração do algoritmo. A versão do AED_{MMG} que utiliza o treinamento *online* do modelo de mistura é denominada aqui de AED_{MMGO}.

O algoritmo AED_{MMGO} incorpora os conceitos utilizados no AED_{MMG}, provendo maior flexibilidade com a possibilidade de reutilizar boas soluções encontradas em um passado recente. Ademais, sua representação interna dos dados é concisa, sem a necessidade de manter o modelo de mistura em todas as gerações do algoritmo evolutivo, apenas armazenando o vetor de estatísticas suficientes.

Nos experimentos realizados no Capítulo 6, o AED_{MMGO} será considerado para análises e comparações com outras ferramentas já existentes.

5.4 Considerações finais

Neste capítulo, foi discutida a aplicação de AEDs junto a problemas de otimização em ambientes variantes no tempo. Foram apresentadas algumas propostas encontradas na literatura, tanto para o espaço de variáveis discretas quanto para o espaço de variáveis contínuas.

Para o caso contínuo, que é o foco desta pesquisa, o número de propostas de algoritmos de estimação de distribuição ainda é restrito. Dentre os motivos, é possível destacar: a dificuldade em trabalhar com modelos gaussianos em ambientes multimodais, dado seu custo computacional; o desafio de controlar o número de modelos gaussianos a serem considerados, de tal forma a evitar um número excessivo de componentes que acabe aumentando o custo computacional; e a necessidade de prover ao algoritmo a adaptabilidade necessária às mudanças que ocorrem no espaço de busca.

Face a isto, esta pesquisa propõe AEDs flexíveis e com baixo custo computacional. O algoritmo AED_{MMG} foi apresentado, além de uma versão (AED_{MMGO}) que utiliza uma variante *online* do algoritmo *Expectation-Maximization*, a qual é capaz de incorporar informações do histórico da busca no modelo de mistura corrente.

Os algoritmos propostos serão experimentados no Capítulo 6 e os resultados obtidos serão comparados aos fornecidos por outras metodologias existentes na literatura, incluindo outros AEDs e também outras abordagens populacionais.

Capítulo 6

Experimentos

6.1 Considerações iniciais

Como foi mostrado no Capítulo 5, diversas propostas de algoritmos evolutivos para tratar de problemas de otimização em ambientes dinâmicos foram introduzidas nos últimos anos. Para que seja possível analisar o desempenho destes algoritmos, é necessário definir um conjunto de problemas de teste (*benchmark*) que sejam capazes de por à prova a eficiência de cada algoritmo.

Nos experimentos aqui realizados, dois geradores de espaços de busca contínuos e dinâmicos foram utilizados: o *benchmark de picos móveis* proposto por Branke (1999) e uma variação que emprega rotação dos picos, ao invés de deslocamento, desenvolvido por Li & Yang (2008), os quais serão devidamente detalhados. Estes geradores possuem um conjunto de parâmetros ϕ que definem um espaço de busca. Logo, alterando os valores destes parâmetros é possível conceber diferentes ambientes a serem otimizados. Como serão feitas e quão severas serão estas alterações é determinado por um procedimento matemático controlado por um parâmetro de severidade.

Os algoritmos desenvolvidos ao longo desta pesquisa terão suas performances analisadas sob ambos os geradores mencionados e os resultados serão comparados com outras técnicas encontradas na literatura.

6.2 Gerador de ambientes dinâmicos

A dinâmica de ambientes que modificam sua estrutura interna no decorrer do tempo, pode ser modelada por um conjunto de parâmetros ϕ , que, por sua vez, sofrem alterações ao longo do processo, definindo uma sequência de novos cenários a serem enfrentados. O grau de intensidade destas alterações influenciará diretamente na complexidade do problema a ser tratado. Logo, é possível definir um problema de otimização variante no tempo F como mostra a equação 6.1:

$$F = f(\mathbf{x}, \phi, t) \quad (6.1)$$

onde f é a função-objetivo, \mathbf{x} é uma solução factível do conjunto de soluções \mathbf{B} (espaço de busca), t é o tempo e ϕ é um vetor de parâmetros que controla o dinamismo do sistema.

Com o objetivo de analisar o desempenho das metodologias propostas para tratar deste tipo de problemas, foram construídos *frameworks* geradores de ambientes dinâmicos artificiais, os quais são controlados pelo conjunto de parâmetros ϕ e um procedimento para alterar estes parâmetros.

O gerador de ambientes dinâmicos proposto por Branke (1999), denominado de *Benchmark de Picos Móveis* (BPM) (do inglês *Moving Peaks Benchmark*) foi amplamente utilizado para comparação de algoritmos populacionais propostos para tratar problemas de otimização em ambientes dinâmicos. O BPM produz cenários de otimização que consistem de um certo número de picos com posição, altura e largura variantes no tempo. Aqui, as alterações são realizadas por intermédio de um vetor de deslocamento aleatório.

Outro gerador recentemente proposto por Li & Yang (2008) é uma variante do primeiro e utiliza um procedimento de rotação dos picos, ao invés de deslocamentos aleatórios. Com isso, o autor afirma que é possível eliminar a desigualdade de complexidade dos cenários gerados a cada alteração, promovida principalmente quando os picos alcançam as bordas do espaço de busca e retornam para seu interior. Estes dois geradores, BPM e BPM com rotação, serão formalmente apresentados a seguir.

6.2.1 Benchmark de Picos Móveis

Conceitualmente, o BPM, desenvolvido por Branke (1999), é uma simulação de um ambiente dinâmico configurável que se altera no decorrer do tempo. A superfície de otimização é composta por um conjunto picos, podendo ser representados por cones ou funções gaussianas, com posições, alturas e larguras variantes. A dimensão do espaço de busca é prefixada como um parâmetro do *benchmark*.

Dado $\phi = (\vec{H}, \vec{W}, \vec{X})$, onde \vec{H} , \vec{W} e \vec{X} denotam a altura, largura e posição dos picos, respectivamente, a função $f(\mathbf{x}, \phi, t)$ é definida como segue (Branke, 1999):

$$f(\mathbf{x}, \phi, t) = \max_{i=1}^m \frac{H_i(t)}{1 + W_i(t) \cdot \sum_{j=1}^n (x_j - X_j^i(t))^2}, \quad (6.2)$$

onde m é o número de picos e n é o número de dimensões. Os parâmetros dos picos são gerados aleatoriamente, dentro de um intervalo pré-estabelecido. Após um período de tempo, medido pelo número de avaliações de função realizadas, esses parâmetros têm seus valores alterados por algum

procedimento. A figura 6.1 ilustra um espaço de busca bidimensional gerado pelo BPM com 50 picos utilizando funções cone.

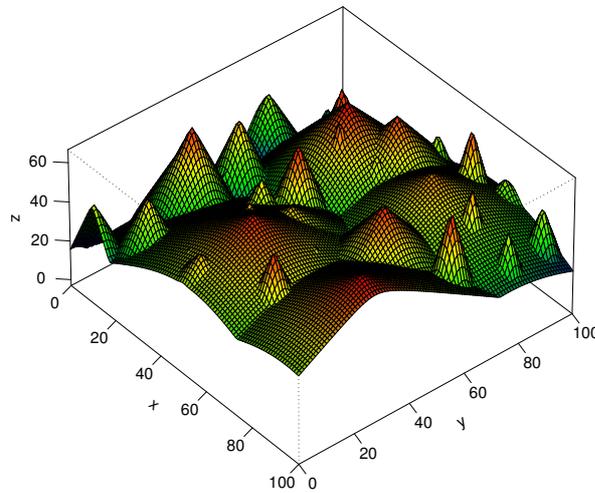


Fig. 6.1: Espaço de busca contínuo com 50 picos gerado pelo BPM.

Este procedimento de modificação dos parâmetros ϕ define como o ambiente se altera ao longo do tempo. Branke (1999) propôs que as posições dos picos sejam alteradas por meio de um *vetor deslocamento*, sendo este adicionado às suas posições atuais. Para as alturas e larguras, uma variável aleatória gaussiana escalada produz a aleatoriedade do ambiente. Mais formalmente, uma alteração pode ser descrita como mostra a equação 6.3:

$$\begin{aligned} H_i(t) &= H_i(t-1) + r\Delta H \\ W_i(t) &= W_i(t-1) + r\Delta W \\ \vec{X}(t) &= \vec{X}(t-1) + \mathbf{v}(t) \end{aligned} \quad (6.3)$$

onde $r \sim \mathcal{N}(0,1)$ e ΔH e ΔW são as severidades das alterações das alturas e larguras, respectivamente. Branke (1999) utilizou os seguintes valores pra estes parâmetros: $\Delta H = 7$ e $\Delta W = 0,01$. A variável \vec{v} é um vetor deslocamento, obtido pela combinação linear de um vetor aleatório e o vetor deslocamento anterior. Assim, o vetor deslocamento pode ser definido como mostra a equação 6.4:

$$\mathbf{v}_i(t) = \frac{s}{\|\vec{r} + \mathbf{v}_i(t-1)\|} \cdot (1 - \lambda)\mathbf{r} + \lambda\mathbf{v}_i(t-1) \quad (6.4)$$

onde \mathbf{r} é um vetor aleatório, s é um parâmetro que controla a severidade da alteração, $\lambda \in [0, 1]$ é um fator de correlação que define o balanço entre movimento aleatório e previsível do vetor deslocamento e $\|\cdot\|$ é a norma euclidiana. Os tipos e número de picos, juntamente com suas posições, alturas e larguras iniciais, a dimensão, os limites do espaço de busca, a severidade das alterações e a frequência

com que elas ocorrem, devem ser informadas. Esta configuração de parâmetros define um *cenário de otimização*.

A figura 6.2 ilustra a movimentação do máximo global em um espaço de busca bi-dimensional. Nota-se, claramente, que, com diferentes valores do parâmetro s , é possível simular ambientes com dinâmicas e complexidade de otimização bem distintos.

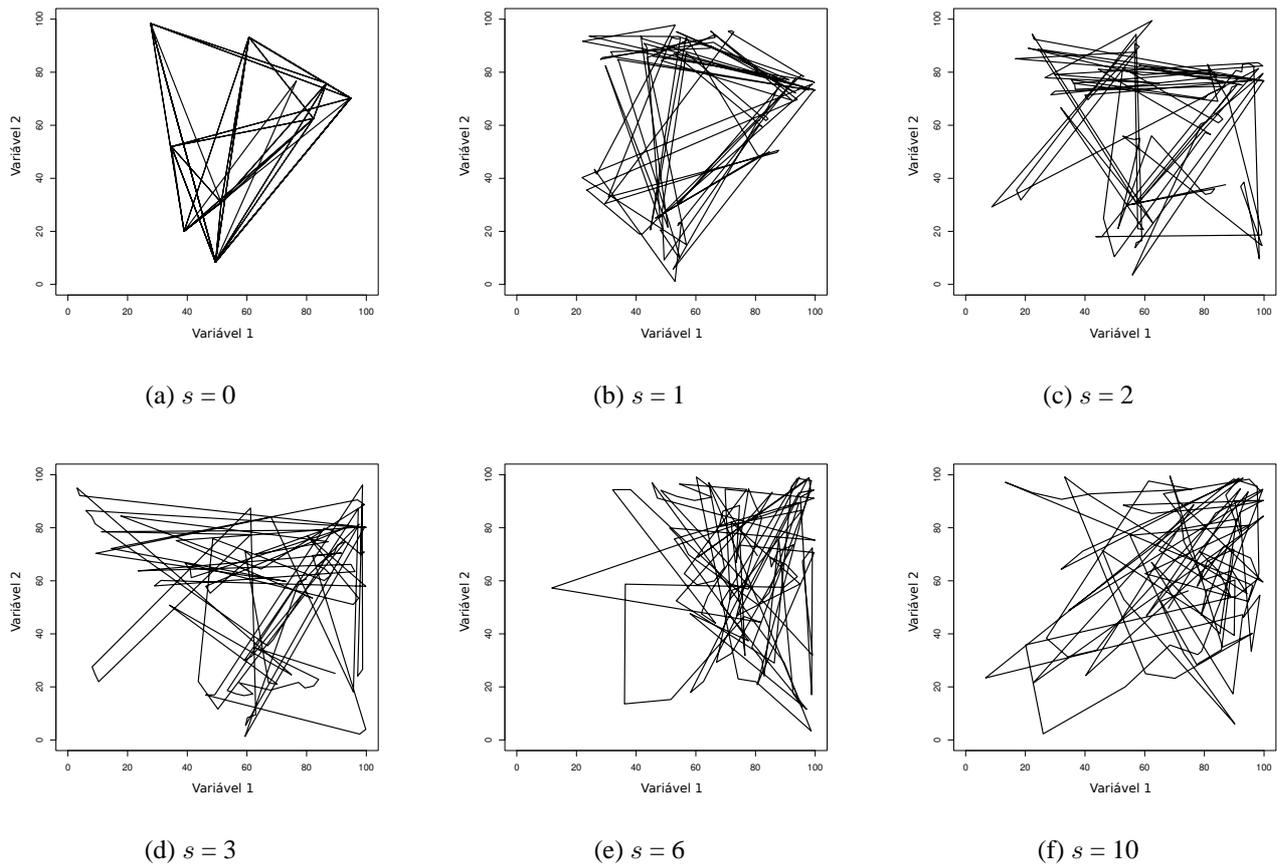


Fig. 6.2: Movimentação do máximo global em um espaço bidimensional, sujeito a 100 alterações, com diferentes valores de s e tomando $\lambda=0,5$.

6.2.1.1 Picos móveis com rotação e novas formas de alteração

Segundo Li & Yang (2008), o BPM proporciona ambientes com diferentes níveis de dificuldade para o algoritmo de otimização, a cada alteração, uma vez que a posição dos picos alcançam os limites do espaço de busca. Sendo assim, o autor sugere um procedimento em que os picos são rotacionados, ao invés de sofrerem um deslocamento aleatório.

