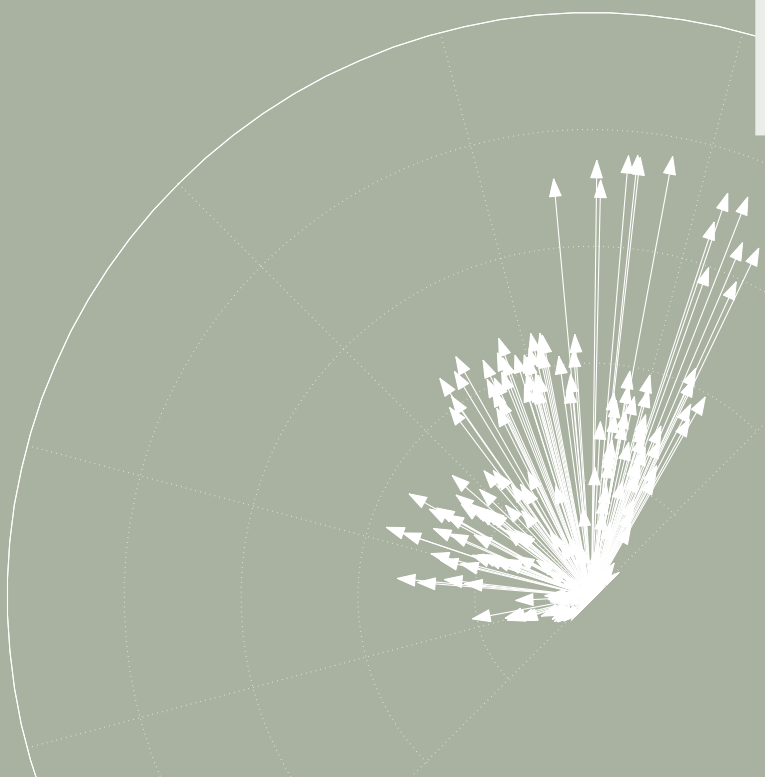
MANUAL DE COMPUTAÇÃO EVOLUTIVA E METAHEURÍSTICA

ANTÓNIO GASPAR-CUNHA
RICARDO TAKAHASHI
CARLOS HENGGELER ANTUNES
COORDENADORES



IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDITORA ufmg

CAPÍTULO 15

Otimização em Ambientes Incertos

Fabrício O. de França [*] *Fernando J. Von Zuben* [**]

[*] *Centro de Matemática, Computação e Cognição*
Universidade Federal do ABC

[**] *Departamento de Engenharia de Computação e Automação Industrial*
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

A incerteza de um parâmetro, por definição, indica a faixa de dispersão de valores aceitáveis que podem ser atribuídos ao parâmetro. Logo, a presença de incerteza em processos de tomada de decisão implica na necessidade de agir sob variação dinâmica, conhecimento aproximado ou incompleto, e ausência de controle sob certas variáveis que influenciam na decisão. Este capítulo trata de algoritmos computacionais para otimização iterativa em ambientes incertos, de modo que a incerteza do ambiente implica na imprevisibilidade do resultado da otimização, caso os efeitos das incertezas sejam negligenciados. O resultado se torna imprevisível no sentido de que, quando um mesmo algoritmo é aplicado diversas vezes a um dado problema de otimização com incertezas, a qualidade da solução encontrada em cada caso pode ser muito mais variável do que ocorre naturalmente em algoritmos estocásticos. Um fator agravante aqui é que todo problema real de otimização e de interesse prático vai expressar algum grau de incerteza. Sendo assim, o propósito deste capítulo é estudar algoritmos voltados para otimização e que apresentem uma maior robustez na presença de incertezas, sendo que a computação evolutiva fornece soluções bastante competitivas para este fim. Ao longo deste capítulo, será mostrado

como identificar as incertezas em processos de otimização e como a computação evolutiva pode ser empregada para tratar desse tipo de problema.

1. Introdução

Ao longo dos demais capítulos desta obra, foi feito um esforço coordenado visando mostrar como, quando e onde aplicar computação evolutiva e também outras meta-heurísticas voltadas para a solução de problemas de interesse prático, que desafiam profissionais das mais diversas áreas. Imagina-se que o leitor deve ter se surpreendido com a diversidade de cenários de aplicação e com a flexibilidade dos algoritmos evolutivos e seus derivados e afins. No entanto, este capítulo vem mostrar que ainda há espaço para novos cenários, embora voltados apenas para processos de otimização, e apresenta desafios adicionais para esta flexibilidade dos algoritmos evolutivos.

São tratados aqui problemas de otimização em ambientes incertos, os quais são formalizados matematicamente de acordo com a origem da incerteza. As incertezas podem ter diversas origens: (*i*) ausência de acuidade em sensores de medida; (*ii*) variação temporal de parâmetros e de objetivos; e (*iii*) erros, simplificações e aproximações no processo de modelagem matemática do problema de otimização, incluindo aqui a própria definição da função-objetivo.

Negligenciar essas incertezas pode ser desastroso em otimização de processos, pois decisões equivocadas podem ser tomadas, causadas pela diferença entre a situação idealizada que guia a tomada de decisão e a situação real do processo. Com isso, os efeitos cumulativos desses potenciais equívocos, na qualidade do resultado final, tornam-se imprevisíveis.

E os algoritmos de otimização, os de computação evolutiva inclusos, mesmo em versões mais elaboradas, normalmente não se mostram capazes de lidar de forma apropriada com incertezas durante o processo de otimização. Visando maior robustez de desempenho, é necessário monitorar o ambiente, propor operadores mais robustos às incertezas e empregar recursos computacionais adicionais visando minorar os efeitos potencialmente danosos das incertezas sobre o desempenho dos algoritmos de otimização, particularmente dos algoritmos evolutivos. Em linhas gerais, a ideia é fazer o melhor uso possível da informação disponível, ciente de que o que se dispõe de informação pode não ser suficiente para auxiliar na escolha da melhor decisão a cada passo.

Portanto, este capítulo trata de ferramentas capazes de tornar os algoritmos evolutivos mais efetivos no tratamento de otimização em ambientes incertos. Para tanto, optou-se por dividir o capítulo em quatro seções principais, o que se justifica pelo fato de que os problemas incertos são geralmente classificados em quatro tipos: *problemas ruidosos*, em que há incerteza na medição do valor da função-objetivo; *problemas robustos*, em que as decisões são tomadas admitindo-se uma faixa de tolerância nos valores do vetor-solução; *problemas de aproximação de função*, quando não existem meios de se definir precisamente uma função-objetivo, ou o cálculo dela é muito custoso; *problemas variantes no tempo*, quando a superfície de otimização muda sua conformação em função do tempo, provocando deslocamentos dos pontos de ótimo. Mas antes dessas quatro seções principais, existe uma seção dedicada a descrever em linhas gerais problemas relevantes que conduzem a processos de otimização em ambientes incertos, e que são bons candidatos a serem abordados via computação evolutiva.

2. A incerteza presente em problemas reais de otimização

Em lugar de representar uma exceção, a presença de algum grau de incerteza em qualquer atividade humana é regra. Fazem parte desta regra os problemas de otimização, com incertezas que vão desde a definição do que se quer ver presente na solução até no que vai suportar a decisão a cada passo de um processo iterativo de solução.

Já foi discutido ao longo dos demais capítulos desta obra que, dentre todos os problemas que

existem para serem resolvidos pela humanidade, parte deles é computável, ou seja, admite solução empregando recursos computacionais. Será tratada aqui a computação digital e não serão consideradas outras formas de computação. Dentre os problemas computáveis, parte deles é de otimização. E dentre os problemas de otimização, parte deles é intratável computacionalmente, no sentido de que a garantia de obtenção da solução ótima via algoritmos exatos de resolução requer recursos computacionais, mais especificamente tempo de processamento e memória, além do razoável. Há problemas computáveis de otimização, presentes no dia-a-dia das pessoas, que requerem bilhões de anos de processamento computacional para serem resolvidos por algoritmos exatos (que garantem obter a solução ótima), isso empregando os computadores digitais mais poderosos já concebidos. Certamente trata-se de um tempo de processamento além do razoável.

E o que já foi visto também ao longo dos demais capítulos desta obra é que as meta-heurísticas representam alternativas de solução para esses tipos de problemas de otimização computáveis, mas intratáveis. Elas são capazes de produzir propostas de solução empregando recursos computacionais razoáveis, mas ao preço de não mais garantir três condições ao longo da otimização: (*i*) não garantem obter a solução ótima; (*ii*) não permitem antecipar o custo até a convergência; e (*iii*) não garantem sequer a convergência. Na ausência de alternativas mais poderosas, as meta-heurísticas reinam soberanas, apesar deste preço a pagar. E dentre as meta-heurísticas, encontram-se todas as metodologias apresentadas nesta obra, ao longo de seus capítulos, incluindo obviamente as técnicas de computação evolutiva.

A computação evolutiva, então, é uma meta-heurística que deve ser empregada junto a certos problemas de otimização que não admitem solução a partir de algoritmos exatos, mesmo que estes algoritmos exatos existam. Este é o cenário de onde se parte neste capítulo.

Repare que não se falou ainda, nos parágrafos iniciais desta seção, de incerteza. É hora de afirmar então que a presença de incerteza amplia ainda mais os desafios do processo de otimização junto a esses problemas computáveis, mas intratáveis. Os desafios para se chegar a uma solução já eram significativos, e com incertezas ficam ainda mais expressivos.

Mas será que problemas de otimização em ambientes incertos, com as características levantadas acima, são comuns no dia-a-dia das pessoas e das instituições? Nos parágrafos que se seguem, será apresentada uma resposta afirmativa para esta questão, a partir de exemplos descritos de forma genérica (não-matemática).

Na formulação de problemas de otimização que envolvem diversas entidades e condições a serem atendidas, como no caso de escalonamento de recursos ou tarefas, é sabido que muitos esforços devem ser investidos na definição do modelo matemático que descreve o problema. Por vezes, devido à complexidade dos processos reais envolvidos, algumas simplificações e aproximações são adotadas. Em outras circunstâncias, aspectos relevantes do processo podem ser negligenciados involuntariamente ao longo da formulação. Quando o modelo matemático não consegue retratar aspectos decisivos da realidade do processo que se pretende modelar, é possível que se obtenha a solução certa para o problema errado. Essas simplificações, aproximações e erros de modelagem podem, por sua vez, ser interpretados como fontes de incertezas, a serem devidamente incorporadas à formulação matemática do problema de otimização. Um exemplo aqui pode ser a decisão de onde posicionar, considerando todo o Território Nacional ou dentro de algum Estado da Federação, uma nova unidade de uma grande rede de hipermercados, visando maximizar o retorno do investimento no menor tempo possível. Como considerar matematicamente todos os principais aspectos de cada localidade candidata, visando estimar adequadamente o retorno de investimento? Outro exemplo vinculado ao anterior seria como definir a capacidade de atendimento deste hipermercado (tamanho das instalações, número de funcionários, etc.). Como estimar apropriadamente a demanda esperada de cada localidade candidata?

Em outros problemas de otimização, tanto os objetivos quanto os parâmetros do processo podem variar ao longo do tempo, durante a etapa de resolução. Um exemplo aqui é o roteamento de veículos em grandes cidades, visando minimizar o tempo de percurso, sendo que as ruas e avenidas podem estar

sujeitas a níveis diferentes de congestionamento ao longo do trajeto dos veículos. Os mecanismos de otimização devem então ser adaptados às diversas fontes de variação que ocorrem durante a operação, que também caracterizam ambientes incertos.

Custos e ausência de recursos tecnológicos também podem impactar na ocorrência de incerteza em ambientes de otimização. É muito comum a necessidade de tomar decisões ignorando-se o comportamento de variáveis que sabidamente influenciam no andamento do processo, mas que não puderam ser monitoradas pelo custo dos sensores ou por ainda não existir tecnologia disponível para o projeto e implementação desses sensores. Custos e ausência de recursos tecnológicos também podem impactar na qualidade da atuação, de modo que mesmo que a decisão ótima seja conhecida a atuação não vai corresponder exatamente à decisão ótima que deveria ser tomada. Como exemplo para ambos os problemas (sensoriamento e atuação), pode-se citar toda e qualquer empresa prestadora de serviço e sua interação com os clientes. Como atender otimamente à demanda dos clientes e como perceber (sensoriar) qual é essa demanda? Como atrair novos clientes tendo em vista que a melhor maneira de atraí-los não é completamente viável? Qual o impacto de fazer isso de forma simplificada?

Por último, cabe mencionar a dificuldade de avaliação das soluções candidatas em problemas de otimização. Os algoritmos evolutivos e muitas outras meta-heurísticas requerem a atribuição de graus de qualidade às soluções, e isso pode ser muito custoso, em certos casos até impedindo uma avaliação precisa. Tem-se, assim, uma fonte importante de incertezas. Como exemplo, pode-se citar o processo de avaliação de potencial de novos produtos a serem lançados no mercado. Em qual ordem os produtos devem ser lançados de forma a causar maior impacto nos consumidores, dado um leque de opções? Pode-se até supor que o custo de lançamento de cada novo produto já seja conhecido, ou então esta passa a ser outra fonte de incerteza que também impacta na avaliação do potencial do produto.

Espera-se que os exemplos apresentados até aqui já permitam ao leitor identificar o quão comuns e relevantes são os problemas de otimização em ambientes incertos que precisam ser resolvidos no dia-a-dia das pessoas e das instituições. No entanto, para concluir esta seção serão contrastados dois problemas de otimização em ambientes incertos: um tratável e o outro intratável computacionalmente (já exemplificado acima, inclusive).

Considere o uso simples de um Navegador GPS, onde o usuário quer apenas traçar a menor rota entre sua residência e seu trabalho. Para isso, o sistema encontra a solução do problema do caminho mínimo, da Teoria de Grafos, para determinar a melhor rota entre dois pontos. Esse é um problema bem conhecido e para o qual existem diversas soluções de complexidade polinomial para resolvê-lo. Logo, é possível obter a solução de ótimo global em tempo razoável. Imaginem agora que, ao seguir o caminho indicado pelo GPS, que é a solução ótima, o usuário acaba parando em um congestionamento. Esse congestionamento pode ser encarado como um agente externo que mudou parâmetros do problema de otimização. A rota atual não é mais o ótimo global e, portanto, é preciso encontrar outra solução. Para esse exemplo, o procedimento é simples: basta atribuir um custo alto para os trechos que contêm congestionamento e pedir para o Navegador GPS encontrar uma nova solução.

