Técnicas de aumento de eficiência para metaheurísticas aplicadas a otimização global contínua e discreta

Vinícius Veloso de Melo

Técnicas de aumento de eficiência para metaheurísticas aplicadas a otimização global contínua e discreta [1]

Vinícius Veloso de Melo

Orientador: *Prof. Dr. Alexandre Cláudio Botazzo Delbem*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

USP - São Carlos
Outubro/2009

[1] Trabalho realizado com apoio financeiro da Capes.

Dedicatória

A Deus, por permitir que eu chegasse até aqui.

À minha família, que soube entender a minha ausência desde que ingressei no Doutorado, até a conclusão desta Tese.

À minha namorada Grazieli Luiza Costa Carosio, por me apoiar nos momentos de ansiedade e estresse.

Agradecimentos

Ao meu orientador, Prof. Dr. Alexandre Delbem, cuja orientação foi de substancial importância para a conclusão deste trabalho.

Aos meus amigos e colegas durante o Doutorado, com quem passei momentos de descontração para aliviar um pouco a tensão.

Aos colegas do laboratório, que me ajudaram sempre que precisei.

Aos docentes do ICMC e aos integrantes da minha banca de Qualificação, que deram sugestões importantes para a realização deste trabalho de Doutorado.

Aos Professores Drs. Dorival Leão e Fernando Fernandes, pelas contribuições na área estatística.

Aos funcionários do ICMC, pela atenção.

À CAPES, pelo suporte financeiro essencial a este trabalho.

Resumo

Vários problemas do mundo real podem ser modelados como problemas de otimização global, os quais são comuns em diversos campos da Engenharia e Ciência. Em geral, problemas complexos e de larga-escala não podem ser resolvidos de forma eficiente por técnicas determinísticas. Desse modo, algoritmos probabilísticos, como as metaheurísticas, têm sido amplamente empregados para otimização global. Duas das principais dificuldades nesses problemas são escapar de regiões sub-ótimas e evitar convergência prematura do algoritmo. À medida que a complexidade do problema aumenta, devido a um grande número de variáveis ou de regiões sub-ótimas, o tempo computacional torna-se grande e a possibilidade de que o algoritmo encontre o ótimo global diminui consideravelmente. Para solucionar esses problemas, propõe-se o uso de técnicas de aumento ou melhoria de eficiência. Com essas técnicas, busca-se desenvolver estratégias que sejam aplicáveis a diversos algoritmos de otimização global, ao invés de criar um novo algoritmo de otimização ou um algoritmo híbrido. No contexto de problemas contínuos, foram desenvolvidas técnicas para determinação de uma ou mais regiões promissoras do espaço de busca, que contenham uma grande quantidade de soluções de alta qualidade, com maior chance de conterem o ótimo global. Duas das principais técnicas propostas, o Algoritmo de Otimização de Domínio (DOA) e a arquitetura de Amostragem Inteligente (SS), foram testadas com sucesso significativo em vários problemas de otimização global utilizados para *benchmark* na literatura. A aplicação do DOA para metaheurísticas produziu melhoria de desempenho em 50% dos problemas testados. Por outro lado, a aplicação da SS produziu reduções de 80% da quantidade de avaliações da função objetivo, bem como aumentou a taxa de sucesso em encontrar o ótimo global. Em relação a problemas discretos (binários), foram abordados problemas nos quais existem correlações entre as variáveis, que devem ser identificadas por um modelo probabilístico. Das duas técnicas de aumento de eficiência propostas para esses problemas, a técnica denominada Gerenciamento do Tamanho da População (PSM) possibilita a construção de modelos probabilísticos mais representativos. Com o PSM foi possível atingir uma redução de cerca de 50% na quantidade de avaliações, mantendo a taxa de sucesso em 100%. Em resumo, as técnicas de aumento de eficiência propostas mostram-se capazes de aumentar significativamente o desempenho de metaheurísticas, tanto para problemas contínuos quanto para discretos.

Palavras-chave: Técnicas de aumento de eficiência; Metaheurísticas; Otimização Global.

Abstract

Several real-world problems from various fields of Science and Engineering can be modeled as global optimization problems. In general, complex and large-scale problems can not be solved efficiently by exact techniques. In this context, Probabilistic algorithms, such as metaheuristics, have shown relevant results. Nevertheless, as the complexity of the problem increases, due to a large number of variables or several regions of the search space with sub-optimal solutions, the running time augments and the probability that the metaheuristics will find the global optimum is significantly reduced. To improve the performance of metaheuristics applied to these problems, new efficiency-enhancement techniques (EETs) are proposed in this thesis. These EETs can be applied to different types of global optimization algorithms, rather than creating a new or a hybrid optimization algorithm. For continuous problems, the proposed EETs are the Domain Optimization Algorithm (DOA) and the Smart Sampling (SS) architecture. In fact, they are pre-processing algorithms that determine one or more promising regions of the search-space, containing a large amount of high-quality solutions, with higher chance of containing the global optimum. The DOA and SS were tested with significant success in several global optimization problems used as benchmark in the literature. The application of DOA to metaheuristics produced a performance improvement in 50% of problems tested. On the other hand, the application of SS have produced reductions of 80% of the evaluations of the objective function, as well as increased the success rate of finding the global optimum. For discrete problems (binary), we focused on metaheuristics that use probabilistic models to identify correlations among variables that are frequent in complex problems. The main EET proposed for discrete problems is called Population Size Management (PSM), which improves the probabilistic models constructed by such algorithms. The PSM produced a reduction of 50% of function evaluations maintaining the success rate of 100%. In summary, the results show that the proposed EETs can significantly increase the performance of metaheuristics for both discrete and continuous problems.

Keywords: Efficiency-Enhancement Techniques; Metaheuristics; Global Optimization.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

ANN	*Artificial Neural Networks*
BD	*Bayesian Dirichlet*
BOA	*Bayesian Optimization Algorithm*
BIP	*Biased Initial Population*
BB	*Building Block*
C_c	*Combined Complexity Criterion*
CGA	*Compact Genetic Algorithm*
C_p	*Compressed Population Complexity*
CHA	*Continuous Hybrid Algorithm*
CA	*Cultural Algorithms*
DOE	*Design of Experiments*
DE	*Differential Evolution*
DOA	*Domain Optimization Algorithm*
EET	*Efficiency-Enhancement Technique*
EDA	*Estimation of Distribution Algorithm*
EA	*Evolutionary Algorithm*
ECS	*Evolutionary Clustering Search*
ECGA	*Extended Compact Genetic Algorithm*
GA	*Genetic Algorithm*
JPD	*Joint Probability Distribution*
KNN	*K-Nearest Neighbors*
LEM	*Learnable Evolution Model*
LS	*Local Search*
ML	*Machine Learning*
MPM	*Marginal Product Model*
MB	*Minimum Bias*
C_m	*Model Complexity*
NFLT	*No-Free-Lunch Theorem*

ODE	*Opposition-Based Learning*
PSO	*Particle Swarm Optimization*
PBIL	*Population Based Incremental Learning*
PSM	*Population Size Management*
PMBGA	*Probabilistic Model Building Genetic Algoritm*
PS	*Problem Size*
QODE	*Quasi-Opposition-Based Learning*
RFE	*Ratio of Function Evaluations*
RPOPS	*Ratio of POPulation Size*
RBL	*Rule Based Learner*
SRA	*Search-space Reduction Algorithm*
SS	*Smart Sampling*
SSDE	*Smart Sampling with Differential Evolution*
SMB	*Sporadic Model Building*
SVM	*Support Vector Machine*
UMDA	*Univariate Marginal Distribution Algorithm*

Lista de Artigos Publicados

Lista de artigos publicados como resultado desse doutorado:

1. MELO, V. V.; DUQUE, T. S. P. C.; DELBEM, A. C. B.. Efficiency Enhancement of ECGA Through Population Size Management. In: *International Conference on Intelligent Systems Design and Applications* (ISDA'09), 2009, Pisa, Italy. (aceito para publicação)

2. MELO, V. V.; DELBEM, A. C. B.. Using Smart Sampling to Discover Promising Regions and Increase the Efficiency of Differential Evolution. In: *International Conference on Intelligent Systems Design and Applications* (ISDA'09), 2009, Pisa, Italy. (aceito para publicação)

3. MELO, V. V.; DELBEM, A. C. B.. On Promising Regions and Optimization Effectiveness of Continuous and Deceptive Functions. In: *Proceedings of 2008 IEEE Congress on Evolutionary Computation* (CEC 2008), 2008, Hong Kong. 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), 2008.

4. MELO, V. V.; DELBEM, A. C. B.. Easy efficiency-enhancement techniques for the ECGA. In: *Brazilian Symposium on Neural Networks* (SBRN'08), 2008, Salvador. Brazilian Symposium on Neural Networks.: IEEE Computer Press, 2008. p. 69-74.

5. MELO, V. V.; DELBEM, A. C. B.; PINTO JUNIOR, D. L.; FEDERSON, F. M.. Improving Global Numerical Optimization using a Search-space Reduction Algorithm. In: *Genetic and Evolutionary Computation Conference* (GECCO'07), 2007, Londres. Genetic and Evolutionary Computation Conference. Londres: Association for Computing Machinery (ACM), 2007. p. 1195-1202.

6. MELO, V. V.; DELBEM, A. C. B.; PINTO JUNIOR, D. L.; FEDERSON, F. M.. Discovering Promising Regions to help Global Numerical Optimization Algorithms. In: *6th Mexican International Conference on Artificial Intelligence* (MICAI'07), 2007, Aguascalientes. 6th Mexican International Conference on Artificial Intelligence, 2007. p. 72-82.

1

Introdução

Em diversas áreas, como engenharia e ciência, existem inúmeros problemas que podem ser considerados como de otimização global. Para uma grande parte desses problemas, têm sido investigados algoritmos probabilísticos, heurísticas, metaheurísticas, dentre outros, devido à dificuldade de solucionar tais problemas de maneira computacionalmente eficiente para instâncias relativamente grandes.

Como exemplos de problemas de otimização global pode-se citar: projetos de antenas (Linden, 2002), roteamento em redes de comunicação (Arabshari, Gray, Kassabalidis, & A, Arabshari et al.), otimização de processos químicos (Bhaskar et al., 2000), reconhecimento de padrões (Bandyopadhyay et al., 2005; Lane & Gobet, 2005), dentre vários outros (Alba et al., 2009).

Um problema de *otimização global* em geral é modelado por uma função que possua ou não restrições de domínio, com variáveis definidas em $\mathbb{R}$. O problema pode possuir restrições de domínio, que equivalem aos valores que as variáveis podem assumir, determinados a partir de conhecimentos prévios sobre o problema em questão. Por exemplo, pode-se definir que a temperatura ideal de um forno para o derretimento de um determinado material deve estar entre 700^oC e 800^oC e a função de otimização deve encontrar a temperatura que derreta o material com o menor custo em termos de tempo e/ou consumo de energia. Em problemas do mundo real a restrição do domínio pode eventualmente facilitar a solução do problema, uma vez que restringe o espaço de busca. O *espaço de busca* ou *espaço de soluções*, é o espaço de todas as soluções possíveis, conjunto no qual se encontra a solução procurada.

A função (f) a ser otimizada é chamada *função meta* ou *função objetivo*, a qual é normalmente fornecida pelo usuário como um procedimento de caixa-preta (função

cujo comportamento é desconhecido para o algoritmo de otimização (Alba et al., 2009)) que retorna o valor calculado de uma solução informada como entrada ou argumento do procedimento. A apresentação geral desse tipo de problema é dada pela Equação 1.1:

$$min/max \;\; f(x), \;\; x = (x_1, x_2, x_3, ..., x_D) \in \mathbb{R}^D, \tag{1.1}$$

sujeita às restrições de domínio equivalentes a $x_i^{inf} < x_i < x_i^{sup}$, $1 \leq i \leq D$, tal que D é a quantidade de dimensões (variáveis) do problema em questão.

Em geral, as restrições de domínio (os limites x_i^{sup} e x_i^{inf}) que cada variável pode assumir, são definidas *a priori* pelo usuário, de acordo com suas necessidades. No caso das funções teste para algoritmos de otimização (funções de *benchmark*) encontradas na literatura, é comum que as restrições sejam determinadas de maneira a tornar o problema mais difícil, com o objetivo de avaliar a robustez do algoritmo de otimização.

Em um problema de minimização, dado um conjunto de soluções X de um problema de otimização f e uma vizinhança Vz, uma solução factível $x \in X$ é chamada ótima local com respeito a Vz (ou simplesmente ótimo local) se:

$$f(x) \leq f(g) \text{ para todo } g \in Vz(x).$$

Caso Vz contenha todas as possíveis soluções do problema, x é chamada ótimo global (Papadimitriou & Steiglitz, 1998).

Para problemas de otimização combinatoriais, cada variável é discreta, ao invés de real, limitando consideravelmente a quantidade de soluções possíveis. Contudo, problemas combinatoriais de larga-escala demandam alto custo computacional, e pode não haver técnicas específicas para melhoria de eficiência para esses problemas.

Se todas as possíveis soluções forem testadas tem-se um algoritmo de *busca exaustiva* (Nievergelt, 2000; Haupt & Haupt, 2004). Esse tipo de técnica é inadequado devido à grande quantidade de avaliações (cálculos da função objetivo) que pode ser requerida. Por exemplo, suponha um problema combinatorial binário do tipo caixa-preta que possua 100 variáveis. A quantidade de soluções desse problema é dada por $2^{100} = 1,267651e + 30$. Pode-se perceber que, mesmo se o tempo necessário para se avaliar uma única solução for da ordem de 1 nano-segundo (10^{-9}), avaliar todas as possibilidades seria impraticável, levando cerca de $4,019694e + 13$ anos.

Em otimização de problemas do tipo caixa-preta, a única maneira de garantir que o ótimo global foi encontrado seria calcular ou avaliar todas as possíveis soluções. Quando esse processo não é prático, como no exemplo acima, técnicas capazes de encontrar soluções próximas ao ótimo global têm sido investigadas. Essas técnicas efetuam uma quantidade consideravelmente menor de avaliações da função objetivo utilizando alguma

informação para evitar soluções que possuam determinada característica ou para orientar o algoritmo de busca a seguir determinado comportamento. Dentre as técnicas que efetuam esse tipo de procedimento estão as *técnicas heurísticas*, conforme mostrado na Seção 1.1.

1.1 Técnicas Heurísticas

Métodos heurísticos para otimização são desenvolvidos com o intuito de resolver rapidamente problemas complexos de larga-escala, para os quais métodos exatos necessitariam de tempo computacional impraticável. Para tanto, tais métodos utilizam algum conhecimento sobre o problema (ou sobre algum conjunto de soluções) com o objetivo de tomar atalhos e encontrar, rapidamente, soluções de alta qualidade, não necessariamente o ótimo global. Desse modo, heurísticas produzem equilíbrio entre o esforço computacional e a qualidade da solução que se busca. Para obter esse equilíbrio, é importante que se tenha algum conhecimento sobre o funcionamento do problema, de modo a explorar mais adequadamente o espaço de busca. Esse conhecimento orienta o algoritmo, reduzindo o espaço de busca investigado.

A Figura 1.1 exemplifica uma superfície de busca fictícia de um problema contínuo. Supondo que seja um problema de maximização, pode-se identificar nessa Figura três características importantes, apesar de não serem as únicas características possíveis de serem categorizadas:

1. Platôs: regiões praticamente planas. Dentro dessas regiões, um algoritmo de otimização possui dificuldade em identificar qual seria a melhor direção a seguir, visto que a superfície possui valores muito similares. O algoritmo passa, então, a realizar uma busca cega;

2. Ótimos locais: quando o algoritmo identifica um ou mais pontos que possuem resultados de maior qualidade, a busca passa a ser realizada nas proximidades desses pontos. Há problemas práticos nos quais ótimos locais são considerados satisfatórios, ou seja, soluções de alta qualidade;

3. Cumes: esses são ótimos locais acentuados. Em ótimos locais comuns, pequenas mudanças nas coordenadas da superfície produzem pequenas mudanças no valor da função a ser otimizada. Esse comportamento possibilita uma melhor exploração da região. Por outro lado, em regiões do tipo cume, pequenas mudanças nas coordenadas produzem mudanças bruscas no valor da função. Desse modo, se a qualidade das soluções piorar, o algoritmo tende a evitar tais mudanças e a ficar aprisionado nos cumes, que podem ser ótimos locais indesejados ou inadequados.

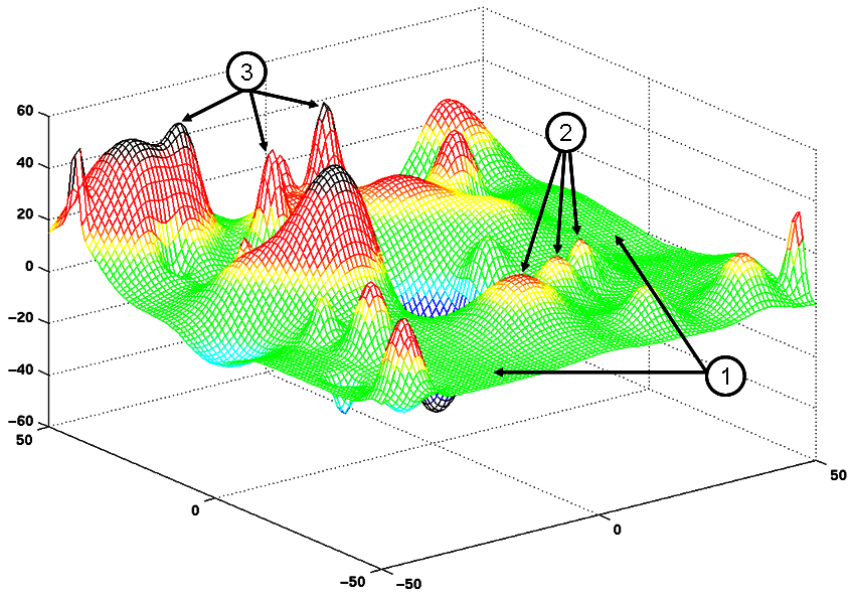


Figura 1.1: Exemplo de superfície de busca de um problema contínuo: 1) Platôs, 2) Ótimos locais e 3) Cumes.