Inicialmente, define-se a função $f(\mathbf{x}, \phi, t)$ a ser otimizada, como segue:

$$f(\mathbf{x}, \phi, t) = \max_{i=1}^m \frac{H_i(t)}{(1 + W_i(t) \cdot \sqrt{\sum_{j=1}^n \frac{(x_j - X_j^i(t))^2}{n}})}, \quad (6.5)$$

onde m é o número de picos e n é o número de dimensões do problema.

Em comparação com a equação 6.2, é possível ressaltar algumas diferenças, como a normalização do termo, no denominador, que calcula a diferença entre o ponto x e os picos. Além da utilização da raiz quadrada deste termo, diferentemente do termo absoluto, como ocorre na equação 6.2. A figura 6.3 ilustra o espaço de busca com 50 picos, em duas dimensões, gerado pelo BPM com rotação.

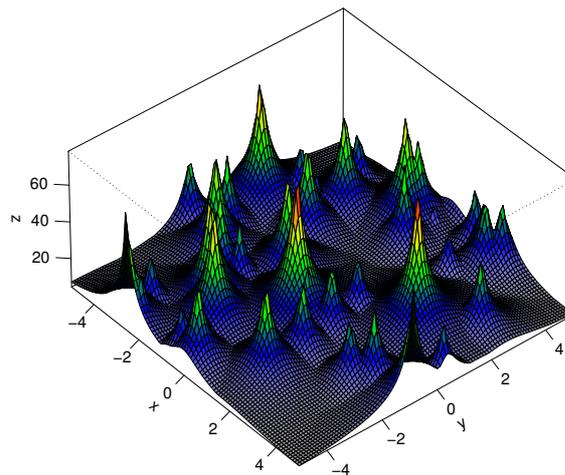


Fig. 6.3: Espaço de busca contínuo com 50 picos gerado pelo BPM com rotação.

Na modificação das posições dos picos, utiliza-se uma matriz de rotação $R_{ij}(\theta)$ (Salomon, 1996), que é obtida pela rotação da projeção de \vec{X} no plano $i - j$ por um ângulo θ do i -ésimo eixo para o j -ésimo eixo. Assim, a posição do pico \vec{X} é modificada pelo Algoritmo 8.

Algoritmo 8: Algoritmo de rotação dos picos.

1 início

- 2 Selecione aleatoriamente l dimensões, dentre as n dimensões do problema, sendo l um número par e $l \leq n$, para compor o vetor $r = [r_1, r_2, \dots, r_l]$;
- 3 Para cada par de dimensões $(r[i], r[i+1])$, construa uma matriz de rotação $R_{r[i],r[i+1]}(\theta(t))$, $\theta(t) = \text{AlteraçãoDinâmica}(\theta(t-1))$;
- 4 A matriz de transformação $A(t)$ é obtida por:

$$A(t) = R_{r[1],r[2]}(\theta(t)) \cdot R_{r[3],r[4]}(\theta(t)) \cdots R_{r[l-1],r[l]}(\theta(t)) \quad (6.6)$$

$$\vec{X}(t+1) = \vec{X}(t) \cdot A(t)$$

6 fim

No Algoritmo 8, à *severidade* da alteração de θ ($\phi_{\text{severidade}}$) é atribuído valor igual a 1 na equação 6.7, o intervalo de valores de θ é $\|\phi_{\text{severidade}}\|$, com $\|\phi_{\text{severidade}}\| \in (-\pi, \pi)$. Na escolha das l dimensões, caso l seja escolhida ser igual a n e n for um número par, então $l = n$, se n é ímpar, então $l = n - 1$.

Além da substituição do vetor de deslocamento pelo método de rotação, Li & Yang (2008) propôs um novo procedimento para alteração nas alturas e larguras dos picos. Foram apresentadas sete formas de alteração, sendo elas denominadas de: *pequeno passo*, *grande passo*, *aleatória*, *caótica*, *recorrente*, *recorrente ruidosa* e *aleatória com variação na dimensão do problema*. Matematicamente, estas alterações são descritas como segue:

Pequeno passo:

$$\Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{\text{severidade}} \quad (6.7-1)$$

Grande passo:

$$\Delta\phi = \|\phi\| \cdot (\alpha \cdot \text{sign}(r) + (\alpha_{\text{max}} - \alpha) \cdot r) \cdot \phi_{\text{severidade}} \quad (6.7-2)$$

Aleatória:

$$\Delta\phi = N(0, 1) \cdot \phi_s \quad (6.7-3)$$

Caótica:

$$\phi(t+1) = A \cdot (\phi(t) - \phi_{\text{min}}) \cdot (1 - (\phi(t) - \phi_{\text{min}})/\|\phi\|) \quad (6.7-4)$$

Recorrente:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 \quad (6.7-5)$$

Recorrente ruidosa:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 + \mathcal{N}(0, 1) \cdot R_s \quad (6.7-6)$$

onde $\|\phi\|$ é o intervalo de valores dos parâmetros de controle ϕ , ϕ_s é uma constante que indica a severidade das alterações de ϕ , ϕ_{min} é o valor mínimo de ϕ , $R_s \in (0; 1)$ é a severidade do ruído na alteração recorrente com ruído. As constantes $\alpha \in (0; 1)$ e $\alpha_{max} \in (0; 1)$ são definidas como 0.04 e 0.1, respectivamente. Uma função logística é utilizada na alteração caótica, onde A é uma constante positiva entre (1; 4). Os valores iniciais de ϕ são definidos como diferentes valores dentro do intervalo $\|\phi\|$. P é o período das alterações recorrente e recorrente com ruído, φ é a fase inicial, r é um valor aleatório entre $(-1; 1)$, $sign(x)$ é uma função sinal, como mostra a equação 6.8, e por fim, $\mathcal{N}(0, 1)$ denota um valor amostrado por uma distribuição normal univariada, com média zero e desvio padrão igual a um.

$$sign(x) = \begin{cases} -1 & : x < 0 \\ 0 & : x = 0 \\ 1 & : x > 0 \end{cases} \quad (6.8)$$

Para alterações recorrente e recorrente ruidosa, o intervalo de θ ($\|\phi_\theta\|$) é definido como sendo igual a $(0, \pi/6)$. A figura 6.4 apresenta a movimentação da solução ótima (máximo global), em um espaço de busca bidimensional, para cada um dos seis tipos de alterações definidos no BPM com rotação.

Observa-se que o ótimo sempre permanece na região central do espaço e raramente alcança as bordas do espaço de busca, tendo visivelmente um caráter rotativo. Na alteração caótica, o ótimo concentra-se em um determinada região do espaço de busca, e a função sinal utilizada provê um caráter alternante ao posicionamento da solução ótima. Outros tipos de situações podem ser simuladas, alterando os valores dos parâmetros das funções de alteração do ambiente (equações 6.7).

Por meio da alteração T_7 é possível simular alguns problemas reais, onde o número de variáveis a serem otimizadas pode ser alterado no decorrer do processo, como, por exemplo, em um problema de roteamento um novo nó em uma rota já planejada deve ser necessariamente adicionado. Nestes cenários, os algoritmos podem sofrer uma queda repentina de desempenho quando a dimensão do problema aumenta, isso devido ao aumento do espaço de busca e, conseqüentemente, da

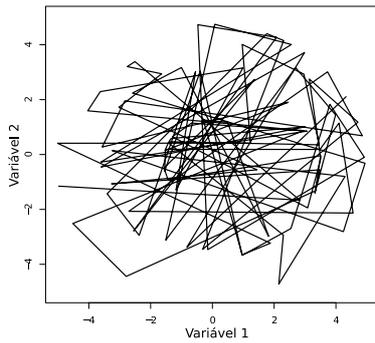
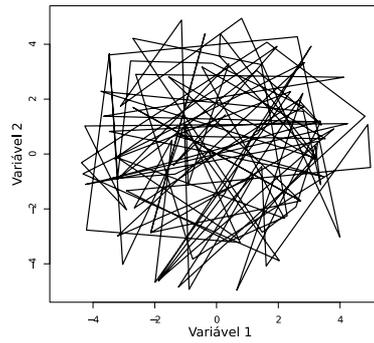
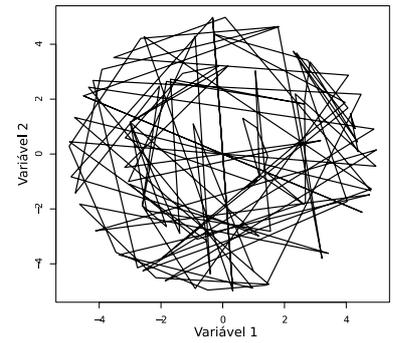
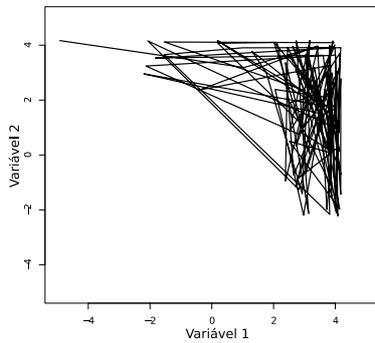
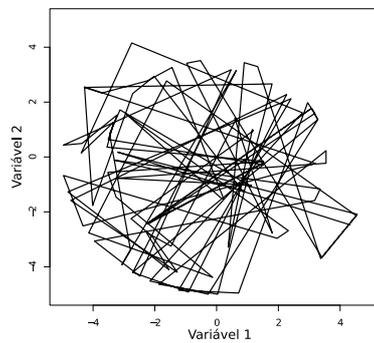
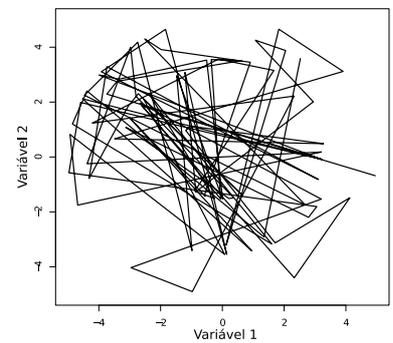
(a) Alteração *pequeno passo*.(b) Alteração *passo largo*.(c) Alteração *aleatória*.(d) Alteração *caótica*.(e) Alteração *recorrente*.(f) Alteração *recorrente com ruído*.

Fig. 6.4: Movimentação do máximo global em um espaço bidimensional, sujeito a 100 alterações, para os seis tipos de alterações (exceto para a alteração com variação na dimensão) definido no BPM com rotação.

complexidade do problema.

A variação na dimensão do espaço de busca é implementada pelo gerador, aumentando e diminuindo o número de dimensões do problema, conforme a equação 6.9:

$$D(t+1) = D(t) + \text{sign} \cdot \Delta D, \quad (6.9)$$

onde ΔD é uma constante pré-definida, cujo valor padrão é 1. Se $D(t) = \text{Max}_D$, $\text{sign} = -1$ e caso $D(t) = \text{Min}_D$, $\text{sign} = 1$. Max_D e Min_D são, respectivamente, o número máximo e mínimo de dimensões que o sistema pode alcançar. Quando o número de dimensões é reduzido de uma unidade, a última dimensão é removida do espaço de busca e, quando uma nova dimensão é adicionada, a esta é atribuída um valor aleatório. Uma alteração da dimensão é realizada somente após uma alteração não-dimensional (equações 6.7).

Ambos os autores disponibilizaram os códigos-fonte dos *benchmarks* citados. O BPM original, disponível em linguagem C, foi adquirido por contato direto com o autor, Jürgen Branke. Já para o BPM com rotação os códigos estão disponíveis em linguagem C++ e para o ambiente Matlab[®], no endereço <http://www.cs.le.ac.uk/people/syang/ECiDUE/DBG.tar.gz>.

6.3 Critérios de avaliação

Dado que os algoritmos estudados possuem um caráter estocástico, duas execuções independentes podem produzir resultados distintos. Para realizar uma análise mais confiável da performance do algoritmo, diversas execuções independentes devem ser realizadas e, então, algumas métricas podem ser obtidas. Os critérios utilizados nesta pesquisa são obtidos a partir de um conjunto de execuções do algoritmo, com diferentes inicializações.

A seguir, serão destacadas as duas métricas comumente utilizadas para comparação entre algoritmos de otimização para ambientes variantes no tempo.

Média do Offline error

É a mais utilizada para comparação das performances de algoritmos aplicados à otimização de ambientes dinâmicos. O *offline error* é a média do erro absoluto entre a melhor solução encontrada pelo algoritmo e o ótimo global (conhecido) a cada instante de tempo t . Seja T o tempo de execução do algoritmo, o *offline error* (OE) é obtido como segue:

$$OE = \frac{1}{T} \sum_{t=0}^T |f(\mathbf{x}_{best}, t) - f(\mathbf{x}^*, t)|, \quad (6.10)$$

onde $f(\mathbf{x}_{best}, t)$ é o valor de *fitness* da melhor solução, \mathbf{x}_{best} , encontrada no tempo t e $f(\mathbf{x}^*, t)$ é o ótimo global (conhecido) no mesmo instante de tempo t . Para cada um das N execuções independentes do algoritmo, o *offline error* é calculado e, então, a média é obtida:

$$M_{OE} = \frac{1}{N} \sum_{i=1}^N OE_i. \quad (6.11)$$

Dado que a média do *offline error* é um valor médio do erro absoluto, o erro padrão desta estatística mede a discrepância entre o valor médio calculado e o valor verdadeiro. O erro padrão da média do *offline error* é obtido de acordo com a equação 6.12:

$$SE_{MOE} = \frac{s}{\sqrt{N}}, \quad (6.12)$$

onde s é o desvio padrão dos erros absolutos e N é o número de execuções do algoritmo. Quanto menor for o erro padrão, mais estável é o algoritmo e mais confiável é o valor da média do *offline error*.