Agora, considere um exemplo um pouco diferente: imagine que o usuário do Navegador GPS é uma empresa de distribuição de mercadorias. Essa empresa contém ao seu dispor diversos caminhões-baús para transportar e entregar lotes de mercadorias em n pontos de entrega. Cada caminhão, equipado com um navegador GPS, agora não quer apenas saber a menor rota entre os diversos pontos de entrega, considerados par-a-par. O objetivo a ser cumprido aqui é encontrar a sequência de n pontos de entrega que minimize o caminho total percorrido ou o tempo de entrega total. Esse problema é conhecido como o problema do caixeiro viajante (PCV) e, ao contrário do problema do caminho mínimo, não existe um algoritmo exato que retorne o ótimo global em um tempo razoável, isso quando n é grande. Esse é um problema em que o recomendado é o uso de meta-heurísticas que, embora não garantam produzir a solução ótima, fornecem uma solução de boa qualidade e que pode ser obtida antes dos caminhões partirem para as entregas. Se, no entanto, ao executar uma rota determinada pelo sistema o caminhão enfrentar o mesmo problema de congestionamento do exemplo anterior, resulta um novo

problema que já não é tão simples de se resolver. De fato, com essa perturbação de parâmetros do problema, não só a rota atual deixa de ser ótima mas, possivelmente, a sequência de pontos de entrega pode se alterar. Dado o custo para se resolver esse problema, não é sempre viável mandar o Navegador GPS recalcular a melhor solução. Este tipo de situação será tratado ao longo das próximas seções deste capítulo.

Nas próximas seções, quatro tipos de ambientes incertos em problemas de otimização serão explorados individualmente e de forma mais detalhada, com exemplos de casos reais. Será dada ênfase aos procedimentos geralmente adotados para adaptar algoritmos de computação evolutiva a cada contexto, buscando evitar degradação de desempenho no tratamento dos problemas de otimização resultantes. Adicionalmente, recomendações de leituras serão dadas em cada seção, visando complementar esta abordagem introdutória.

3. Problemas Ruidosos

Problemas ruidosos são aqueles em que existe uma incerteza no valor da função-objetivo durante a medição. Isso pode ocorrer, por exemplo, devido à ausência de acuidade nos sensores, à interferência de fatores externos e à presença de propriedades estocásticas do próprio problema de otimização. Usualmente, a função-objetivo para esse tipo de problema é representada conforme a Eq. 15.1:

$$F(x) = f(x) + \delta, \tag{15.1}$$

onde a função-objetivo original é acrescida de uma perturbação, geralmente de natureza gaussiana. Pode-se interpretar, então, $f(x)$ como a função-objetivo ideal, $F(x)$ como a função-objetivo medida e δ como uma distribuição normal com média zero e variância σ^2 definida experimentalmente.

Exemplos Ilustrativos

Para ilustrar esse tipo de ambiente incerto, pode-se considerar o problema de escalonamento de processos, muito comum em Ciência da Computação, o qual deve ser resolvido por sistemas operacionais que gerenciam ambientes multiprocessados. Esse problema consiste em, dados P processos a serem executados por M processadores de igual eficiência (tratar processadores homogêneos representa uma simplificação do problema), é requisitado que esses processos sejam distribuídos e enfileirados de forma a minimizar o tempo total de execução. Nesse problema, cada processo tem um dado tempo de execução nos processadores, conhecido a priori. Embora esse tempo de execução seja considerado determinístico, na prática ele pode apresentar graus de incerteza, sofrendo pequenas flutuações em torno do tempo esperado. É fácil constatar que cada perturbação causa uma diferença pequena no valor da função-objetivo, mas que, em conjunto, podem ser suficientes para impedir que a proposta de escalonamento considerada ótima (solução ideal) não corresponda à melhor solução possível para o problema real.

Outro exemplo pertinente é a otimização de parâmetros em algoritmos de Computação Evolutiva (Lobo et al., 2007). Nesses algoritmos, conforme o leitor deve ter observado em capítulos anteriores, existem diversos parâmetros que devem ser ajustados visando bom desempenho em aplicações práticas, com o agravante de que o melhor ajuste para uma certa aplicação pode ser bem diferente do melhor ajuste para uma outra aplicação do mesmo algoritmo. Além dos diversos parâmetros, existe uma grande diversidade de operadores de mutação e recombinação, os quais inclusive podem ser aplicados em conjunto, em diversas configurações e a diferentes taxas. Diante disso, fica caracterizado o problema da escolha da combinação ótima de parâmetros e operadores que conduzam ao melhor resultado em cada aplicação de um algoritmo evolutivo. Devido ao número de parâmetros que alguns algoritmos têm, sendo alguns deles com valores contínuos, e a diversidade de operadores, esse processo de escolha se configura como uma tarefa complicada de se resolver. Uma proposta para tornar esta tarefa automática

é a incorporação da própria combinação de parâmetros e operadores ao problema de otimização. Logo, como os algoritmos evolutivos são estocásticos por natureza, a avaliação das propostas de combinação de parâmetros e operadores torna-se incerta, em algum grau. O efeito é um ruído na função-objetivo do problema, visto que ela incorpora a avaliação de parâmetros do próprio algoritmo de otimização.

Nas próximas seções, o leitor poderá constatar que esses dois exemplos podem também ser interpretados como fontes de outros tipos de incerteza junto ao processo de otimização.

Reduzindo o Ruído

Média Explícita

Lidar com ruído na função-objetivo de algoritmos evolutivos pode ser relativamente simples, dadas algumas características dessa meta-heurística. A maneira mais direta para se aproximar do valor real da função-objetivo é a chamada *média explícita*, que consiste em calcular várias amostras do valor da função-objetivo em um dado ponto e, em seguida, extrair a média desse valor, conforme ilustrado na Eq. 15.2:

$$\hat{F}(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{15.2}$$

onde $f_i(x)$ é o i-ésimo valor medido da função-objetivo, $\hat{F}(x)$ é a média amostral, ou seja, o valor estimado para a função-objetivo, e N é o número de amostras utilizadas. Conceitualmente, a existência de N amostras da função-objetivo num dado ponto leva à redução do desvio-padrão da média amostral correspondente por um fator de $\sqrt{N}$. Em outras palavras, o valor médio da função-objetivo torna-se mais confiável naquele ponto. Uma vez que tende a ocorrer um aumento da qualidade da avaliação de função ao aumentar o número de amostras, intuitivamente aumentar o número N de amostras, com a consequente redução do desvio-padrão da média, pode ser o caminho a seguir.

Mas, voltando ao exemplo de escalonamento citado acima, imagine que o problema é de larga escala, com milhões de processos a serem executados por milhares de processadores. O custo para computar a função-objetivo uma única vez já é alto e, como se trata de um algoritmo populacional, esse custo aumenta proporcionalmente ao tamanho da população (número de soluções candidatas). Assim, os custos inerentes dessa classe de algoritmos podem se ampliar ainda mais com a reamostragem da função-objetivo, trazendo problemas de escalabilidade aos algoritmos evolutivos.

Uma primeira alternativa, capaz de reduzir custos, seria calcular a média durante as iterações, ou seja, a cada iteração é feito um novo recálculo da função-objetivo para indivíduos remanescentes de gerações anteriores.

Em alguns artigos na literatura, encontram-se outras estratégias para se ter um número de amostras suficientes sem aumentar consideravelmente o custo do algoritmo. A solução mais simples, proposta por Aizawa e Wah (1994, 1993), consiste em definir o tamanho da amostragem adaptativamente durante a execução do algoritmo. Para tanto, eles adotaram duas estratégias: aumentar o número de amostras a cada geração ou usar um número de amostras maior para as soluções candidatas (indivíduos da população) com maior desvio amostral.

Uma outra maneira foi explorada em (Stagge, 1998), em que a média do valor da função-objetivo é calculada apenas quando dois indivíduos, i e j, são comparados entre si durante o processo de seleção. Como a avaliação dos dois indivíduos sofre o efeito de ruído, a comparação relativa entre ambos deve se dar por meio de testes estatísticos. Considerando a maximização da função-objetivo, o autor trabalhou com a hipótese nula

$$H_0 : \hat{F}(x_i) - \hat{F}(x_j) \geq 0, \tag{15.3}$$

que é a hipótese do indivíduo i ser melhor que o indivíduo j e que pode ser testada contra a hipótese alternativa

$$H_1 : \hat{F}(x_i) - \hat{F}(x_j) < 0. \tag{15.4}$$

Definidas as hipóteses, e dado um valor N_0 inicial de amostras (avaliações da função-objetivo), é realizado o teste-t de hipóteses (Student, 1908) que indica se a hipótese H_0 pode ser rejeitada com um erro probabilístico $\leq \alpha \in [0, 1]$, ou alternativamente, com grau de confiança $p = 1 - \alpha$. Se esse grau de confiança for baixo, então mais avaliações serão necessárias para i, j ou ambos. As amostragens continuam a ser realizadas até que o grau de confiança seja maior ou igual ao limiar ou um número N_{max} de amostragens seja atingido. Dessa forma, é possível deixar o número de avaliações por indivíduo no mínimo necessário para termos um grau de confiança acima de um limiar acerca da aceitação ou rejeição da hipótese. Outros testes de hipóteses podem ser adotados em substituição ao teste-t, dependendo do comportamento estatístico do ruído detectado no problema.

Uma estratégia parecida com essa foi adotada por Branke e Schmidt (2003, 2004); Cantú-Paz (2004), os quais fizeram o uso da técnica de amostragem sequencial. Esta técnica consiste em aumentar incrementalmente o número de amostras até que seja possível tomar uma decisão baseada na diferença de valores médios e na variância entre os dois conjuntos de amostras. Um exemplo é a amostragem sequencial baseada no valor observado da função-objetivo. Nesta técnica, a diferença entre $\hat{F}(x_i)$ e $\hat{F}(x_j)$ é calculada conforme a Eq. 15.5:

$$d^*_{ij} = \frac{\hat{F}(x_i) - \hat{F}(x_j)}{\sqrt{s_i^2 + s_j^2}}, \tag{15.5}$$

onde s_k é o desvio-padrão observado para o indivíduo k. Este procedimento está descrito na forma de pseudocódigo no Algoritmo 1.

Algoritmo 1 Pseudocódigo para o método de amostragem sequencial

```
Gere N_0 amostras para cada indivíduo (i e j)
Determine a diferença d*_ij entre os indivíduos i e j, conforme Eq. 15.5.
para k = 1 até K − 1 faça
    se |d*_ij| < ε_k então
        Retorne o melhor indivíduo
    fim se
    Gere uma amostra adicional para cada indivíduo
    Atualize d*_ij
fim para
Gere N_k amostras adicionais para cada indivíduo
Retorne o melhor indivíduo
```

O algoritmo começa gerando a amostragem inicial para cada indivíduo, calculando em seguida a diferença entre valores dessas duas amostragens. A partir daí, o algoritmo entra em um ciclo em que é verificado se a diferença é menor que um dado ϵ_k e, em caso afirmativo, retorna o melhor entre os dois indivíduos. Caso contrário, gera-se uma nova amostra para cada um. Após completarem um total de K amostras, caso a diferença ainda não seja suficiente para afirmar qual dos dois indivíduos é o melhor, são geradas mais N_k amostras e, então, o melhor indivíduo é retornado, mesmo sem informações suficientes. Pode-se verificar que, no pior caso, o algoritmo gera $K + N_k$ amostras para cada indivíduo.

Em lugar de realizar diversas amostragens da função-objetivo em um único ponto, uma proposta diferente é procurar estimar o valor da função-objetivo em um ponto a partir do valor obtido para ela

em outros pontos na vizinhança. Passa-se então de uma abordagem de média no tempo para outra de média no espaço, através de um critério de máxima verossimilhança (Kay, 1993). Neste caso, leva-se em conta um histórico de pontos previamente amostrados e sua distância em relação ao ponto a ser estimado. Essa técnica foi aplicada em (Sano e Kita, 2000; Sano et al., 2000; Sano e Kita, 2002) e tem como pré-requisito que o ruído tenha as mesmas características na vizinhança do ponto sendo avaliado e que a função-objetivo seja localmente suave.

Já Jaskowski e Kotlowski (2008) idealizaram 4 métodos diferentes de média explícita para selecionar o melhor indivíduo da população, dado um limite k máximo de amostras. O primeiro deles, chamado ingênuo (do inglês *naïve procedure*), distribui as k amostras igualmente para cada indivíduo da população, ou seja, se existem n indivíduos, cada um terá $\frac{k}{n}$ amostras para o cálculo da média. Após as amostragens serem feitas, o indivíduo que tem a média com maior valor (no caso de maximização) é retornado.

O segundo método, um pouco mais elaborado, é chamado de torneio (do inglês *tournament procedure*), e começa de maneira parecida com o procedimento anterior, amostrando uma quantidade igual para cada indivíduo. Só que, dessa vez, o número de amostras é metade do caso anterior, $\frac{k}{2n}$. Após a amostragem, os $n/2$ melhores indivíduos são selecionados para a fase seguinte, em que eles ganharão novas amostragens e o processo é repetido até que sobre apenas um único indivíduo. A ideia por trás desse método é que os melhores indivíduos tenham um número maior de amostras do que os piores indivíduos, já que estes últimos, mesmo com o efeito do ruído, não poderiam atingir o desempenho dos melhores indivíduos.

O terceiro método é chamado de seleção de candidatos (do inglês *candidate selection*) e consiste em amostrar os indivíduos com menor grau de confiança no teste de hipóteses, em comparação com o melhor indivíduo atual. O seu pseudocódigo está descrito em maiores detalhes no Algoritmo 2.