Nesse contexto de platôs, ótimos locais e cumes fica evidente que uma heurística restringindo o espaço de busca, mesmo que de maneira probabilística, pode evitar que o algoritmo atinja os cumes e encontre o ótimo global. Além disso, algoritmos que realizam uma análise local do comportamento da superfície podem direcionar um algoritmo para soluções sub-ótimas. Esse efeito torna-se mais acentuado se o espaço de busca for grande e/ou o número de platôs, ótimos locais e cumes for maior. Como heurísticas e outras técnicas de busca local costumam ser aplicadas para se encontrar o ótimo de uma vizinhança, estes tornam-se inadequados para serem usados em problemas de otimização global.

Nas últimas décadas, diversas pesquisas têm sido realizadas no desenvolvimento de algoritmos simples e genéricos (Osman & Kelly, 1996; Siarry & Michalewicz, 2008), buscando resolver problemas complexos e de larga-escala para os quais não há heurísticas adequadas e mesmo algoritmos de busca local são incapazes de encontrar soluções satisfatórias. Dentre as principais técnicas que têm sido pesquisadas para superar esse desafio estão as *metaheurísticas*, descritas na Seção 1.2.

1.2 Metaheurísticas

De maneira similar a outras técnicas de otimização, as metaheurísticas iniciam com uma ou mais soluções iniciais que são melhoradas até ser atingido um critério de parada definido pelo usuário. As metaheurísticas que utilizam um conjunto (*população*) com mais de uma

solução a cada etapa do processo de otimização têm sido chamadas de metaheurísticas populacionais.

Uma estratégia de busca simples e eficiente é a exploração do espaço de busca em pontos bem espalhados na etapa inicial do processo. Essa estratégia possibilita ao algoritmo mapear o espaço de busca e determinar regiões que poderão ser mais bem exploradas em iterações subseqüentes. Nessas iterações, novos pontos de qualidade cada vez melhor tendem a ser amostrados mais próximo do ótimo global.

Muitas metaheurísticas populacionais seguem o padrão de funcionamento da Computação Evolutiva do tipo Darwiniana (Julstrom, 1999). Esse modelo de computação pode ser visto como um processo de busca paralela de propósito geral. Basicamente, um conjunto inicial de pontos (indivíduos) no espaço de busca é gerado aleatoriamente. Após a avaliação de cada indivíduo pela função objetivo, um subconjunto dos melhores é selecionado utilizando algum critério. Tal subconjunto é modificado por operadores de modificação como *mutação* ou *recombinação.* Em geral, esses operadores trabalham de maneira probabilística, sem uso de qualquer heurística sobre o domínio do problema que garanta a geração de melhores indivíduos.

Os novos indivíduos gerados são avaliados e inseridos na população. O procedimento comum é que os melhores indivíduos sejam selecionados para a próxima geração, enquanto que os indivíduos de baixa qualidade sejam eliminados. Essa etapa de um algoritmo de Computação Evolutiva corresponde, na Natureza, ao processo de sobrevivência dos indivíduos melhor adaptados, que transmitem seus genes para seus descendentes. Esse processo de seleção dos melhores indivíduos, modificação e substituição dos demais continua até que uma condição de parada seja satisfeita. Mais detalhes sobre essa classe de algoritmo são apresentados na Seção 2.1.

Uma das principais desvantagens das metaheurísticas, mas não somente delas, é o fato de poderem terminar sua execução antes de alcançarem uma solução aceitável. Esse problema ocorre devido a uma variedade de fatores como: uso de uma solução ou um conjunto de soluções iniciais de baixa qualidade, configuração inadequada de parâmetros do algoritmo e, principalmente, operadores ineficazes para determinar o próximo conjunto de soluções.

Os operadores modificadores de soluções desses algoritmos possuem dificuldade em determinar subconjuntos de variáveis de um problema que possuem forte correlação que, portanto, não poderiam ter seus valores modificados de maneira independente. Mais recentemente, metaheurísticas populacionais capazes de detectar tais relações têm sido investigadas, apresentando significativo aumento de desempenho, conforme mostra a Seção 1.3.

1.3 Algoritmos de Estimação de Distribuição

Variáveis que se relacionam fortemente são fundamentais na composição de soluções parciais de um problema maior. Pode-se dizer que tais variáveis formam blocos de construção (BBs, do inglês *Building Blocks*, ver Seção 6.1) associados a subproblemas que comporiam o problema original (Goldberg, 2002). Várias metaheurísticas comumente utilizadas na literatura ignoram essas interações, tratando cada variável como independente. Desse modo, as soluções parciais podem ser desfeitas pelos operadores modificadores responsáveis por gerar novas soluções a partir das atuais. Por esse motivo, tem-se investigado técnicas capazes de manter esses BBs. Assim, uma nova classe de algoritmos de otimização global, denominada Algoritmos de Estimação de Distribuição (EDAs, do inglês *Estimation of Distribution Algorithms*, ver Seção 2.3.4) tem recebido significativa atenção de pesquisadores. EDAs com características similares ao de Algoritmos Genéticos (GAs, do inglês *Genetic Algorithms*) são comumente chamados de PMBGAs (do inglês, *Probabilistic Model Building Genetic Algorithms*) (Sastry et al., 2004). Esses algoritmos apresentam uma taxa de sucesso relativamente alta na obtenção do ótimo global para uma baixa quantidade de avaliações da função objetivo.

A estratégia utilizada por EDAs é a construção de modelos de relacionamentos entre variáveis e a posterior geração de novas soluções a partir desse modelo, denominado processo de reamostragem. Esses modelos de correlação de alta qualidade devem ser construídos a partir de uma quantidade relativamente grande de soluções, permitindo que a reamostragem oriente o algoritmo para regiões mais promissoras do espaço de busca.

Técnicas de melhoria de eficiência podem ser elaboradas com base na capacidade de direcionar o algoritmo mais rapidamente para regiões promissoras. Para esse tipo de problema, uma técnica de determinação de regiões promissoras pode ser responsável pela construção de modelos de alta qualidade reduzindo o custo computacional para a resolução do problema.

Ao se utilizar esse tipo de estratégia, é interessante que haja um equilíbrio entre Qualidade e Custo Computacional do modelo. A quantidade de avaliações da função objetivo realizadas pelo algoritmo e o tempo computacional do processo de otimização como um todo, são utilizados para avaliar o desempenho do algoritmo de otimização. Por esse motivo, é importante recordar que quanto maiores o domínio do problema e a quantidade de variáveis, maior deverá ser a quantidade de soluções amostradas. Com isso, o modelo tende a ser mais complexo e requerer maior tempo para ser construído.

1.4 Motivação

O aumento da eficiência computacional de metaheurísticas populacionais pode possibilitar que seja encontrado, em tempo aceitável, o ótimo global de problemas complexos de larga-escala, isto é, pode tornar problemas intratáveis em tratáveis. Para reduzir o tempo computacional de algoritmos de otimização pode-se orientar a busca, por exemplo, utilizando relações entre variáveis obtidas de dados amostrais capazes de direcionar o algoritmo para soluções de melhor qualidade com uma chance maior do que o uso de operadores modificadores aleatórios. Outra possibilidade é encontrar regiões mais promissoras do espaço de busca (por exemplo, com alta concentração de pontos com valores relativamente bons para a função objetivo) e realizar uma exploração mais minuciosa dentro dessas regiões.

Uma região promissora (Yan & Changrui, 2007) pode ser definida como um subconjunto do espaço de busca de um problema, que contenha uma substancial quantidade de soluções de alta qualidade. Pode-se dizer que um subconjunto de soluções S extraídas da população X dentro de uma vizinhança Vz possui qualidade acima da média de X. Isso pode significar que essa região deve ser melhor explorada pois existe maior probabilidade de conter o ótimo global do que uma região com soluções de baixa qualidade.

Durante o processo de otimização, metaheurísticas geram soluções de qualidade cada vez maior. Desse modo, o processo seria guiado naturalmente a regiões promissoras. Existe, porém, a necessidade de identificar o quanto uma região é promissora, visto que uma busca minuciosa pode resultar na realização de muitas avaliações da função objetivo.

Por outro lado, a determinação da própria região promissora pode requerer uma quantidade alta de avaliações da função objetivo. Em diversos problemas do mundo real que exigem simulação ou uma elevada quantidade de cálculos complexos, a avaliação/cálculo da função objetivo é, geralmente, a etapa de maior custo computacional. Assim, quanto menor a necessidade de cálculos dessa função para se encontrar o ótimo, melhor é o algoritmo. Dada a grande quantidade de soluções que podem ser avaliadas por uma metaheurística e a grande quantidade de iterações necessárias para se encontrar essa região promissora, técnicas que possam reduzir esse esforço são, em geral, bem recebidas pela comunidade científica.

Metaheurísticas capazes de identificar regiões promissoras em menos iterações têm apresentado resultados relevantes. Como exemplos importantes dessas técnicas tem-se o CHA (do inglês, *Continuous Hybrid Algorithm*) (Chelouah & Siarry, 2003) e o ECS (do inglês, *Evolutionary Clustering Search*) (Oliveira, 2004). O CHA é um algoritmo evolutivo padrão. Quando a população torna-se muito parecida, ou seja, a distância Euclidiana entre o melhor indivíduo e o outro mais distante deste for menor do que um

certo *raio*, é definida uma região promissora em torno do melhor indivíduo e iniciada uma busca mais intensa nessa região. Ao término da busca intensiva, o processo do CHA é finalizado, mesmo que o ótimo global não tenha sido encontrado.

De modo a melhorar o CHA, foi desenvolvido o ECS. No ECS, um processo de agrupamento iterativo trabalha simultaneamente com um algoritmo de Computação Evolutiva, contabilizando as operações (seleção ou atualização de indivíduos) ocorridas em regiões do espaço de busca e identificando aquelas que merecem especial interesse. Em outras palavras, o processo de agrupamento (do inglês *clustering*) (Mitchell, 1997) identifica grupos de indivíduos com similaridades, significando um padrão predominante. Identificadas tais regiões, uma busca intensiva é iniciada dentro delas. Se nenhum critério de parada for alcançado, o ECS continua o processo de busca.

Neste trabalho, investiga-se o desenvolvimento de métodos capazes de identificar tais regiões promissoras antes da execução de uma metaheurística. Dessa maneira, propõe-se métodos genéricos, não atrelados a um algoritmo específico. No caso de problemas contínuos, um dos métodos pertence à classe de algoritmos de redução do espaço de busca (SRA, do inglês *Search-space Reduction Algorithms*) (Srinivas & Patnaik, 1991; Chen & Smith, 1999; Guo & Matta, 2003). Um SRA é uma técnica para determinar uma região promissora antes de utilizar um algoritmo de busca propriamente dito, ao invés de detectá-la durante o processo de otimização. Obtida a região promissora, pode-se gerar o conjunto inicial de soluções dentro de limites definidos pelo SRA (a região promissora), onde existe uma maior possibilidade de se encontrar o ótimo global.

Por outro lado, em problemas discretos (binários), o processo de redução do espaço de busca torna-se mais complexo, visto que não existe um intervalo de valores entre dois limites e sim apenas um conjunto restrito de possibilidades (duas) de valores para cada variável do problema. Portanto, ao invés de limitar o espaço, utiliza-se a probabilidade da ocorrência de cada um desses valores. Com isso, a identificação da região promissora é substituída pela determinação de uma probabilidade para cada valor que a variável pode assumir. Para problemas binários, uma probabilidade maior do que 0,5 pode indicar uma chance maior da variável ser 1 do que ser 0. No caso de problemas com BBs, determina-se que uma dada combinação de valores de variáveis (uma instância do BB) possui probabilidade de ocorrência significativamente maior do que outras. É importante destacar que o objetivo dessas técnicas é reduzir a quantidade de avaliações necessárias para se encontrar o ótimo global nos problemas de teste (*benchmark*) de alta complexidade, mantendo a taxa de sucesso em 100%. Vale lembrar que essa taxa não é necessariamente mantida em problemas do mundo real.

Para problemas com BBs, pode ser mais interessante a aplicação de GAs competentes, que são PMBGAs (ver Seção 1.3) capazes de resolver, em um tempo polinomial, problemas que são intratáveis para métodos tradicionais de otimização. Apesar de possuírem maior

desempenho do que metaheurísticas que não utilizam construção de modelos de correlações entre as variáveis do problema, os GAs competentes costumam empregar populações de tamanho relativamente maior, aumentando, nesse sentido, o seu custo computacional. Além disso, a própria construção do modelo é uma etapa computacionalmente complexa. Para superar essas desvantagens, diversas Técnicas de Melhoria de Eficiência (EETs, do inglês *Efficiency-Enhancement Techniques*) têm sido propostas (Sastry et al., 2004; Duque et al., 2008a; Melo & Delbem, 2008a; Pelikan et al., 2008).

As EETs podem transformar problemas tratáveis em problemas práticos, isto é, em problemas para os quais um GA competente encontraria soluções rapidamente, mesmo para problemas de larga-escala. O aspecto principal das EETs é o fato de não serem específicas para um determinado algoritmo, podendo ser utilizadas em EDAs de uma forma geral. Essa flexibilidade é importante pois estabelece que uma EET não é apenas um ajuste no método de otimização e sim uma estratégia real para melhoria do desempenho.

Como exemplos de conceitos aplicados em EETs, pode-se citar:

1. Uso de funções de objetivos aproximadas, ao invés de avaliar uma solução utilizando uma função de alta complexidade (Sastry, 2001);

2. Busca local interna aos BBs. Identificado um BB, pode-se testar todas as soluções parciais e utilizar a que der melhor resultado (Sastry & Goldberg, 2004);

3. Construção esporádica do modelo de correlações entre variáveis, ao invés de construir um modelo a cada geração (Pelikan et al., 2008), visto que um modelo de alta qualidade pode ser mantido por mais tempo sem atualizações.

Com os estudos sobre identificação de regiões promissoras e EETs buscou-se explorar os modelos mais simples e computacionalmente eficientes que produzissem um ganho relativamente alto de desempenho, conforme apresentado na Seção 1.5.

1.5 Métodos Propostos

A base teórica para as técnicas e abordagens aqui propostas advém da investigação de técnicas de otimização global, EDAs, Técnicas Heurísticas para identificação de regiões promissoras, EETs, Planejamento de Experimentos (DoE, do inglês *Design of Experiments*) (Box et al., 1978; Montgomery, 1997) e Aprendizagem de Máquina (ML, do inglês *Machine Learning*) (Mitchell, 1997).

Para determinação de regiões promissoras, duas técnicas foram desenvolvidas. Como essas técnicas atuam em uma etapa de pré-processamento, é possível que sejam empregadas com diversas metaheurísticas, sem necessidade de adaptações.

A primeira técnica, denominada Algoritmo de Otimização de Domínio (DOA, do inglês *Domain Optimization Algorithm*) (Melo et al., 2007b,a) utiliza conceitos de DoE para eliminar do espaço de busca do problema com menor probabilidade de conter o ótimo global, orientando o algoritmo a regiões mais promissoras. Os testes utilizando DOA em conjunto com três metaheurísticas mostraram que é possível encontrar regiões promissoras e que essas podem melhorar o desempenho de metaheurísticas. No caso do DOA, uma melhoria significativa foi obtida para cerca de 50% dos testes realizados.

A segunda técnica proposta é uma arquitetura genérica para determinação de uma ou mais regiões promissoras. Tal arquitetura foi denominada Amostragem Inteligente (SS, do inglês *Smart Sampling*). A SS emprega técnicas de ML para verificar se uma nova solução apresenta características de uma solução de alta qualidade de modo a evitar que soluções de baixa qualidade sejam avaliadas pela função objetivo. O uso iterativo do SS retorna um conjunto de soluções de alta qualidade, separadas no espaço de busca em uma ou mais regiões promissoras. A arquitetura permite que seus módulos sejam substituídos por outras técnicas de mesma finalidade, de modo a torná-la mais eficiente para lidar com outros tipos de problemas, caso necessário.

A SS foi testada em conjunto com a metaheurística DE (ver Seção 2.3.3) que foi mais beneficiada pela aplicação do DOA. Foram realizadas avaliações em 62 casos de teste de problemas de otimização global contínua e os resultados comparados com outros algoritmos avançados de otimização global, obtendo resultados significativamente superiores em 51 casos, ou cerca de 82%.

Para trabalhar com problemas binários em EDAs, foram desenvolvidas duas EETs denominadas Gerenciamento do Tamanho da População (PSM, do inglês *Population Size Management*) e População Inicial Tendenciosa (BIP, do inglês *Biased Initial Population*). Ambas as técnicas têm como objetivo viabilizar melhores modelos probabilísticos capazes de, implicitamente, orientar a busca para regiões mais promissoras do espaço de busca. Com o PSM foi possível reduzir o tamanho da população pela metade, bem como a quantidade final de avaliações da função objetivo. Com a BIP, foi possível identificar comportamento de redução linear na quantidade de gerações, à medida que se aumenta o valor da tendência.

A análise desses resultados não é, exatamente, uma tarefa trivial. Técnicas de otimização são altamente dependentes do conjunto de soluções iniciais. Um conjunto inicial distante do ótimo global pode fazer com que a técnica necessite de mais tempo para atingi-lo, ou seja, mais avaliações da função objetivo. Isso reduz a probabilidade do ótimo global ser encontrado. Por outro lado, se o conjunto inicial possuir soluções mais próximas ao ótimo global, as chances de que ele seja encontrado aumentam, além de reduzir a quantidade de avaliações necessárias.