Gráficos de convergência

Os gráficos de convergência apresentam, de forma concisa, a informação da performance do algoritmo. O gráfico mostra a performance mediana do valor relativo $r(t) \in [0, 1]$ dado pela razão entre $f(x_{melhor}(t))$ e $f(x^*(t))$, sobre o total de execuções realizadas. Para problemas de maximização (picos móveis) $r(t)$ é obtido por:

$$r(t) = \frac{f(\mathbf{x}_{melhor}, t)}{f(\mathbf{x}^*, t)}. \quad (6.13)$$

A forma comumente utilizada para comparar a convergência de algoritmos evolutivos é analisar o erro em relação ao número de gerações. No entanto, dado que, em alguns algoritmos evolutivos, o número de indivíduos na população pode variar no decorrer da execução, estes algoritmos poderiam vir a realizar um maior ou menor número de avaliações de função do que algoritmos com população fixa. Sabendo que quanto maior o número de avaliações de função realizadas, mais informações o algoritmo detém sobre o espaço de busca, este critério poderia não penalizar devidamente algoritmos computacionalmente mais custosos, ou favorecer algoritmos menos custosos.

Face a este cenário, os critérios de avaliação dos algoritmos evolutivos aqui utilizados analisam a convergência em relação ao número de avaliações de função (AFs), o que equivale ao número de atribuições de valores de *fitness*, ao invés do número de gerações.

6.4 Resultados experimentais

As simulações realizadas buscam analisar e validar os métodos propostos, além de comparar com outros algoritmos encontrados na literatura. Além disso, permite a investigação da sensibilidade do algoritmo e direcionar a configuração dos parâmetros do algoritmo. Esta seção aborda as seguintes investigações:

- Estudo sobre a sensibilidade dos parâmetros η e γ do algoritmo AED_{MMGO};

- Comparar a versão de aprendizagem completa do modelo de mistura com a versão de aprendizagem ao longo do tempo, discutida no Capítulo 5;
- O algoritmo AED_{MMGO} será confrontado, em uma grande diversidade de cenários, com outros algoritmos populacionais desenvolvidos para otimização em ambientes dinâmicos.

Cabe ressaltar que, em cada cenário considerado, o elenco de algoritmos sob comparação é distinto. Isso se deve à disponibilidade de resultados na literatura junto a cada cenário.

6.4.1 Análise de sensibilidade dos parâmetros do AED_{MMGO}

A escolha adequada de valores dos parâmetros de um algoritmo é essencial para o seu bom desempenho. No entanto, para usuários sem experiência, essa escolha pode não ser tão direta. Para isso, referências sobre quais valores utilizar em determinados tipos de cenários se mostram relevantes.

Como foi discutido no Capítulo 5, o parâmetro $\eta \in [0, 1]$ controla a quantidade dos melhores indivíduos que serão utilizados para estimar o modelo probabilístico.

Conceitualmente, para valores altos de η , o algoritmo realizará uma maior exploração local, tendendo a ter uma taxa de convergência mais alta, pois a mesma quantidade ($100\eta\%$ da população) de indivíduos utilizados na estimação serão amostrados pelo modelo estimado, como pode ser vista no Algoritmo 7. Logo, poucos indivíduos serão inseridos na população de forma aleatória, diminuindo, assim, a diversidade da população. Já para valores baixos do parâmetro η , o algoritmo tende a convergir mais lentamente, uma vez que uma grande quantidade de novos indivíduos, aleatoriamente gerados, serão inseridos na população a cada geração, contribuindo para o aumento da diversidade da população e, assim, reduzindo a taxa de convergência.

Para verificar esta hipótese, um cenário de otimização deve ser definido. Nesta análise, utilizou-se um cenário muito frequente em análise de performance, denominado de *Scenario2*, proposto por Branke (2001b). A configuração dos parâmetros para este cenário é apresentada na tabela 6.1.

Os resultados obtidos pelo algoritmo AED_{MMGO} , utilizando diferentes valores do parâmetro η , quando aplicado ao *Scenario2*, são apresentados na figura 6.5. Os critérios considerados nesta análise são: a média do *offline error* e uma medida de diversidade da população no decorrer da execução, obtida pela média do desvio-padrão dos valores de *fitness* das soluções geradas pelo modelo de mistura. Esses critérios foram calculados tomando 50 execuções independentes do algoritmo e 100 alterações em cada execução. Neste experimento, foi utilizada uma população de 80 indivíduos e ao parâmetro γ (quantidade de informação do passado recente preservada) foi atribuído o valor 0,1.

Parâmetro	Valor
Intervalo de busca	[0,100]
Número de picos p	10
Número de dimensões	5
Altura dos picos	$\in [30, 70]$
Largura dos picos	$\in [1, 12]$
Número de AFs entre alterações	5000
Severidade s	1.0
Coefficiente de correlação λ	0

Tab. 6.1: Configuração do *Scenario2* definido para o *benchmark* de picos móveis.

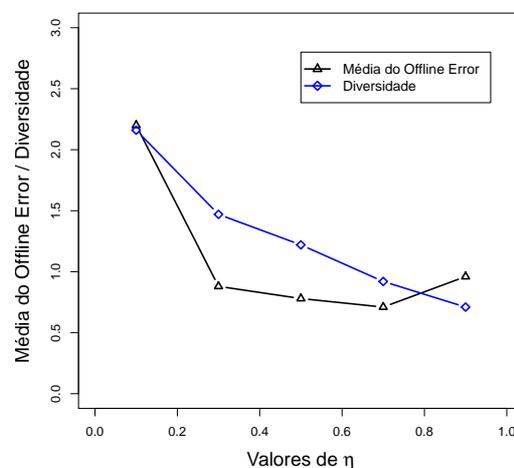


Fig. 6.5: Comportamento do algoritmo utilizando diferentes valores do parâmetro η .

O comportamento do algoritmo perante a variação do parâmetro η corrobora com as premissas levantadas anteriormente: um balanço entre a exploração local (maior quantidade de indivíduos utilizados para estimação e gerados pelo modelo) e a exploração global (novos indivíduos inseridos de forma aleatória) é a alternativa mais adequada. Nota-se que, com valores igualmente distribuídos entre exploração local e global, o algoritmo obteve melhores resultados para o *Scenario2*.

O parâmetro $\gamma \in [0, 1]$ determina o quanto as melhores soluções encontradas nas gerações anteriores influenciarão o modelo de mistura corrente. Valores baixos indicam que o modelo dará mais importância para as soluções atuais, enquanto que valores maiores fazem com que o modelo privilegie soluções passadas. A figura 6.6 ilustra o comportamento do algoritmo perante a variação do parâmetro γ . Para este caso de teste, foi utilizado $\eta = 0,5$.

É possível observar pela figura 6.6, uma redução na performance, em relação ao *Offline error*, quando aumenta-se a participação de soluções anteriores no modelo de mistura. Para valores de γ

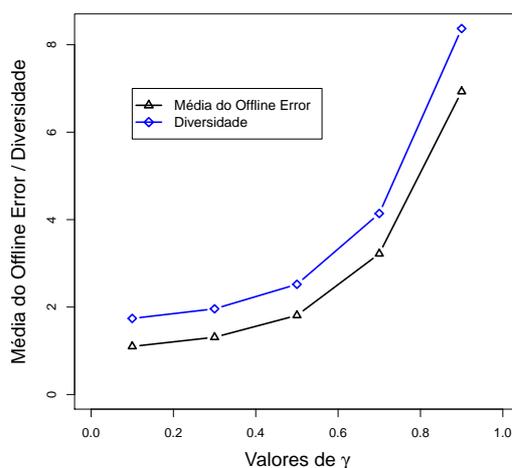


Fig. 6.6: Comportamento do algoritmo em relação à variação do parâmetro γ .

maiores, o modelo torna-se mais rígido, levando mais tempo para se adaptar às mudanças e, conseqüentemente, em ambientes com alterações mais frequentes, como o caso do problema testado, acaba obtendo resultados piores.

Nota-se também que a diversidade da população cresce com valores maiores de γ , indicando que para estes casos o algoritmo leva mais tempo para convergir, realizando uma maior exploração, o que pode ser interessante em ambientes onde as alterações ocorrem em intervalos de tempo maiores.

Com base nos resultados das simulações, é possível prover algumas diretivas sobre a configuração dos parâmetros η e γ . Para o parâmetro η , valores intermediários são mais indicados, dado que estes permitem que a exploração local e global sejam feitas de forma balanceada. Por outro lado, em ambientes onde as alterações ocorrem em intervalos de tempo maiores, é possível utilizar valores menores para η , o que contribui para a manutenção de diversidade na população.

Em situações em que o cenário de otimização é alterado muito rapidamente, valores baixos para o parâmetro γ são mais indicados, pois, com isso, o algoritmo alcança uma taxa de convergência mais alta. Carregar muitas informações do passado, acopladas às informações atuais, pode trazer benefícios tanto para a estimação do modelo de mistura, pois uma massa maior de dados estará disponível para tal, quanto para a exploração de regiões promissoras identificadas em gerações passadas e agregadas ao modelo atual. No entanto, valores altos podem “engessar” o algoritmo, reduzindo sua capacidade de adaptação ao novo ambiente. Sendo assim, valores menores e medianos são indicados para ambientes onde as alterações ocorrem com menor frequência.

6.4.2 Treinamento ao longo do tempo ou treinamento completo

As propostas de algoritmos de otimização em ambientes dinâmicos, apresentadas nesta pesquisa, empregam uma forma de aprendizado no decorrer do tempo. A cada iteração, estimacões simples do modelo de mistura são realizadas e, dado o caráter seletista e de melhorias contínuas dos algoritmos evolutivos, a estimacão tende a se tornar cada vez mais acurada.

A seguir, serão comparadas as duas metodologias de treinamento de modelos de mistura, quando empregados em algoritmos de otimização: treinamento ao longo do tempo, onde a convergência é obtida no decorrer do tempo; e treinamento completo do modelo a cada geração. Aqui, foi utilizada uma versão simples do AED_{MMG} , que emprega o EM padrão e não utiliza o módulo de controle de convergência. Foi utilizada a função `gmdistribution.fit()`, que implementa a estimacão dos parâmetros de um modelo de mistura gaussiano, através do algoritmo EM, a qual está disponível no Matlab®.

A média do *offline error* de 50 execuções são comparadas através da aplicacão de um teste de hipótese. Além disso, são analisados os tempos médios de execução dos algoritmos em uma geração, com o objetivo de verificar a reduçã do tempo computacional na abordagem de treinamento ao longo do tempo.

O teste t de *Student* é comumente utilizado quando os dados seguem uma distribuicão gaussiana. No entanto, quando esta suposiçã não é válida, um teste de hipótese não-paramétrico deve, então, ser utilizado, como o teste de Wilcoxon-Mann-Whitney (Bussab & Morettin, 2010), que não faz suposições sobre a distribuicão dos dados. Assim, a definiçã de qual método empregar passará pela aplicacão de um teste de aderência, que visa investigar se os dados seguem uma dada distribuicão teórica, como a gaussiana. Um teste de aderência gaussiana bastante utilizado é o teste de Shapiro-Wilk (Shapiro & Wilk, 1965).

A comparaçã aqui realizada será feita com base nestes três testes. Para cada configuraçã de espaço de busca analisado, valores de *severidade* e *número de picos*, o teste Shapiro-Wilk é aplicado e caso os dados seguirem uma distribuicão gaussiana o teste t é utilizado para comparaçã, caso contrário o teste de Wilcoxon-Mann-Whitney é empregado. Isto é realizado pois, quando os dados seguem uma distribuicão gaussiana, o teste t é mais poderoso, no entanto, quando esta não é verificada o teste de Wilcoxon-Mann-Whitney é mais eficiente (Bussab & Morettin, 2010). Para ambos os testes de hipótese, t de *Student* e Wilcoxon-Mann-Whitney, para o nível de significância foi utilizado valor igual a 5%.

A tabela 6.2 apresenta o ganho obtido (diferença relativa do *offline error* entre treinamento ao longo do tempo e treinamento completo) e o p -valor, entre parênteses, do teste t de *Student* ou Wilcoxon-Mann-Whitney (fundo destacado). Se o p -valor for menor do que 0,05 (5% de significância) a hipótese H_0 (de que as médias são iguais) é rejeitada e a hipótese H_1 (médias do

treinamento ao longo do tempo é menor) é aceita. Como em todos os experimentos realizados, o *offline error* obtido pelo treinamento ao longo do tempo foi menor do que com o treinamento completo, o ganho obtido é um valor entre [0,1) que informa a melhoria obtida pelo primeiro em relação ao segundo. Em outras palavras, informa o ganho proporcional obtido quando utilizou-se o treinamento ao longo do tempo ao invés do treinamento completo. Assim, quanto mais próximo de 1 for o ganho relativo, maior é a diferença entre as abordagens.

s	Número de picos						
	1	10	20	30	40	50	100
0	0,97 (0,000)	0,19 (0,000)	0,22 (0,000)	0,15 (0,000)	0,15 (0,004)	0,09 (0,018)	0,14 (0,004)
1	0,57 (0,000)	0,36 (0,000)	0,38 (0,000)	0,34 (0,000)	0,37 (0,000)	0,37 (0,000)	0,40 (0,000)
2	0,46 (0,000)	0,30 (0,000)	0,29 (0,000)	0,29 (0,000)	0,30 (0,000)	0,31 (0,000)	0,33 (0,000)
3	0,38 (0,000)	0,25 (0,000)	0,23 (0,000)	0,24 (0,000)	0,25 (0,000)	0,26 (0,000)	0,27 (0,000)
4	0,33 (0,000)	0,21 (0,000)	0,21 (0,000)	0,22 (0,000)	0,24 (0,000)	0,23 (0,000)	0,24 (0,000)
5	0,29 (0,000)	0,18 (0,000)	0,19 (0,000)	0,19 (0,000)	0,20 (0,000)	0,20 (0,000)	0,22 (0,000)
6	0,25 (0,000)	0,16 (0,000)	0,18 (0,000)	0,17 (0,000)	0,18 (0,000)	0,18 (0,000)	0,20 (0,000)

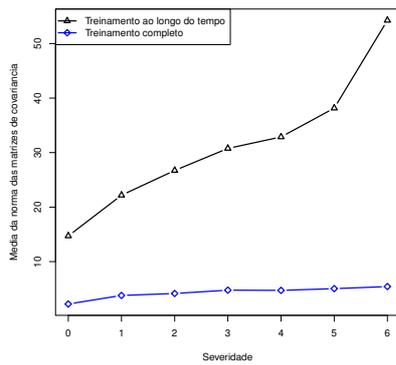
Tab. 6.2: Ganho obtido e resultados da aplicação do teste t de *Student* ou Wilcoxon-Mann-Whitney (fundo destacado) para comparação entre treinamento ao longo do tempo e treinamento completo a cada geração.