Algoritmo 2 Pseudocódigo para o método de seleção de candidatos

```
Gere 1 amostra para cada indivíduo
k ← k − n
enquanto k > 0 faça
    a_max ← elemento com maior valor médio
    a_prox ← elemento com segundo maior valor médio
    a_min ← elemento que minimiza o grau de confiança em relação a a_max
    se conf(a_min, a_max) < conf(a_max, a_prox) então
        Gere nova amostra de a_min
    senão
        Gere nova amostra de a_max
    fim se
    k ← k − 1
fim enquanto
Retorne o melhor indivíduo
```

Inicialmente, como já mencionado, é gerada uma amostra para cada indivíduo, restando apenas $k - n$ amostras a serem distribuídas. Em seguida, o algoritmo entra em seu laço principal onde escolhe três indivíduos: aquele com o maior valor médio (a_{max}, melhor candidato), aquele com o segundo maior valor médio (a_{prox}, concorrente direto) e aquele que tem um menor grau de confiança no teste de hipóteses em relação ao melhor indivíduo (a_{min}). Caso o grau de confiança entre esse último e o melhor indivíduo seja menor do que o grau de confiança entre o melhor e o segundo melhor indivíduo, uma nova amostra é gerada para a_{min}. Caso contrário, uma nova amostra é gerada para a_{max}. O processo se repete até que o número máximo k de amostragens seja alcançado, quando então é retornado o indivíduo com melhor média. A ideia por trás desse procedimento é distribuir as amostras

de tal forma a maximizar o grau de confiança de que o indivíduo a_{max} é o melhor da população. Para facilitar os cálculos, o grau de confiança é aproximado seguindo a Eq. 15.6:

$$conf(x, y) = (media(x) - media(y))^2 \cdot N_x, \tag{15.6}$$

onde N_x é o número de amostras de x.

Finalmente, o último procedimento, denominado bayesiano de um estágio à frente (do inglês *Bayesian one-stage ahead procedure*) e baseado numa técnica de amostragem denominada *Bayesian look ahead one-stage sampling*, requer o conhecimento dos parâmetros reais do ruído (média e variância) e de propriedades estatísticas da função-objetivo no espaço de busca. Em (Jaskowski e Kotlowski, 2008), esse algoritmo foi utilizado apenas como base de comparação com os outros três métodos. Ele dificilmente tem utilidade prática, uma vez que geralmente não se conhecem os parâmetros e propriedades requeridos. Com isso, e dada a complexidade desse procedimento, que foge do escopo introdutório desse texto, os autores remetem o leitor para a literatura específica.

Outras Abordagens

Uma alternativa para a média explícita, abordada na seção anterior, é o uso da *média implícita*, a qual consiste simplesmente em aumentar o tamanho da população. Em algoritmos populacionais, as áreas mais promissoras são mais exploradas e, nessas áreas, muitos indivíduos similares são gerados. Quando tem-se uma população muito grande, o efeito do ruído em um indivíduo é compensado pela avaliação da função-objetivo de todos os indivíduos na mesma região. Como consequência disso, em (Miller et al., 1996) foi demonstrado que, para uma população infinita, a seleção proporcional não é afetada pelo ruído.

Surge então a questão de qual abordagem é a melhor: aumentar o número de amostras por indivíduo (média explícita) ou aumentar o tamanho da população (média implícita). As investigações feitas até o momento mostram que cada técnica é melhor sob determinadas condições. Por exemplo, a média explícita mostrou-se melhor nos casos de uma estratégia evolutiva $(1, \lambda)$ (Beyer, 1993). Já a média implícita teve vantagens ao ser aplicada em uma determinada versão de algoritmo genético (Fitzpatrick e Grefenstette, 1988). Em (Miller et al., 1996; Miller, 1997) foram determinadas metodologias para calcular o valor ideal do tamanho da população e número de amostras para reduzir otimamente o efeito do ruído. Para tanto, Aizawa e Wah (1994) empregaram uma formulação de problema de escalonamento. Nessa formulação, o número de processos se torna o número de avaliações ou amostragens por indivíduo, o número de processadores é o tamanho da população e o tempo de execução é a duração de cada geração do algoritmo genético.

Outra abordagem que pode ser feita para atenuar o efeito do ruído é modificar o processo de seleção. A forma mais simples, e utilizada por Markon et al. (2001), é de simplesmente aceitar os filhos de certos indivíduos caso o valor da função-objetivo seja melhor que os dos pais acrescidos de um limiar. Em (Branke e Schmidt, 2003), os autores adotam o uso do cálculo de probabilidade de diferentes maneiras, para compensar o efeito do ruído no momento da seleção proporcional.

Casos Reais

Casos reais de problemas com função-objetivo ruidosa já foram estudados por Pietro et al. (2002), onde os autores levaram em conta a física ruidosa dos sensores de movimento do futebol de robôs, por Darwen (2000); Barone e While (2000), onde foi estudado o efeito do ruído em jogos de azar. O efeito do ruído também foi estudado em outros algoritmos bio-inspirados, por exemplo em otimização por colônia de formigas (Gutjahr, 2003) e otimização por enxame de partículas (Fernandez-Marquez et al., 2009), e também em algumas outras meta-heurísticas não-populacionais, como recozimento simulado (Gutjahr et al., 1996; Prudius e Andradóttir, 2005; Branke et al., 2008) e busca tabu (Costa e Silver, 1998).

4. Problemas Robustos

Problemas robustos são aqueles em que não se busca necessariamente a solução ótima, mas sim uma boa solução que apresente o mínimo de variações quando parâmetros e outros atributos do problema variam. Essa situação ocorre quando não há precisão na determinação dos valores dos parâmetros ou existe algum ruído na determinação dos mesmos, tolerâncias de fabricação, alterações sob diferentes condições (temperatura, umidade, armazenamento). A função-objetivo para esse tipo de problema é representada conforme a Eq. 15.7:

$$F(x) = f(x + \vec{\delta}), \tag{15.7}$$

onde $f(x)$ é a função-objetivo ideal, $F(x)$ é a função-objetivo medida e δ é a perturbação ou ruído aditivo, geralmente considerado ser de distribuição normal, com média zero e variância σ^2 definida experimentalmente. Repare que agora a perturbação está no argumento da função-objetivo e não no valor calculado a cada ponto.

Pode-se pensar nesse problema como um problema ruidoso, só que com o ruído agindo sobre o vetor de parâmetros que caracteriza o argumento da função-objetivo. Apesar disso, vale lembrar que o objetivo aqui não é atenuar o efeito do ruído na determinação de valores mais confiáveis para a função-objetivo, mas sim encontrar um ponto do espaço de busca que representa uma solução de boa qualidade e para a qual o efeito do ruído sobre a função-objetivo é mínimo.

Exemplos Ilustrativos

A aplicação mais óbvia para esse tipo de problema é na fabricação de produtos (Parmee et al., 1994; Thompson, 1998; Wiesmann et al., 1998; Kumar et al., 2006; Loyer e Jézéquel, 2009), onde as peças ou produtos projetados sofrem alterações durante ou após a fabricação devido a fatores como imprecisão na manufatura, temperatura e umidade do ar. O objetivo do fabricante é então encontrar bons parâmetros e que sofram uma menor perda de qualidade diante desses efeitos.

Um outro exemplo, que foi apresentado na seção anterior como um problema ruidoso, é o de escalonamento de processos, também visto por alguns autores como um problema robusto (Leon et al., 1994; Sevaux e Sörensen, 2002a,b). A razão disso é que, além de termos ruídos externos que influenciam o tempo gasto na alocação dos processos, existem também ruídos que afetam os parâmetros internos da função-objetivo, como diferenças na finalização prevista de um determinado passo ou disponibilidade de um componente. Logo, para este tipo de problema, pode ser indicado não buscar a solução ótima (considerando ausência de ruído), mas sim uma boa solução que tenha o mínimo de degradação de desempenho ao sofrer essas alterações.

Outros tipos de problemas tratados como problemas robustos são aqueles de natureza econômica e financeira (Schied, 2006; Pictet et al., 1996), pois eles envolvem taxas de juros e de câmbio que flutuam em certa faixa de valores, assim como outros índices que tendem a sofrer variações dentro de certos intervalos, quando imprevistos de grandes consequências não ocorrem. Cabe antecipar aqui que, quando as variações de parâmetros e outros atributos são mais acentuadas, resulta um outro tipo de problema de otimização a ser tratado mais adiante: problema dinâmico.

Encontrando uma Solução Robusta

Médias Explícitas e Implícitas

Visando obter soluções robustas para os tipos de problemas que acabam de ser descritos, geralmente são utilizadas duas abordagens diferentes. A primeira delas é o uso de técnicas de média explícita e implícita, da mesma forma que nos problemas ruidosos. A diferença aqui é que o ruído é introduzido de forma controlada no vetor de parâmetros, de forma a cobrir a área de tolerância estipulada. As

mesmas técnicas mostradas na seção anterior podem ser imediatamente aplicadas aqui para encontrar uma solução robusta, como o uso da média da aptidão com várias amostras geradas a partir de perturbações aleatórias (Thompson, 1998; Wiesmann et al., 1998). Em (Loughlin e Ranjithan, 1999; Markon et al., 2001) é utilizada a amostragem denominada *Amostragem por Hipercubo Latino* (do inglês *Latin Hypercube Sampling* - LHS) (McKay et al., 2000) para gerar as amostras de perturbação dos parâmetros. Considerando em termos estatísticos, uma matriz $n \times n$ é um *Quadrado Latino* (do inglês *Latin Square*) se existir apenas um único valor não nulo em cada linha e em cada coluna (ver figura 15.1). O Hipercubo Latino é uma generalização do Quadrado Latino para dimensões maiores e pode ser representado como uma matriz $n \times d$, onde n é o número de amostras e d é a dimensão do vetor de parâmetros, e cada coluna apresenta uma permutação de valores, dentre os valores possíveis de cada variável, de forma a se ter n amostras diferentes. Isso traz a vantagem de não repetir amostras e ainda assim explorar toda a área de distribuição do ruído.

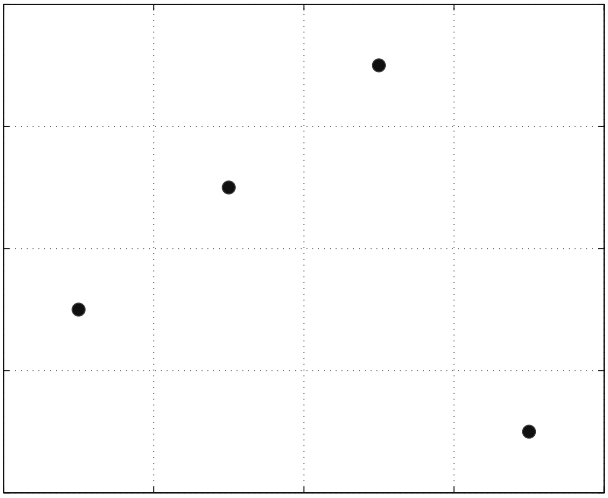


Figura 15.1: Exemplo de um Quadrado Latino, repare que cada linha e cada coluna contêm apenas uma amostra (representada por um círculo fechado).

Em (Branke, 1998), foi utilizada uma ideia similar à abordagem de média no espaço (Kay, 1993), já descrita no caso de problemas ruidosos. O valor da função-objetivo de um indivíduo é estimado através de uma média ponderada das soluções já calculadas, conforme Eq. 15.8:

$$f_{mod}(x_j) = \frac{\sum_i w_i.f(x_i)}{\sum_i w_i}, \tag{15.8}$$

onde x_j é a solução que se quer estimar, x_i são as soluções armazenadas no histórico e w_i é o peso que pondera o valor da função-objetivo para x_i. O peso é calculado em relação à distância do indivíduo que se quer estimar seguindo a Eq. 15.9:

$$w_i = max\{0, 1 - d \times b\}, \tag{15.9}$$

onde d é a distância entre a solução i e a solução j e b é um parâmetro definido pelo usuário. Essas soluções já calculadas podem ser tanto da população atual quanto de um histórico armazenado

referente a diversas iterações passadas.

Seguindo as mesmas soluções para Problemas Ruidosos, também foram feitos testes com média implícita (Tsutsui et al., 1996; Tsutsui e Ghosh, 1997; Branke, 1998), onde o aumento da população e a introdução de ruído nas variáveis de cada solução realizam uma amostragem equivalente à média explícita. De qualquer forma, novamente foi percebido que, dependendo da situação, uma proposta de tratamento pode ser melhor que a outra.

Formulação Multiobjetivo

Uma desvantagem em usar apenas a média, explícita ou implícita, do valor da função-objetivo de cada indivíduo é desprezar a sua variância (Das, 2000; Jin e Sendhoff, 2003). Uma alternativa é considerar dois critérios: o valor esperado para a função-objetivo e a sua variância. Mas para tanto, converte-se um problema de otimização mono-objetivo em um problema multiobjetivo. O leitor é convidado a consultar o capítulo que trata de Otimização Multiobjetivo para se familiarizar com os conceitos envolvidos, particularmente com os conceitos de soluções não-dominadas e de fronteira de Pareto e com a forma com que algoritmos evolutivos podem ser empregados no seu tratamento. Note que esses dois critérios não são necessariamente conflitantes, dependendo da natureza do problema e do ponto em que se encontram as soluções candidatas no espaço de busca. Outros critérios também podem ser adicionados ao problema multiobjetivo para permitir uma melhor escolha de soluções. Em (Ray, 2002), o próprio valor real de aptidão foi adicionado como critério. Outra opção é otimizar a função-objetivo e uma medida de *robustez* calculada como a razão entre o desvio padrão do valor da função-objetivo e o desvio padrão do ruído gerado no argumento da função-objetivo.