Dada essa dependência, a análise da qualidade de metaheurísticas e técnicas de otimiza-

ção em geral é realizada a partir de métricas calculadas sobre várias execuções de uma mesma técnica, com um mesmo conjunto de parâmetros, sobre um mesmo problema. Porém, com diferentes conjuntos de soluções iniciais.

Para cada tipo de problema, seja ele contínuo ou discreto, existe um conjunto de métricas que podem ser utilizadas. Não existem normas de como as comparações entre metaheurísticas devem ser realizadas, nem quais testes estatísticos devem ser aplicados. Em geral, o pesquisador escolhe as métricas mais comumente empregadas na literatura, ou aquelas utilizadas no trabalho de outro pesquisador, com o qual ele deseja comparar seus resultados. Além disso, a aplicação de métricas de maneira inadequada, bem como a falta de comparações por testes estatísticos, podem dificultar uma comparação adequada entre trabalhos distintos.

Buscando realizar comparações mais confiáveis para problemas de larga-escala, propõe-se também uma métrica baseada na Redução do Espaço de Busca para comparação de desempenho de metaheurísticas. Essa métrica possibilita comparar adequadamente metaheurísticas com características relativamente bem diferentes da área de otimização global.

1.6 Organização do Trabalho

Este texto está organizado da seguinte maneira:

- No Capítulo 2 é apresentada uma visão geral de metaheurísticas populacionais para otimização global, bem como conceitos necessários para seu entendimento, além de uma introdução de três metaheurísticas comumente utilizadas na literatura;

- No Capítulo 3 são apresentados os resultados do estudo sobre métricas para comparação de metaheurísticas, bem como as métricas propostas e a metodologia empregada para comparação;

- No Capítulo 4 são apresentados conceitos e métodos relacionados à identificação de regiões promissoras em problemas de otimização global. Além disso, são apresentados resultados sobre dois estudos realizados para avaliar a importância e o potencial da pesquisa de técnicas de busca por regiões promissoras: o primeiro baseado no princípio de redução do espaço de busca e o segundo fundamentado na análise de tendência da amostra;

- No Capítulo 5 é apresentada a arquitetura de Amostragem Inteligente (SS), baseada em técnicas de ML determinar regiões promissoras do espaço de busca;

- No Capítulo 6 são introduzidos os GAs competentes, capazes de lidar com problemas discretos em que existem correlações entre variáveis;

- No Capítulo 7, os estudos realizados envolvendo EETs para EDAs são descritos. Além disso, são propostas duas técnicas de melhoria de eficiência;
- O Capítulo 8 apresenta os resultados dos experimentos referentes às técnicas de determinação de regiões promissoras e de melhoria de eficiência propostas nos Capítulos 5 (para problemas contínuos) e 7 (para problemas discretos, binários);
- Por fim, no Capítulo 9, são apresentadas as conclusões, destacando as maiores contribuições para a melhoria da eficiência de algoritmos de otimização global, bem como são sugeridos trabalhos de pesquisa futuros.

2

Metaheurísticas Populacionais para Otimização Global

Neste Capítulo é apresentada uma visão geral de metaheurísticas populacionais para otimização global, bem como conceitos necessários para seu entendimento e uma análise crítica. Após isso, são introduzidas brevemente quatro metaheurísticas comumente utilizadas para otimização global: algoritmos genéticos (Seção 2.3.1), otimização por enxame de partículas (Seção 2.3.2), evolução diferencial (Seção 2.3.3) e algoritmos de estimação de distribuição (Seção 2.3.4). Por fim, a Seção 2.4 apresenta as considerações finais do Capítulo.

2.1 Metaheurísticas para Otimização Global

Uma metaheurística é um conjunto de conceitos que podem ser utilizados para definir métodos de otimização aplicáveis a um extenso conjunto de problemas. Em outras palavras, uma metaheurística pode ser vista como uma arquitetura genérica de algoritmos que podem ser aplicados a diferentes problemas de otimização com relativamente poucas modificações, capazes torná-los adaptados a um problema específico[1].

Por ser um algoritmo mais geral do que algoritmos de otimização numérica (Nocedal & Wright, 2006), evita-se o uso de técnicas de resolução de problemas específicos que façam uso de conhecimento prévio do problema (heurísticas). Essa característica é importante pois possibilita o uso das metaheurísticas na resolução de problemas do tipo *caixa-preta*.

[1] http://www.metaheuristics.org/index.php?main=1

Em um problema do tipo caixa-preta, o usuário da técnica não tem acesso à função objetivo a ser resolvida. Apenas informa uma solução e obtém a resposta calculada. Portanto, a construção de uma heurística para solução mais eficiente do problema torna-se difícil. Desse modo, metaheurísticas de otimização global são aplicadas a problemas para os quais não há um método exato ou heurístico satisfatório capaz de resolvê-los, ou quando não é viável aplicar tal método, por exemplo se o tempo de computação for excessivamente grande. Apesar disso, metaheurísticas não garantem que o ótimo global será encontrado.

Em geral, metaheurísticas são aplicadas a problemas de otimização combinatorial ou problemas contínuos com função objetivo multimodal, nos quais existe uma grande quantidade de soluções de alta qualidade, em relação a uma vizinhança, (podendo ser ótimos locais) e poucos ou apenas um ótimo global. Métodos heurísticos costumam ser eficientes em encontrar ótimos locais, não sendo aplicados globalmente devido à dependência de informações do domínio do problema para restringir o espaço de busca.

Existem métodos de otimização que utilizam heurísticas como informação de derivadas parciais e gradientes para identificar a direção da busca (Dennis & Schnabel, 1996; Hyvarinen et al., 2001). Essa informação é extremamente útil quando o algoritmo encontra-se em uma região tendenciosa, isto é, que possui ótimos locais ou cumes (ver Seção 1.1). Nessas circunstâncias, tais métodos são capazes de encontrar o melhor ponto dessa região, ou chegar bem próximo dele. Esses métodos são denominados otimizadores locais (Xu, 2003; Alexandrov et al., 1998). Como o próprio nome sugere, não é garantido que otimizadores locais encontrem o ótimo global. Para esse fim, diversas metaheurísticas têm sido desenvolvidas (Siarry & Michalewicz, 2008).

2.2 Generalização do Problema e Técnicas de Solução

O objetivo da otimização combinatorial é encontrar um objeto matemático discreto (como uma cadeia de *bits* ou permutação desses) que maximiza (ou minimiza) uma função arbitrária especificada pelo usuário. Esses objetos são comumente chamados *estados* ou *soluções* do espaço de busca. A natureza desses estados e o próprio espaço de busca são normalmente específicos do problema em questão.

Algumas metaheurísticas mantém um único estado atual durante todo o processo de otimização, podendo substituí-lo por um novo a cada iteração. Essa alteração de estado (PB, Passo Básico) é denominada um *movimento* ou uma *transição de estado*. O movimento pode ser uma escalada ou uma descida, dependendo se o valor da função objetivo aumenta ou diminui. O novo estado pode ser construído sem informação *a priori* por um procedimento *gerador* ou derivado do estado atual por um procedimento *modificador*. Em geral, procedimentos geradores e modificadores são de natureza probabilística. Se

o novo estado preservar características do estado atual, o que ocorre com procedimentos modificadores, ele pode ser denominado *estado vizinho*. Desse modo, o conjunto de novos estados que podem ser produzidos pelo procedimento modificador corresponde à *vizinhança* do estado atual.

Diversas metaheurísticas mantém, em vez de um único estado atual, um conjunto atual que contém vários estados, também conhecido como *população*, dando origem às metaheurísticas populacionais. Nesse contexto, uma alteração de estado pode inserir ou remover estados dessa população, gerando um novo conjunto atual. Podem ser utilizados procedimentos para selecionar os estados a serem descartados com o objetivo de gerar os novos estados a serem adicionados. Esses últimos podem ser gerados combinando características de dois ou mais estados da população.

Uma característica importante das metaheurísticas é a utilização de um ou mais critérios de parada. Esses critérios são adotados para evitar a execução da metaheurística por um tempo muito longo. Como o espaço de busca normalmente é muito grande, se não existir um critério de parada algumas metaheurísticas poderão verificar todas as possibilidades e usar métodos heurísticos somente para escolher a ordem da pesquisa. Nesse sentido, visto que a metaheurística poderá verificar todas as soluções ela sempre encontrará o ótimo global, se o tempo de execução for longo o suficiente (Rudolph, 1999; Nelles, 2000).

Análise Crítica de Metaheurísticas

Pesquisadores contrários ao uso de metaheurísticas (Wolpert & Macready, 1995, 1997) afirmam que o objetivo geral da metaheurística não pode ser realizado. Tal objetivo, que seria a otimização eficiente de uma função caixa-preta arbitrária, pode ser evitado de maneira relativamente fácil, visto que para qualquer metaheurística M pode-se construir uma função objetivo f do tipo "agulha-no-palheiro" que poderá forçar M a vasculhar o espaço de busca inteiro para encontrar o ótimo global. Nesse sentido, o "teorema de nenhum-almoço-grátis" (NFLT, do inglês *No-Free-Lunch Theorem*) (Wolpert & Macready, 1995), cuja representação gráfica pode ser vista na Figura 2.1, afirma que, sobre o conjunto de todos os problemas matematicamente possíveis, cada metaheurística terá desempenho mediano como qualquer outra.

Desse modo, no melhor caso, uma determinada metaheurística pode ser eficiente apenas para classes restritas de funções objetivo. Fora dessas classes, ou estão excluídas a maioria das aplicações de interesse no mundo real ou o problema é ameno o suficiente para a aplicação de métodos de solução específicos (com heurísticas do domínio do problema), que são em geral muito mais eficientes do que uma metaheurística.

Portanto, algumas implicações do NFLT são:

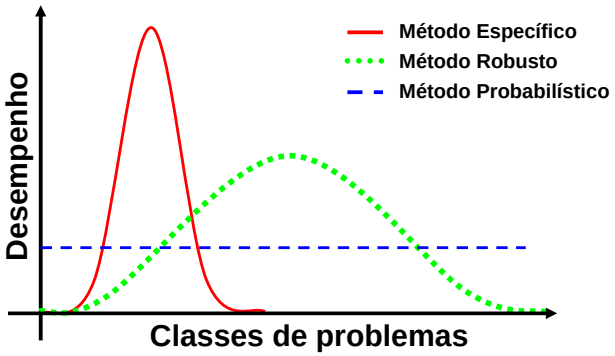


Figura 2.1: Desempenhos relativos de métodos de otimização desenvolvidos para problemas específicos, robustos e de pesquisa aleatória.

- Métodos de otimização específicos para um determinado problema normalmente alcançam excelente desempenho, visto que eles são projetados especificamente para aquele problema, explorando o espaço de busca mais limitado pelo conhecimento adicional do domínio do problema. Algoritmos de otimização local utilizando método do gradiente podem ser vistos em (Hyvarinen et al., 2001);

- Métodos robustos podem resolver um montante relativamente grande de problemas com custo computacional aceitável, visto que utilizam princípios de busca que aumentam sua eficiência. Exemplo: metaheurísticas híbridas combinadas com algoritmos de busca local, como os Algoritmos Meméticos (Merz, 2003);

- Métodos de pesquisa probabilística, como as metaheurísticas, não utilizam informações sobre a natureza do problema e são capazes de resolver praticamente todos os possíveis problemas com pequenas adaptações em seus procedimentos de modificação. Porém podem ser extremamente ineficazes, de modo que encontrar o ótimo global pode não ser garantido, em tempo aceitável.

Os primeiros métodos computacionais eram específicos para determinados problemas e sua aplicação era interessante, visto que computadores eram equipamentos demasiadamente caros e os recursos computacionais eram limitados. Nesse contexto, métodos de busca aleatória ou probabilísticos eram impraticáveis.

Com a evolução tecnológica das últimas décadas, algoritmos robustos, que demandam um custo computacional moderado, tornaram-se práticos. Porém, até mesmo algoritmos robustos precisam ser implementados utilizando heurísticas para resolver certas classes de problemas como: contínuos multimodais (Noman & Iba, 2008), inteiros (Beraldi & Ruszczyński, 2005), de permutação (Tsutsui et al., 2006; Bożejko & Wodecki, 2006) ou deceptivos (Deb & Goldberg, 1993).

Nesse sentido, o problema principal residiria no poder computacional. Se o usuário possuísse capacidade computacional substancialmente elevada, capaz de avaliar uma função altamente complexa (uma simulação de clima ou de mecânica de fluídos) em um tempo consideravelmente pequeno (milissegundos ou menos) ou realizar uma grande quantidade (milhares, milhões) de avaliações em paralelo em alguns segundos, pode-se usar um método de pesquisa estocástica ou até mesmo avaliar todas as possibilidades em vários problemas do mundo real.

Contudo, esse poder computacional é, atualmente, acessível a apenas um grupo muito restrito de pesquisadores. Além disso, há instâncias de problemas que não podem ser resolvidas mesmo com tal infra-estrutura computacional. Portanto, para a grande maioria de pesquisadores e demais usuários de algoritmos de otimização, existe a necessidade de que esses algoritmos sejam mais eficientes.

Um dos principais gargalos no desempenho de um método de otimização global é a quantidade de avaliações (cálculos da função objetivo) que este requer para encontrar o ótimo global. Assim, o caminho para o desenvolvimento de métodos mais eficientes exige a redução do número de avaliações. Por outro lado, quanto mais complexo o problema, mais pontos do espaço de busca precisam ser investigados e avaliados para se atingir o ótimo global. Desse modo, o projeto de um método de otimização global deve conseguir um compromisso entre esses dois aspectos conflitantes.

Além desses aspectos, todas as metaheurísticas dependem do conjunto de soluções iniciais e de procedimentos modificadores, também chamados operadores. As soluções iniciais costumam ser construídas aleatoriamente dentro do domínio do problema, independentemente da metaheurística populacional utilizada. Por outro lado, os procedimentos modificadores podem ser bastante particulares da metaheurística, estando diretamente relacionados à estrutura de dados empregada.

Apesar disso, dadas duas metaheurísticas distintas, M_1 e M_2, e uma função objetivo f, é geralmente possível escrever um conjunto de operadores que farão M_1 encontrar o ótimo global de maneira muito mais eficiente do que M_2, por muitas ordens de magnitude; ou vice-versa. Como exemplo, suponha um problema discreto cujas variáveis são 0 ou 1. Uma metaheurística que trata o problema como discreto provavelmente irá encontrar a solução muito mais rapidamente do que uma metaheurística que trata o problema como contínuo no intervalo de 0 a 1.

Outra crítica sobre a flexibilidade das metaheurísticas está relacionada com o fato de que pode ser possível transformar M_1 em M_2 com poucas modificações. Tal facilidade resulta na possibilidade de criação de uma gama de metaheurísticas extremamente similares, em termos de suas estruturas de dados e seus operadores. Poderia-se então concluir que, apesar de possuírem nomes distintos, suas funcionalidades seriam praticamente as mesmas.

Embora metaheurísticas sejam freqüentemente usadas para solucionar problemas com função objetivo discreta, não-diferenciável, ou caixa-preta, não se pode esperar que a metaheurística seja útil a menos que haja alguma correlação entre valores da função objetivo de soluções candidatas vizinhas, de modo que tendências possam ser identificadas e seguidas. É preciso que a função objetivo tenha uma superfície contínua que seja suave de um nível de observação global e que tal suavidade seja mais ou menos escondida por saltos criados por restrições de discretizações, entre outros fatores. A Figura 2.2 ilustra os diferentes perfis de uma função objetivo: a nível local, com saltos; e a nível global com certa suavidade. O comportamento global da superfície pode ser identificado por operadores diferentes do que os que detectam perfis locais. Desse modo, pode-se ter operadores específicos para cada caso, sendo que no perfil local é mais adequada a aplicação de um operador de busca mais minuciosa.

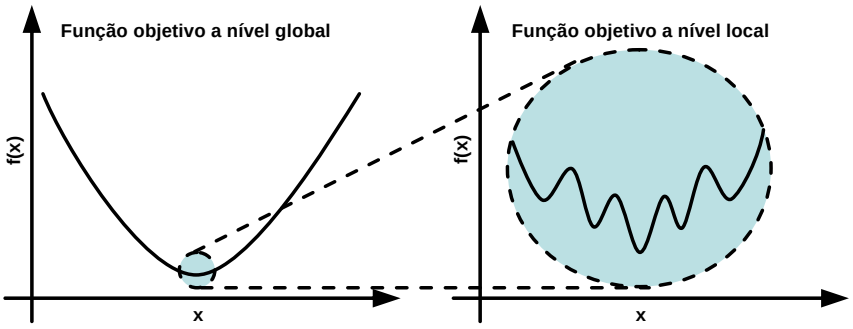


Figura 2.2: Diferentes níveis de uma função objetivo.

Dado que o desempenho de uma metaheurística é extremamente dependente dos procedimentos geradores (da população inicial, por exemplo) e modificadores, muitos pesquisadores concentram-se em melhorar esses procedimentos, em vez de apenas encontrar seus melhores parâmetros. Uma metaheurística trivial, com um ou mais modificadores de qualidade adequada, tende a ser mais eficiente do que uma metaheurística mais sofisticada com um modificador incapaz de gerar soluções de melhor qualidade. Por sua vez, uma heurística específica para o problema em questão será provavelmente muito melhor que ambas.

O uso de metaheurísticas tem produzido soluções que superam as melhores soluções já produzidas por seres humanos especialistas, apesar de anos de teoria e investigação[2]. Domínios de problema que entram nessa categoria são freqüentemente problemas de otimização combinatorial e incluem o planejamento de redes de comunicação (Chou et al., 2001) e de energia elétrica (Granelli & Montagna, 2007), projeto de antenas (Linden, 2002), dentre outros.