Em todos os casos, o algoritmo de otimização que utiliza treinamento do modelo de mistura ao longo do tempo foi superior. Na tabela 6.2, os p -valores foram truncados em três casas decimais. Dadas as 50 execuções e a grande diferença em termos da média do *offline error*, os teste t de *Student* ou de Wilcoxon-Mann-Whitney apontam com alto grau de certeza que existem diferenças significativas entre o treinamento ao longo do tempo e o treinamento completo, quando estes são utilizados para a estimação de modelos de mistura empregados como guia para algoritmos evolutivos. O ganho obtido pelo treinamento ao longo do tempo é elevado para todos os casos, destoando-se ainda mais em ambientes unimodais.

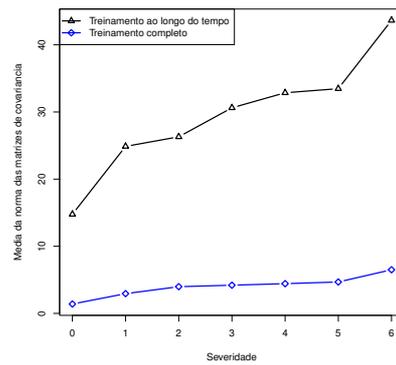
Uma possível explicação acerca do melhor desempenho obtido pela aprendizagem ao longo do tempo é que, devido às estimativas menos acuradas nas primeiras gerações, os componentes gaussianos cobrem regiões maiores do espaço de busca, contribuindo para uma melhor exploração do espaço de busca.

Para verificar esta hipótese, foi calculada a média da norma euclidiana das matrizes de covariância dos componentes do modelo, tanto na utilização do treinamento completo, quanto para o treinamento ao longo do tempo. A figura 6.7 apresenta o comportamento dos valores para cada um dos cenários analisados.

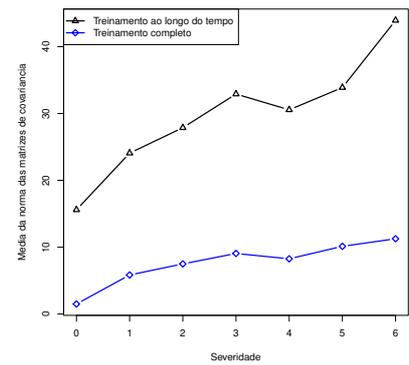
Em todas as situações analisadas, a média da norma da matriz de covariância dos componentes é sempre maior para o treinamento ao longo do tempo, quando comparado ao treinamento completo.



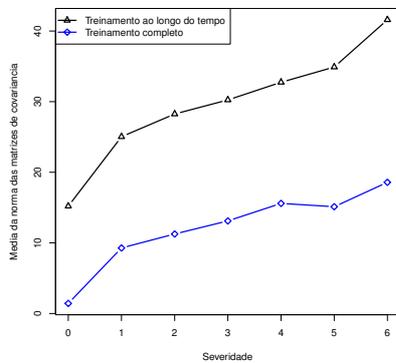
(a) Ambiente com 1 pico.



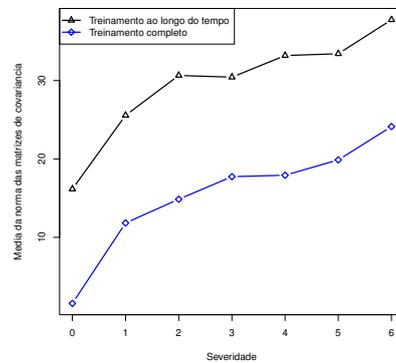
(b) Ambiente com 10 picos.



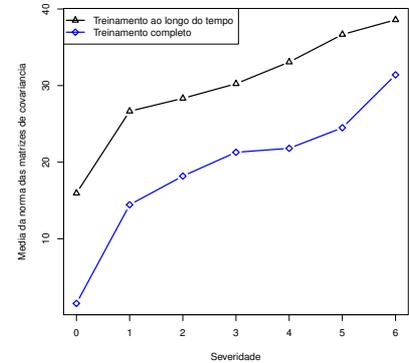
(c) Ambiente com 20 picos.



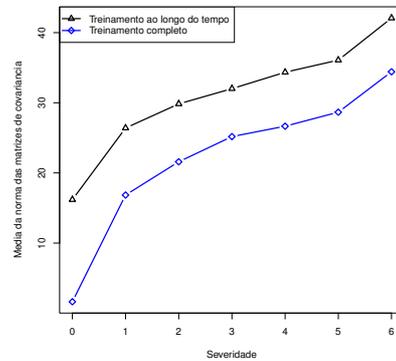
(d) Ambiente com 30 picos.



(e) Ambiente com 40 picos.



(f) Ambiente com 50 picos.



(g) Ambiente com 100 picos.

Fig. 6.7: Análise da média da norma da matriz de covariância dos componentes do modelo de mistura, quando empregado treinamento completo e ao longo do tempo. Quanto maior este valor, mais abertos são os componentes e, conseqüentemente, eles cobrem uma maior área do espaço de busca.

Este resultado corrobora a hipótese de melhor exploração do espaço busca pelo algoritmo de treinamento ao longo do tempo, devido à presença de componentes da mistura com maiores aberturas. Embora seja um indício da obtenção de melhores resultados, outras análises necessitam ainda ser realizadas, para verificar tal afirmação.

Além das melhorias apresentadas no contexto de otimização (média do *offline error*), o algoritmo de treinamento ao longo do tempo possui um menor custo computacional do que o treinamento completo, uma vez que executa apenas uma iteração (um passo E e um passo M) do algoritmo EM. A tabela 6.3 mostra o ganho computacional obtido pelo treinamento ao longo do tempo em relação ao treinamento completo (*speedup*).

s	Número de picos						
	1	10	20	30	40	50	100
↓							
0	1,78	1,76	1,74	1,74	1,74	1,73	1,71
1	1,92	1,91	1,90	1,90	1,90	1,89	1,87
2	1,90	1,89	1,88	1,88	1,87	1,86	1,84
3	1,91	1,87	1,86	1,85	1,84	1,84	1,81
4	1,89	1,86	1,84	1,83	1,81	1,80	1,77
5	1,89	1,83	1,82	1,79	1,78	1,79	1,74
6	1,89	1,81	1,79	1,77	1,76	1,75	1,71

Tab. 6.3: *Speedup* do algoritmo com treinamento ao longo do tempo em relação ao treinamento completo.

Utilizando o treinamento ao longo do tempo, o algoritmo AED_{MMGO} alcançou uma redução no custo computacional de, no mínimo, 70%. Esta melhoria é de extrema importância para problemas com alta dimensionalidade e/ou com muitas amostras (indivíduos na população de soluções), onde o algoritmo de treinamento EM demandará um maior custo computacional.

6.4.3 Comparação com outras propostas

A seguir, uma comparação ampla entre a abordagem proposta por esta pesquisa e algoritmos existentes na literatura de otimização em ambientes dinâmicos é realizada. Resultados empregando os dois geradores, BPM e BPM com rotação, são aqui considerados.

6.4.3.1 Análise sobre o BPM: variando a severidade, número de picos e dimensão do espaço de busca

Nesta subseção, uma análise comparativa é realizada sobre o BPM proposto por Branke (1999). As comparações serão realizadas em diferentes cenários de otimização, os quais foram escolhidos

devido ao fato de que, para estes cenários, há resultados de outros algoritmos publicados na literatura. Inicialmente, o *Scenario2* é considerado. Para este cenário, os algoritmos utilizados para comparação estão descritos a seguir:

- DynDE (Mendes & Mohais, 2005), algoritmo baseado em Evolução Diferencial;
- mCPSO (Blackwell & Branke, 2006), algoritmo baseado em Otimização por Enxame de Partículas (OEP);
- SPSO (Li et al., 2006), algoritmo baseado em OEP;
- CESO (Lung & Dumitrescu, 2007), algoritmo baseado em OEP;
- MMEO (Moser & Hendtlass, 2007), algoritmo baseado em Otimização Extrema;
- ESCA (Lung & Dumitrescu, 2010), algoritmo baseado em OEP;
- MS_{FGBF} (Kiranyaz et al., 2011), algoritmo baseado em OEP.

A tabela 6.4 apresenta os valores da média do *offline error* e o erro padrão obtido por cada algoritmo, quando aplicados no *Scenario2*. Exceto quando explicitamente indicado, em todas as simulações realizadas nesta seção a seguinte configuração de parâmetros do AED_{MMGO} foi utilizada: $\eta = 0,5$; $\gamma = 0,1$; população de indivíduos de tamanho 80 e, para o método de seleção por roleta, foram utilizados cinco competidores. O mecanismo de controle de convergência permanece inativo nestas simulações, sendo ativado apenas em espaços de busca com alta dimensionalidade.

Nota-se que a quantidade de indivíduos na população para cada algoritmo não é levada em consideração, uma vez que todos os algoritmos deverão realizar a mesma quantidade de avaliações de função, em cada intervalo de tempo que o ambiente se mantém estático. Se for utilizado um grande número indivíduos, uma melhor exploração do espaço de busca pode ser feita. No entanto, poucas gerações do algoritmo serão realizadas. Todos os algoritmos são executados sobre o mesmo ambiente dinâmico, em 50 execuções independentes, e em cada execução, 100 alterações no ambiente são consideradas.

Embora o algoritmo MMEO apresentou melhor performance (0,66) neste cenário, a proposta de algoritmo desenvolvida nesta pesquisa (AED_{MMGO}) se mostrou bastante competitiva frente a outros algoritmos analisados. Destaca-se também a estabilidade do algoritmo, o qual obteve o menor valor do erro padrão dentre os algoritmos analisados.

Variando a severidade das alterações

Algoritmo	Metaheurística base	$M_{OE}(SE_{M_{OE}})$
DynDE	ED	1,75 ($\pm 0,03$)
mCPSO	OEP	1,72 ($\pm 0,06$)
SPSO	OEP	1,93 ($\pm 0,06$)
CESO	OEP	1,38 ($\pm 0,02$)
MMEO	OE	0,66 ($\pm 0,20$)
ESCA	OEP	1,38 ($\pm 0,02$)
MS_{FGBF}	OEP	1,03 ($\pm 0,35$)
AED_{MMGO}	AED	1,01 ($\pm 0,01$)

Tab. 6.4: Performance dos algoritmos em relação ao *Scenario2* definido para o BPM.

A drasticidade das alterações impactam fortemente na complexidade do problema de otimização. Para verificar o desempenho frente cenários com diferentes severidades nas alterações, diversos valores do parâmetro s foram testados e os resultados estão presentes na tabela 6.5. Foram considerados os três algoritmos descritos, dados que estes apresentam resultados sobre os mesmos cenários, sendo possível a comparação.

s	CESO	ESCA	mCPSO	AED_{MMGO}
0	0,85 ($\pm 0,02$)	1,72 ($\pm 0,03$)	1,18 ($\pm 0,07$)	0,63 ($\pm 0,01$)
1	1,38 ($\pm 0,02$)	1,53 ($\pm 0,01$)	1,75 ($\pm 0,06$)	1,01 ($\pm 0,01$)
2	1,78 ($\pm 0,02$)	1,57 ($\pm 0,01$)	2,40 ($\pm 0,06$)	1,38 ($\pm 0,02$)
3	2,03 ($\pm 0,03$)	1,67 ($\pm 0,01$)	3,00 ($\pm 0,06$)	1,66 ($\pm 0,03$)
4	2,23 ($\pm 0,05$)	1,72 ($\pm 0,03$)	3,59 ($\pm 0,10$)	1,84 ($\pm 0,04$)
5	2,52 ($\pm 0,06$)	1,78 ($\pm 0,06$)	4,24 ($\pm 0,10$)	1,84 ($\pm 0,05$)
6	2,74 ($\pm 0,10$)	1,79 ($\pm 0,03$)	4,79 ($\pm 0,10$)	1,69 ($\pm 0,04$)

Tab. 6.5: Performance dos algoritmos para ambientes com diferentes níveis de severidade nas alterações.

O algoritmo AED_{MMGO} se mostrou bastante competitivo em todos os cenários analisados. Dada a rápida convergência de AEDs, em cenários onde o ótimo global alterna em determinadas regiões do espaço, ou para regiões próximas da posição anterior, métodos de otimização baseados neste princípio podem rapidamente alcançar a nova solução ótima. A tabela 6.5 mostra que, em ambientes com estas características (severidade baixa), o AED_{MMGO} obteve excelentes resultados. Além disso, alcançou bons resultados também em ambientes com alta severidade.

Variando o número de picos

Em geral, é mais simples encontrar o máximo global quando o número de picos é baixo. Para tornar o espaço de busca mais complexo, variou-se o número de picos no *Scenario2* e os resultados são mostrados na tabela 6.6. Assim como na comparação sobre a variação da severidade (tabela 6.5), os algoritmos citados foram escolhidos por publicarem seus resultados nestes mesmos cenários de otimização.