Com o uso de formulações multiobjetivo para resolver o problema de robustez em otimização em ambientes incertos, surgiu também o interesse em encontrar soluções para problemas multiobjetivos em que a fronteira de Pareto é robusta. Em (Deb e Gupta, 2005), foram examinados quatro casos diferentes:

- **A fronteira Pareto-ótima original é única e já é robusta:** o caso mais simples em que nada precisa ser feito;
- **Apenas parte da fronteira Pareto-ótima é robusta:** quando apenas parte da fronteira Pareto-ótima faz parte de uma região sensível a perturbações no espaço de variáveis. Neste caso, é necessário tratar apenas essa parte da fronteira de Pareto, buscando soluções mais robustas;
- **Uma fronteira Pareto-ótima local é robusta enquanto a global não é:** no caso em que a média das perturbações na fronteira local domina a média das perturbações na fronteira global, é preferível retornar a fronteira local, mais robusta.
- **Parte da fronteira Pareto-ótima global é robusta:** quando parte da fronteira global é robusta e na região sensível existe uma fronteira local que é robusta, o mais indicado seria combinar essas duas fronteiras de forma a constituir uma fronteira mais robusta.

Para lidar com esses problemas, foi utilizado o algoritmo genético multiobjetivo NSGA-II (Deb et al., 2002), substituindo as funções-objetivo pela média das amostragens de perturbações feitas, de forma similar ao procedimento do hipercubo latino. Outra forma utilizada é manter as funções-objetivo originais e adicionar uma restrição. Com esta restrição, são permitidas apenas soluções que tenham uma variância mínima no valor de cada função-objetivo, quando as variáveis são perturbadas. A formulação é apresentada a seguir:

$$\left.\begin{array}{ll} Minimizar & (f_1(x), f_2(x), \ldots, f_M(x)), \\ s.a. & \frac{\left\| f_i^{aprox.} - f_i(x) \right\|}{\|f_i(x)\|} \leq \eta, \qquad \forall i = 1..M, \\ & x \in S. \end{array}\right\}, \tag{15.10}$$

onde $f_i^{aprox.}$ é a média de n amostras de perturbação na solução para a função-objetivo i, M é o número de objetivos, S é o espaço de busca e $\|.\|$ pode ser qualquer medida de norma.

Em (Luo e Zheng, 2008), é explorado um algoritmo denominado *Eff-MOEA – Effective objective function based multi-objective evolutionary algorithm* – substituindo as funções-objetivo originais pela média dessas funções com perturbações no espaço de variáveis, amostradas através da técnica de *Monte Carlo*.

Uma outra forma de considerar o problema robusto é recorrendo à teoria de aritmética intervalar (Moore, 1966), geralmente utilizada para lidar com imprecisões numéricas computacionais. Nessa abordagem, o intervalo de tolerância que cada variável pode ter é codificado diretamente nos vetores de soluções. Dessa forma, um vetor de soluções passa a ser representado por:

$$[x] = [\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2], ..., [\underline{x}_n, \overline{x}_n] \text{ para } x \in \Re^n, \tag{15.11}$$

onde $\underline{x}_i$ e $\overline{x}_i$ são, respectivamente, o limitante inferior e o limitante superior do intervalo da variável i.

Em (Soares et al., 2009), essa abordagem foi explorada em um algoritmo evolutivo multi-objetivo denominado [I]RMOEA – *Interval Robust Multi-Objective Optimization Evolutionary Algorithm* – onde o tamanho de cada intervalo é definido antes da otimização, de acordo com a precisão real do sistema. A avaliação das soluções é feita de acordo com um parâmetro de incerteza que define as amostras que serão geradas dentro do intervalo determinado junto a cada variável e, então, são retornados os piores valores obtidos dentre essas amostras. Nesse método, os operadores de dominância também foram alterados para lidar com a aritmética intervalar, de tal forma que uma solução deve dominar a outra dentro de todo o conjunto de parâmetros de incerteza atribuídos a eles.

De forma diferente, em (Forouraghi, 2000) foi feita a otimização não só dos parâmetros mas também do intervalo de tolerância. Dessa forma, é possível avaliar a melhor solução de um problema para diferentes tolerâncias. Outra novidade nessa abordagem foi o uso de medição de sinal-ruído como função de aptidão. Nessa metodologia, a aptidão de uma solução é calculada de acordo com a medição da avaliação do ruído, ou seja, da distribuição de amostras dentro de cada intervalo. Essas amostras são definidas através de um método de vetores ortogonais. A equação da aptidão é dada por:

$$\eta = -10 \log \left(\frac{\sum_{i=1}^{n} y_i^2}{n}\right), \tag{15.12}$$

onde y_i é o experimento i dentro do intervalo definido e n é o número de amostras.

Finalmente, em (Sanseverino, 2005), foi otimizado o "hipercubo" que envolve a tolerância de uma solução, ou seja, o quanto cada variável de uma solução pode ser alterada de forma a manter a diferença da função-objetivo dentro de um determinado limiar.

Uma forma alternativa de tratar as incertezas é aplicando lógica nebulosa (Pedrycz e Gomide, 2007) nos parâmetros. Dessa forma, cada parâmetro assume uma faixa variável de valores com diferentes graus de pertinência. O trabalho desenvolvido por Wang (2004) focou especificamente no problema de escalonamento, onde as incertezas foram modeladas na duração de cada processo, e o tempo entre o término de um processo e início de outro. Transformando esses valores em números nebulosos (Pedrycz e Gomide, 2007) é possível definir um cálculo de aptidão que reflete a possibilidade de uma solução não ser tão robusta, tendo em vista as imprecisões e incertezas.

Outras Aplicações e Técnicas

Outra situação em que Problemas Robustos podem ser estudados é quando as restrições são tratadas de forma robusta, ao invés da função-objetivo. A ideia aqui, ao contrário de soluções robustas em que se quer minimizar o prejuízo na aptidão quando há perturbações na solução, é garantir que a solução continuará viável mesmo na presença de perturbações. As mesmas técnicas descritas acima podem ser aplicadas aqui, mas com o cuidado em definir quando uma solução será considerada infactível (por

exemplo, quando o valor amostrado da restrição for infactível ou quando pelo menos uma amostra de perturbação for infactível). Esse problema é abordado por Deb et al. (2009) e denominado *Otimização Baseada em Confiabilidade*.

Aplicações recentes de otimização robusta podem ser encontradas em (Deb, 2008), onde foi feito um estudo de caso no problema de distribuição de energia hidrotérmica; Roy et al. (2009) estudaram o problema de projetar um sistema de resfriamento por bobinas; enquanto Lee et al. (2008) resolveram o problema de projetar a aerodinâmica em veículos de combate aéreo não-tripulados.

Outras técnicas bio-inspiradas já utilizadas para resolver problemas robustos são a *Otimização por Sistemas de Partículas Multiobjetivo* (Ono e Nakayama, 2009), que formula o problema como um problema bi-objetivo, conforme descrito anteriormente, e o *Sistema Hormonal Artificial* (Moioli et al., 2009) utilizado para controle autônomo de robôs móveis, onde é necessário estar imune a efeitos de ruídos em suas variáveis de decisão.

5. Aproximação de Função

Quando se modelam matematicamente problemas reais, nem sempre é possível modelá-los de forma completa ou precisa. Algumas vezes a razão está na complexidade intrínseca do problema, outras vezes está no custo do conhecimento e em outras circunstâncias está na ausência de recursos tecnológicos. Também podem ser inseridos neste contexto os erros involuntários ao longo do processo de modelagem. Em muitos casos, na ausência de uma formulação matemática para a função-objetivo, a atribuição de um valor de adaptação para as soluções candidatas é feita por operadores humanos, o que representa uma tarefa exaustiva, sujeita a erros e à incorporação de aspectos subjetivos na avaliação. A função-objetivo, nesses casos, é dada pela Eq. 15.13:

$$F(x) \approx E(x), \tag{15.13}$$

onde $E(x)$ é a função-objetivo estimada ou aproximada.

Exemplos Ilustrativos

Para os casos em que o custo de avaliação é alto, um problema muito estudado é a Otimização de Projetos Aerodinâmicos, que envolvem o cálculo da fluidodinâmica computacional. Há a necessidade de avaliar equações de Navier-Stokes em três dimensões, sendo que um computador moderno pode levar horas de cálculos para tratar uma única solução. Como, geralmente, em algoritmos de Computação Evolutiva é necessário avaliar milhares ou até milhões de soluções para obtermos uma solução de boa qualidade, esse problema pode se tornar inviável. Uma saída é utilizar aproximações dessa função original, o que acaba gerando um problema aproximado a um custo menor. Essa aproximação pode ser feita avaliando algumas amostras da função-objetivo real e, então, aproximando esta função por meio de métodos como modelo Kriging (Jeong e Murayama, 2004) e metamodelos (El-Beltagy e Keane, 1999) e por meio de técnicas de aprendizado de máquina como redes neurais artificiais (Rai, 2002; Hüsken et al., 2005) e máquinas de vetores-suporte (do inglês *Support Vector Machines*) (SVM) (Fan et al., 2005).

Já quando se fala em funções-objetivo que não se pode quantificar apropriadamente em termos determinísticos e não-subjetivos, citam-se como exemplos composição musical, síntese sonora, painéis e esculturas. No caso de composição musical, o objetivo do algoritmo é encontrar uma sequência de tons e timbres que gerem uma composição musical qualitativamente agradável para o ser humano ouvinte. Embora esse objetivo possa parecer bem claro, não existe uma função-objetivo bem definida para avaliar cada solução candidata. Intuitivamente, esse processo pode ser feito através da interação humana (Biles, 2002), onde existe um ou mais avaliadores que escutarão a a sequência sonora gerada pelo algoritmo e, em seguida, atribuirão um valor de acordo com a qualidade da solução gerada.

Um problema com essa abordagem é novamente a quantidade de avaliações que os algoritmos de Computação Evolutiva requerem e, portanto, a interação do avaliador pode se tornar cansativa, tender a erros e variações de critério. Outro problema é a subjetividade e arbitrariedade com que o avaliador atribuirá as notas. Uma maneira de reduzir esse problema é utilizar algum modelo de aprendizado para tentar entender a forma com que o avaliador atribui notas às composições. Um exemplo disso foi utilizado em (Johanson e Poli, 1998), onde os autores aplicaram uma rede neural para modelar as preferências do avaliador. Outras formas para resolver essa questão é a utilização de metamodelos criados a partir de certas estruturas de teoria musical (Wiggins et al., 1998). Uma linha de pesquisa bem estabelecida e que envolve operadores humanos na avaliação de soluções candidatas propostas por computador, particularmente em projetos artísticos, é a de algoritmos evolutivos interativos (Bentley, 1999; Takagi et al., 2001; Moroni et al., 2002; Machwe e Parmee, 2007).

Aproximando as Funções

Os métodos utilizados para aproximar funções podem ser divididos em três frentes, segundo Jin (2005). Duas dessas frentes são tradicionais da área de otimização e uma é específica da área de computação evolutiva:

- *Aproximação do Problema*: quando um método é dito aproximar um problema, significa que ele procura encontrar um problema simplificado aproximado do original mas com um custo menor. Uma maneira de implementar esse método é negligenciando certos aspectos da formulação do problema, como restrições e efeitos externos (Engquist e Runborg, 2002; Jaisankar e Raghurama Rao, 2007). Outra forma seria através de simulações numéricas contando com diferentes níveis de acuidade, impactando na qualidade da avaliação.

- *Aproximação Funcional e Meta-modelos*: a aproximação funcional refere-se à construção de um modelo matemático explícito e equivalente ao original. Tal modelo descreve um espaço de busca aproximado enquanto economiza em custo computacional. Quando esse tipo de aproximação é feita através de um modelo matemático, sua implementação se torna específica para um determinado problema e muitas vezes sua criação é difícil e custosa. Uma forma mais comum de aproximação funcional são os *meta-modelos*, que são sintetizados a partir de abordagem bottom-up, geralmente empregando aprendizado a partir de dados amostrados.

- *Aproximação Evolucionária*: esse tipo de aproximação foi concebida para algoritmos evolutivos e explora a abordagem populacional para gerar conhecimento sobre o espaço de busca. Esse tipo de aproximação leva em conta a distribuição espacial entre os indivíduos de uma geração e sua prole, de forma que seja possível utilizar aproximações com maior ou menor acuidade, dependendo da quantidade de informação existente na região do novo indivíduo. Métodos comuns a esse tipo de aproximação são: herança de aptidão (do inglês *fitness inheritance*), imitação de aptidão (do inglês *fitness imitation*) e atribuição de aptidão (do inglês *fitness assignment*).

Visando manter o foco nos algoritmos evolutivos, apenas os métodos dos dois últimos itens serão descritos a seguir.

Construção de meta-modelos por regressão não-paramétrica

Existem, na literatura, diversos métodos para construção de meta-modelos baseados em regressão não-paramétrica, ou seja, sem conhecimento a priori da forma da função a ser estimada. Esses modelos se diferenciam de acordo com o nível de precisão obtido em relação à função original e à quantidade de amostras que cada um requer. A seguir, são apresentados os métodos mais comuns da literatura.

Modelo Linear

O modelo linear de regressão gera uma combinação linear de diversas funções (lineares ou não) de forma a aproximar o máximo possível a curva gerada aos pontos de amostragem da função-objetivo real $f(x)$. Esse modelo assume a seguinte forma:

$$F(x) = \sum_{j=1}^{m} w_j h_j(x), \tag{15.14}$$

onde x pertence a uma região compacta (fechada e limitada), $F(x)$ é a função aproximada, h_j é a j-ésima função-base, w_j é o peso atribuído à j-ésima função-base e m é o número de funções-base.

A Eq. 15.14 representa uma combinação linear de m funções-base, onde sua estrutura e parâmetros são pré-definidos. Mesmo que as funções-base sejam não-lineares, o modelo ainda é considerado linear nos parâmetros, pois a flexibilidade de obtenção de $F(x)$ está relacionada apenas à escolha dos coeficientes da combinação linear $(w_j, j = 1, ..., m)$. Funções-base comumente usadas são polinomiais, wavelets, senoides e sigmoides. Os coeficientes da combinação linear são geralmente definidos com base em métodos de quadrados mínimos e restrições de suavidade podem ser adicionadas ao modelo.