[2]"HUMIES" AWARDS FOR HUMAN-COMPETITIVE RESULTS PRODUCED BY GENETIC AND EVOLUTIONARY COMPUTATION: *http://www.genetic-programming.org/hc2009/cfe2009.html*

Como exposto acima, o uso de metaheurísticas é indicado para otimização global, apesar de poder necessitar de uma grande quantidade de avaliações da função objetivo e não garantir o ótimo global. O objetivo desta Tese é desenvolver técnicas capazes de diminuir essas duas desvantagens para qualquer metaheurística. Algumas dessas metaheurísticas são apresentadas a seguir.

2.3 Algumas das Principais Metaheurísticas

Dentre as metaheurísticas (populacionais ou não) mais conhecidas e utilizadas, podem-se destacar:

- *Ant Colony Optimization* (Dorigo, 2004): simula inteligência de enxames baseada no funcionamento de colônias de formigas, que se movimentam guiadas pelo feromônio liberado por outras formigas;

- *Cultural Algorithms* (Reynolds & Sverdlik, 1994): aplica técnicas de aprendizagem de máquina (Mitchell, 1997) em um subconjunto de soluções de alta qualidade para adquirir conhecimento sobre o problema, gerando regiões reduzidas denominadas *células-de-crença*. Essas regiões são criadas com o objetivo de restringir o modo como indivíduos são modificados por operadores genéticos. Desse modo, novos indivíduos devem manter as crenças dos antepassados, ou seja, são criados dentro das células.

- *Differential Evolution* (Price et al., 2005): utiliza diferença entre vetores de soluções para obter informações de distância e direção entre eles. Com a combinação dessas informações existe uma maior chance de gerar pontos de maior qualidade;

- *Estimation of Distribution Algorithms* (Larrañaga & Lozano, 2002): modela as correlações entre as variáveis, ao invés de tratá-las de maneira independente e usa o modelo para gerar a nova população.

- *Evolution Strategy* (Bäck et al., 1991): utiliza conceitos de evolução natural Lamarckiana (melhoramento de indivíduos) (Julstrom, 1999). Nesse caso, características adquiridas por um indivíduo ao longo de sua vida são passadas aos seus descendentes. Um indivíduo (compostos por genes) gera um descendente por meio de mutação. Se o filho for tão bom ou melhor do que o pai, ele o substitui. Esse processo é repetido até que algum critério de parada seja alcançado.

- *Genetic Algorithms* (Goldberg, 1989): utiliza conceitos de evolução natural Darwiniana (Julstrom, 1999) (cruzamento e mutação de soluções), tratando soluções

como indivíduos (compostos por genes) que competem pelo direito de propagarem suas características genéticas para seus filhos;

- *Greedy Randomized Adaptive Search Procedures* (Resende & Ribeiro, 2003): metaheurística de busca estocástica com vários reinícios em regiões distintas, quando necessário, que utiliza um procedimento guloso para gerar soluções de alta qualidade que são repassadas a um algoritmo de otimização local;

- *Learnable Evolution Model* (Coletti et al., 1999): utiliza métodos de aprendizagem simbólica (Mitchell, 1997) para guiar o processo evolutivo. A idéia central é a formulação de hipóteses sobre o problema. A criação de novas soluções é realizada por meio de instanciação a partir das hipóteses determinadas como verdadeiras, ao invés de mutações/recombinações aleatórias de soluções, como ocorre no modelo de Computação Evolutiva.

- *Particle Swarm Optimization* (Clerc, 2006): simula inteligência de enxames baseada no movimento de cardumes ou bandos, que se movimentam seguindo um líder, evitando se chocarem;

- *Simulated Annealing* (Laarhoven & Aarts, 1987): baseado em princípios físicos da termodinâmica, segundo os quais o resfriamento lento de uma matéria que se encontrava em estado de fusão pode levá-la a uma estrutura estável com o nível de energia mais fraco possível;

- *Tabu Search* (Glover & Laguna, 1997): utiliza uma lista de movimentos proibidos para ir em direção ao ótimo, evitando as últimas s soluções já encontradas;

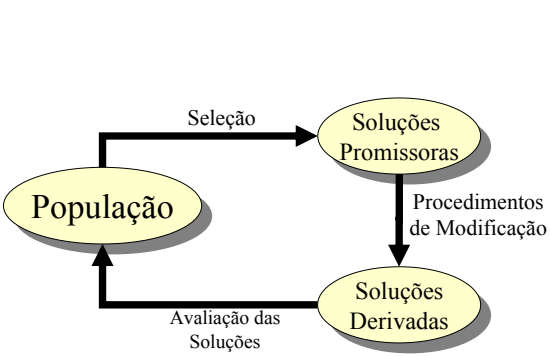
Várias dessas metaheurísticas possuem inspiração biológica e pertencem a uma classe denominada Algoritmos Evolutivos (Bäck et al., 2000a,b) (EA, do inglês *Evolutionary Algorithms*). As principais diferenças entre esses algoritmos estão relacionadas aos procedimentos auxiliares empregados (ver Seção 2.2) aqui chamados operadores evolutivos. Cada solução é chamada, por exemplo, *indivíduo*, *partícula*, *vetor* ou *agente*, de acordo com a metaheurística em questão.

Ao longo dos anos tem sido proposta uma grande quantidade de variantes e híbridos dessas técnicas. Além disso, diversas aplicações de metaheurísticas em problemas específicos têm sido apresentadas. Portanto, esse é um campo ativo de investigação, com uma literatura considerável, uma grande comunidade de investigadores e usuários e uma gama extensiva de aplicações, o que indica que trabalhos capazes de melhorar o desempenho de metaheurísticas costumam ser bem recebidos pela comunidade.

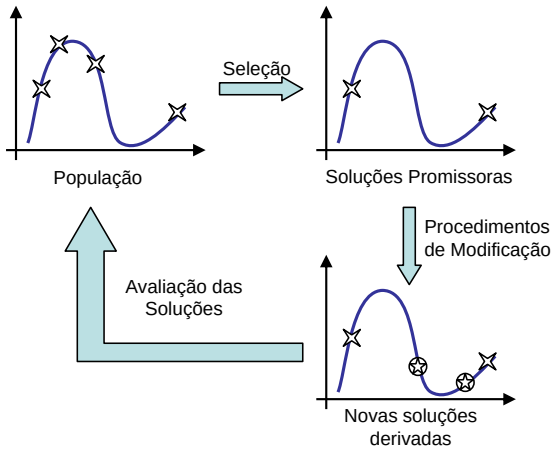
A seguir, são apresentados mais detalhes sobre as seguintes metaheurísticas populacionais, que possuem características significativamente diferentes no processo de geração de novas soluções: Algoritmo Genético (Seção 2.3.1), Otimização por Enxame de Partículas (Seção 2.3.2), Evolução Diferencial (Seção 2.3.3) e Algoritmos de Estimação de Distribuição (Seção 2.3.4).

2.3.1 Algoritmo Genético

Em GAs ocorre um processo no qual os indivíduos da população competem uns contra os outros pela sobrevivência e pelo direito de se reproduzirem. Enquanto os melhores indivíduos propagam suas características pelos indivíduos sobreviventes, os indivíduos menos promissores tendem a ser eliminados. O resultado desse processo é a evolução de uma população de soluções em direção a melhores regiões do espaço de busca no ambiente. Do espaço de busca no qual a população evolui (pontos da superfície amostrada), obtém-se a informação de avaliação, indicando o quão adequado é uma determinada solução. Com base nessa adequação, pode-se decidir se o processo de busca deve continuar e a partir de quais soluções deve-se derivar novas. (Bäck et al., 2000a,b). O funcionamento básico de um GA é sintetizado na Figura 2.3.



(a) Funcionamento sintetizado de um GA.



(b) Exemplo de funcionamento do GA na minimização de uma função contínua.

Figura 2.3: Funcionamento básico de um EA.

O funcionamento simplificado de um GA segue o procedimento mostrado na Figura 2.4. Em geral, uma população de soluções candidatas é iniciada aleatoriamente dentro do domínio do problema e evolui em direção às melhores regiões do espaço de busca por meio de operações como seleção, recombinação e mutação. O ambiente no qual a população evolui é a superfície de resposta da função objetivo. Uma informação de avaliação/adaptação (*fitness*) quantifica o quão adequado é um indivíduo da população de soluções em relação a esse ambiente. O processo de seleção favorece os indivíduos com melhor avaliação e o processo de recombinação mistura a informação de dois indivíduos para a criação de dois novos indivíduos (duas novas soluções candidatas). O operador de mutação introduz modificações em características dos indivíduos buscando manter a diversidade da população (Bäck et al., 2000a,b).

Os GAs clássicos utilizando codificação binária dos dados focam uma otimização local;

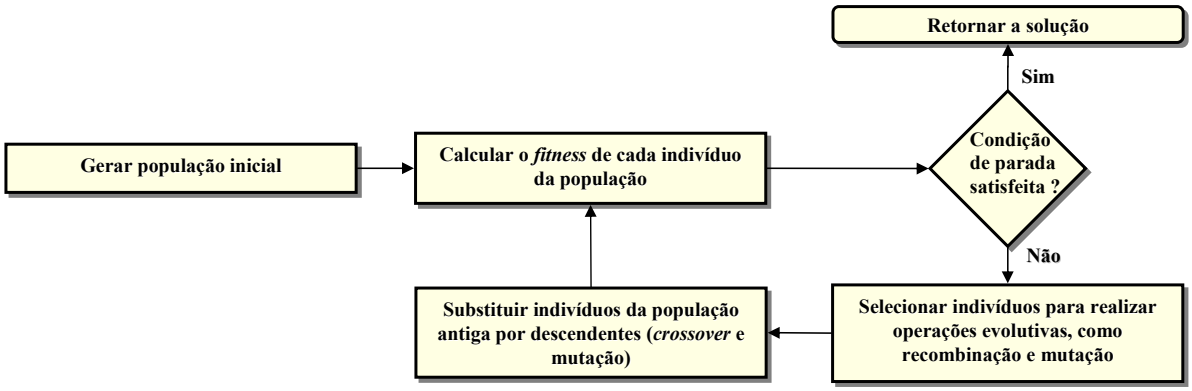


Figura 2.4: Funcionamento básico do GA.

entretanto, a evolução é controlada para tentar percorrer todo o espaço de busca, o que pode tornar a otimização global. Com apenas o operador de recombinação clássico, os GAs podem vasculhar apenas uma parte do espaço de busca onde exista um ótimo local. Como o operador de mutação permite ao algoritmo saltar para uma nova solução em outro contexto, o algoritmo pode, por meio de mutações suficientes, conseguir atingir um valor de ótimo global. Nas Figuras 2.5 e 2.6 tem-se um exemplo de funcionamento desses operadores em problemas com soluções binárias.

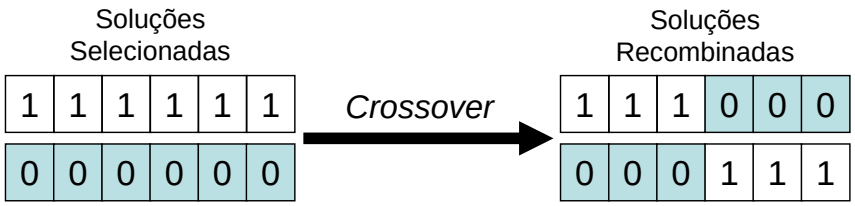


Figura 2.5: Exemplo de funcionamento de operador de recombinação.

Figura 2.6: Exemplo de funcionamento de operador de mutação.

Modificações posteriores no GA permitiram uma melhor exploração do espaço de busca. A codificação em reais pode explorar o espaço de busca com mais facilidade, principalmente utilizando operadores de recombinação como o BLEND *Crossover* (BLX-α) (Eshelman & Schaffer, 1993). Esse operador gera valores aleatórios (usando uma distribuição uniforme), para cada variável, que encontram-se na faixa de valores entre os valores da variável nos pais, mas pode ultrapassar essa faixa igualmente em qualquer extremo, de acordo com o valor de α. Porém, com um valor de α grande, o GA pode

aproximar-se do ótimo rapidamente e não conseguir atingi-lo, devido ao tamanho da modificação gerada por α. Dados dois pais x_a e x_b, esse operador produz um descendente pela recombinação linear aleatória dos pais como segue:

$$\left((x_a)_{i\in[1,D]}, (x_b)_{i\in[1,D]}\right) \longrightarrow \left(\phi_i x_a + (1-\phi_i)x_b\right)_{i\in[1,D]}, \tag{2.1}$$

tal que D é o tamanho do genoma (quantidade de variáveis do problema) e $\phi_i = U(\alpha, 1+\alpha)$.

2.3.2 Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) (Eberhart & Kennedy, 1995; Bonabeau et al., 2000; Kennedy et al., 2003), utiliza conceitos de vida artificial e psicologia social, bem como engenharia e ciência da computação. A PSO difere de outros EAs uma vez que as partículas[3] movimentam-se pelo espaço de busca assumindo que elas são sociáveis, o que implica em uma interação entre as partículas dentro de uma determinada vizinhança. Foi desenvolvida para trabalhar com problemas contínuos. Posteriormente, foram criadas versões para tratar problemas binários (Chuang et al., 2008).

Quando a população inicial é gerada, além de atribuir valores aleatórios às características das partículas, são também atribuídos valores de velocidade. A cada iteração, cada partícula tem sua posição estocasticamente movimentada na direção combinada entre a melhor posição já encontrada por ela (onde foi encontrado o melhor valor de *fitness*) e a melhor posição já encontrada no processo até o momento, por qualquer partícula ou dentro da vizinhança. O tamanho da movimentação é dado pelo valor da velocidade de cada característica.

Portanto, existem duas informações principais disponíveis para uma partícula: 1) sua própria experiência passada e 2) o conhecimento sobre o desempenho dos seus vizinhos, ou apenas da melhor partícula. Essa influência do sucesso de uma partícula em outra não ocorre no GA clássico. O funcionamento básico da PSO é sintetizado na Figura 2.7.

Um indivíduo na da população da PSO é dado por $x = [x_1, x_2, ..., x_D]$, sendo D a quantidade de variáveis do problema. Seu vetor de velocidades é dado por $v = [v_1, v_2, ..., v_D]$ e sua melhor solução (posição com melhor valor de função objetivo) já encontrada é dada por $MP = [MP_1, MP_2, ..., MP_D]$. O deslocamento das partículas na iteração j, baseado na sua própria experiência e na "troca" de conhecimento entre a população, é dado por:

[3]Denominação usada na PSO para o termo *indivíduos*.

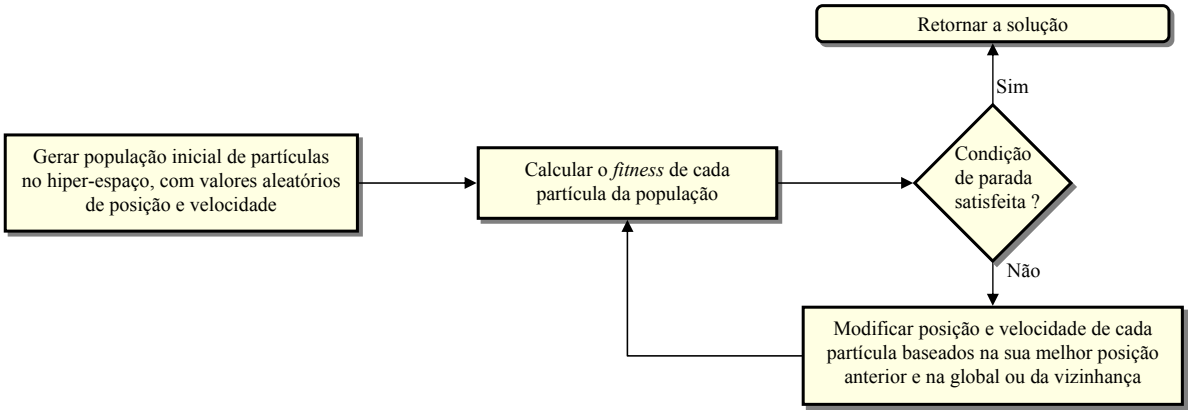


Figura 2.7: Funcionamento da PSO.

$$x^{j+1} \quad = \quad x^j + v^{j+1}, \text{ sendo que} \tag{2.2}$$

$$v^{j+1} = v^j + rand_1(MP^j - x^j) + rand_2(MP_g^j - x^j), \tag{2.3}$$

sendo que MP_g é a melhor posição já encontrada por todas as partículas. Uma representação gráfica do deslocamento de uma partícula é apresentada na Figura 2.8.

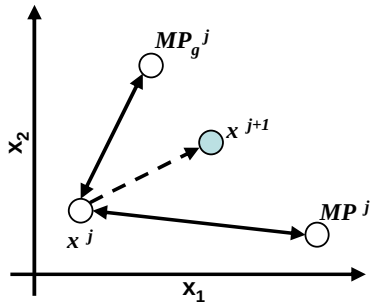


Figura 2.8: Deslocamento de partícula na PSO.

A PSO tem demonstrado ser uma técnica poderosa, pois é simples de compreender, implementar e computacionalmente mais eficiente do que o GA. O algoritmo principal corresponde a apenas duas linhas de código. Segundo os autores (Kennedy et al., 2003), a PSO pode ser uma ordem de magnitude mais rápidos (em quantidade de avaliações da função objetivo) do que o GA em problemas de otimização contínua e costuma ser mais resistente a ótimos locais do que os GAs. Outra característica importante da PSO é que não necessita de um grande número de partículas, pois consegue manter a diversidade populacional por mais gerações. Isso pode tornar a busca menos custosa por precisar de menos tempo computacional e menos avaliações da função objetivo (Carlisle & Dozier, 2001).