No. de picos	CESO	ESCA	mCPSO	AED _{MMGO}
1	1,04 ($\pm 0,00$)	0,98 ($\pm 0,00$)	4,93 ($\pm 0,07$)	0,91 ($\pm 0,01$)
10	1,38 ($\pm 0,02$)	1,54 ($\pm 0,01$)	1,75 ($\pm 0,06$)	1,14 ($\pm 0,03$)
20	1,72 ($\pm 0,02$)	1,89 ($\pm 0,04$)	2,42 ($\pm 0,06$)	1,37 ($\pm 0,05$)
30	1,24 ($\pm 0,01$)	1,52 ($\pm 0,02$)	2,48 ($\pm 0,06$)	1,32 ($\pm 0,04$)
40	1,30 ($\pm 0,02$)	1,61 ($\pm 0,02$)	2,55 ($\pm 0,10$)	1,42 ($\pm 0,04$)
50	1,45 ($\pm 0,01$)	1,67 ($\pm 0,02$)	2,50 ($\pm 0,10$)	1,58 ($\pm 0,04$)
100	1,28 ($\pm 0,02$)	1,61 ($\pm 0,03$)	2,36 ($\pm 0,10$)	1,75 ($\pm 0,06$)

Tab. 6.6: Performance dos algoritmos para espaços de busca com diferentes quantidades de picos e $s = 1$.

Para o cenário com severidade baixa ($s = 1$) o algoritmo AED_{MMGO} obteve bons resultados em todos os cenários, sendo melhor do que as propostas de algoritmos analisadas, em ambientes com menores quantidades de mínimos locais (1, 10 e 20 picos).

O mesmo cenário de otimização foi considerado, mas agora com a severidade das alterações aumentada ($s = 5$). Os resultados estão dispostos na tabela 6.7.

No. de picos	CESO	ESCA	AED _{MMGO}
1	1,26 ($\pm 0,00$)	1,17 ($\pm 0,00$)	1,65 ($\pm 0,05$)
10	2,52 ($\pm 0,06$)	1,80 ($\pm 0,06$)	1,76 ($\pm 0,04$)
20	3,22 ($\pm 0,12$)	2,16 ($\pm 0,04$)	1,81 ($\pm 0,04$)
30	2,62 ($\pm 0,04$)	1,96 ($\pm 0,04$)	1,79 ($\pm 0,04$)
40	2,23 ($\pm 0,05$)	1,94 ($\pm 0,09$)	1,76 ($\pm 0,05$)
50	2,52 ($\pm 0,05$)	1,88 ($\pm 0,04$)	1,82 ($\pm 0,04$)
100	2,49 ($\pm 0,05$)	1,92 ($\pm 0,04$)	1,82 ($\pm 0,05$)

Tab. 6.7: Performance dos algoritmos para espaços de busca com diferentes quantidade de picos e $s = 5$.

Neste cenário o algoritmo AED_{MMGO} se mostrou menos sensível à quantidade de picos no ambiente, em relação às outras abordagens, e alcançou melhores resultados na grande maioria dos ambientes considerados. Aqui, os resultados foram comparados apenas com os algoritmos CESO e ESCA, pois estes apresentaram seus resultados para o cenário com $s=5$.

Variando a dimensão do espaço de busca

Outro parâmetro que exerce grande impacto na performance dos métodos de otimização é a dimensão do espaço de busca. Em cenários com alta dimensionalidade, maior é o espaço de busca e mais difícil é a tarefa de obter o ótimo global. Simulações com problemas de diferentes dimensões foram realizadas e a tabela 6.8 apresenta os resultados, para $s=5$.

Para os espaços de busca de 50 e 100 dimensões, o módulo de controle de convergência está ativado, e ao parâmetro δ foi atribuído valor 0,005.

Dimensão	CESO	ESCA(6)	AED _{MMGO}
10	6,26 ($\pm 0,25$)	4,64 ($\pm 0,43$)	2,33 ($\pm 0,08$)
20	3,20 ($\pm 0,01$)	3,18 ($\pm 0,03$)	7,32 ($\pm 0,12$)
50	12,32 ($\pm 0,09$)	10,9 ($\pm 0,14$)	13,93 ($\pm 0,25$)
100	40,39 ($\pm 0,26$)	41,07 ($\pm 1,28$)	40,57 ($\pm 0,61$)

Tab. 6.8: Performance dos algoritmos em problemas com média e alta dimensionalidades, tomando $s = 5$.

Quanto maior a quantidade de amostras disponíveis para a estimação de um experimento gerador, por meio de um modelo probabilístico, maior será a confiança na representatividade do modelo. Logo, quanto mais amostras estiverem disponíveis, melhor será a estimativa da distribuição que os gerou.

Em problemas com alta dimensionalidade, o AED_{MMGO} necessita de uma quantidade maior de amostras do espaço de busca para realizar uma estimativa confiável. Assim, para os ambientes com 10 e 20 dimensões (média dimensionalidade), uma população com 120 indivíduos foi aqui utilizada. Já para problemas com alta dimensionalidade, 50 e 100, foram considerados 200 indivíduos.

Com base na tabela 6.8, é possível verificar o bom desempenho em problemas com muitas dimensões, principalmente com 10 e 100 dimensões, onde a proposta alcançou melhores resultados ou equivalentes, dado o erro padrão, aos obtidos pelos outros algoritmos analisados.

6.4.3.2 Análise sobre o BPM com rotação: variando o número de picos e frequência das alterações

Inicialmente, o algoritmo proposto é comparado com outros dois AEDs desenvolvidos para tratar de problemas em que a função-objetivo é variante no tempo, os quais são eles: *Improved Univariate Marginal Distribution Algorithm* (IUMDA) (Liu et al., 2008) e *Tri-EDA_G* (Yuan et al., 2008), descritos no Capítulo 5.

Os parâmetros do BPM com rotação, considerados neste experimento, exceto o intervalo entre as alterações, são os mesmos utilizados na competição de “Computação Evolutiva em Ambientes

Dinâmicos e Incertos”, ocorrida no *Congress on Evolutionary Computation* (CEC) de 2009 (Li et al., 2008), os quais estão descritos na tabela 6.9. Na competição, após transcorridas 100.000 AFs, o ambiente é alterado e a contagem de AFs é reiniciada. Nos experimentos aqui realizados, as alterações ocorrem com maior frequência, a cada 5.000 e 50.000 AFs o ambiente é alterado, produzindo ambientes mais dinâmicos.

Parâmetro	Valor
Intervalo de busca	[-5,5]
Número de picos p	10, 50
Número de dimensões	10, [5-15]
Altura dos picos	$\in [10, 100]$
Largura dos picos	$\in [1, 10]$
Número de AFs entre alterações	5.000, 50.000

Tab. 6.9: Configuração do BPM com rotação utilizado para comparação entre as propostas de algoritmo.

Quatro cenários de otimização, com diferentes níveis de complexidade em relação à frequência das alterações (medidos em número de AFs) e multimodalidade do espaço de busca (número de picos), foram construídos e estão especificados na tabela 6.10.

Cenário	Intervalo entre alterações (τ)	Multimodalidade (p)
1	5.000	10
2	5.000	50
3	50.000	10
4	50.000	50

Tab. 6.10: Configuração dos ambientes dinâmicos.

Em cada um dos cenários, os sete tipos de alterações (T_1 - T_7), implementadas pelo BPM com rotação, (veja Seção 6.2.1.1) foram aplicados, totalizando um conjunto de 28 casos de teste. Os cenários 1 e 2 são altamente dinâmicos, ou seja, a função-objetivo é alterada rapidamente e, se o algoritmo possui uma baixa taxa de convergência, provavelmente obterá uma baixa performance. Por outro lado, nos cenários 3 e 4 as alterações ocorrem em um intervalo de tempo maior. Assim, o algoritmo dispõe de mais tempo para encontrar a solução ótima. No entanto, deve ser capaz de resolver um problema de otimização global.

Para a seleção dos melhores indivíduos, AED_{MMGO} utilizou o método de seleção por torneio com cinco competidores, enquanto o $IUMDA$ e o $Tri-EDA_G$ empregam o método determinístico de seleção. Aqui, o módulo de controle de convergência permanece inativo em todas as simulações

realizadas. Os valores dos parâmetros livres de cada algoritmo, utilizado neste experimento, estão descritos na tabela 6.11.

Algoritmo	Parâmetro	Valor
IUMDA	μ	0,2
	P_{mut}	0,01
	σ_{mut}	0,05
Tri-EDA _G	ρ	0,3
	λ	0,8
AED _{MMGO}	η	0,5
	γ	0,1

Tab. 6.11: Configuração de parâmetros dos algoritmos analisados.

No algoritmo IUMDA, μ denota a porcentagem de indivíduos que serão utilizados em uma população temporária do algoritmo. Vários valores para μ foram testados e $\mu=0,2$ (ou seja, 20% dos melhores indivíduos) produziram melhores resultados, dado o tamanho da população considerado. O parâmetro de mutação (P_{mut}) foi determinado de forma similar.

Em relação ao Tri-EDA_G, ρ é a porcentagem dos melhores indivíduos selecionados para atualizar os parâmetros de uma distribuição gaussiana multivariada, utilizada pelo algoritmo para realizar exploração local. Já o parâmetro λ é a porcentagem da população da próxima geração, a ser gerada por esta distribuição gaussiana multivariada. Para ambos os parâmetros, foram utilizados os valores indicados pelos próprios autores (Yuan et al., 2008).

Uma característica compartilhada por todos os algoritmos é a aplicação de um mecanismo que garante que o melhor indivíduo da população corrente é sempre inserido na população seguinte.

Em todos os cenários estudados, a população foi composta por 100 indivíduos e o tempo de execução (medido pelo número de alterações) foi de 60 alterações.

As tabelas 6.12 e 6.13 apresentam a média do *offline error* e o erro padrão, para 20 execuções independentes, em cada um dos quatro cenários considerados. Os melhores resultados, para cada problema teste, estão destacados na tabela.

Em ambientes onde as alterações ocorrem com alta frequência, o AED_{MMGO} claramente supera os dois outros algoritmos. Uma grande vantagem da metodologia proposta nesta pesquisa é a exploração de vários picos simultaneamente, assim, aumentando a probabilidade de encontrar o ótimo global, ou pelo menos um ótimo local melhor.

Assim como nos outros cenários, em ambientes com menor frequência das alterações, o AED_{MMGO} obteve melhor performance, quando comparado às outras duas abordagens de AEDs também propostas para atacar problemas de otimização em ambientes dinâmicos. Nota-se que, em alguns casos, a performance do AED_{MMGO} é considerada equivalente ao Tri-EDA_G, dado o erro

	Cenário 1			Cenário 2		
	IUMDA	Tri-EDA _G	AED _{MMGO}	IUMDA	Tri-EDA _G	AED _{MMGO}
T ₁	77,43(±0,75)	47,64(±1,26)	6,89(±0,94)	80,18(±1,37)	48,34(±0,49)	17,60(±1,57)
T ₂	76,78(±1,13)	48,75(±1,23)	16,66(±2,43)	73,00(±0,94)	37,79(±1,34)	14,43(±1,48)
T ₃	62,09(±1,30)	35,91(±1,54)	18,24(±1,89)	62,98(±1,01)	35,86(±1,27)	20,36(±2,12)
T ₄	76,21(±0,23)	53,06(±0,48)	9,93(±1,33)	76,36(±0,19)	47,44(±0,57)	18,06(±1,70)
T ₅	77,08(±0,50)	74,51(±0,32)	37,53(±1,05)	78,87(±0,62)	73,47(±0,49)	44,50(±1,14)
T ₆	73,11(±0,83)	68,34(±0,74)	38,82(±1,36)	74,93(±0,96)	64,61(±0,46)	49,74(±1,63)
T ₇	60,61(±1,42)	35,92(±2,48)	24,89(±2,11)	65,46(±1,02)	36,80(±1,12)	32,77(±1,66)

Tab. 6.12: Média do *offline error* e erro padrão para os cenários 1 e 2.

	Cenário 3			Cenário 4		
	IUMDA	Tri-EDA _G	AED _{MMGO}	IUMDA	Tri-EDA _G	AED _{MMGO}
T ₁	10.47(±0.85)	6.48(±0.57)	1.93(±0.24)	14.17(±2.30)	8.62(±1.16)	2.61(±0.43)
T ₂	14.29(±0.97)	11.64(±0.86)	8.89(±0.85)	12.96(±1.53)	8.93(±1.00)	5.93(±0.60)
T ₃	18.16(±2.09)	15.65(±2.00)	14.00(±1.95)	16.19(±1.18)	14.10(±1.81)	12.33(±1.65)
T ₄	8.13(±0.11)	7.01(±0.16)	5.07(±0.07)	7.34(±0.30)	6.11(±0.12)	3.37(±0.10)
T ₅	43.32(±0.01)	40.36(±1.10)	10.00(±0.41)	45.03(±0.01)	32.54(±0.86)	9.39(±0.51)
T ₆	41.91(±0.65)	36.25(±0.93)	21.85(±0.90)	39.72(±1.63)	35.79(±0.83)	32.85(±0.87)
T ₇	16.63(±2.37)	15.49(±2.31)	12.77(±2.28)	17.42(±0.81)	15.58(±0.60)	12.97(±0.53)

Tab. 6.13: Média do *offline error* e erro padrão para os cenários 3 e 4.

padrão.

Em problemas de otimização global, o modelo de mistura, inicialmente, cobre vários picos e, por meio dos operadores de seleção e amostragem, procura pelas melhores regiões dentro desta cobertura inicial. Uma vez encontrados, aplica-se nestas regiões promissoras uma exploração auxiliada por uma distribuição gaussiana com o intuito de alcançar o ótimo local.

As figuras 6.8(a), 6.8(b) e 6.8(c) mostram o comportamento relativo mediano, sobre 20 execuções independentes, de cada algoritmo, quando aplicados ao Cenário 2 com alterações do tipo T₇. A performance é medida pela razão entre a melhor solução encontrada e o ótimo global. Assim, a performance ideal deve estar tão próxima quanto possível do valor 1.