Modelo Kriging

O modelo Kriging é um modelo generalizado do modelo linear de interpolação que considera a existência de correlação espacial entre amostras próximas. A equação básica de um modelo Kriging apresenta poucas diferenças em relação ao modelo linear, assumindo a forma:

$$F(x) = \sum_{j=1}^{m} w_j(x) h_j(x), \tag{15.15}$$

onde agora os coeficientes da combinação linear são dependentes de x e as funções-base dependem dos dados amostrados.

Os pesos são calculados de forma que a média dos erros entre os valores reais e os estimados seja nula e a variância seja mínima. Na verdade, Kriging se refere a um conjunto de métodos, sendo que um dos mais utilizados é aquele que obtém os pesos na forma:

$$w(x) = \begin{pmatrix} c(x_1, x_1), & \dots & c(x_1, x_n) \\ \vdots & \ddots & \vdots \\ c(x_n, x_1), & \dots & c(x_n, x_n) \end{pmatrix}^{-1} \begin{pmatrix} c(x_1, x) \\ \vdots \\ c(x_n, x) \end{pmatrix}, \tag{15.16}$$

onde $c(x_i, x_j)$ é um índice que estima a covariância entre duas amostras.

Redes Neurais Artificiais

Redes Neurais Artificiais (Haykin, 1999) (RNAs) representam uma técnica de inteligência computacional aplicada basicamente a quatro tipos de problemas de aprendizado de máquina: memória endereçável por conteúdo, classificação de padrões, agrupamento de dados e síntese de mapeamentos não-lineares multidimensionais. Haverá ênfase aqui neste último tipo de problema de aplicação, em que as redes neurais operam como modelos de regressão não-paramétrica.

A capacidade de aprendizado e a estrutura conexionista de redes neurais resultam da modelagem matemática de neurônios e suas conexões sinápticas no cérebro. O princípio de operação das redes neurais artificiais se fundamenta na interação de unidades de processamento simples e similares, denominadas neurônios artificiais. Os vários padrões já propostos para conexão entre neurônios artificiais

definem estruturas de processamento de informação com grande poder de síntese de mapeamentos não-lineares multidimensionais, a partir do ajuste dos pesos associados às conexões sinápticas. Se existirem conexões recorrentes, o mapeamento exibe propriedades dinâmicas, senão ele é estático. Serão considerados aqui mapeamentos estáticos e aprendizado supervisionado, em que estão disponíveis amostras do mapeamento a ser aproximado.

Apenas duas versões de RNAs voltadas para aprendizado a partir de dados amostrados serão consideradas a seguir, devido à relevância e ampla aplicação de ambos os modelos, além do que eles admitem uma representação similar àquela já adotada para descrever os modelos lineares. O primeiro deles é o perceptron de múltiplas camadas (MLP, do inglês *multilayer perceptron*) (Lippmann, 1987), o qual apresenta capacidade de aproximação universal (Cybenko, 1989), ou seja, aproxima com grau arbitrário de precisão qualquer mapeamento contínuo em uma região compacta do espaço de aproximação. No entanto, por não se conhecer a priori a forma do mapeamento que se deseja aproximar, é geralmente um desafio especificar o número de neurônios e o quanto a rede neural deve ser treinada para produzir uma boa capacidade de generalização, ou seja, para responder bem a amostras não empregadas durante a fase de treinamento supervisionado (ajuste dos pesos sinápticos). Como critério de parada para o treinamento supervisionado, pode-se separar parte do conjunto de amostras disponível para treinamento, criando um conjunto de dados de validação. O mapeamento realizado por uma rede neural do tipo perceptron com uma camada intermediária de neurônios é dado na forma, sendo $F_k(.)$ a saída produzida pelo k-ésimo neurônio da camada de saída:

$$F_k(x) = \sum_{j=1}^{n} w_{kj} f(\sum_{i=1}^{m} v_{ij} x_i + v_{0j}) + w_{k0}, \tag{15.17}$$

onde m é o número de entradas, n é o número de neurônios na camada intermediária, $f(.)$ é a função de ativação desses neurônios, k é o índice da saída, w_{kj} é o peso sináptico que conecta a saída do j-ésimo neurônio da camada intermediária ao k-ésimo neurônio da camada de saída e v_{ij} é o peso sináptico conectando a i-ésima entrada ao j-ésimo neurônio da camada intermediária. A função de ativação dos neurônios da camada intermediária geralmente é tomada como sendo a função tangente hiperbólica. Os índices i e j se iniciam em zero porque é necessário considerar uma entrada constante, denominada entrada de polarização, para todos os neurônios. Se apenas os pesos w_{kj} fossem ajustáveis, resultaria um modelo linear nos parâmetros ajustáveis. No entanto, os pesos v_{ij} são também ajustáveis, resultando um modelo não-linear de aproximação.

Uma outra forma bem conhecida de arquitetura de RNA que pode ser usada para a síntese de mapeamentos não-lineares é a rede neural com funções de ativação de base radial (RBF, do inglês *radial basis function*) (Broomhead e Lowe, 1988). A principal diferença entre as redes neurais MLP e RBF está no formato das funções de ativação dos neurônios da camada intermediária ($f(.)$) e na forma como as entradas são processadas pelos neurônios da camada intermediária. Na MLP é feito um produto interno, enquanto que na RBF é calculada a distância euclidiana. O mapeamento realizado por uma rede neural com funções de ativação de base radial é dado na forma, sendo $F_k(.)$ a saída produzida pelo k-ésimo neurônio da camada de saída:

$$F_k(x) = \sum_{j=1}^{n} w_{kj} \phi(\|x - c_j\|) + w_{k0}, \tag{15.18}$$

onde $\phi(.)$ é a função de base radial com centro em c_j e $\|.\|$ é a norma euclidiana. A rede neural RBF também apresenta capacidade de aproximação universal (Park e Sandberg, 1991).

Máquinas de Vetores-Suporte

As máquinas de vetores-suporte (SVM, do inglês *support vector machines*) (Vapnik, 1998) compõem um método de aprendizado estatístico que procura superar a limitação apresentada pelos modelos com capacidade de aproximação universal, ou seja, a dificuldade de definir o grau de flexibilidade do modelo não-paramétrico. Ao mapear os dados do problema original para um espaço de maior dimensão, empregando métodos de kernel, o que se busca é um hiperplano que apresente margem máxima (no caso de problemas de classificação) ou que melhor aproxime os dados, no caso de problemas de regressão. Ao retornar ao espaço original, produz-se o mapeamento não-linear que resolve o problema, com grau de flexibilidade adequada. As máquinas de vetores-suporte foram originalmente concebidas para resolver problemas de classificação binária, sendo posteriormente estendidas para regressão e agrupamento de dados. Os vetores-suporte, neste caso de classificação binária, são as amostras que definem a posição do hiperplano de máxima margem de separação. Em qualquer caso, encontrar o hiperplano no espaço expandido implica em resolver um problema de otimização quadrática com restrições de igualdade e/ou desigualdade, o qual tem solução única. Além de ter solução única, a complexidade do modelo não depende do número de entradas, mas sim do número de vetores-suporte.

Extraindo mais da população a cada geração

A população gerada pelos algoritmos de computação evolutiva são geralmente ricas em informação do espaço de busca a ser otimizado, além de tender a amostrar mais regiões com maior potencial de produção de indivíduos com bons níveis de avaliação. Diante disso, algumas técnicas foram desenvolvidas para minimizar o número de avaliações reais da função-objetivo e, assim, reduzir o custo quando se depara com funções-objetivo muito custosas. Essas técnicas serão descritas brevemente a seguir.

Herança de aptidão

Herança de aptidão (do inglês *fitness inheritance*) (Smith et al., 1995) é uma técnica que estima o valor da função-objetivo (*fitness*) de parte dos novos indivíduos a partir do valor já atribuído às soluções-pais, geralmente se baseando nos seus blocos construtivos (do inglês *building blocks*) (Goldberg, 1989). Blocos construtivos correspondem a um conceito empregado para justificar a eficácia dos algoritmos evolutivos e remetem a pedaços de boas soluções, ou padrões dentro da codificação, que contribuem para a avaliação dos indivíduos que os contêm. Sendo assim, ao longo das gerações, esses blocos construtivos tendem a se espalhar pela população. Com isso, é possível inferir que a dimensão do problema original é reduzida para o número de blocos construtivos.

Para se avaliar os indivíduos a partir de blocos construtivos, deve-se inicialmente obter a média da aptidão de todos os indivíduos que contêm cada bloco construtivo, de acordo com a Eq. 15.19:

$$F(x) = \frac{f}{l} + (l-1)p, \tag{15.19}$$

onde f é a aptidão real do bloco construtivo, l é o tamanho do bloco construtivo e p é a proporção de blocos construtivos corretos dentro do indivíduo.

Em (Sastry et al., 2001), foi avaliado o ganho de desempenho ao usar herança de aptidão com a proporção ótima de indivíduos que herdam a aptidão e aqueles que têm sua aptidão calculada com a função original. Nesse estudo, concluiu-se que, com o uso de uma população fixa, o número de avaliações da função real pode ser reduzido em até 70%.

O desempenho da herança de aptidão foi também avaliado para problemas multiobjetivo (Chen et al., 2002; Ducheyne et al., 2008). Chen et al. (2002) obtiveram um ganho na redução das avaliações de um algoritmo multiobjetivo com compartilhamento de aptidão (explicado a seguir). Já Ducheyne et al. (2008) concluíram que o uso de herança de aptidão em problemas não-convexos e não-contínuos pode afetar negativamente o desempenho do algoritmo.

Uma outra estratégia possível, que não trabalha diretamente com os blocos construtivos, é utilizar uma média ponderada da aptidão dos pais em relação ao quanto de cada pai o indivíduo absorveu (Chen et al., 2002; Salami e Hendtlass, 2003).

Imitação de aptidão

Imitação de aptidão (do inglês *fitness imitation*) foi idealizada por Kim e Cho (2001) e consiste em agrupar indivíduos da população em vários grupos distintos, através de uma medida de dissimilaridade. Ao agrupar os indivíduos, um deles será selecionado para ser o centro do grupo, e terá sua aptidão avaliada pela função-objetivo real. Os outros indivíduos pertencentes ao grupo terão sua aptidão estimada com base na aptidão do centro (imitação) e proporcional à dissimilaridade entre eles. O algoritmo de agrupamento e a fórmula de dissimilaridade podem ser escolhidos de acordo com o problema tratado. Em (Kim e Cho, 2001), foram utilizados o método k-médias de agrupamento (Cover e Hart, 1967) e a medida de distância euclidiana.

Em (Bhattacharya e Lu, 2003), foi criado um algoritmo evolutivo que combina os métodos de imitação de aptidão com o modelo *Support Vector Machine* (SVM) de regressão. Ele inicia o procedimento amostrando 5 vezes mais soluções do que o tamanho total da população e calculando o valor real da função-objetivo desses pontos de forma a gerar um modelo de regressão utilizando o SVM. Após esse passo, as N melhores soluções são selecionadas para fazer parte da população inicial. A seguir, aplica-se o processo de seleção de pais, reprodução e mutação, gerando novos indivíduos. Esses indivíduos terão sua aptidão calculada através do modelo de regressão do SVM. Em seguida, a partir da nova população, são gerados m agrupamentos e a aptidão de cada novo indivíduo é recalculada a partir da seguinte fórmula:

$$F(x) = f_{SVM}(x) - \rho_1\sigma_i - \rho_2 d_i - \rho_3 s_i, \tag{15.20}$$

onde ρ_1, ρ_2, ρ_3 são pesos atribuídos a cada um dos fatores da equação, $f_{SVM}(x)$ é a aptidão calculada via SVM, σ_i é o desvio padrão do valor da função-objetivo do cluster i ao qual o indivíduo pertence, d_i é a distância euclidiana mínima normalizada entre a solução e os elementos do cluster i que tiveram sua aptidão avaliada pela função-objetivo original, e s_i é a dispersão dos elementos do cluster i, calculada pela Eq. 15.21:

$$s_i = \frac{no.\_de\_individuos\_no\_cluster\_i}{dimensão\_do\_individuo}. \tag{15.21}$$

Com as aptidões aproximadas calculadas, encontra-se o valor ótimo atual aproximado e calcula-se o seu valor real e também o de seus k-vizinhos mais próximos. Repete-se esse procedimento para cada um dos m centróides referentes aos agrupamentos construídos. Esses novos pontos calculados serão adicionados ao modelo do SVM para ser feita uma atualização do modelo de aproximação. Finalmente, o processo se repete até que o critério de convergência seja alcançado.

Em (Jin e Sendhoff, 2004), foi criado um algoritmo parecido com o descrito acima, mas utilizando um *ensemble* de redes neurais para aproximar a função, calculando a função real apenas dos centroides dos grupos.

Atribuição de Aptidão - Algoritmos coevolutivos

O conceito de atribuição de aptidão (do inglês *fitness assignment*) está diretamente ligado aos algoritmos coevolutivos. Esses algoritmos representam um sistema de competição entre duas populações que evoluem em paralelo, onde a aptidão de cada indivíduo de uma população vai depender dos indivíduos da outra população. Esses algoritmos são particularmente úteis quando não existe um objetivo definido, ou seja, quando ocorre uma situação em que se quer saber, dados dois problemas correlatos, quando a solução de um problema compete bem, ou se aplica melhor, na solução do outro problema.

Dessa forma a pressão seletiva impõe alterações na avaliação durante a evolução, tornando o processo dinâmico e competitivo.

Uma forma de utilizar esse conceito em avaliação de indivíduos é imaginar que uma população é formada pelas soluções candidatas do problema a ser resolvido e a outra por uma série de modelos de predição capazes de aproximar a aptidão. Em (Schmidt e Lipson, 2008), essa ideia é estendida adicionando uma terceira população que evolui o conjunto de amostras de treinamento dos preditores.

Esse algoritmo começa com a população de soluções candidatas, onde várias iterações de um algoritmo evolutivo são efetuadas utilizando o melhor preditor atual, até que o erro de aproximação da melhor solução obtida seja menor que um limiar. Em seguida, passa-se para a população de preditores, onde eles sofrem também um processo de evolução (o treinamento de uma rede neural do tipo MLP ou RBF pode ser feito através de algoritmos evolutivos também) e, periodicamente, evoluem também a população de amostras de treinamento, aumentando a variabilidade de soluções. Idealmente, esses processos de evolução são feitos em paralelo.