2.3.3 Evolução Diferencial

O algoritmo Evolução Diferencial (DE, do inglês *Differential Evolution*)(Storn & Price, 1997) é um dos EAs existentes mais eficientes para resolver problemas de otimização contínua global. A DE extrai a informação diferencial da população atual de soluções para guiar a busca posterior. Essa informação entre um indivíduo de alta qualidade e dois de baixa qualidade, escolhidos aleatoriamente, permite determinar a distância entre eles, pelo cálculo da diferença entre os valores de cada posição e, também a direção, pois a diferença pode ser positiva ou negativa. Porém, a DE não possui nenhum mecanismo para extrair e usar informação global sobre o espaço de busca, sendo guiada sempre na direção do melhor indivíduo, com pouca exploração global. Um esquema básico da DE é apresentado na Figura 2.9.

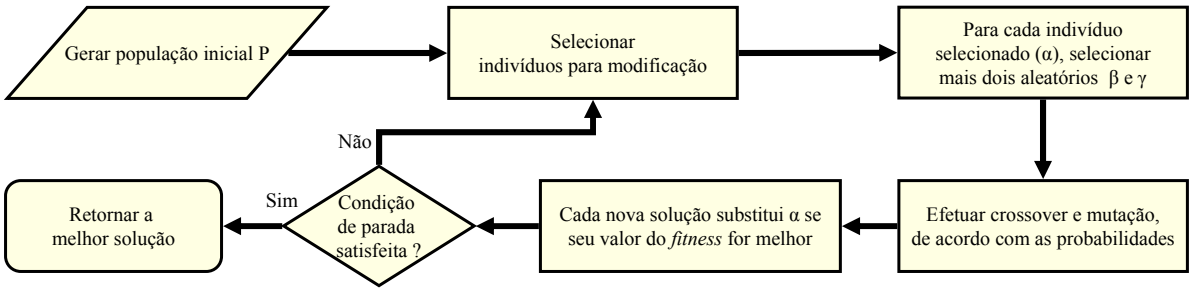


Figura 2.9: Funcionamento do DE.

Como o GA, a DE possui operadores de seleção, mutação e recombinação. O operador de mutação utiliza 3 soluções da população atual para modificar a solução x_a em questão, gerando uma solução x_c. O ponto x_c é gerado combinando x_a com um outro ponto com melhor valor de *fitness* (x_b), que é selecionado aleatoriamente da população atual, e com o vetor de diferenças entre outros dois pontos quaisquer, também selecionados aleatoriamente.

Como discutido em (Storn & Price, 1997), a utilização dos diferenciais dos vetores amostrados tem várias vantagens. Em primeiro lugar, como a média da distribuição de diferenciais sempre é zero, não há nenhum viés de amostragem. Isso é útil para preservar a diversidade da população. Em segundo lugar, o desvio-padrão da distribuição do diferencial poderia mudar junto com o tamanho e forma da população no espaço de busca, o que é interessante para problemas cujos intervalos são muito grandes. Em terceiro lugar, o esquema trabalha bem como um otimizador local, visto que o diferencial em uma população que está convergindo tenderá a zero. A capacidade de busca local é importante quando se encontra uma região promissora do espaço de busca (um cume, por exemplo - ver Seção 2.1).

O operador de recombinação equivale ao clássico *crossover* uniforme (Davis, 1991) entre a solução x_c, gerada pelo operador modificador, e a solução x_a. A cada posição i

da nova solução recombinada NS_i, atribui-se aleatoriamente o valor de $x_{c,i}$ ou $x_{a,i}$.

Devido à sua habilidade em manter a diversidade e fazer uma busca local de boa qualidade, a DE tende a obter resultados melhores do que outros EAs (Storn & Price, 1997; Chakraborty, 2008). Contudo, esse algoritmo não possui nenhum mecanismo para usar informação global diretamente sobre o espaço de busca de modo a guiar a população para áreas promissoras. Por esse motivo, o estudo de algoritmos capazes de encontrar regiões promissoras pode melhorar o desempenho da DE em diversos problemas.

2.3.4 Algoritmos de Estimação de Distribuição

Algoritmos de Estimação de Distribuição (EDAs, do inglês *Estimation of Distribution Algorithms*) (Mühlenbein & Paass, 1996; Larrañaga & Lozano, 2002; Pelikan et al., 2002), também conhecidos como algoritmos evolutivos de construção de modelos probabilísticos, têm chamado uma crescente atenção de pesquisadores durante os últimos anos. O uso de um modelo probabilístico dos genes foi introduzido por Baluja no algoritmo PBIL (do inglês, *Population-based incremental learning*)(Baluja, 1994). O PBIL trabalha com problemas binários analisando a probabilidade dos valores de cada uma das variáveis, independentemente, para gerar novos indivíduos que contenham os valores de maior freqüência. EDAs em problemas contínuos foram introduzidos no campo de EA por Mühlenbein (Mühlenbein & Paass, 1996), incorporando métodos para a identificação da distribuição de genes independentes para melhorar o procedimento de recombinação dos GAs. Posteriormente, outros estudos incorporaram métodos para aprendizagem automatizada de correlações entre variáveis codificadas nas soluções do problema (Larrañaga & Lozano, 2002). Variáveis correlacionadas podem ser vistas como BBs.

A identificação de variáveis correlacionadas por meio de um modelo probabilístico possibilita a criação da descendentes baseados nesse modelo, buscando respeitar as correlações identificadas. Desse modo, evita-se o rompimento de importantes blocos de construção na grande maioria das vezes, ao contrário do que tende a ocorrer com o operador de recombinação do GA, por exemplo. Como as novas soluções são amostradas do modelo, não há necessidade de especificar parâmetros como taxas de cruzamento e mutação do GA.

Porém, a construção do modelo probabilístico usado para guiar a busca por meio da amostragem da descendência é uma das principais dificuldades desse método, devido ao custo computacional requerido para se obter modelos significativamente representativos de uma população(Heckerman et al., 1994). Uma das desvantagens é que quanto mais complexo o modelo, maior deve ser a população. Portanto, comparado ao GA, o EDA pode necessitar de uma população muitas vezes maior. Por outro lado, espera-se uma menor quantidade de gerações para atingir o ótimo global.

Diversas abordagens para explorar modelagem de problemas têm sido estudadas (Pelikan et al., 2000; Paul & Iba, 2003; Ding et al., 2008). O primeiro algoritmo apresentado

por Mühlenbein é o UMDA (do inglês, *Univariate Marginal Distribution Algorithm*) (Mühlenbein & Paass, 1996). No UMDA, assume-se que as variáveis são independentes entre si mas que existe uma distribuição dos pontos no espaço de busca que seguem um determinado padrão. O padrão adotado é o de uma distribuição normal (Patel & Read, 1996). Assim, a nova amostragem é realizada baseada na média e no desvio padrão de cada uma das variáveis das soluções promissoras na população atual. Desse modo, os novos pontos tendem a ser criados na região localizada entre as soluções promissoras (ponto médio). O grau de exploração é determinado pelo desvio padrão de cada variável. Logo, à medida que a população converge para uma região a busca é refinada, visto que o desvio padrão é reduzido. Essa característica importante torna o algoritmo auto-adaptativo. Um fluxograma simplificado de um EDA padrão é apresentado na Figura 2.10. A diferença básica dos EDAs é o modelo probabilístico, sua construção e amostragem. Um exemplo do funcionamento dessa diferença básica é apresentado na Figura 2.11. $p(x_i) = 1$ é a probabilidade do bit i da solução x ser igual a 1, de acordo com as ocorrências no conjunto de soluções selecionadas. $P(x)$ é o modelo probabilístico de distribuição dos bits das soluções selecionadas. Mais detalhes podem ser vistos na Seção 6.2.1.

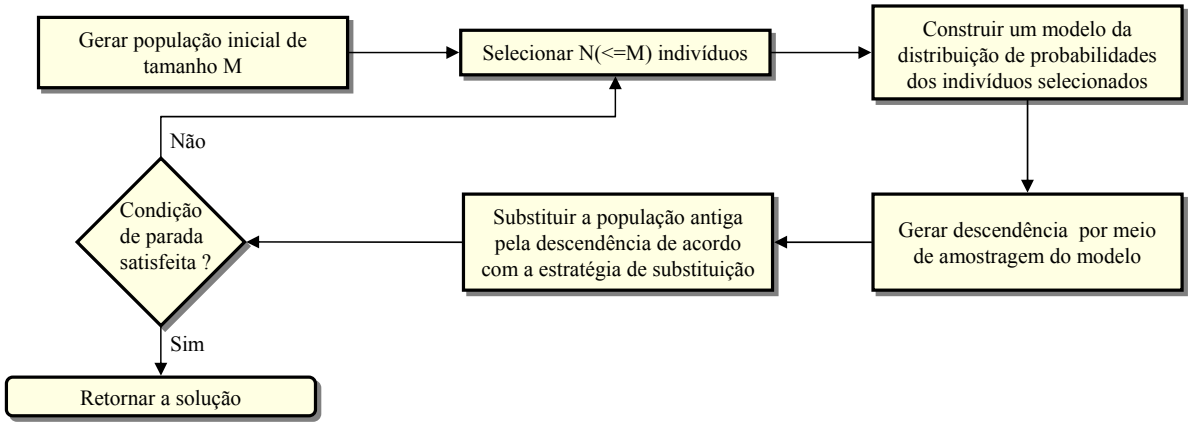


Figura 2.10: Funcionamento do EDA.

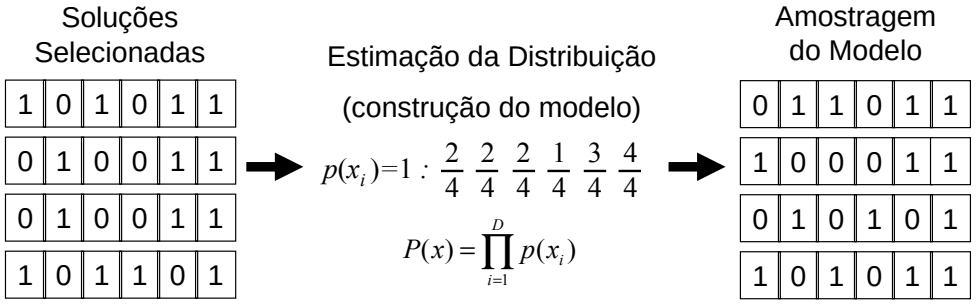


Figura 2.11: Exemplo de construção e amostragem do modelo do UMDA.

De acordo com Larrañaga e Lozano (Larrañaga & Lozano, 2002), é mais adequado aplicar EDA em lugar do GA convencional em problemas contínuos pois o EDA seria

capaz de identificar correlações entre as variáveis e necessitaria de menos avaliações da função objetivo. Contudo, a estimação da distribuição de probabilidade associada ao conjunto das soluções promissoras é o gargalo dessa nova técnica. Não existe um método eficiente para construção modelos complexos em problemas de larga-escala com grande quantidade de correlações entre variáveis. Se o modelo for mais confiável, os resultados serão melhores. Contudo a construção do modelo pode ser complexa e requerer um elevado tempo computacional. Além disso, fazer a amostragem a partir do modelo pode não ser uma tarefa trivial.

2.3.5 ECS

O algoritmo Busca por Agrupamento Evolutivo (ECS, do inglês *Evolutionary Clustering Search*) (Oliveira, 2004) é um algoritmo de otimização global avançado, composto por (ver Figura 2.12):

1. um algoritmo evolutivo (AE): esse componente, um GA, trabalha gerando soluções candidatas por meio de seleção e recombinação (*crossover* e mutação), independentemente dos outros módulos;

2. um agrupador iterativo (AI): opera agrupando indivíduos similares, mantendo uma solução representativa de cada grupo (cluster);

3. um módulo analisador de agrupamentos (AA): dentro de intervalos regulares de gerações, esse módulo verifica cada um dos clusters encontrados pelo módulo AI para determinar quais são promissores;

4. um algoritmo de otimização local (AO): aplica um algoritmo de busca local nas regiões supostamente promissoras.

Em resumo, o ECS funciona da seguinte maneira. Um processo de agrupamento iterativo trabalha simultaneamente com um EA. Esse processo é responsável por contabilizar as operações de modificação ocorridas em regiões do espaço de busca e identificar aquelas que merecem atenção especial. Em outras palavras, o processo de agrupamento identifica grupos de indivíduos com similaridades, que possuem um padrão predominante de genótipos (esquemas) e, conseqüentemente, concentram-se em uma certa região do espaço de busca. Quando o algoritmo detecta que existe uma baixa diversidade nos indivíduos de um determinado grupo, significando que soluções candidatas estão relativamente próximas, um algoritmo de busca local é aplicado para tentar identificar o ponto mínimo da região.

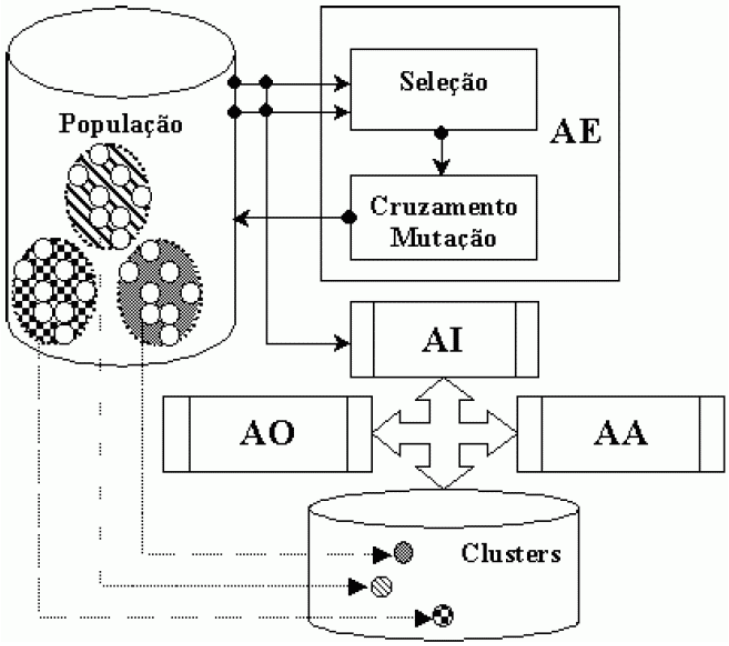


Figura 2.12: Diagrama conceitual do ECS (Oliveira, 2004).

2.3.6 C-GRASP

Feo e Resende (Resende & Ribeiro, 2003) descrevem a metaheurística GRASP, como um procedimento de busca local com recomeços (*multistart*), em que cada iteração do algoritmo é composta por uma fase de construção e uma fase de busca local a partir dessa solução. O C-GRASP (Hirsch et al., 2007) é uma extensão do GRASP para trabalhar com problemas contínuos. É um método de busca estocástica simples de implementar, largamente aplicável e não utiliza informação de derivadas, o que o torna adequado para resolver problemas de otimização global, quando reiniciado várias vezes em diferentes regiões do espaço de busca. As fases do funcionamento do C-GRASP são apresentadas a seguir.

1. Construção: constrói uma solução factível para o problema. Essa construção é realizada elemento (variável) a elemento por meio de uma seleção aleatória de candidatos em uma lista restrita, ordenada por uma função gulosa adaptativa. Isso significa que, inicialmente, todos os elementos são livres, e à medida que a solução vai sendo construída, os elementos vão sendo fixados. A solução é considerada concluída quando todos os elementos estiverem fixados.

2. Busca Local: dada a solução construída, esse procedimento busca pela melhor solução da vizinhança. A busca é feita utilizando movimentos em direções "quase canônicas" escolhidas aleatoriamente. As direções consideradas são da forma: [0, -1, 1, ..., 1], [1, 1, -1, ..., 0], etc; ou seja, vetores cujas componentes são -1, 0 ou 1,

tal que -1 indica que o valor da variável deve ser reduzido, 0 que deve ser mantido e 1 que deve ser aumentado.

2.4 Considerações Finais

Neste Capítulo foram apresentadas técnicas para solução de problemas de otimização global, as metaheurísticas, suas características gerais, prós e contras.

Uma metaheurística é um método de busca por soluções de alta qualidade utilizado para resolver uma classe ampla de problemas computacionais de otimização. Por ser um algoritmo mais geral do que algoritmos de otimização numéricos, evita-se o uso de técnicas de resolução de problemas específicos que façam uso de conhecimento prévio do problema (heurísticas). Essa característica é importante pois possibilita o uso das metaheurísticas na resolução de problemas do tipo caixa-preta.

Foram apresentadas quatro metaheurísticas para otimização global: GA, PSO, EDA e DE. O GA é uma das metaheurísticas mais utilizadas atualmente. É de fácil implementação e tem apresentados resultados relevantes na resolução de problemas. Porém, para que os resultados sejam satisfatórios, podem ser necessários operadores evolutivos e heurísticas específicas para o problema em questão.

Em otimização contínua, a PSO costuma apresentar resultados em geral de qualidade tão boa ou melhor do que o GA, com a vantagem de necessitar de uma população menor e possivelmente menos gerações, portanto, menos avaliações. Contudo, a exploração local não é um ponto forte deste algoritmo.

A DE, por outro lado, tem apresentado resultados ainda melhores do que a PSO em problemas nos quais o ótimo global não é tão pronunciado, necessitando de uma exploração local mais eficiente. Por outro lado, essa eficiência só ocorre quando a DE encontra a região do ótimo global. Por esse motivo, é interessante o uso de técnicas capazes de encontrar essa região. Esse é o foco desta Tese de doutoramento.