É possível identificar que a provável perda de diversidade na população do algoritmo IUMDA impacta fortemente em sua performance. No tempo $t=1$, onde a população é gerada aleatoriamente (logo, possui alta diversidade nas soluções) o algoritmo encontra bons ótimos locais, mas para $t>1$ esta situação não se repete. Um mecanismo de inserção de variabilidade no conjunto de soluções seria de extrema importância para este método de otimização.

O efeito oscilatório da complexidade do problema, causado pelas modificações no número de dimensões do espaço de busca, é claramente refletida no desempenho do algoritmo Tri-EDA_G. Este

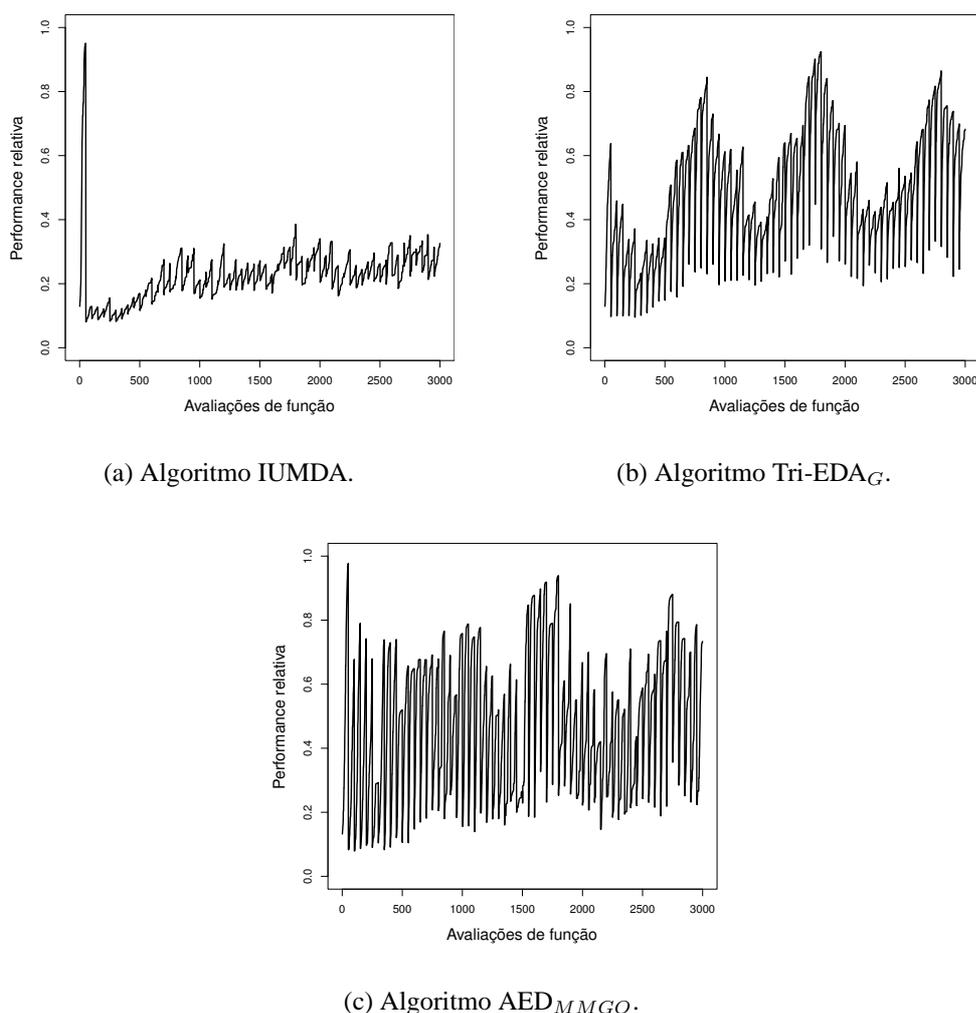


Fig. 6.8: Performance relativa dos algoritmos para o Cenário 2 com alterações T_7 .

efeito não é evidente no comportamento do AED_{MMGO} , indicando que este último algoritmo obtém melhores resultados em problemas com dimensões mais elevadas.

Como discutido anteriormente, o algoritmo AED_{MMGO} é capaz de controlar automaticamente o número de componentes no modelo de mistura. A figura 6.9 mostra a variação do número de componentes do modelo de mistura empregado pelo AED_{MMGO} , no Cenário 1 (10 picos e 5.000 AFs entre as alterações). Em ambientes onde as alterações são mais suaves, como as alterações T_1 e T_2 , o modelo de mistura não emprega muitos componentes, exceto em fases iniciais onde ainda não há conhecimento sobre o espaço de busca e um maior número de componentes se faz necessário. Isto se deve ao fato de que, após encontrado boas soluções, e possivelmente a solução ótima, pouco esforço é necessário para rastrear o novo ótimo, dado que alterações suaves levam a indivíduos relativamente próximos aos antecessores. Já em alterações mais drásticas, um maior esforço é necessário para que

o novo ótimo seja encontrado, utilizando assim um maior número de componentes no modelo de mistura.

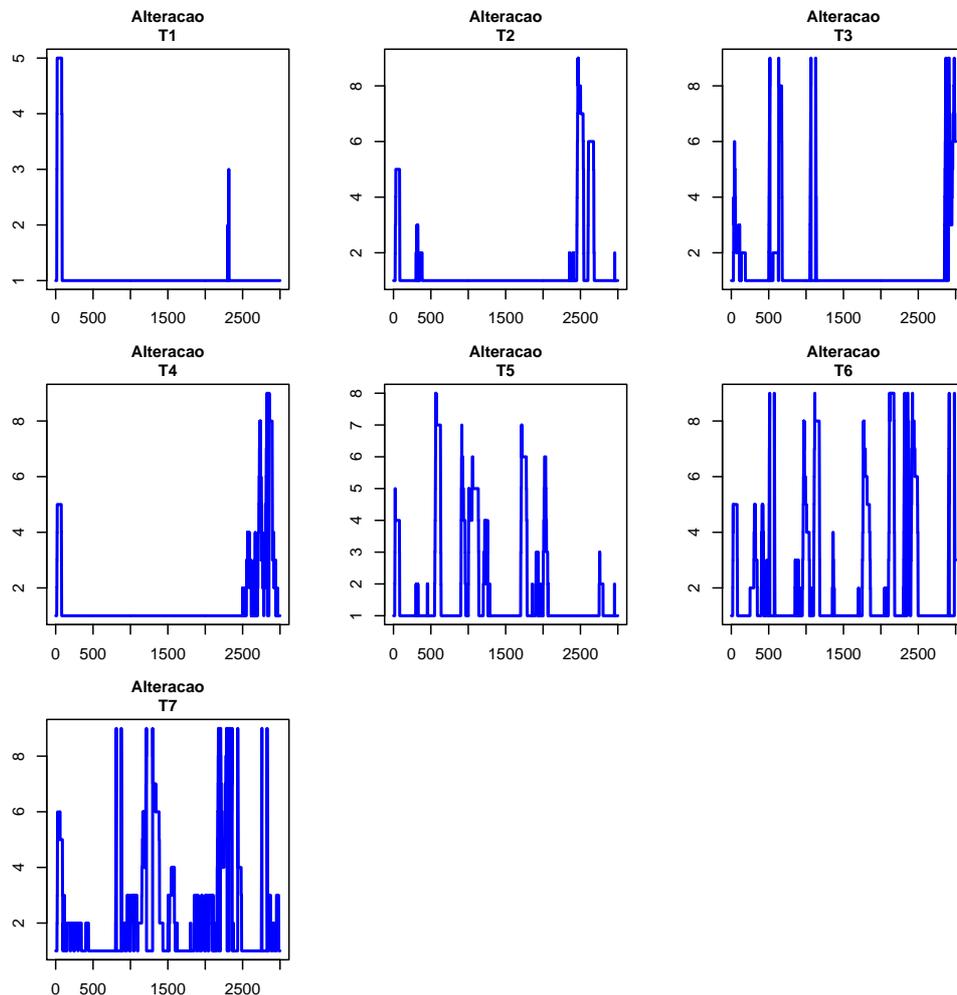


Fig. 6.9: Variação do número de componentes do modelo de mistura utilizado no AED_{MMGO} (eixo y) no decorrer do tempo (número de avaliações de função - eixo x) para o Cenário 1 (10 picos). A variação em cada um dos sete tipos de alterações, descritas no BPM com rotação, estão ilustradas.

Na figura 6.10 é ilustrada a variação para Cenário 2 (50 picos e 5.000 AFs entre as alterações). É possível notar que, neste cenário, há uma maior quantidade de componentes no modelo de mistura, dado que existem mais picos a serem explorados. Sendo nos ambientes com alterações mais drásticas as maiores quantidades.

Análise em ambientes com baixa frequência de alterações

Esta análise comparativa é feita empregando o BPM com rotação, utilizado na competição de

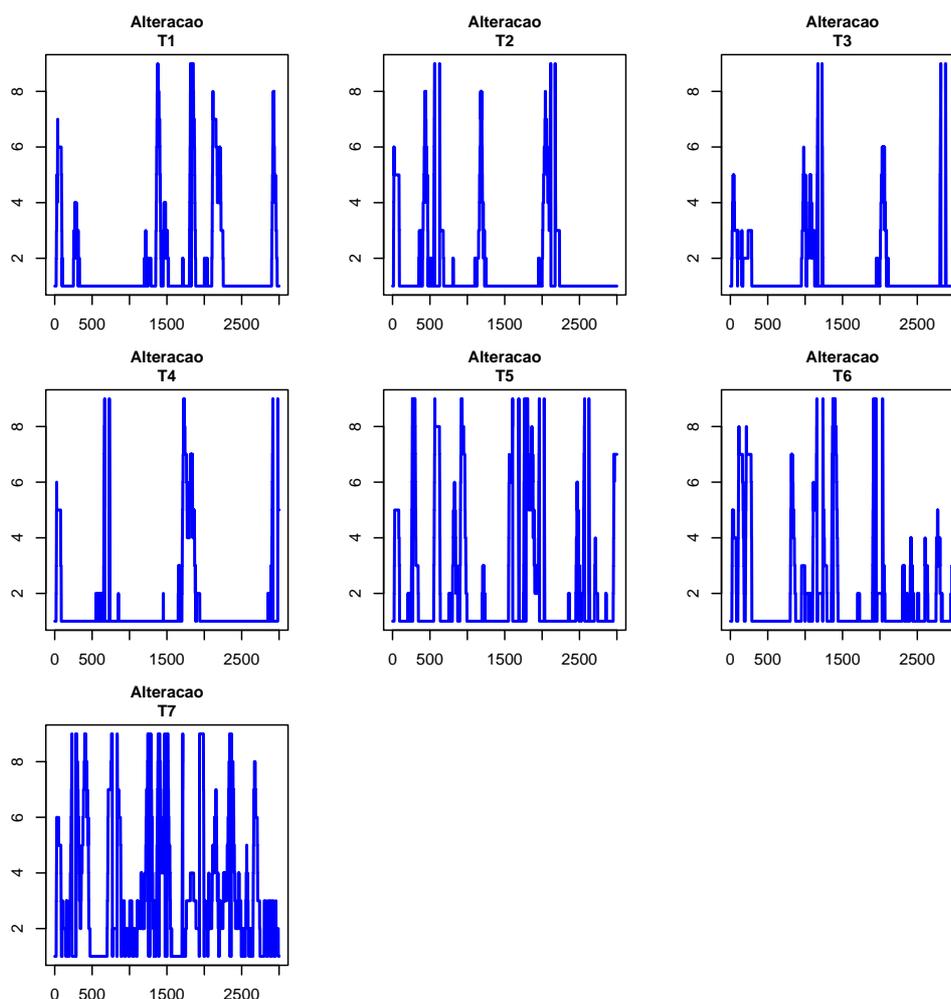


Fig. 6.10: Variação do número de componentes do modelo de mistura utilizado no AED_{MMGO} (eixo y) no decorrer do tempo (número de avaliações de função - eixo x) para o Cenário 2 (50 picos). A variação em cada um dos sete tipos de alterações, descritas no BPM com rotação, estão ilustradas.

otimização em ambientes dinâmicos, ocorrida no *Congress on Evolutionary Computation - CEC* de 2009 (Li et al., 2008).

Neste cenário de otimização, as alterações ocorrem a cada 100.000 AFs e as configurações dos parâmetros do BPM com rotação são as mesmas especificadas na tabela 6.9.

Os algoritmos que participaram da competição, e cujas performances serão comparadas com aquela produzida pelo AED_{MMGO} , estão listados abaixo:

- DASA (Korošec & Šilc, 2009), algoritmo baseado em Otimização por Colônia de Formigas;
- CPSO (Li & Yang, 2009), algoritmo baseado em Otimização por Enxame de Partículas;

- dopt-aiNet (de França & Von Zuben, 2009), algoritmo baseado em Sistemas Imunológicos Artificiais;
- UEP (Yu & Suganthan, 2009), algoritmo baseado em Programação Evolutiva;
- jDE (Brest et al., 2009), algoritmo baseado em Evolução Diferencial.

Nestas simulações, o módulo de controle de convergência está ativo e ao parâmetro δ foi atribuído valor 0,005. Em ambientes com baixa frequência de alterações, ao algoritmo é dado um maior tempo para explorar o espaço de busca. Nesse sentido, o módulo de controle de convergência contribui para esta exploração, sem que o algoritmo perca a informação dos melhores indivíduos encontrados até o momento.

Vários critérios de avaliação foram considerados para competição e, dentre eles, o mais informativo, o qual engloba diversas análises do comportamento do algoritmo no decorrer da execução, é denominado de *medida de marcação* (do inglês “*marking measure*”).