Falsos Ótimos

Muitas vezes os pontos amostrados não são suficientes para dar uma indicação de onde se encontra o ótimo global ou, ao menos, boas soluções em uma região do espaço de busca que está sendo explorada. Mesmo que o cálculo real do valor da função-objetivo (aptidão) seja feito para alguns indivíduos da população, a baixa amostragem em regiões em que a superfície de aptidão sofre variações mais intensas pode conduzir a aproximações equivocadas e, por consequência, a falsos ótimos.

Em outros casos, a taxa de amostragem pode estar adequada, mas a distribuição não-uniforme de amostras pela superfície de aptidão pode induzir uma aproximação equivocada, a qual novamente pode guiar o processo evolutivo a falsos ótimos.

Um exemplo ilustrativo considerando um espaço de busca de dimensão 1 é apresentado na Figura 15.2.

Outras Aplicações e Técnicas

É evidente que essas estratégias de aproximação de funções podem ser empregadas junto a outras meta-heurísticas populacionais que operam em espaços de busca. Um exemplo é a otimização por enxame de partículas (PSO, do inglês *particle swarm optimization*) (Hendtlass, 2007).

Aproximações da função-objetivo também podem ser empregadas na presença de ambientes ruidosos (Branke et al., 2001), visando reduzir o efeito do ruído, e em otimização robusta (Paenke et al., 2006).

Uma outra contribuição possível da aproximação de funções-objetivo pode ocorrer em cenários em que os ótimos locais impedem ou dificultam a localização do ótimo global. Nesses casos, é possível substituir a função-objetivo original por uma aproximação que procura eliminar ótimos locais ou ao menos reduzir a quantidade destes, favorecendo a busca pelo ótimo global (Yang e Flockton, 1995; Liang et al., 2000, 1999).

6. Problemas Dinâmicos

Até este ponto do capítulo, foram apresentados problemas de otimização em ambientes incertos em que: (*i*) existe uma variação no valor da função-objetivo, em torno de seu valor real (ruído com propriedades estatísticas invariantes); (*ii*) a otimização deve levar em conta intervalos de valores admissíveis para certas variáveis e parâmetros, buscando a robustez da solução; (*iii*) a otimização deve se dar em situações em que é inviável ou impossível obter o valor real da função-objetivo, requerendo assim a definição de valores aproximados. Agora, será apresentado o quarto tipo de incerteza que pode surgir

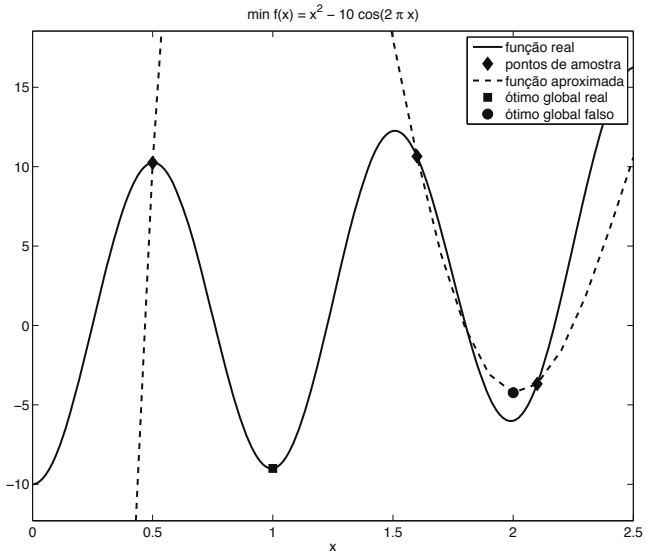


Figura 15.2: Exemplo da criação de um falso ótimo global por conta da aproximação de função com amostragens insuficientes em um problema de minimização. Note que a forma aproximada da função-objetivo (em linha tracejada) difere da forma real (em linha sólida) em algumas regiões onde faltaram pontos de amostra. Nessa situação, o ótimo global da função aproximada se torna o ponto localizado pelo círculo sólido, enquanto o ótimo global real, representado pelo quadrado sólido, está em um ponto diferente da função. Esse problema poderia ser solucionado com a obtenção de um número suficiente de amostras bem localizadas, o que geralmente não é fácil de conseguir.

quando se resolve um problema de otimização: a incerteza causada por variações dinâmicas ao longo do tempo. Esse tipo de problema difere bastante dos outros três, pois não só a intensidade de variação do valor da função-objetivo é maior, como também podem ocorrer mudanças estruturais no espaço de busca. Com uma maior intensidade de variação da função-objetivo, podem resultar fenômenos como mudanças na localização de um ou mais ótimos, surgimento de novos ótimos e desaparecimento de ótimos existentes. Já um exemplo de mudança estrutural está na variação da dimensão do problema de otimização.

Para representar esse tipo de problema, geralmente é utilizada a notação da Eq. 15.22, onde consta que a função é dependente, não só do vetor de variáveis, mas também do tempo:

$$F(x) = f(x, t), \tag{15.22}$$

onde $f(x, t)$ é a função-objetivo que varia em relação ao tempo t. Note que aqui o termo tempo não necessariamente representa o conceito de tempo físico, mas sim uma sequência de eventos discretos, de origem interna ou externa ao sistema.

Exemplos Ilustrativos

Um exemplo de problema de otimização que varia com o tempo, e que ocorre no dia-a-dia de boa parte das pessoas que vivem em grandes cidades, é o problema de definição de rotas na presença de congestionamentos do tráfego urbano, citado brevemente na introdução desse capítulo. Hoje em dia, muitos veículos já contam com um aparelho de GPS para facilitar a localização e navegação pela cidade. Esses aparelhos têm a capacidade de encontrar a rota que tem o menor custo (distância)

entre o ponto em que o veículo se encontra e o destino desejado. Esses algoritmos conseguem traçar essa rota quase instantaneamente, auxiliando o motorista no caminho a seguir. Mas os aparelhos de navegação não contêm informações sobre tráfego e possíveis congestionamentos, nem mesmo a localização e estado dos semáforos, uma vez que todos esses requisitos dependem do instante em que o motorista está passando pelos locais. Com isso, a solução dada pelo navegador, embora ótima quando se consideram condições de tráfego uniforme em todas as vias, pode vir a ter uma qualidade ruim por conta dos imprevistos e das condições não-uniformes de tráfego. Em uma situação ideal, o algoritmo seria capaz de perceber o mais breve possível a mudança no ambiente de busca e reagir traçando uma rota alternativa que evitasse imprevistos ou fatores que possam reduzir a qualidade da solução. Claro que, no caso de uma rota entre dois pontos, o navegador poderia facilmente recalcular uma nova rota partindo do ponto atual em que o motorista avistou o congestionamento, uma vez que a solução desse problema é relativamente barata. Mas quando se trata de rotas para múltiplos veículos, sob requisitos de otimização global (Ahn e Shin, 1991; Fleischmann et al., 2004), ou ciclos hamiltonianos (Guntsch e Middendorf, 2001; Zhou et al., 2003), o problema se torna $\mathcal{NP}$-difícil. Nessas circunstâncias, reiniciar o processo de otimização (supondo que o problema é estático) para cada novo estado do problema torna-se inviável, o que remete a técnicas de otimização capazes de operar sob condições e requisitos que variam dinamicamente.

Outro exemplo, já citado em seções anteriores, é o problema de escalonamento de processos computacionais (Savic et al., 1997; Dellaert et al., 2000; Ingolfsson et al., 2002). Como visto, esse problema trabalha com um número pré-fixado de processadores e processos, mas durante a execução da solução obtida pode ocorrer uma mudança em uma etapa de algum processo, ocasionando atraso na execução desse, um ou mais processadores podem sair de operação e processos novos e urgentes podem surgir de última hora. Todas essas mudanças causam uma alteração na dimensão do problema e, portanto, no espaço de busca, além de mover as boas soluções ao longo do espaço de busca. Adicionalmente, o custo de execução de cada processo pode se alterar com o passar do tempo, tornando algumas soluções infactíveis.

Lidando com Mudanças

Um ponto comum das meta-heurísticas populacionais, incluindo os algoritmos evolutivos, é a tendência de convergir para uma solução de ótimo local, que pode até ser o ótimo global. Em ambientes variantes no tempo, isso é algo indesejável, uma vez que o algoritmo perde a capacidade de reação ao se confrontar com uma mudança mais intensa de características do problema. Outro ponto importante é a detecção de que realmente houve uma mudança na função-objetivo e, portanto, deve-se reavaliar as soluções com as quais o algoritmo está trabalhando nesse momento. Note que simplesmente reavaliar todas as soluções a cada passo do algoritmo pode não ser desejável pelo custo computacional. Em revisão feita por Jin e Branke (2005), foram destacadas características incorporadas por algoritmos evolutivos para resolver essa situação, concluindo que o uso de uma ou mais dessas características, quando possível, são desejáveis para lidar com esse tipo de problema. Essas características são:

- *Geração de Diversidade*: um modo simples de evitar a convergência do algoritmo é simplesmente inserir diversidade de tempos em tempos, ou quando uma mudança no problema de otimização for detectada.

- *Manutenção de Diversidade*: de forma similar ao item anterior, mas um pouco mais elaborada, pode-se utilizar mecanismos que promovam a capacidade de exploração do espaço de busca, a todo momento. Isso obviamente implica em custos computacionais adicionais para o processo de otimização, visto que haverá um compromisso entre o investimento voltado para localizar o ótimo global e aquele voltado para manter soluções diversas na população.

- *Memória de Soluções*: o algoritmo pode também implementar uma memória de soluções passadas que pode ser utilizada na inserção de diversidade ou de forma a reinicializar a busca diante de mudanças no ambiente. Essa abordagem é particularmente útil em situações em que a mudança é cíclica, e o ótimo tende a passar sempre pelos mesmos lugares, embora nem sempre numa sequência previsível.

- *Abordagem Multipopulacional*: ao dividir a população original em várias sub-populações, pode-se incentivar as populações a se manterem afastadas entre si, explorando assim regiões distintas do espaço de busca. É evidente que cada sub-população pode ser composta por suas próprias sub-populações, e assim por diante, numa organização hierárquica e aninhada.

Uma outra característica, que inicialmente foi concebida com os Sistemas Imunológicos Artificiais (de Castro e Timmis, 2002), mais especificamente o algoritmo aiNet (de Castro e Von Zuben, 2001), e recentemente tem sido empregada em alguns algoritmos para otimização em ambientes variantes no tempo, é o controle dinâmico do tamanho da população (de França et al., 2005; Blackwell, 2007). Uma vez que é necessário manter diversidade sem perder o poder de exploração, os algoritmos tendem a necessitar de uma população com um tamanho maior ou menor que o inicialmente atribuído. Tendo um controle dinâmico desse tamanho, a partir de uma proposta inicial, pode-se minimizar o acréscimo do custo computacional gerado pelas técnicas apontadas anteriormente.

Além de incorporar tais características no algoritmo, pode ser desejável também alterar alguns mecanismos internos, como aqueles vinculados a operadores de mutação, reprodução e seleção, nos algoritmos evolutivos.

A seguir, será descrito, em linhas gerais, como cada uma dessas características desejáveis é implementada na literatura.

Introdução de Diversidade

Para gerar diversidade em algoritmos evolutivos, conta-se com algumas opções. Em (Cobb, 1990), por exemplo, toda vez que uma mudança é detectada, por reavaliação de indivíduos, a taxa de mutação é aumentada drasticamente de forma que a variabilidade das soluções aumente. Isso gera algumas questões, como o quanto essa taxa deve ser aumentada. Se aumentada demasiadamente, a busca se torna aleatória. Se for pequeno demais, pode não ter o efeito desejado.

Uma alternativa é aumentar gradativamente a taxa de mutação (Vavak et al., 1997), de forma controlada, e reduzi-la quando atingir níveis muito altos. De forma complementar, essas estratégias podem ser feitas também em relação à intensidade da mutação ou ao investimento de mais recursos na execução de etapas de busca local.

Outra estratégia é substituir indivíduos por novos amostrados em regiões distintas daquelas que estão sendo exploradas pelo algoritmo. Os indivíduos substituídos podem ser simplesmente os n piores ou os mais similares, por exemplo. Essa estratégia é conhecida como *imigração* (Grefenstette, 1992).

Manutenção de Diversidade

A ideia de manter diversidade não é exclusiva de ambientes variantes no tempo. Esse conceito já foi amplamente utilizado junto a meta-heurísticas populacionais para otimização, de forma a evitar a convergência prematura para ótimos locais. Em algoritmos evolutivos isso é feito geralmente por meio de compartilhamento de aptidão (do inglês *fitness sharing*) e métodos de nicho (Sareni e Krahenbuhl, 1998).

Compartilhamento de aptidão é uma metodologia aplicada principalmente em algoritmos genéticos em que a aptidão de cada indivíduo é influenciada não só pela função-objetivo mas, também, pelos indivíduos com aptidões parecidas. Nesse caso, como se quer manter diversidade, reduz-se o valor

dessa aptidão proporcional à distância em relação à aptidão dos indivíduos mais próximos. A fórmula de aptidão então se torna:

$$F(x) = \frac{f(x)}{m(x)}, \tag{15.23}$$

onde $F(x)$ é a aptidão calculada por meio do compartilhamento de aptidão, $f(x)$ é a aptidão original e $m(x)$ é a chamada contagem de nichos, a qual indica em quanto a aptidão será reduzida.