Os EDAs são desenvolvidos objetivando a resolução de outro tipo de problemas: os problemas nos quais existe correlação entre variáveis (BBs) e, por isso, é importante que elas sejam trabalhadas em conjunto. Esses algoritmos fazem uso de modelos probabilísticos capazes de identificar e explorar mais adequadamente os BBs, evitando a quebra de soluções parciais. Essa característica pode tornar os EDAs substancialmente mais eficientes que outros EAs em problemas dessa natureza e são apresentados em mais detalhes no próximo Capítulo. Contudo, para conseguir gerar um modelo de alta qualidade, é requerida uma população que pode ser substancialmente maior do que a do GA. Porém, ao encontrar regiões promissoras, foco deste trabalho, é possível que a população seja reduzida, visto que não será necessária uma grande exploração no espaço de busca.

Outros dois algoritmos recentes que têm apresentado resultados relevantes na liter-

atura são o ECS e o C-GRASP. O ECS é um algoritmo que emprega uma técnica de agrupamento para detecção de regiões promissoras e com isso executar uma busca pela melhor solução dessas regiões reduzidas. Essa abordagem apresenta um ganho de desempenho significativo.

O C-GRASP é um método de busca local que pode ser aplicado globalmente, bastando para isso reiniciar o algoritmo em outra região do espaço de busca. Por ser um algoritmo de busca local, possui uma maior probabilidade de encontrar o ótimo global. Contudo, a quantidade de avaliações necessárias tende a ser significativamente alta.

Com o desenvolvimento de metaheurísticas surge, também, a necessidade de verificar sua eficácia e eficiência em relação a outras técnicas. Esse assunto é tratado com mais detalhes no Capítulo 3.

3

Métricas para Comparação de Algoritmos de Otimização Global

Neste Capítulo são apresentadas métricas de avaliação de algoritmos de otimização global e conceitos de comparação estatística. Existe uma grande diversidade de métricas de avaliação de desempenho desses algoritmos, baseada em indicadores. Dentre as métricas mais comuns podem-se citar: valor da melhor solução encontrada, tempo de execução e quantidade de avaliações da função objetivo (ver Seção 3.1). A comparação baseada nessas métricas é realizada, em geral, por meio da média e desvio padrão. Contudo, essa comparação pode estar equivocada, do ponto de vista estatístico. Além disso, por não existir um padrão ou norma adotado pela comunidade acadêmica, a comparação dos resultados entre duas publicações pode ser impossível, visto que os resultados apresentados por um pesquisador podem estar incompletos ou inadequados. Uma outra característica infelizmente não rara é realizar uma comparação desleal entre um novo método, contendo os mais recentes avanços do estado-da-arte, com um método clássico e, comumente, de baixo desempenho. Uma comparação adequada deve ser feita entre algoritmos de características similares.

Para determinados tipos de problema de teste, utilizados na avaliação de metaheurísticas, não basta saber o valor da solução encontrada, é importante saber se essa solução está ou não *física* ou *matematicamente* próxima do ótimo global. Essa informação tem sido pouco tratada nas pesquisas da área de metaheurísticas. Outro problema está relacionado à capacidade de exploração do espaço de busca pelos algoritmos de otimização global. Nenhuma das métricas permite essa avaliação, que pode dar indícios do comportamento do algoritmo em diversos tipos de problema. Esses dois itens serviram de

motivação para a métrica proposta na Seção 3.2.

A análise de média e desvio padrão, adotada por grande parte da comunidade acadêmica, faz sentido apenas em dados com um tipo especial de comportamento. Caso contrário, como explicado na Seção 3.3, sua interpretação será equivocada e as conclusões estatisticamente inválidas. Por esse motivo, é recomendável o uso de testes estatísticos (testes de hipótese) adequados para comparação de algoritmos, principalmente para problemas complexos de larga-escala. É interessante notar que mais recentemente o embasamento estatístico dos experimentos tem ganhado mais importância na área de metaheurísticas, conforme mostram as seguintes referências de trabalhos incentivando sua aplicação (Bartz-Beielstein, 2006; Zlochin et al., 2004). Por outro lado, as métricas de avaliação merecem também uma maior investigação, conforme mostra a Seção 3.1, apesar de alguns trabalhos terem tratado desse aspecto.

3.1 Métricas de Avaliação

Esta Seção sintetiza as métricas mais utilizadas na área para avaliação de desempenho de algoritmos de otimização global. Na Seção 3.2 é apresentada a métrica de avaliação proposta nesse trabalho, que busca diferenciar melhor o desempenho de técnicas, possibilitando, por exemplo, avaliar também o grau de aproximação do ótimo global quando esse não é atingido. É importante ressaltar que algumas dessas informações são possíveis de serem obtidas apenas em problemas de testes, nos quais se conhece o ótimo global.

Medidas de desempenho quantitativas são usadas para estimar o número de avaliações necessárias para atingir o ótimo da função objetivo. Os termos a seguir serão usados para explicar as métricas de avaliação:

- A: um algoritmo populacional de otimização global;
- n: o tamanho da população de A;
- f: uma função objetivo a ser otimizada;
- D: o número de dimensões (variáveis) de f;
- r: quantidade de repetições da execução de A para otimizar f;
- S^*: conjunto das melhores soluções possíveis para f;
- x^*: uma solução de S^*;
- x: a melhor solução encontrada para f, em uma determinada execução de A;
- O^*: o valor da melhor solução possível para f, equivalente a $f(x^*)$;

- O_i: o valor da melhor solução encontrada para f, em uma determinada execução i de A;

- $O = [O_1, ..., O_r]$: conjunto de tamanho r contendo O_i de cada repetição i de A;

- $Aval_i$: a quantidade de avaliações calculadas em uma determinada execução i de A;

- $Aval = [Aval_1, ..., Aval_r]$: conjunto de tamanho r contendo $Aval_i$ de cada execução i de A;

- MAX_{aval}: a quantidade limite de avaliações para cada execução de A.

Quantidade de Avaliações (QA)

O valor de $Aval_i$ de f que A necessita para encontrar O^*, limitado a MAX_{aval}, é uma das métricas mais empregadas. Como, em geral, o conjunto $Aval$ pode não possuir uma distribuição Normal, deve-se evitar a apresentação apenas do valor de sua média. Para possibilitar uma comparação adequada com outros algoritmos, essa medida deve ser apresentada com algumas estatísticas do conjunto $Aval$. Dentre essas estatísticas, podem-se destacar: mínimo, 1^o quartil, mediana, média, 3^o quartil, máximo e desvio padrão (Pfeiffer, 1965). Contudo, é comum serem apresentados apenas os estimadores de média e desvio padrão.

- Prós: essa medida é importante pois fornece um indicativo do custo computacional do algoritmo, visto que tal custo está relacionado diretamente ao tempo de cálculo de f. Como exemplo, pode-se supor uma f com complexidade computacional que requeira 1 segundo para ser calculada em um PC dos dias atuais (cerca de 2GHz de *clock* e 4GB de memória RAM). Se A necessitar de muitas avaliações, o tempo necessário poderá ser inaceitável do ponto de vista de uma aplicação prática;

- Contras: o custo final de A poderá ser drasticamente subestimado se A utilizar alguma técnica com alto custo computacional que extraia informações dos dados para guiar a busca, sem a necessidade de muitas avaliações da função objetivo.

Quantidade de Gerações

Trata-se de uma medida que apresenta a quantidade de gerações (ou a média $\overline{Ger}_{O*}$) que A requer para encontrar O^*. Em geral, são informadas apenas média e desvio padrão de um conjunto de r repetições de A. Contudo, é importante que sejam informados outros estimadores estatísticos, como apresentados no item anterior (QA). Os principais aspectos referentes ao uso dessa métrica seriam:

- Prós: quanto menor o número de gerações que A necessitar para encontrar O^*, melhor será o seu desempenho computacional. É uma métrica simples de interpretar, principalmente para usuários que não são da área de pesquisa em EAs;

- Contras: na comparação entre trabalhos pode não existir uma relação direta com o número de avaliações por geração, pois As distintos podem utilizar populações de tamanhos diferentes. Além disso, o tamanho da população pode variar ao longo das gerações para um mesmo A. Portanto, essa comparação não é válida para As distintos, sendo adequada somente quando for comparar duas configurações para um mesmo A, por exemplo.

Tempo de processamento

Trata-se da medida do tempo físico que A requer para encontrar O^* em um certo computador. Em geral, apresenta-se também essa medida por meio de estatísticas de média e desvio padrão. Pode-se resumir sua análise da seguinte maneira:

- Prós: em problemas nos quais o tempo computacional é crítico, como tomada de decisão em tempo real (Resende & Sousa, 2004), essa medida possui uma relevância de extrema importância. Além disso, pode ser usada para comparar As quando a quantidade de avaliações deles é muito similar. Outra característica importante é que essa pode ser a única medida importante em problemas do mundo real. Por exemplo, quanto tempo um algoritmo leva para encontrar uma solução com valor de função objetivo desejado, não importando a quantidade de avaliações ou gerações. Além disso, essa é a única medida que leva em consideração o custo computacional do algoritmo completo (*overhead*), não apenas da função objetivo;

- Contras: essa medida tem sido empregada para As rápidos, mas que possuem MAX_{aval} relativamente altos. De acordo com esse aspecto, tal medida pode mascarar a ineficiência de A, sendo inadequada para a comparação de trabalhos, visto que as configurações de *hardware* e *software* raramente são iguais.

Valor da Melhor solução encontrada

Essa medida apresenta apenas o melhor O encontrado nas r repetições. Em síntese:

- Prós: é adequada quando não se sabe O^* de f, ou quando procura-se por uma solução que tenha, no mínimo, um valor de referência informado pelo usuário, não necessitando ser O^*. Essa medida pode ser útil em problemas do mundo real quando existe um limite de tempo;

- Contras: publicações que utilizam essa métrica costumam apresentar apenas o melhor resultado encontrado, sem qualquer outra informação como: quantidade de vezes que o valor foi alcançado, o valor médio encontrado, desvio padrão, etc.

Valor Médio das melhores soluções encontradas

Com esse critério, em geral, apresentam-se apenas estimadores de média e desvio padrão sobre o conjunto O. Como, em geral, O pode não possuir uma distribuição Normal, deve-se evitar a apresentação desses dados isoladamente. Para possibilitar uma comparação adequada com outros algoritmos, esse critério deve ser apresentado com outras informações como: mínimo, 1^o quartil, mediana, média, 3^o quartil, máximo e desvio padrão.

- Prós: essa medida é importante pois está relacionada à eficácia de A. Com ela, é possível ter uma estimativa do comportamento de A em f e concluir se A é capaz de fornecer, na média, soluções de alta qualidade ou se A é inadequado para resolver *f;*

- Contras: a comparação direta entre médias ou medianas deve ser evitada, visto que As com comportamentos (função de distribuição) consideravelmente diferentes podem apresentar médias similares (ver item 3.3).

Taxa de sucesso (TS)

A taxa de sucesso informa quantas vezes O^* foi encontrado em r repetições. Se MAX_{aval} forem realizadas sem A encontrar O^*, então assume-se que não houve sucesso. Em síntese:

- Prós: essa medida possibilita uma avaliação da robustez de A em termos de porcentagem de sucesso. Para problemas de otimização contínua em que se conhece o ótimo global, em geral, $O^* \equiv f(x^*) + \varepsilon$, ou seja, aceita-se um erro ε para se concluir que O^* foi alcançado. Isso permite ao pesquisador graduar o nível de rigor na avaliação de A;

- Contras: precisa ser combinada com uma medida de custo computacional, preferencialmente as estatísticas da quantidade de avaliações, para que uma análise adequada possa ser realizada. Isso evita, por exemplo, que um algoritmo A_1 que possua 100% de sucesso, utilizando 10^6 avaliações seja considerado melhor do que um algoritmo A_2 que possua 90% de sucesso, em apenas 10^5 avaliações. Obviamente, existem problemas em que Max_{aval} pode ser ignorado.

Desempenho de Sucesso (DS)

Suganthan et. al. (Suganthan et al., 2005) introduziram a métrica denominada Desempenho de Sucesso (DS) que combina TS e a média da QA necessárias para se atin-

gir $O^*(\overline{Aval}_{O*})$:

$$DS = \frac{\overline{Aval}_{O*}}{TS} \tag{3.1}$$

Em alguns trabalhos os autores apresentam a média de avaliações de todos os testes, não apenas os que obtiveram sucesso. Desse modo, para se calcular a DS, é necessário estimar a média de avaliações realizadas apenas quando o algoritmo encontrou o ótimo global. Para tanto, pode ser utilizado o seguinte procedimento: dado que os autores informam a média de avaliações total ($\overline{Aval}$), a quantidade máxima de avaliações (MAX_{aval}), a quantidade de repetições (r) e a taxa de sucesso (TS), a estimativa de $\overline{Aval}_{O*}$ pode ser calculada como:

$$\overline{Aval}_{O*} = \frac{(\overline{Aval} * r) - (1 - TS) * MAX_{aval} * 100}{TS * r} \tag{3.2}$$

- Prós: essa métrica facilita a comparação, para evitar que um algoritmo seja superior em um critério e inferior em outro. Por exemplo, A_1 que possua TS=1 e $\overline{Aval}_{O*} = 1000$ e A_2 que possua TS=0,5 e $\overline{Aval}_{O*} = 500$ são considerados iguais;

- Contras: por outro lado, não existem testes estatísticos para comparar valores de DS entre dois ou mais As, sendo necessário realizar comparações diretas.

3.2 Métrica proposta

Neste trabalho foi proposta uma métrica, a análise em domínio reduzido, que envolve o conceito de regiões promissoras (ver Capítulo 4). Essa métrica, que deve ser utilizada em problemas de teste nos quais O^* é conhecido, é apresentada a seguir.

Redução de domínio

O processo de redução do domínio de um problema para intervalos cada vez mais próximos ao ótimo global pode ser útil tanto como técnica de otimização (ver Capítulo 5) quanto como uma métrica para comparar desempenho de algoritmos.

Para comparar A_1 e A_2 em f, ambos são executados r vezes em diferentes níveis de redução do domínio, por exemplo, reduções sucessivas de 10% em cada dimensão. Para cada nível é aplicada a TS. O algoritmo mais eficiente conseguirá atingir O^* com maior freqüência a um nível menor de redução do domínio. Essa métrica possibilita distinguir os desempenhos de A_1 e A_2 mesmo para os problemas mais difíceis, pois para algum nível de redução maior, A_1 ou A_2 começará a obter sucesso, enquanto o outro precisará de reduções adicionais.

- Prós: Se a TS de A não aumentar com a redução, provavelmente será devido: à incapacidade de A em otimizar a função objetivo, que pode ser excessivamente complexa; ou a problemas relacionados à configuração de A;

- Contras: deve-se tomar cuidado com relação à posição do ótimo global em relação ao centro do domínio, pois se o mesmo estiver no centro, reduções sucessivas podem tender ao centro, mesmo que A não seja um método de busca eficiente. Com isso, para um nível alto de redução, qualquer A seria capaz de encontrar O^* caso esse esteja posicionado no centro do domínio.

A seguir são apresentados testes estatísticos para problemas de *benchmark*, mostrando como as métricas desta Seção permitem que se diferencie o desempenho de dois As.

3.3 Comparação Estatística

Ao realizar experimentos, é necessário seguir uma metodologia para validar os resultados e determinar se existe uma diferença significativa entre as várias abordagens investigadas, ou se a diferença entre elas é insignificante, o que impede a conclusão de superioridade de uma sobre a outra (Bartz-Beielstein, 2006).

Os resultados de uma metaheurística que é executada várias vezes sobre o mesmo problema podem variar bastante, devido aos diferentes pontos de partida e decisões estocásticas subseqüentes. Por essa razão, trabalhos nessa área costumam apresentar estatísticas como média e desvio padrão obtidos a partir dos resultados das execuções, para determinar a superioridade de um algoritmo em relação a outro. De modo a realizar uma comparação mais adequada, é interessante utilizar testes de hipótese na verificação de igualdade das médias ou medianas dos resultados.

Nesta Seção é recordado brevemente o teste de hipótese e são introduzidos os conceitos de Estatísticas Paramétricas e Não-Paramétricas.

Teste de Hipótese

O teste de hipótese é uma técnica aplicada na inferência estatística sobre o conjunto de todas as possibilidades de um problema (conhecido estatisticamente como população), a partir do subconjunto dessa população (denominado amostra). Esse teste é um procedimento utilizado para aceitar ou não uma hipótese estatística com base nas amostras e pode ser de dois tipos: 1) Paramétrico, quando a amostra segue um comportamento conhecido e 2) Não-Paramétrico, caso contrário. Nesse procedimento, destaca-se os seguintes componentes:

1. **Hipótese Estatística**: Corresponde a uma suposição quanto ao valor de um dos parâmetros da população, ou quanto à natureza da distribuição de probabilidade de uma determinada variável populacional;

2. **Tipos de Hipótese**: Designa-se por H_0, denominada hipótese nula, a hipótese estatística a ser testada (ex.: as médias de duas amostras são iguais), e por H_1, a hipótese alternativa (ex.: as médias - μ_1 e μ_2 - de duas amostras são diferentes). A hipótese nula expressa uma igualdade, enquanto a hipótese alternativa é dada por uma desigualdade. Exemplo de diferença entre duas médias:

   (a) $H_0 : \mu_1 - \mu_2 = 0$;

   (b) $H_1 : \mu_1 - \mu_2 \neq 0$.

3. **Tipos de Erro de Hipótese**: Existem dois possíveis erros de hipótese:

   (a) Erro do tipo 1: rejeição de uma hipótese quando ela é verdadeira;

   (b) Erro do tipo 2: aceitação de uma hipótese quando ela é falsa.