Basicamente, esta métrica define um conjunto de pontos de marcação igualmente espaçados (a cada 100 AFs) na curva de performance do algoritmo e, em cada ponto, calcula a razão entre a melhor solução encontrada até o momento e o ótimo global. Com estas informações, é possível fazer uma análise sobre toda a curva de performance do algoritmo, verificando sua convergência em diversos pontos da execução. A medida de marcação retorna um valor definido no intervalo [0,1], sendo que quanto mais próximo de 1, melhor é o desempenho do algoritmo. As tabelas 6.14 e 6.15 apresentam os resultados das cinco propostas de algoritmos que participaram da competição e do AED_{MMGO}, em ambientes com 10 e 50 picos, respectivamente, para cada um dos sete tipos de alteração (T₁-T₇).

	DASA	CPSO	dopt-aiNet	UEP	jDE	AED _{MMGO}
T ₁	0.981	0.942	0.902	0.854	0.985	0.981
T ₂	0.905	0.892	0.756	0.783	0.913	0.982
T ₃	0.853	0.869	0.764	0.769	0.922	0.841
T ₄	0.944	0.977	0.776	0.923	0.981	0.989
T ₅	0.931	0.889	0.706	0.821	0.929	0.972
T ₆	0.903	0.882	0.674	0.834	0.942	0.826
T ₇	0.885	0.857	0.770	0.730	0.911	0.830

Tab. 6.14: Medidas de marcação dos algoritmos para o cenário com 10 picos.

Com base nos resultados obtidos, o algoritmo AED_{MMGO} se mostra bastante competitivo frente às outras abordagens em ambos os cenários. Em ambientes com alterações moderadamente drásticas (*passo largo* - T₂), caóticas (T₄) e recorrentes (T₅), apresentou performance superior aos algoritmos analisados. Destaque-se também o desempenho alcançado em situações em que as alterações ocorrem de maneira suave (T₁), onde obteve resultados praticamente equivalentes ao algoritmo jDE.

	DASA	CPSO	dopt-aiNet	UEP	jDE	AED _{MMGO}
T ₁	0.970	0.941	0.894	0.853	0.979	0.978
T ₂	0.893	0.888	0.798	0.749	0.906	0.981
T ₃	0.827	0.838	0.791	0.706	0.902	0.844
T ₄	0.949	0.975	0.806	0.930	0.980	0.984
T ₅	0.959	0.918	0.742	0.900	0.958	0.967
T ₆	0.897	0.873	0.707	0.901	0.925	0.795
T ₇	0.832	0.873	0.786	0.752	0.899	0.837

Tab. 6.15: Medidas de marcação dos algoritmos para o cenário com 50 picos.

Estas simulações e comparações provêm indícios de que a proposta desenvolvida nesta pesquisa é capaz de tratar problemas de otimização global, uma vez que as alterações nos cenários acima ocorrem de forma bastante espaçada no tempo.

6.5 Considerações finais

A experimentação é uma etapa fundamental em uma pesquisa, onde são definidos um conjunto diversificado de cenários e situações, junto aos quais a metodologia proposta deve ser aplicada. Assim, é possível avaliar seu desempenho e compará-lo com resultados obtidos por outras metodologias encontradas na literatura.

Com este objetivo, neste capítulo foram descritos os principais cenários utilizados para análise de algoritmos populacionais de otimização, desenhados para trabalhar com funções-objetivo variantes no tempo.

Os algoritmos estado-da-arte da área de pesquisa foram utilizados e comparados com a proposta de algoritmo desenvolvido nesta pesquisa, denominado AED_{MMGO}. Os resultados mostraram que o AED_{MMGO} é competitivo e alcançou bons resultados em diferentes cenários de otimização. Em ambientes com baixa frequência de alterações, assim como em espaços com alta dimensionalidade, a abordagem de controle de convergência se mostra promissora, pois promove uma melhor exploração do espaço de busca, permitindo ao algoritmo alcançar melhores resultados.

A proposta de treinamento do modelo de mistura ao longo do tempo se mostrou promissora, pois, além do ganho computacional, obteve melhorias estatisticamente significativas nos cenários analisados, em relação ao treinamento completo a cada geração, no sentido da média do *offline error*.

Capítulo 7

Conclusões

Problemas de otimização cuja função-objetivo varia no decorrer do tempo têm ganhado bastante atenção nos últimos anos, devido à grande variedade de problemas encontrados no mundo real que compartilham desta característica.

Nesta pesquisa foi abordada a aplicação de uma classe de algoritmos populacionais que utilizam modelos probabilísticos como guia para a exploração de regiões promissoras do espaço de busca, chamados Algoritmos de Estimação de Distribuição (AEDs). Uma versão parcimoniosa de modelos de mistura gaussianos foi aqui utilizada, sendo esta capaz de controlar automaticamente o número de componentes no modelo, dependendo da multimodalidade do espaço de busca, além de utilizar uma forma de treinamento ao longo do tempo, onde a convergência do algoritmo *Expectation-Maximization* (EM) se dá no decorrer das iterações do algoritmo de otimização. Mecanismos de controle de convergência e de manutenção de diversidade também são empregados para auxiliar a adaptação do algoritmo às variações do ambiente. Uma variante *online* do algoritmo EM também foi analisada. Esta metodologia permite ao modelo de mistura incorporar boas soluções encontradas em um passado recente.

O *Benchmark* de Picos Móveis (BPM) foi utilizado para gerar ambientes dinâmicos. Este gerador tem sido amplamente utilizado para a análise de performance de algoritmos de otimização em ambientes dinâmicos e, assim, diversos resultados foram publicados considerando este gerador.

Os algoritmos propostos foram comparados a métodos bem conhecidos na literatura, junto a ambientes dinâmicos com diferentes níveis de complexidade, em relação a severidade das alterações, multimodalidade e dimensionalidade do espaço de busca. Os resultados mostraram que as abordagens propostas alcançaram performances destacadas nestes cenários, quando comparadas aos métodos encontrados na literatura.

Algoritmos de estimação de distribuição, assim como outras meta-heurísticas, são plenamente capazes de lidar com problemas de otimização em ambientes dinâmicos, uma vez a eles acoplados

operadores específicos para estes cenários, como: controle de convergência, manutenção de diversidade da população em toda a execução do algoritmo, ou ainda utilização de memória e múltiplas populações.

Em relação a outras meta-heurísticas, os AEDs com modelos de mistura possuem uma importante vantagem: a capacidade de explorações simultâneas de regiões promissoras, por meio da utilização de vários componentes gaussianos. E neste trabalho procurou-se, ainda, definir automaticamente o número de componentes da mistura.

Esta pesquisa abordou vários aspectos acerca da aplicação de algoritmos de estimação de distribuição em problemas de otimização com variáveis contínuas e função-objetivo variante no tempo. Dentre suas principais contribuições, é possível destacar:

- estudo sobre algoritmos populacionais, destacando os métodos evolutivos e principalmente a classe de Algoritmos de Estimação de Distribuição e suas aplicações na otimização de ambientes variantes no tempo;
- proposição de operadores específicos a serem empregados em algoritmos de otimização populacionais para trabalhar em ambientes dinâmicos;
- investigação da aplicabilidade de modelos probabilísticos mais flexíveis e parcimoniosos do que os modelos encontrados na literatura de AEDs, para otimização de espaços contínuos e função-objetivo dinâmica;
- proposta de treinamento do modelos de mistura ao longo do tempo, o qual mostrou ser capaz de reduzir o tempo computacional do algoritmo e ainda alcançar melhores resultados no contexto estudado, quando comparado ao treinamento completo a cada geração;
- emprego de treinamento *online* do modelo de mistura, o qual permite que o algoritmo incorpore novidades e também preserve junto ao modelo atual informações adquiridas no passado;
- desenvolvimento de ferramentas de otimização em ambientes dinâmicos baseadas em AEDs, as quais se mostraram bastante competitivas quando comparadas com as técnicas disponíveis atualmente.

Embora a aplicação de meta-heurísticas para otimização em ambientes dinâmicos vem sendo bastante estudada no últimos anos, poucos avanços foram feitos quanto ao emprego de AEDs para otimização de espaços contínuos e funções-objetivo variantes no tempo. Logo, esta área de pesquisa ainda se mostra em fase inicial e diversas melhorias junto aos algoritmos propostos podem ser realizadas. A seguir, algumas direções a serem seguidas são destacadas.

- Em espaços de busca muito grandes, nem sempre é possível realizar uma boa estimativa baseando-se em poucas amostras. Assim, a divisão do espaço de busca em sub-regiões e a alocação de uma sub-população em cada região, promovendo uma abordagem multipopulacional, pode ser benéfica ao algoritmo, particularmente quanto à sua capacidade de exploração do ambiente;
- Definição de novas regras para a inserção de componentes no modelo de mistura, como a proposta baseada em distância introduzida por Priebe (1994);
- Emprego de memórias explícitas para armazenar e resgatar boas soluções encontradas em cenários anteriores;
- Utilização de outras variantes *online* do algoritmo *Expectation-Maximization*, como as propostas de Sato & Ishii (2000) e de Cappé & Moulines (2009);
- O modelo de mistura considera todos os dados com a mesma importância. No contexto de otimização, a qualidade das soluções (valor de *fitness*) é uma informação que pode ser considerada pelo modelo probabilístico. Os chamados *modelos de mistura ponderados* (do inglês, *weighted mixture model*) (Markatou, 2000), que levam em consideração a importância relativa dos dados, surgem como uma interessante abordagem a ser investigada.

Trabalhos Publicados Pelo Autor

1. A.R Gonçalves & F.J. Von Zuben. Hybrid evolutionary algorithm guided by a fast adaptive Gaussian mixture model applied to dynamic optimization problems. In: *Proceedings of III Workshop on Computational Intelligence (WCI'2010)- Joint Conference*, São Bernardo do Campo, São Paulo, pag. 553–558, ISBN: 978-8-5766-9250-8, Outubro 2010.
2. A.R Gonçalves & F.J. Von Zuben. Online learning in estimation of distribution algorithm for dynamic environments. In: *Proceedings of 2011 IEEE Congress on Evolutionary Computation (CEC'11)*, New Orleans, USA, pag. 1–8, ISBN: 978-1-4244-7833-11, June 2011.

Referências Bibliográficas

- C.S. Adjiman, I.P. Androulakis, C.A. Floudas & A. Neumaier. A global optimization method, α -BB, for general twice- differentiable constrained NLPs - I. Theoretical advances. *Computers & Chemical Engineering*, 22:1137–1158, 1998.
- R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J.L. Flores, J.A. Lozano, V.Y. Peer, R. Blanco, V. Robles, C. Bielza & P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData mining*, 1(1):1–12, 2008.
- T. Bäck & H.P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.
- S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.
- S. Baluja & S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of 14th International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann, 1997.
- S. Baluja & S. Davies. Fast Probabilistic Modeling for Combinatorial Optimization. In *Proceedings of the American Association for Artificial Intelligence*, pages 469–476, 1998.
- M.S. Bazaraa & J.J. Jarvis. *Linear programming and network flows*. John Wiley & Sons, 1977.
- R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- A.D. Bethke. *Genetic algorithms as function optimizers*. PhD thesis, University of Michigan, 1981.
- L.T. Biegler & I.E. Grossmann. Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192, 2004.

- C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st edition, October 2007.
- T. Blackwell & J. Branke. Multiswarm, exclusion, and anti-convergence in dynamic environments. *IEEE Trans Evol Comput*, 10(4):459–472, 2006.
- P.A.N. Bosman & D. Thierens. An algorithmic framework for density estimation based evolutionary algorithms. Technical Report UU-CS-1999-46, Utrecht University, 1999.
- P.A.N. Bosman & D. Thierens. Mixed IDEAs. Technical Report UU-CS-2000-45, Utrecht University, 2000a.
- P.A.N. Bosman & D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In M. Pelikan, H. Mühlenbein & A.O. Rodriguez, editors, *Optimization by Building and Using Probabilistic Models (OBUPM) - Workshop at the Genetic and Evolutionary Computation Conference GECCO'00*. Morgan Kaufmann Publishers, 2000b.
- S. Boyd & L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.
- P.S. Bradley, U. Fayyad & C. Reina. Scaling EM (expectation-maximization) clustering to large databases. Technical Report MSR-TR-9, Microsoft Research, Redmond, WA, 1998.
- J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the Congress on Evolutionary Computation CEC'99*, pages 1875–1882, 1999.
- J. Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *GECCO - Workshop on evolutionary algorithms for dynamic optimization problems*, pages 134–137, 2001a.
- J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Norwell, MA, USA, 2001b.
- J. Brest, A. Zamuda, B. Bošković, M.S. Maučec & V. Žumer. Dynamic optimization using self-adaptive differential evolution. In *Proceedings of the Congress on Evolutionary Computation CEC'09*, pages 415–422, 2009.
- W.O. Bussab & P.A. Morettin. *Estatística básica*. Editora Saraiva, 6a. edition, 2010.
- O. Cappé & E. Moulines. On-line Expectation–Maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.
- A. Cervantes & L.T. Biegler. Optimization strategies for dynamic systems. *Encyclopedia of Optimization*, 5:400–413, 2001.

- A.C. Chiang. *Elements of Dynamic Optimization*. Waveland Pr Inc, December 1999.
- C. Chow & C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.
- J. Clausen. Branch and bound algorithms - principles and examples. *Parallel Computing in Optimization*, pages 239–267, 1997.
- H.G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Naval Research Laboratory, 1990.
- G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, August 1998.
- C. Darwin. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. *New York: D. Appleton*, 1859.
- J.S. De Bonet, C.L. Isbell & P. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 4, pages 424–430. The MIT Press, 1997.
- F.O. de França & F.J. Von Zuben. A dynamic artificial immune algorithm applied to challenging benchmarking problems. In *Proceedings of the Congress on Evolutionary Computation CEC'09*, pages 423–430. IEEE Press, 2009.
- F.O. de França, F.J. von Zuben & L.N. de Castro. An artificial immune network for multimodal function optimization on dynamic environments. In *Proceedings of the Conference on Genetic and Evolutionary Computation GECCO'05*, pages 289–296. ACM, 2005.
- A.P. Dempster, N.M. Laird & D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
- W. Dong & X. Yao. NichingEDA: Utilizing the diversity inside a population of EDAs for continuous optimization. In *Proceedings of the Congress on Evolutionary Computation CEC'08*, pages 1260–1267. IEEE Press, 2008.
- W. Dong & Xin Yao. Covariance matrix repairing in Gaussian based EDAs. In *Proceedings of the Congress on Evolutionary Computation CEC'07*, pages 415 –422, 2007.
- M. Dorigo, M. Birattari & T. Stützle. Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique. *IEEE Comput. Intell. Mag*, 1:28–39, 2006.