O cálculo da contagem de nichos é baseado em uma medida de distância capaz de apontar o quão próximos são dois valores de aptidão. Note aqui que não se está calculando a distância das soluções, mas sim do valor de suas funções-objetivo. A fórmula para contagem de nichos é dada por:

$$m(x) = \sum_{i=1}^{N} sh(dist(f(x), f(x_i)), \tag{15.24}$$

onde N é o tamanho da população, $dist(.,.)$ é a diferença entre dois valores de aptidão e $sh(.)$ é a medida de compartilhamento, a qual deve retornar 1 se os valores de aptidão dos dois indivíduos forem idênticos e 0 se a diferença entre eles for maior que um limiar. Uma forma comum de calcular essa medida é dada por:

$$sh(d) = \begin{cases} 1 - (d/\sigma_s)^{\alpha} & \text{se } d < \sigma_s \\ 0 & c.c. \end{cases}, \tag{15.25}$$

onde d é a distância entre os dois valores de aptidão, σ_s é o limiar de distância e α é um parâmetro que regula o formato da função de compartilhamento.

Apesar de inicialmente ser concebida para manter a diversidade de aptidão na população, essa técnica também pode ser utilizada para manter a diversidade de soluções, substituindo a medida de distância entre os valores de aptidão por uma medida de distância entre soluções, no espaço de busca.

Outra forma de nicho é conhecida como *crowding* e funciona de forma semelhante à substituição de indivíduos da população por novos indivíduos, descrito anteriormente no item de geração de diversidade. A diferença é que esse processo é feito constantemente e os indivíduos escolhidos para substituição são aqueles que têm uma similaridade alta com qualquer outro indivíduo. Essa escolha pode ser feita deterministicamente em toda a população ou em um subconjunto da população.

Finalmente, cabe mencionar também a técnica de *clearing*, similar ao compartilhamento de aptidão na formação dos nichos. Essa técnica preserva a aptidão original dos k melhores indivíduos de cada nicho, definidos pela distância entre eles em termos de valor de aptidão ou em termos de suas posições no espaço de busca, e reduz a aptidão de todos os outros. Dessa forma o algoritmo ainda mantém uma certa quantidade de bons indivíduos, mesmo que parecidos, para não perder o potencial de evolução, mas ao mesmo tempo desfavorece a manutenção de muitos indivíduos similares.

Um outro método que cabe mencionar é a chamada seleção termodinâmica (Mori et al., 1996, 1998) que procura manter a diversidade na etapa de seleção, ao invés de alterar a aptidão do indivíduo. Para tanto, esse método utiliza uma equação chamada de medida de energia livre, definida pela Eq. 15.26 para problemas de minimização. Os indivíduos da geração atual são então selecionados um a um para participarem da próxima geração de forma a minimizar o seguinte critério:

$$F = \langle E \rangle - TH, \tag{15.26}$$

onde F é o valor da energia livre da população, $\langle E \rangle$ é a média da aptidão da população, H é uma medida de diversidade e T, chamado de temperatura, é um controle da taxa de diversidade desejada para o problema.

O termo H, referente à diversidade, é calculado baseado na frequência com que cada indivíduo, ou parte dele, aparece na população. Para o caso de codificação binária, os autores definiram essa equação como sendo:

$$H = \sum_{i=1}^{D} H_i, \tag{15.27}$$

onde D é a dimensão do problema e H_i é dada na forma:

$$H_i = - \sum_{j \in \{0,1\}} p_j^i \log p_j^i, \tag{15.28}$$

com p_j^i sendo a frequência de ocorrência do valor j (0 ou 1, nesse caso) na posição i do vetor solução, calculada pela razão do número de ocorrências desse valor pelo tamanho atual da população já selecionada para a próxima geração.

Outras formas de manutenção de diversidade são propostas por Coelho et al. (2009) para o algoritmo evolutivo denominado evolução diferencial, onde a diversidade das soluções é monitorada e, dependendo do quanto cada indivíduo contribui para a diversidade, ele pode sofrer tipos diferentes de mutação.

Memória de Soluções

O uso de memória, em um primeiro momento, pode remeter apenas aos casos em que se tem um ambiente que muda de forma cíclica, ou seja, de tempos em tempos a solução ótima retorna a posições já visitadas. No entanto, o uso de memória pode ser útil também em casos em que o ótimo se aproxima de situações já experimentadas anteriormente, além de permitir manter a diversidade da população, ao preservar indivíduos que exerceram papéis relevantes num passado recente. O uso de memória pode ser classificado como explícita, quando se tem uma forma bem definida de como e quando armazenar e restaurar a memória, e implícita, usando, por exemplo, o conceito de dominância entre soluções alternativas.

Memória Explícita

Um método mais direto para fazer uso da memória, proposto por Ramsey e Grefenstette (1993), é utilizar uma base de soluções para armazenar os melhores indivíduos em diferentes situações. De tempos em tempos, o melhor indivíduo da população é armazenado juntamente com dados que permitam identificar o estado atual do espaço de busca. Toda vez que uma mudança no ambiente é percebida, o algoritmo é reinicializado com metade da população sendo obtida através da base de soluções, sendo que são selecionados aqueles indivíduos que foram melhores em situações mais parecidas com a atual. Esse tipo de metodologia implica que existe uma forma de comparar as várias configurações do ambiente dinâmico, o que não é viável na maioria dos problemas reais. Por exemplo, para saber se o ambiente retornou a uma configuração já experimentada no passado, podem ser exigidos mais recursos computacionais do que aqueles alocados para se buscar a solução ótima.

Uma abordagem semelhante foi adotada por Louis e Xu (1996), junto a problemas de escalonamento. O algoritmo armazena o melhor indivíduo de tempos em tempos e, ao detectar uma mudança no ambiente de otimização, o algoritmo é reinicializado com parte dos indivíduos pertencentes à memória. Essa metodologia tenta permitir que o algoritmo reinicializado contenha propostas de soluções que tiveram alta qualidade em instantes anteriores do processo de otimização. Espera-se que alguma dessas propostas de solução sirva como um bom ponto de partida para se encontrar o novo ótimo.

Vale citar também o modelo proposto por Trojanowski et al. (1997), que faz com que cada indivíduo tenha uma memória particular. Nessa memória, são armazenados os n últimos ancestrais do indivíduo

e, quando ocorre a mudança no ambiente, todos esses indivíduos e suas respectivas memórias são reavaliados. Após a reavaliação, o melhor entre o conjunto de ancestrais e o próprio indivíduo se torna o indivíduo atual.

Um outro tipo de memória explícita foi utilizado por de França et al. (2005) em um algoritmo de sistemas imunológicos artificiais. Esse algoritmo tem uma natureza multimodal, ou seja, localiza e mantém paralelamente múltiplos ótimos. Para esse algoritmo, toda vez que um indivíduo supostamente encontra algum ótimo local, ele é movido imediatamente para a memória e não passa mais por novas otimizações. Quando o ambiente apresenta uma mudança, os indivíduos da memória são reavaliados e copiados para a população principal, sofrendo novas otimizações em busca de novos ótimos locais. A ideia aqui é de que, ao armazenar múltiplos ótimos locais, o algoritmo está separando amostras distribuídas pelo espaço de busca e, quando ocorre uma mudança, mesmo que o novo ótimo global não esteja armazenado na memória, esse mecanismo irá inserir diversidade de forma distribuída, auxiliando no reinício do algoritmo.

Memória Implícita

Nos métodos de memória implícita, as técnicas mais comuns envolvem o uso de poliploides (Ng e Wong, 1995; Hadad e Eick, 1997) na codificação dos indivíduos. Poliploides são uma generalização do conceito de diploides em algoritmos genéticos (Goldberg e Smith, 1987). Esse tipo de representação conta com múltiplos cromossomos por indivíduo, cada qual associado a um diferente fenótipo. O fenótipo que será expresso é a combinação dos genes dominantes de cada um dos cromossomos do indivíduo. A determinação da dominância dos genes é feita através de um vetor que indica, para cada posição dos cromossomos, qual deles tem o gene dominante. Na hora da reprodução, é feita uma recombinação entre os múltiplos cromossomos de cada indivíduo, gerando indivíduos haploides. Esses indivíduos haploides são agregados através de um processo de seleção, gerando novos indivíduos poliploides.

O uso desse método para ambientes variantes no tempo permite que, ao considerar soluções compostas por múltiplos cromossomos em conjunto com um vetor de dominância, cada indivíduo acabe armazenando diversas soluções distintas, que podem representar memórias passadas de boas soluções. Essas soluções, ao mudar o ambiente, podem assim se tornar dominantes e ajudar na localização do novo ótimo.

Multipopulação

O conceito de multipopulação já vem sendo usado muito antes de ser empregado em propostas para ambientes variantes no tempo. O principal foco de abordagens multipopulacionais é tratar problemas multimodais – ou seja, executar a busca simultânea por múltiplos ótimos (Tsutsui et al., 1997). O uso desse conceito pode ser muito útil em ambientes dinâmicos, pois, se bem elaborado, vai conter implícita ou explicitamente todos os atributos considerados relevantes para algoritmos de otimização em ambientes dinâmicos, brevemente descritos nas seções anteriores.

Um exemplo de multipopulação para algoritmos genéticos em ambientes variantes no tempo foi proposto por Branke et al. (2000). Esta proposta recebeu a denominação de escoteiros auto-organizados (do inglês *self-organizing scouts*) e foi baseada num algoritmo genético de bifurcação (FGA, do inglês *forking genetic algorithm*) (Tsutsui et al., 1997). O processo do FGA funciona da mesma forma que um algoritmo genético comum, mas, quando a população converge para um ótimo local, ocorre uma bifurcação do espaço de busca, de modo que uma população fica responsável pela área onde o algoritmo convergiu e uma outra população por todo o resto do espaço. Esse processo gera dois espaços de busca disjuntos. Isso torna a busca pelo ótimo global mais ampla. No entanto, quando se trata de ambientes variantes no tempo, as divisões já realizadas no espaço de busca podem não refletir a atual configuração de regiões promissoras para a busca, fazendo com que algumas subpopulações fiquem

restritas a realizar uma busca onde já não existem boas soluções para serem localizadas. Outra consequência da frequente variação do ambiente é a possível ocorrência de muitas bifurcações do espaço de busca, criando uma dificuldade de alocação de recursos computacionais para um grande número de partições do espaço de busca.

No algoritmo de escoteiros auto-organizados, o autor tenta tratar esses aspectos do FGA com as seguintes alterações:

- O passo de mutação de cada sub-população é restrito à partição do espaço de busca em que ela se encontra.
- Todos os indivíduos são reavaliados a cada iteração.
- Cada sub-população é formada por um indivíduo central (o melhor indivíduo) e um raio de busca. Todos os indivíduos no raio de busca pertencerão a essa sub-população.
- Conforme o melhor indivíduo de cada população se altera, o centro da partição do espaço de busca o acompanha, promovendo partições dinâmicas do espaço de busca.
- É promovida uma redistribuição de indivíduos entre as sub-populações, de forma que as regiões mais promissoras tenham mais indivíduos e o tamanho total da população permaneça fixo.

Outra abordagem multipopulacional é o chamado algoritmo genético multinacional (do inglês *multinational genetic algorithm*) (Ursem, 2000). Nesse algoritmo, as populações são denominadas *nações* e cada uma delas tem um *governo* formado pelos indivíduos mais representativos (nesse caso, os indivíduos com melhor avaliação dentro da nação). Finalmente, esses governantes definem o sub-espaço de busca para a sua nação, conforme ilustrado na figura 15.3, onde são destacadas as curvas de nível da superfície de otimização. O agrupamento dos indivíduos é feito através de um procedimento de detecção de vales e montanhas ao longo da superfície de otimização. Dados dois pontos no espaço de busca, calcula-se a aptidão de algumas amostras situadas na linha formada entre eles. Um vale é detectado quando o *fitness* de uma dessas amostras é menor que aquele associado aos dois pontos. Esse procedimento é usado em três momentos nesse algoritmo: na migração dos indivíduos entre nações, na criação de novas nações e na união de duas nações.

De forma similar ao algoritmo de escoteiros auto-organizados, em que existe uma população-pai que controla a criação das outras populações, o algoritmo genético de balanço variável (SBGA, do inglês *shifting balance genetic algorithm*) (Wineberg e Oppacher, 2000) define uma população central que tem a responsabilidade de explorar as regiões promissoras, enquanto as sub-populações, denominadas *colônias*, têm a função de procurar novas regiões que ainda não foram exploradas. O grande diferencial desse algoritmo é em como ele mantém as sub-populações longe umas das outras. Para isso, ele usa uma medida de distância entre grupos, representada pela média da distância entre cada indivíduo de uma sub-população e todos os indivíduos pertencentes a uma outra sub-população. A partir desse cálculo, é determinada a porcentagem de sobreposição entre duas sub-populações. Essa informação é utilizada na fase de seleção de indivíduos para a próxima geração, a qual visa minimizar essa sobreposição.

População Auto-ajustável

Um problema encontrado com o uso de multipopulação está no tamanho total da população junto a cada problema de otimização. Se esse número for muito pequeno, existirão apenas poucos indivíduos por sub-população, o que pode fazer com que o algoritmo perca sua capacidade de exploração, a qual deveria ser o ponto forte da abordagem multipopulacional. Por outro lado, se esse número for muito grande, o custo de otimização por iteração tende a ser muito alto e, dessa forma, torna a busca pelo ótimo mais lenta, particularmente quando o ambiente varia a taxas elevadas.

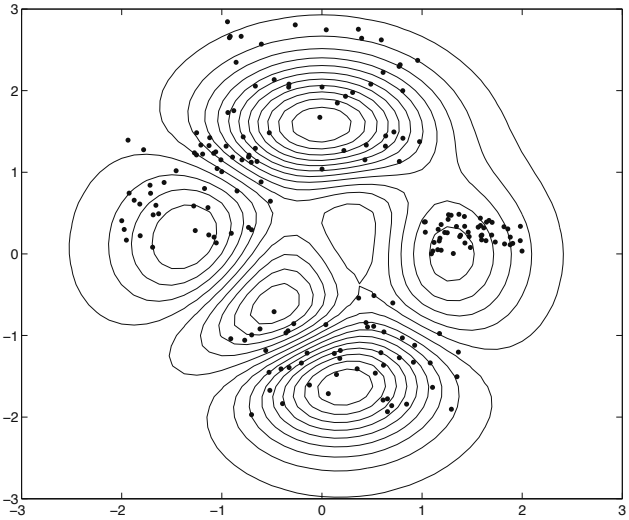


Figura 15.3: Ilustração do comportamento de um algoritmo genético multinacional em um espaço de busca bidimensional, onde são apresentadas as curvas de nível da superfície de otimização (informação esta não disponível para o algoritmo de busca). Cada vale detectado (ótimo local) contém uma população que se separa das demais. No caso da figura, são apresentadas quatro nações diferentes, sendo que a sub-população mais abaixo está identificando dois ótimos locais distintos.