As probabilidades dos erros do tipo 1 e 2 são designadas α e β, respectivamente. A probabilidade α é conhecida como *nível de significância* do teste. Como foi dito inicialmente, o objetivo do teste de hipótese é determinar, por meio de uma estatística, se a hipótese nula deve ou não ser aceita. Essa decisão é tomada considerando uma região de rejeição ou região crítica. Caso o valor observado da estatística pertença à região de rejeição, H_0 deve ser rejeitada e diz-se que não há informações suficientes para aceitar a hipótese nula; caso contrário, H_0 não deve ser rejeitada e diz-se que não há informações suficientes para rejeitar a hipótese nula.

A Lógica do Teste de Significância

Em geral, atribui-se valores pequenos para α, geralmente de 1 a 10%, sendo o mais comum $\alpha = 5\%$. A hipótese H_0 é formulada com o objetivo de que seja rejeitada. Por isso, o nome de hipótese nula. O teste de significância não elimina a probabilidade de erro. Todavia, fornece o valor de probabilidade, denominado *p-valor*)(Schervish, 1996) que permite decidir, em relação a α, se existe evidência suficiente para rejeitar (*p-valor*$< \alpha$) H_0. O *p-valor* diz quão provável seria obter uma amostra tal qual a que foi obtida, quando H_0 é verdadeira. Por esse motivo, os pesquisadores possuem confiança em rejeitar H_0 (assumir que existe a diferença procurada) quando o *p-valor* é pequeno, cujo significado equivaleria dizer que o resultado não ocorreu por acaso. Contudo, deve-se observar que rejeitar H_0 significa que a decisão tomada está correta apenas em relação a α. Caso o teste indique a aceitação de H_0, diz-se que, com o nível de significância α, não se pode rejeitar H_0.

3.3.1 Estatística Paramétrica para Testes de Média

É comum utilizar o teste t de "Student" (Sheskin, 2000) para duas amostras independentes quando se investiga diferenças em médias. Esse tipo de teste é baseado na retirada de amostras aleatórias de duas distribuições independentes e normais. O fundamento teórico desse teste necessita que as seguintes suposições sejam verdadeiras:

- As duas amostras devem ser obtidas obrigatoriamente de uma distribuição Normal;
- As duas amostras devem ser extraídas aleatoriamente de suas respectivas populações.

Para o teste t, as variâncias das duas amostras devem ser estatisticamente iguais, verificadas por um teste de variância, como o de Fisher (Sheskin, 2000). Caso as variâncias sejam significativamente diferentes, deve-se utilizar o teste de Welch (Sheskin, 2000), que é uma adaptação do teste-t para tratar amostras dessa natureza.

A Distribuição Normal

A variação natural de muitos processos industriais é considerada aleatória. Uma peça é fabricada dentro de limites de engenharia. Em uma amostra, o diâmetro da peça, por exemplo, corresponde a uma média mais ou menos uma pequena variação, com nível similar de tendência para mais ou para menos. Logo, a grande maioria das peças encaixa-se dentro dos limites de especificação, e poucas peças ficam abaixo ou acima dos limites. Além disso, há uma simetria na distribuição dos diâmetros em relação ao valor da média.

Embora as distribuições de muitos processos possam assumir uma variedade de formas, muitas variáveis observadas possuem uma distribuição de freqüências que é, aproximadamente, uma distribuição de probabilidade Normal. Vale lembrar que a probabilidade é a chance real de ocorrer um determinado evento, isto é, a chance de ocorrer uma medida em um determinado intervalo. Por exemplo, a freqüência relativa desse intervalo, observada a partir de uma amostra de medidas, é a aproximação da probabilidade. A distribuição de freqüências é uma aproximação da distribuição de probabilidades.

Para determinar a área sob uma curva de distribuição Normal deve-se conhecer dois estimadores estatísticos (também chamados de parâmetros): média μ e desvio padrão σ. O gráfico da Figura 3.1 mostra algumas áreas importantes. A média mais ou menos três vezes o desvio padrão cobre 99,73% dos dados. É comum esse limite ser adotado para identificar ruído nos dados ou erros de medição(Pukelsheim, 1994), mas é utilizado apenas para a distribuição Normal.

A comparação entre duas amostras com distribuição Normal pode ser realizada usando o teste t. Porém, existem outras distribuições, chamadas Não-Normais, como as apresentadas na Figura 3.2. Para tais distribuições, os parâmetros para sua construção não são μ

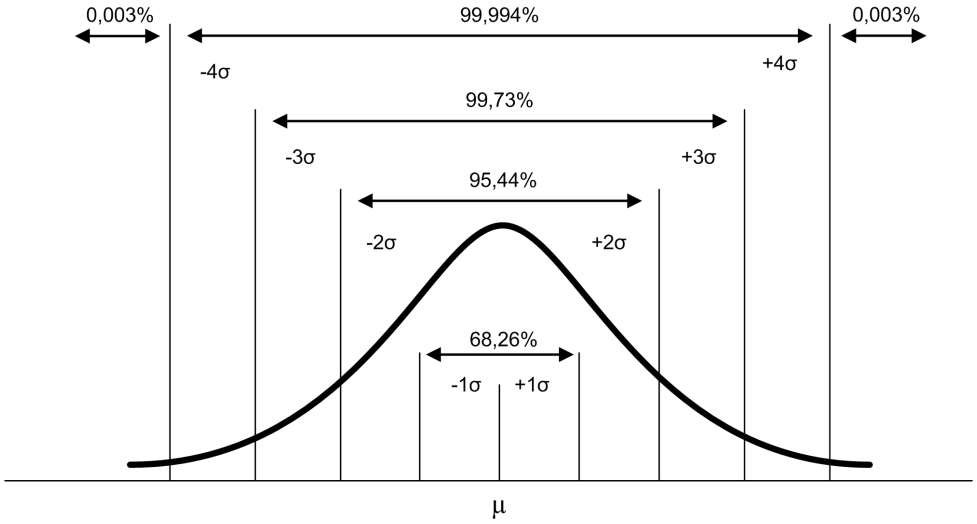


Figura 3.1: Áreas sob uma curva Normal de média μ e desvio padrão σ.

e σ. Portanto, analisar uma distribuição Não-Normal por meio desses parâmetros torna-se inadequado. O mesmo vale para testes de média. Desse modo, para conjuntos de dados em que não há uma aderência adequada (Sheskin, 2000) de funções de distribuição de probabilidades a partir dos parâmetros μ e σ, como a Normal, a t de Student, dentre outras (ver Figura 3.2), é importante que sejam utilizados testes para distribuições Não-Normais ou, mais especificamente, Testes Não-Paramétricos, conforme apresentado a seguir.

3.3.2 Estatística Não-Paramétrica para Testes de Média

Técnicas estatísticas Não-Paramétricas podem ser aplicadas a dados contínuos de qualquer natureza, visto que não exigem suposições quanto à distribuição da população da qual se tenha retirado amostras para análises. Essa característica as torna mais indicadas para análises de dados qualitativos. Isso significa que podem ser aplicadas a dados que se disponham simplesmente em ordem, contrariamente à estatística paramétrica, em que as variáveis são intervalares, na maioria das vezes. Duas características importantes da estatística Não-Paramétrica podem ser destacadas:

1. Exigem poucos cálculos, o que as torna aplicáveis para análise de pequenas amostras;
2. Independem dos parâmetros populacionais e amostrais, como média, variância e desvio padrão.

Como justificativa de uso de técnicas Não-Paramétricas para a comparação de médias, tem-se o seguinte exemplo: suponha dois algoritmos distintos, A_1 e A_2, executados sobre um mesmo problema de minimização, no qual $O^* = 0$. Ambos algoritmos foram executados $r = 1000$ vezes, gerando duas amostras de melhores resultados encontrados em cada

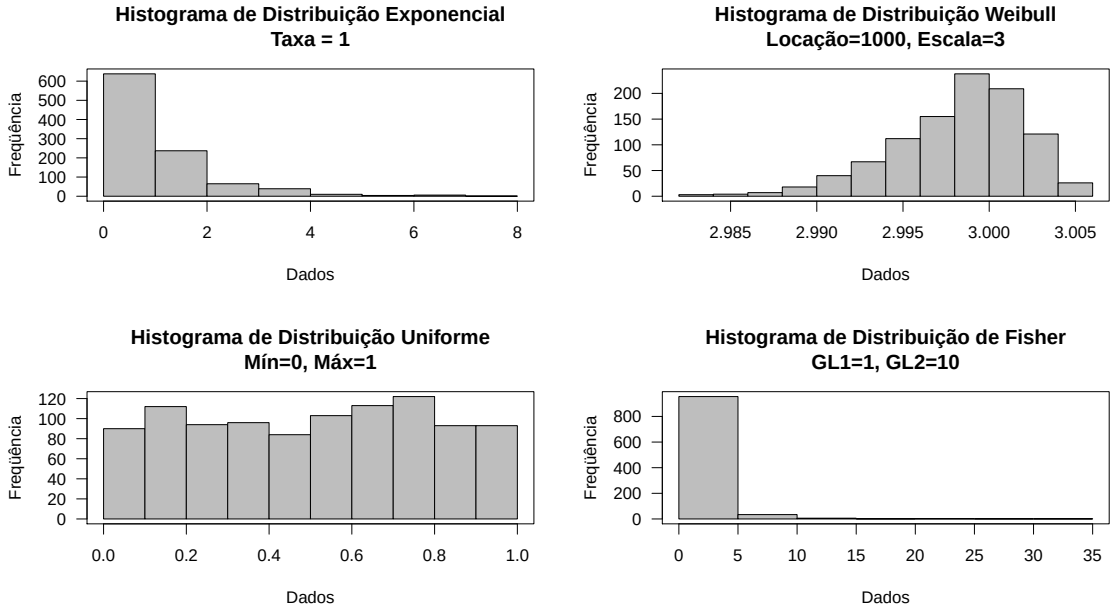


Figura 3.2: Outras distribuições (obtidas experimentalmente) com seus parâmetros (Sheskin, 2000): Exponencial, Weibull (Rinne, 2008), Uniforme e de Fisher (GL = Graus de Liberdade).

repetição, ou seja: O_{A1} e O_{A2}. A média dessas duas amostras é igual. Logo, pode-se supor que o desempenho dos dois algoritmos é igual, com relação à qualidade do resultado. Porém, é interessante avaliar o comportamento da amostra.

Na Figura 3.3 tem-se o histograma das amostras obtidas para A_1 e A_2. É possível perceber que o comportamento de A_1 assemelha-se a uma distribuição Exponencial, enquanto que o comportamento de A_2 assemelha-se a uma distribuição Normal. Como é possível verificar na Figura 3.3, ambas distribuições possuem média igual a $0,15$.

No histograma de A_1, é possível notar que o intervalo dos dados vai de 0 a aproximadamente 1, enquanto que no histograma de A_2, o intervalo dos dados vai de aproximadamente 0,03 a cerca de 0,3. Logo, apesar de possuírem média igual, a variância é bem distinta. Isso significa que não se pode afirmar que os dois conjuntos de dados pertencem a uma mesma distribuição e que possuem comportamento similar. Essa conclusão é evidenciada pelo gráfico da Figura 3.4 contendo os histogramas de A_1 e A_2, juntamente com uma linha reta tracejada indicando o valor da média.

Outra característica importante que pode ser vista na Figura 3.4 é o valor mínimo encontrado por cada algoritmo, juntamente com a freqüência. Como informado na suposição, trata-se de um problema de minimização com $O* = 0$. No histograma de A_1, pode-se perceber uma grande quantidade de resultados com valores bem próximos a 0. O mesmo não ocorre no histograma de A_2, cujo valor mínimo é cerca de $0,03$ e, ainda, ocorre com freqüência muito baixa. Utilizando-se o critério de TS, apresentado na Seção 3.1,

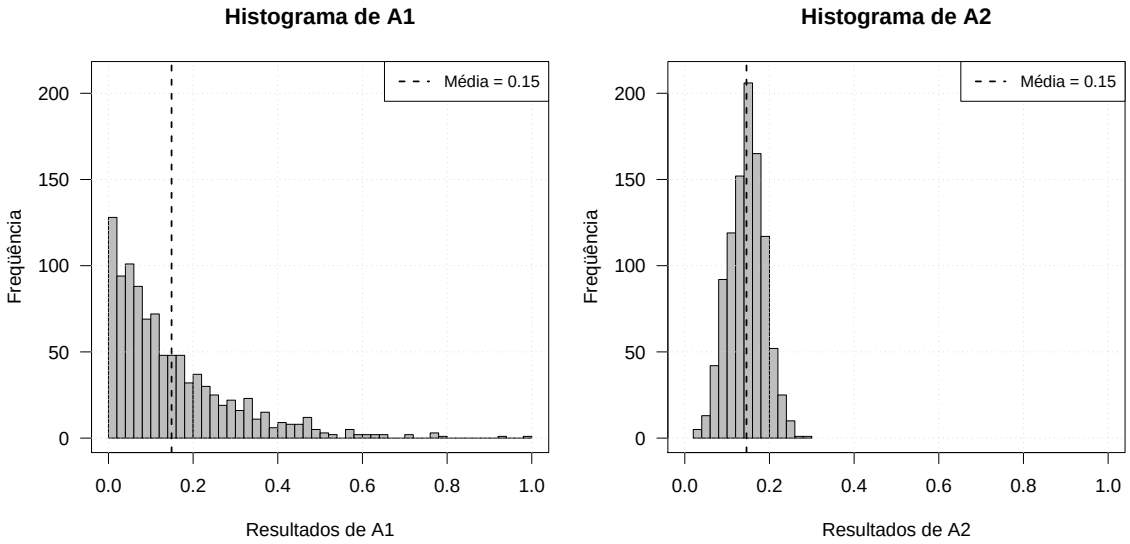


Figura 3.3: Histograma dos resultados de dois algoritmos fictícios A_1 e A_2.

poder-se afirmar que, apesar de uma grande variância nos resultados, o algoritmo A_1 conseguiu encontrar o ótimo global do problema, ao contrário do algoritmo A_2.

Não sendo adequado comparar as médias de maneira direta, visto que os resultados de A_1 não seguem uma distribuição Normal, é indicado o uso de um teste de médias Não-Paramétrico, como o de Wilcoxon Mann-Whitney, descrito na seqüência.

Teste de Wilcoxon Mann-Whitney

O teste de Wilcoxon Mann-Whitney é uma alternativa Não-Paramétrica ao teste t de duas amostras, e considera a magnitude da diferença entre cada par comparado, no caso, os resultados de A_1 contra A_2. É válido para dados a partir de qualquer distribuição, Normal ou não, além de ser menos sensível aos *outliers*[1] do que o teste t de duas amostras. As duas amostras podem ser de tamanhos distintos.

Como a média e o desvio padrão são informativos apenas para dados com distribuição Normal e os resultados das execuções para problemas de otimização contínua são, em geral, Não-Normais (Bartz-Beielstein, 2006), o teste Não-Paramétrico de Wilcoxon Mann-Whitney pode ser usado para medir o nível de confiança da suposição de igualdade das médias de dois conjuntos de resultados. A interpretação do resultado do teste pode ser baseada no *p-valor*. Se *p-valor*$< \alpha$, rejeita-se H_0 e assume-se que as amostras possuem médias diferentes. Nesse caso, o pesquisador pode decidir qual das médias é melhor. Por outro lado, se *p-valor*$\geq \alpha$, as médias são consideradas iguais.

[1] Considerados como ruído na amostra, são observações que estão a alguns desvios padrão (comumente 3σ) de distância da média da amostra. De acordo com (Tan et al., 2005) um outlier é um elemento de dados que, em algum sentido, tem características que são diferentes da maioria dos outros elementos do conjunto de dados.

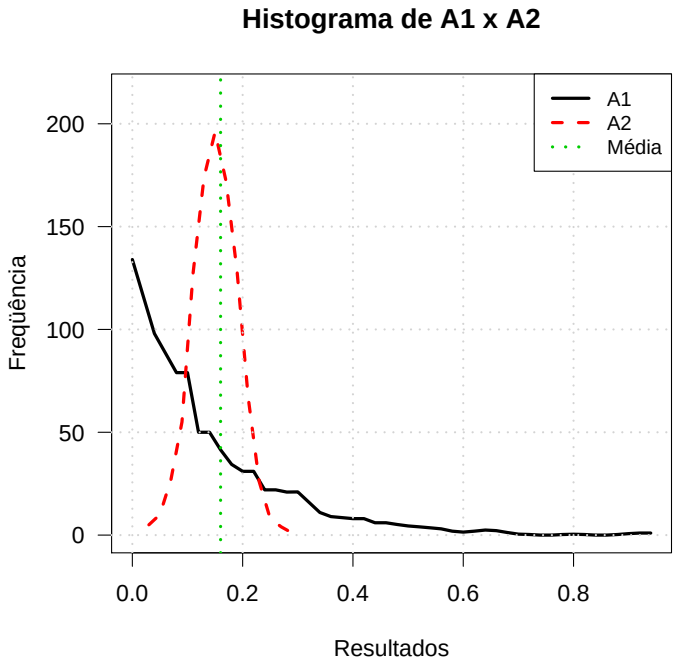


Figura 3.4: Superposição dos histogramas dos resultados de A_1 e A_2.

Para comparação de TSs, não se deve utilizar testes de médias, mas testes de proporção, como apresentado a seguir.

3.3.3 Estatística para Testes de Proporção

O objetivo desse teste é fornecer uma estatística para comparação das proporções de duas amostras (Montgomery & Runger, 1998). Nesse caso, o pesquisador observa as proporções em dois grupos distintos e as compara com o objetivo de verificar se os grupos diferem ou não em relação à resposta de interesse. Um teste comumente utilizado é o teste χ^2 de Pearson (Sheskin, 2000). Para esse tipo de análise, é interessante que o número de observações seja o mesmo nas duas amostras. Contudo, o teste pode ser aplicado com duas amostras de tamanhos distintos.