- R.O. Duda, P.E. Hart & D.G. Stork. *Pattern classification*. Wiley, 2nd edition, November 2001.
- R.G. Duraiski. Otimização dinâmica em tempo real utilizando modelos não-lineares simplificados. Tese de Doutorado. Escola de Engenharia Química. Universidade Federal do Rio Grande do Sul.
- R. Etxeberria & P. Larrañaga. Global optimization using Bayesian networks. In *Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 332–339, 1999.
- T.A. Feo & M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.
- R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 1987.
- L. J. Fogel, A. J. Owens & M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966.
- D. Freedman & P. Diaconis. On the histogram as a density estimator: L2 theory. *Probability Theory and Related Fields*, 57(4):453–476, December 1981.
- K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press Professional, Inc., San Diego, CA, USA, 2nd edition, 1990.
- M. Gallagher, M. Frean & T. Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'99*, pages 840–846, 1999.
- J. Gámez, J. Mateo & J. Puerta. EDNA: Estimation of Dependency Networks Algorithm. In J. Mira, J. and Álvarez, editor, *Bio-inspired Modeling of Cognitive Tasks*, volume 4527 of *Lecture Notes in Computer Science*, pages 427–436. Springer Berlin / Heidelberg, 2007.
- F. Glover & R. Marti. Tabu search. *Metaheuristic Procedures for Training Neural Networks*, pages 53–69, 2006.
- D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
- S.M. Goldfeld, R.E. Quandt & H.F. Trotter. Maximization by quadratic hill-climbing. *Econometrica*, 34(3):pp. 541–551, 1966.
- J. Grefenstette. Genetic algorithms for changing environments. In *Parallel Problem Solving from Nature 2*, pages 137–144. Elsevier, 1992.

- M. Guntsch, J. Branke, M. Middendorf & H. Schmeck. ACO strategies to dynamic optimization problems. In *In Proc. ANTS Workshop*, pages 59–62, 2000.
- M. Guntsch, M. Middendorf & H. Schmeck. An ant colony optimization approach to dynamic TSP. In *In Genetic Evol. Comp. Conference*, pages 860–867, 2001.
- G. Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL 99010, University of Illinois at Urbana-Champaign, 1999.
- G. Harik, E. Cantú-Paz, D.E. Goldberg & B.L. Miller. The Gambler’s Ruin Problem, Genetic Algorithms, and the Sizing of Populations. In *Proceedings of the Conference on Evolutionary Computation CEC’97*, pages 7–12, 1997.
- G. Harik, F. Lobo & K. Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga). In Martin Pelikan, Kumara Sastry & Erick Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling*, volume 33 of *Studies in Computational Intelligence*, pages 39–61. Springer Berlin / Heidelberg, 2006.
- G.R. Harik, F.G. Lobo & D.E. Goldberg. The compact genetic algorithm. In *Proceedings of the IEEE Conference of Evolutionary Computation*, pages 523–528, 1998.
- E. Hart & P. Hoss. An immune system approach to scheduling in changing environments. In *In Proc. Genetic Evol. Comp. Conference*, pages 1559–1566, 1999.
- J.H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Harbor, 1975.
- E.M. Iyoda. Inteligência computacional no projeto automático de redes neurais híbridas e redes neurofuzzy heterogêneas, 2000. Dissertação de Mestrado. Faculdade de Engenharia Elétrica e de Computação (FEEC). Universidade Estadual de Campinas (Unicamp).
- Y. Jin & J. Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3):303–317, 2005.
- N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4): 373–395, December 1984.
- J. Kennedy & R. Eberhart. *Swarm Intelligence*. Morgan Kauffman, San Mateo, CA, 2001.
- S. Kiranyaz, J. Pulkkinen & M. Gabbouj. Multi-dimensional particle swarm optimization in dynamic environments. *Expert Systems with Applications*, 38(3):2212–2223, 2011.

- S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220 (4598):671–680, 1983.
- P. Korošec & J. Šilc. The differential ant-stigmergy algorithm applied to dynamic optimization problems. In *Proceedings of the Congress on Evolutionary Computation CEC'09*, pages 407–414. IEEE Press, 2009.
- J.R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT press, 1992.
- A.H. Land & A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.
- P. Larrañaga. A Review of Estimation of Distribution Algorithms. In P. Larrañaga, J.A. Lozano, editor, *Estimation of Distribution Algorithms: A new tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
- P. Larrañaga, R. Etxeberria, J.A. Lozano & J.M. Pena. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAAIK-4, University of the Basque Country, 1999.
- P. Larrañaga, J.A. Lozano & E. Bengoetxea. Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Technical Report KZZA-IK-1-01, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 2001.
- C. Li & S. Yang. A generalized approach to construct benchmark problems for dynamic optimization. In *Proc. of the 7th Int. Conf. on Simulated Evolution and Learning*, 2008.
- C. Li & S. Yang. A clustering particle swarm optimizer for dynamic optimization. In *Proceedings of the Congress on Evolutionary Computation CEC'09*, pages 439–446. IEEE Press, 2009.
- C. Li, S. Yang, T.T. Nguyen, E.L. Yu, X. Yao, Y. Jin, H.G. Beyer & P.N. Suganthan. Benchmark Generator for CEC'2009 Competition on Dynamic Optimization. Technical report, University of Leicester, 2008.
- H. Li, Y. Hong & S. Kwong. Subspace estimation of distribution algorithms: To perturb part of all variables in estimation of distribution algorithms. *Applied Soft Computing*, 11(3):2974 – 2989, 2011.

- X. Li, J. Branke & T. Blackwell. Particle swarm with speciation and adaptation in dynamic environments. In *Proceedings of the 8th Genetic and Evolutionary Computation Conference GECCO'06*, pages 51–58. ACM, 2006.
- X. Liu, Y. Wu & J. Ye. An Improved Estimation of Distribution Algorithm in Dynamic Environments. In *Fourth International Conference on Natural Computation*, pages 269–272. IEEE Computer Society, 2008.
- S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.
- J.A. Lozano. *Towards a new evolutionary computation: advances in the estimation of distribution algorithms*. Springer-Verlag New York Inc, 2006.
- Q. Lu & X. Yao. Clustering and learning gaussian distribution for continuous optimization. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(2):195–204, 2005.
- R.I. Lung & D. Dumitrescu. A new collaborative evolutionary-swarm optimization technique. In *Proceedings of the 9th Genetic and Evolutionary Computation Conference GECCO'07*, pages 2817–2820, New York, NY, USA, 2007.
- R.I. Lung & D. Dumitrescu. Evolutionary swarm cooperative optimization in dynamic environments. *Natural Computing*, 9:83–94, 2010.
- C.D. Maranas & C.A. Floudas. A deterministic global optimization approach for molecular structure determination. *J. Chem. Phys.*, 10(2), 1994a.
- C.D. Maranas & C.A. Floudas. Global minimum potential energy conformations for small molecules. *J. Glob. Opt.*, 4:135–170, 1994b.
- M. Markatou. Mixture models, robustness, and the weighted likelihood methodology. *Biometrics*, 56(2):pp. 483–486, 2000.
- G.J. McLachlan & D. Peel. *Finite mixture models*. Wiley-Interscience, 2000.
- R. Mendes & A. Mohais. Dynde: a differential evolution for dynamic optimization problems. In *Proceedings of the Congress on Evolutionary Computation CEC'05*, pages 2808–2815. IEEE Press, 2005.

- R. Mendes, J. Kennedy & J. Neves. The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, 8(3):204–210, June 2004.
- Z. Michalewicz. *Genetics Algorithms + Data Structures = Evolution Programs*. Springer-Verlag Berlin Hildelberg, New York, 3rd edition, 1996.
- D.S. Moore, G.P. McCabe & B.A. Craig. *Introduction to the Practice of Statistics*. WH Freeman, 6a. edition, 2009.
- I. Moser & T. Hendtlass. A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In *Proceedings of the Congress on Evolutionary Computation CEC'07*, pages 252–259. IEEE Press, 2007.
- H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.
- H. Mühlenbein & G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature-PPSN IV*, pages 178–187, 1996.
- R. Neal & G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.
- J. Nocedal & S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 2nd edition, July 1999.
- S.J. Nowlan. *Soft competitive adaptation: neural network learning algorithms based on fitting statistical mixtures*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1991.
- P. Nurmi. Mixture models, 2008. Helsinki Institute for Information Technology. CiteSeerX - Scientific Literature Digital Library and Search Engine (United States).
- E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.
- M. Pelikan & H. Mühlenbein. The bivariate marginal distribution algorithm. *Advances in Soft Computing-Engineering Design and Manufacturing*, pages 521–535, 1999.
- M. Pelikan, D.E. Goldberg & E. Cantú-Paz. Linkage problem, distribution estimation and Bayesian networks. *Evolutionary Computation*, 8(3):311–340, 2000a.

- M. Pelikan, D.E. Goldberg & E. Cantú-Paz. Bayesian optimization algorithm, population sizing, and time to convergence. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'00*, pages 275–282, 2000b.
- M. Pelikan, D.E. Goldberg & F.G. Lobo. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, 21(1):5–20, 2002.
- J.M. Pena, J.A. Lozano & P. Larrañaga. Benefits of data clustering in multimodal function optimization via EDAs. In P. Larrañaga, J.A. Lozano, editor, *Estimation of Distribution Algorithms: A new tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
- L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze & E. Mishchenko. *The mathematical theory of optimal processes*. Interscience Publishers, 1962.
- W.H. Press, S.A. Teukolsky, W.T. Vetterling & B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, September 2007.
- C.E. Priebe. Adaptive Mixtures. *Journal of the American Statistical Association*, 89(427):796–806, 1994.
- I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.
- J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- M. Rocha, J. Neves, A.C.A. Veloso, E.C. Ferreira & I. Rocha. *Adaptive and Natural Computing Algorithms*, chapter Evolutionary Algorithms for Static and Dynamic Optimization of Fed-batch Fermentation Processes, pages 288–291. Springer Vienna, 2005.
- M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.
- J.A. Roubos, G. van Straten & A.J.B. van Boxtel. An evolutionary strategy for fed-batch bioreactor optimization; concepts and performance. *Journal of Biotechnology*, 67(2-3):173 – 187, 1999.
- M. Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9(2):65–78, 1982.
- S. Rudlof & M. Köppen. Stochastic hill climbing with learning by vectors of normal distributions. In *Proceedings of the American Association for Artificial Intelligence*, Nagoya, Japan, 1996.

- R. Salomon. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions; a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.
- R. Santana, P. Larrañaga & J.A. Lozano. Learning factorizations in estimation of distribution algorithms using affinity propagation. *Evolutionary Computation*, 18(4):515–546, 2010.
- M-A. Sato & S. Ishii. On-line EM Algorithm for the Normalized Gaussian Network. *Neural Comp.*, 12(2):407–432, February 2000.
- J. Schwarz & J. Očenášek. Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and. In *Proceedings of the Fifth International Conference on Soft Computing*, pages 124–130, 1999.
- H.P. Schwefel. *Numerical optimization of computer models*. John Wiley & Sons, Inc. New York, NY, USA, 1981.
- D.W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, December 1979.
- D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley, September 1992.
- M. Sebag & A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature-PPSN V*, page 418. Springer, 1998.
- S. Shakya & J. McCall. Optimization by estimation of distribution with DEUM framework based on Markov random fields. *International Journal of Automation and Computing*, 4:262–272, 2007.
- S.S. Shapiro & M.B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.
- H. Shimazaki & S. Shinomoto. A Method for Selecting the Bin Size of a Time Histogram. *Neural Computation*, 19(6):1503–1527, June 2007.
- B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1st edition, April 1986.
- B. Srinivasan, S. Palanki & D. Bonvin. Dynamic optimization of batch processes: I. Characterization of the nominal solution. *Computers & Chemical Engineering*, 27(1):1–26, 2003.
- R. Storn & K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, December 1997.

- G.R. Terrell & D.W. Scott. Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265, September 1992.
- R. Tinós & S. Yang. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, 8(3):255–286, 2007.
- V. Van Breusegem & G. Bastin. Optimal control of biomass growth in a mixed culture. *Biotechnology and Bioengineering*, 35(4):349–355, 1990.
- S. Voß. Meta-heuristics: The state of the art. In Alexander Nareyek, editor, *Local Search for Planning and Scheduling*, volume 2148 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin / Heidelberg, 2001.
- S. Voß, I.H. Osman & C. Roucairol. *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Kluwer Academic Publishers Norwell, MA, USA, 1999.
- J. Whittaker. *Graphical models in applied multivariate statistics*, page 167. Wiley Publishing, 1990.
- D. Wierstra, T. Schaul, J. Peters & J. Schmidhuber. Fitness expectation maximization. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni & N. Beume, editors, *Parallel Problem Solving from Nature - PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 337–346. Springer Berlin / Heidelberg, 2008.
- S. Yang & X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comp.*, 9(0):815–834, 2005.
- S. Yang & X. Yao. Population-based incremental learning with associative memory for dynamic environments. *Evolutionary Computation, IEEE Transactions on*, 12(5):542–561, 2008.
- E. L. Yu & P. N. Suganthan. Evolutionary programming with ensemble of explicit memories for dynamic optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 431–438. IEEE Press, 2009.
- B. Yuan, M. Orłowska & S. Sadiq. Extending a class of continuous estimation of distribution algorithms to dynamic problems. *Optimization Letters*, 2(3):433–443, 2008.