A solução para isso é contar com um controle automático do tamanho da população, de forma que o algoritmo aumente o tamanho da população quando detectar a necessidade de explorar mais regiões em paralelo, ou diminua seu tamanho quando houver redundância ou necessidade de concentrar a busca em determinadas regiões. Um algoritmo que tem essa capacidade inerente em seu funcionamento é a já citada *aiNet* (de Castro e Von Zuben, 2001), que implicitamente conecta todos os seus indivíduos em uma rede, onde as ligações entre dois indivíduos podem ser identificadas pela distância entre eles. Note que, nesse algoritmo da área de sistemas imunológicos artificiais, cada indivíduo representa uma sub-população e, diante de uma situação controlada (ex.: convergência), os indivíduos que estão muito próximo de outros são eliminados e, em fases transitórias da busca, novos indivíduos são inseridos dentro da população. Dessa forma, cada região do espaço de busca tende a ser explorada por apenas uma sub-população de tamanho auto-ajustável. Para mais detalhes consulte o capítulo 6, que aborda os Algoritmos Imunoinspirados.

Outro ajuste automático do tamanho da população foi concebido em (Blackwell, 2007), onde um PSO multipopulacional define o número de populações em uma dada iteração t através da seguinte regra:

$$M(0) = 1, \tag{15.29}$$

$$M(t) = \begin{cases} M(t-1)+1 & \text{se } M_{livre} = 0 \\ M(t-1)-1 & \text{se } M_{livre} > n_{excesso} \end{cases}, \tag{15.30}$$

onde $M(.)$ é o número de populações na iteração corrente, M_{livre} é o número de populações livres, que não identificam ótimos ou são redundantes, e $n_{excesso}$ é um limiar para controlar o número desejado de populações livres.

Alteração da Mutação

Em algoritmos de computação evolutiva específicos para otimização em ambientes contínuos, a aplicação da mutação requer a definição de um tamanho de passo para cada iteração. Esse parâmetro, mesmo em ambientes estáticos, é difícil de ser determinado. Geralmente, é desejável um tamanho de passo grande quando ainda se está longe do ótimo, e que esse passo diminua com a proximidade do ótimo. No caso de ambientes variantes no tempo, essa questão é ainda mais desafiadora. Além da dificuldade de se definir uma medida de proximidade do ótimo, o ótimo pode se deslocar de forma arbitrária e repentina. Em (Rossi et al., 2007), foram propostas duas mutações auto-adaptativas para tratar desse problema, considerando que as mudanças seguem um padrão não-aleatório. Essas mutações utilizam uma técnica de detecção de movimento conhecida como filtro de Kalman (Kalman, 1960), utilizado geralmente para estimar o estado completo de um sistema dinâmico através de observações de um subconjunto de variáveis de estado e dispondo de um modelo do sistema. Uma dessas mutações gera um novo indivíduo através de uma distribuição gaussiana centrada na predição de onde o ótimo estará na próxima iteração, segundo o filtro de Kalman. A outra mutação modifica a mutação gaussiana, substituindo o valor estimado pela distribuição gaussiana existente por uma perturbação gaussiana de média 0 em torno da nova previsão do ótimo.

Estratégias para teste de desempenho

Diferente dos outros ambientes incertos estudados, o ambiente variante no tempo pode contar com alterações drásticas da superfície de otimização, incluindo até mudanças estruturais como a variação na dimensão do espaço de busca. Para tanto, diversas estratégias para teste de desempenho de um algoritmo em tais ambientes foram concebidas.

O ambiente de teste mais tradicional para problemas contínuos e, consequentemente, mais utilizado na maioria das aplicações de otimização em ambientes dinâmicos, está vinculado ao problema de picos móveis (Branke, 1999; Morrison e De Jong, 1999). Esse problema consiste em, dado um número pré-definido de picos (ótimos locais), encontrar e perseguir os picos enquanto eles mudam sua altura, largura e localização. Existem diversos parâmetros que podem ser definidos para tornar o problema mais desafiador, e um programa que cria instâncias desse ambiente de teste pode ser encontrado em *http://www.aifb.uni-karlsruhe.de/ jbr/MovPeaks* e *http://www.cs.uwyo.edu/ wspears/morrison/DF1-Overview.html*.

Para problemas multiobjetivos, foi proposta por Farina et al. (2004) uma sistemática bem completa para reutilizar problemas estáticos em ambientes variantes no tempo. Para tanto, esses ambientes foram divididos em quatro tipos:

- Os locais dos ótimos mudam, mas seus valores não.
- Tanto os locais dos ótimos quanto seus valores mudam com o tempo.
- Os valores mudam, mas os locais dos ótimos não.
- A nova superfície de otimização sofre mudanças de grande monta e arbitrárias, as quais impedem o estabelecimento de equivalências com a superfície de otimização anterior, como nos três casos anteriores.

Uma vez definidos esses tipos de ambientes dinâmicos, foram propostas as adaptações necessárias que devem ser feitas nas funções-objetivo do problema para que ele expresse o comportamento descrito.

Recentemente, no congresso *2009 IEEE Congress on Evolutionary Computation* (IEEE CEC 2009), uma nova metodologia para avaliação foi proposta com base em uma competição dentro do próprio congresso. Essa metodologia procurou ser mais completa e desafiadora que as outras já existentes, separando os casos de ambientes variantes no tempo em sete tipos:

- Deslocamentos pequenos nos ótimos.
- Deslocamentos grandes nos ótimos.
- Deslocamentos definidos por uma distribuição gaussiana.
- Mudanças provocadas por dinâmicas caóticas, logo imprevisíveis a médio e longo prazos.
- Mudanças periódicas e cíclicas.
- Mudanças periódicas e cíclicas com ruído.
- Mudanças estruturais, incluindo alteração na dimensão do problema.

Esses sete tipos foram incorporados em seis funções desafiadoras, originalmente estáticas. O desafio nessa competição é de não só o algoritmo estar preparado para reagir à mudança no ambiente, mas também ter um bom desempenho enquanto o ambiente não se altera. Esses problemas podem ser encontrados em *http://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC-09-Dynamic-Opt/CEC09-Dyn-Opt.htm*.

No caso de ambientes discretos, existem também alguns exemplos de ambientes de teste. Em (Mori et al., 1996), foi apresentado, além do algoritmo com seleção termodinâmica, uma proposta de problema da mochila dinâmico, que consiste em diminuir gradativamente o limite de peso da mochila. Como a solução ótima para esse problema geralmente fica próxima a essa restrição de limite, isso faz com que a solução atual torne-se infactível ao ocorrer uma mudança, tornando o problema ainda mais desafiador.

Outra proposta em ambientes discretos é a descrita por de França et al. (2006) para o Problema do Caixeiro Viajante (PCV). Nesse artigo, instâncias estáticas amplamente conhecidas para o PCV são alteradas periodicamente, de forma que uma porcentagem da distância entre cidades é alterada de forma controlada, utilizando um procedimento de λ-opt (Glover e Kochenberger, 2003).

Outras Aplicações e Técnicas

Problemas variantes no tempo são de particular interesse em filtragem de sinais com canais variantes no tempo (Krusienski e Jenkins, 2004; Junqueira et al., 2005, 2006). Esses filtros são utilizados principalmente em transmissão e recepção de sinais sem fio, onde ocorrem interferências temporais das mais variadas formas, como mudança de local das antenas e ruídos externos.

Uma variação do algoritmo de otimização por enxame de partículas (PSO) foi proposta para o problema de sensoriamento de odores (Jatmiko et al., 2006) em robôs móveis. O sensoriamento é afetado por ruídos externos e depende de fatores temporais, como a localização do robô em relação à fonte de odor e a intensidade do odor. O PSO também foi adaptado de outras formas, incluindo a incorporação de reinícios controlados (Carlisle e Dozier, 2000), o emprego de multipopulação (Blackwell e Branke, 2004; Liang e Suganthan, 2005) e mecanismos de manutenção de diversidade ou anti-convergência (Li et al., 2006). Também foram utilizadas técnicas de agrupamento e busca local para aumentar a capacidade de reação do PSO (Li e Yang, 2009).

Outro algoritmo inspirado na natureza que também foi adaptado para casos variantes no tempo é a *dopt-aiNet* (de França et al., 2005; de França e Von Zuben, 2009), que corresponde a uma variação da *aiNet* citada anteriormente, e que incorporou diversos mecanismos para geração e manutenção da diversidade, memória, multipopulação e tamanho auto-ajustável da população, além da alteração de seus operadores de mutação para obter um desempenho melhor durante a otimização.

Algoritmos inspirados em colônia de formigas (*Ant Colony Optimization* - ACO) (Guntsch et al., 2001; Korosec e Silc, 2009) também foram adaptados para otimização em ambientes dinâmicos, tanto para casos contínuos como discretos. As principais alterações nesse algoritmo são feitas diretamente

na chamada matriz de *feromônio*, que é empregada na definição da distribuição de probabilidades de soluções parciais que o algoritmo utiliza para construir uma solução. Uma vez que é detectada uma mudança no ambiente, o algoritmo ativa alguma forma de alteração nessa matriz, visando favorecer novamente a exploração do espaço de busca.

Adicionalmente, conceitos de multipopulação e parâmetros auto-adaptativos foram incorporados em um algoritmo de evolução diferencial (do inglês *differential evolution*) (Brest et al., 2009), sendo este o que obteve o melhor desempenho na competição do *IEEE CEC 09* mencionada na sub-seção anterior.

7. Conclusões

O mundo de hoje é mais complexo e dinâmico e os problemas de otimização de interesse prático tendem a incorporar esta complexidade e esta dinâmica. Além disso, há uma demanda acentuada por automação de processos de tomada de decisão, visando o atendimento de critérios de desempenho mais exigentes. Logo, ao se aplicar algoritmos de otimização para tratar problemas reais, é necessário lidar com incertezas que podem degradar o desempenho de tais algoritmos, o que geralmente não ocorre no caso de simulações experimentais e problemas teóricos bem definidos.

Em face dessa situação, foram descritas neste capítulo abordagens via algoritmos evolutivos para tratar diversos tipos de incertezas comumente encontrados. As iniciativas tendem a tornar o algoritmo mais robusto para lidar com ambientes incertos, geralmente permitindo atenuar os efeitos da incerteza. Cabe mencionar que não se busca eliminar esses efeitos, visto que em muitas situações alguma degradação de desempenho é inevitável.

Primeiramente, foi apresentado o caso de incertezas geradas por ruídos. Quando o algoritmo requisita uma avaliação da função-objetivo, é retornado um valor deslocado em relação ao valor real. Esse tipo de incerteza é comum quando existe imprecisão na medição e quando se trabalha com equipamentos que sofrem interferências externas, por exemplo. Foram apresentadas diversas soluções da literatura, algumas bastante simples e outras mais elaboradas, de forma a minimizar o efeito desse ruído durante a avaliação dos indivíduos da população. Algumas dessas soluções têm como objetivo melhorar a acuidade da medição sem prejudicar o desempenho computacional do algoritmo.

Em seguida, tratou-se de problemas robustos que, diferente dos problemas ruidosos, apresentam um intervalo de valores admissíveis para parâmetros do problema de otimização. Esse tipo de situação surge, por exemplo, em especificações técnicas vinculadas à qualidade de produtos em plantas industriais. Logo, buscam-se soluções de boa qualidade e que têm sua avaliação pouco alterada quando se consideram variações de parâmetros dentro de intervalos admissíveis. Embora tenha sido mostrado que situações desse tipo podem ser abordadas de forma similar aos caso dos problemas ruidosos, existem também soluções dedicadas, geralmente mais elaboradas e mais custosas computacionalmente, que visam aumentar a capacidade do algoritmo em encontrar soluções robustas.

A próxima incerteza apresentada está associada à dificuldade de se obter o valor da função-objetivo, requerendo o emprego de técnicas de aproximação. Esses cenários se dão quando o processo de avaliação da função-objetivo é custoso computacionalmente ou então em situações em que a função-objetivo não pode ser modelada matematicamente, de forma completa ou precisa. Amostragem de valores de parte da população dos algoritmos de otimização e o emprego conjunto de alguma técnica não-paramétrica de aproximação de funções ilustram bem o enfoque das metodologias de tratamento de incertezas neste caso.

Finalmente, como último caso de incerteza, foram apresentados os problemas de otimização em ambientes dinâmicos, em que se apresenta uma superfície de otimização que muda suas propriedades com o passar do tempo. Existem diversos tipos de variação dinâmica que podem ser considerados. Para solucionar esse problema, foram apresentadas características pertinentes que um algoritmo evolutivo deve exibir, visando detectar e reagir a mudanças. Foi mostrado também que alguns algoritmos já

possuem tais mecanismos de forma inerente. Também foi dado destaque ao compromisso existente entre o uso desses mecanismos e o custo computacional: se um procedimento levar muito tempo para ser realizado, a superfície de busca pode mudar nesse meio tempo, reduzindo o desempenho geral do algoritmo. Por fim, foram apresentados os resultados mais recentes e tendências futuras a serem investigadas.

O propósito fundamental desse capítulo foi o de apresentar ao leitor os diversos tipos de incertezas que podem ocorrer em problemas de otimização que admitem tratamento via algoritmos evolutivos e outras meta-heurísticas populacionais. Foi dada ênfase em procedimentos que permitem adaptar esses algoritmos aos diversos tipos de incertezas, visando atenuar os seus efeitos. Por se tratar de uma leitura introdutória, o leitor é convidado a consultar as referências bibliográficas contidas nesse capítulo para obter um conhecimento mais aprofundado sobre otimização em ambientes incertos, particularmente empregando algoritmos evolutivos.