Em geral, utiliza-se o teste bi-caudal, cujo objetivo é testar apenas se as proporções são iguais ou diferentes e não estabelecer qual delas é maior ou menor. Com isso, é possível determinar, dado α, se uma taxa de sucesso de 90% para A_1 é diferente de uma taxa de sucesso de 88% para A_2, sendo ambas amostras com tamanho igual.

3.4 Considerações Finais

Neste Capítulo foram apresentadas métricas de avaliação de metaheurísticas e conceitos de comparação estatística. Sobre as métricas de avaliação, é possível perceber que existem

diversos critérios e que os resultados podem ser apresentados de maneira incompleta, resultando em dificuldades na comparação com resultados encontrados na literatura. Além das métricas apresentadas, foi proposta a análise de domínio reduzido, para analisar a taxa de sucesso do algoritmo quando inicializado com soluções mais promissoras.

Nesta Tese são propostas técnicas para melhorar a eficiência de algoritmos de otimização global. Nesse sentido, são realizados dois tipos de comparações, em problemas contínuos e binários (ver Capítulo 8). Para problemas contínuos foram adotadas os seguintes procedimentos, de acordo com o conjunto de informações disponível:

1. Comparação dos resultados de um algoritmo de otimização global aplicado a diversos problemas de *benchmark* em relação ao mesmo algoritmo quando são utilizadas as técnicas propostas neste trabalho. Nesse caso, são utilizados o DS e um teste estatístico das médias (Wilcoxon, ver Seção 3.3) de $Aval_{O*}$. Além disso, para facilitar a comparação com outras técnicas, são apresentadas, estatísticas de mínimo, 1^o quartil, mediana, média, 3^o quartil, máximo e desvio padrão;

2. Comparação dos resultados com algoritmos avançados de otimização da literatura. Como, para esses algoritmos, os pesquisadores informam, em geral, apenas os valores de Aval e TS, é possível calcular o valor da DS. Contudo, comparações estatísticas não podem ser realizadas.

Para os problemas binários, a comparação será realizada seguindo o procedimento adotado em trabalhos da área, por meio das quantidades de avaliações e de gerações. Essas métricas serão aplicadas para verificar o comportamento do algoritmo original quando aplicadas as técnicas de melhoria de eficiência propostas nesta Tese.

4

Determinação de Regiões Promissoras

Neste Capítulo são apresentados conceitos e métodos relacionados à identificação de regiões promissoras em problemas otimização global. Nesse sentido, este Capítulo está organizado da seguinte maneira: métodos de inicialização de EAs (Seção 4.1), conceito de regiões promissoras (Seção 4.2), EAs híbridos para detecção de regiões promissoras (Seção 4.3) e modelagem do espaço de soluções para orientar a busca (Seção 4.4). Além disso, são apresentados resultados sobre dois estudos realizados para avaliar a importância e o potencial da pesquisa de técnicas de busca por regiões promissoras: um (Seção 4.5) baseado no princípio de Redução do Espaço de Busca, apresentado na Seção 3.1, outro fundamentado na Análise de Tendência da Amostra (Seção 4.6).

4.1 Métodos de Inicialização

A criação da população inicial de um EA é uma etapa importante para o processo de busca. As soluções iniciais devem corresponder a uma amostragem adequada do espaço de soluções, de modo a facilitar a busca pelo ótimo global. Por exemplo, uma amostragem inapropriada pode gerar uma convergência prematura caso as soluções iniciais estejam muito próximas entre si. Por outro lado, pode resultar em um grande número de iterações para se aproximar do ótimo global, caso as soluções iniciais estejam muito distantes entre si e do próprio ótimo global. A gravidade desses efeitos varia para cada EA. Em geral, tais algoritmos têm sua população inicial gerada dentro do domínio do problema. As principais técnicas de inicialização são:

- Aleatória: é a inicialização mais comumente utilizada. Os indivíduos são gerados aleatoriamente dentro de limites pré-estabelecidos (domínio do problema),

amostrando o espaço de busca de maneira aproximadamente uniforme, como na Figura 4.1.

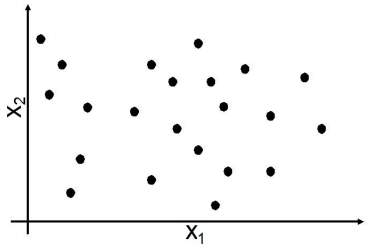


Figura 4.1: Inicialização aleatória.

- Linear: amostra o espaço de busca de maneira exatamente uniforme, utilizando um número regular de pontos amostrados a cada fração do espaço. Esse modo de inicialização não é muito comum, pois o número de indivíduos necessário para uma amostragem linear adequada, em um problema de alta dimensão, pode ser consideravelmente alto. Um exemplo em 2D é mostrado na Figura 4.2.

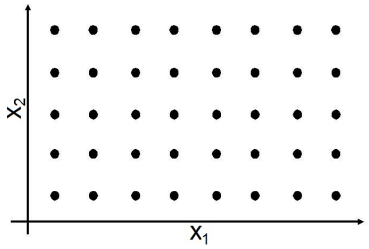


Figura 4.2: Inicialização linear.

- Heurística (Whitley et al., 1998): utiliza-se conhecimento prévio sobre o domínio do problema para gerar amostras em regiões do espaço de busca com maior potencial de conter o ótimo (ver Figura 4.3).

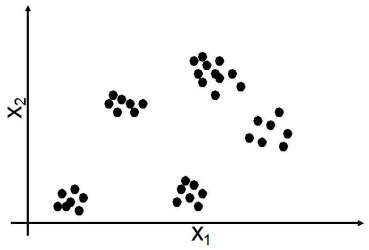


Figura 4.3: Inicialização heurística.

As abordagens de melhoria de algoritmos de otimização propostas nesta Tese (nas Seções 4.5 e 4.6, e nos Capítulos 5 e 7) podem ser consideradas heurísticas de inicialização, pois possibilitam reduzir o espaço de busca a uma ou mais regiões consideradas importantes, ou seja, a uma ou mais regiões promissoras. Isso significa que o algoritmo de otimização será inicializado com um *conhecimento prévio* de onde existe uma maior probabilidade de se encontrar o ótimo global.

4.2 Regiões Promissoras

Uma região promissora (Yan & Changrui, 2007) pode ser descrita como um subconjunto do espaço de busca de um problema, que contenha uma substancial quantidade de soluções de alta qualidade. Isso significa que essa região deve ser melhor explorada, pois existe uma grande chance da melhor solução do problema estar em tal região.

Para identificar regiões promissoras, o espaço de busca precisa ser explorado, inicialmente, sem privilegiar regiões. Para isso, pode-se usar, por exemplo, uma inicialização aleatória (ver Seção 4.1). Quando identificadas, tais regiões devem ser privilegiadas pelo algoritmo de otimização para melhor explorar esses subconjuntos na busca pelo ótimo global. A Figura 4.4 ilustra esse processo. Os pontos pretos compõem o conjunto atual de soluções. Os pontos circulados são as soluções selecionados para recombinação de acordo com algum critério (soluções *pais*, nos termos de GAs), pois estão em regiões promissoras. Os pontos marcados com $\times$ são as soluções que serão substituídas pelas novas soluções, que correspondem aos pontos em cinza, gerados próximos aos pais.

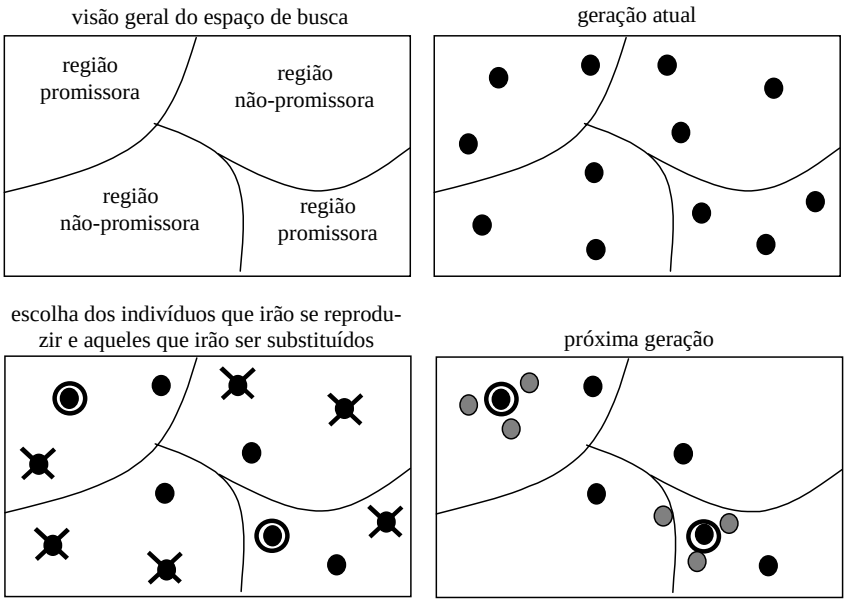


Figura 4.4: Exemplo de divisão do espaço de busca (Von Zuben, 2004).

Algoritmos de busca local são procedimentos que podem ser utilizados para melhorar a

precisão de algoritmos de otimização global visto que podem realizar pequenas melhorias em uma determinada solução de modo a encontrar o ótimo local da vizinhança. Porém, o uso dessas estratégias pode resultar em um considerável aumento do custo computacional, por ocorrer uma intensificação de busca local. Desse modo, são necessárias estratégias para uma aplicação eficiente desses algoritmos em regiões promissoras. Uma estratégia é explorar as regiões em que estão apenas os indivíduos de mais alta qualidade (Yen & Lee, 1997), ignorando indivíduos de qualidade intermediária ou baixa. Indivíduos contidos em regiões promissoras, ou que estejam próximos a soluções de alta qualidade, são denominados *soluções promissoras*, assumindo que ainda não tenham sido avaliados. Após avaliados, podem ser classificados como indivíduos de baixa, intermediária, alta qualidade, etc.

Um problema que surge então é como identificar as regiões promissoras a partir de uma amostra de soluções. Uma possibilidade é o uso de heurísticas que caracterizam uma região como promissora para um certo problema. De acordo com Goldberg (Goldberg, 1989), esquemas (partes do cromossomo compartilhadas entre vários indivíduos) com avaliação acima da média podem ser utilizados para identificar regiões promissoras. Por manterem algumas de suas características genéticas, descendentes produzidos por recombinação tendem a habitar as mesmas regiões dos indivíduos que lhes deram origem. Como os melhores indivíduos tendem a ser selecionados para se criar a próxima população, os esquemas relevantes tendem a ser propagados para as gerações seguintes (Holland, 1962). Esse comportamento reprodutivo é responsável por aumentar a amostragem nas regiões com melhores resultados (regiões promissoras), de modo que o GA intensifica a busca em torno dos melhores indivíduos.

Pode-se supor que características genéticas que são responsáveis pela alta qualidade de um indivíduo tendem a se disseminar na população por meio da reprodução dos indivíduos que as contém. Assim, a população tende a convergir para um determinado perfil de indivíduos em que tais características predominam. Uma população converge para um determinado esquema quando é formada por cópias/instâncias desse esquema. Esse problema, chamado clonagem, causa uma perda na diversidade populacional, fazendo com que o algoritmo deixe de explorar outras regiões do espaço de busca. Durante o processo de evolução, indivíduos podem se aglomerar em determinadas regiões. A localização desses grupos pode fornecer informação sobre quais regiões têm maior potencial de possuírem soluções de alta qualidade. Nesse sentido, pode-se dizer que regiões que possuam uma grande quantidade de indivíduos de alta qualidade podem ser consideradas promissoras.

Como pode ser visto na Figura 4.5, as regiões promissoras funcionam como atratoras de indivíduos. Isso pode ser explicado pelo fato dos melhores indivíduos serem mais selecionados para reprodução e, portanto, gerarem mais descendentes com suas características. Logo, os descendentes tendem a se localizar em regiões próximas a seus pais,

aumentando a amostragem nessas regiões.

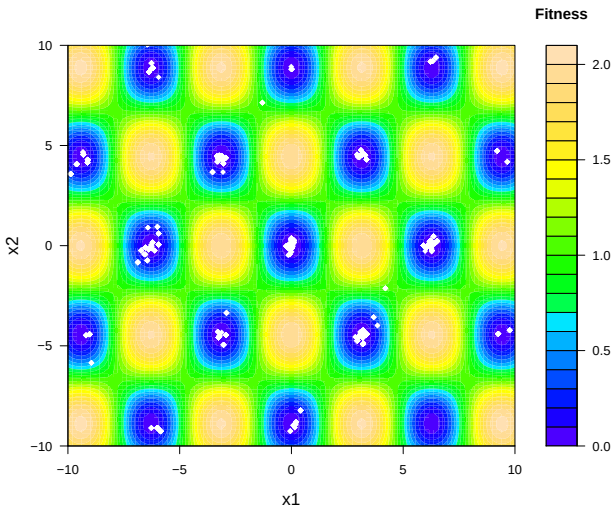


Figura 4.5: Curvas de nível para a função Griewangk e concentração de indivíduos (pontos brancos) em regiões promissoras (ver Seção 4.2). Uma representação dessa função em 3D pode ser vista na Figura 4.9.

Métodos heurísticos de melhoria iterativa (como algoritmos de busca local) podem ser combinados com EAs de diferentes maneiras para se resolver problemas de otimização, dando origem a algoritmos híbridos, possibilitando maior eficácia e precisão do processo de busca. Por outro lado, trabalhos envolvendo algoritmos híbridos podem causar um aumento no número de avaliações da função objetivo e, conseqüentemente, no custo computacional (Birru et al., 1999). Esses aspectos são discutidos na Seção 4.3.

4.3 Métodos Híbridos e Identificação de Regiões Promissoras

O principal desafio dos métodos híbridos que utilizam algoritmos de busca local é encontrar estratégias eficientes para cobrir o espaço de busca, de modo que a busca local seja realizada somente em regiões ou indivíduos realmente promissores, evitando o alto custo computacional gerado pela intensificação da busca. No Algoritmo Híbrido Contínuo (CHA, do inglês *Continuous Hybrid Algorithm*) (Chelouah & Siarry, 2003), o processo evolutivo ocorre até ser detectada uma região promissora, a qual é identificada pela perda de diversidade populacional quando a maior distância entre o melhor indivíduo da população e outros indivíduos é menor do que um determinado raio. A partir desse estágio é realizada uma busca local na região encontrada.

Uma outra abordagem proposta em (Liang et al., 1999) parte do princípio de que a superfície da função objetivo correspondente ao espaço de busca, ou parte dele, pode ser aproximada e suavizada sem mudar drasticamente sua natureza (o valor e a posição do ótimo global). O propósito inicial é que, dado um modelo simplificado (aproximado) como

uma parábola, cuja posição de mínimo é facilmente calculada, construído a partir de uma amostra, é possível prever uma região promissora no espaço de busca original, localizada em torno da posição do mínimo do modelo. A partir desse ponto, é realizada uma busca intensificada, utilizando a função objetivo original.

No algoritmo ECS (ver Seção 2.3.5) é utilizado um algoritmo de agrupamento para identificar grupos de indivíduos muito próximos entre si, o que pode significar uma região de alta qualidade que deve ser melhor explorada por um algoritmo de busca local.

No caso de um modelo de maior complexidade, é necessário o uso de algum algoritmo de otimização para encontrar o ponto mínimo desse modelo. Nesse caso, porém, não existe custo da função objetivo. Contudo, alguns problemas com essa abordagem são apresentados na Seção 4.4.

4.4 Modelagem do Espaço de Busca

Uma das grandes dificuldades na aplicação de EAs em problemas do mundo real de larga-escala é a grande quantidade de avaliações da função objetivo usualmente requeridas para atingir um resultado satisfatório. Contudo, essas avaliações não são sempre triviais de serem realizadas. Por exemplo, uma função objetivo explícita pode não existir ou a sua avaliação ser computacionalmente custosa. Em ambos os casos, pode reduzir o número de avaliações por meio de uma estimativa da função objetivo usando um modelo aproximado. Diversos modelos têm sido utilizados para realizar aproximação de funções. Dentre os mais utilizados, podem-se citar: Polinômios (também conhecido como Metodologia de Superfície de Resposta) (Myers & Montgomery, 2002), Redes Neurais Artificiais (ANN, do inglês *Artificial Neural Network*)(Haykin, 1999) e Máquinas de Vetores de Suporte (SVM, do inglês *Support Vector Machines*) (Schlkopf & Smola, 2001).

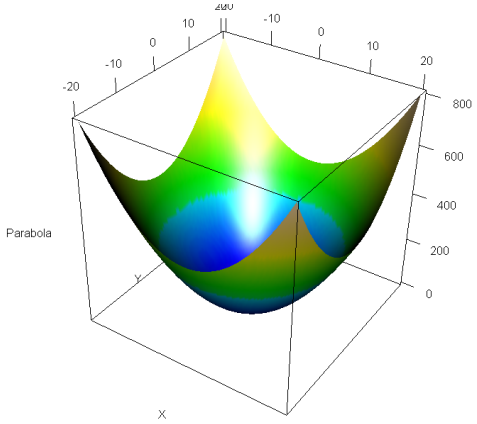
Após testes experimentais utilizando as três técnicas de aproximação citadas anteriormente em diversas funções de *benchmark*, foi verificado que a SVM apresentou uma aproximação mais precisa da superfície do espaço de busca. Uma revisão bibliográfica sobre diversas técnicas de aproximação de funções pode ser vista em (Jin, 2005).

Para ilustrar o funcionamento da modelagem de funções, são apresentados a seguir os resultados da aplicação de modelagem utilizando SVM em 4 funções matemáticas comumente utilizadas como *benchmark* na literatura (ver Tabela 4.1). Para esse teste foram utilizadas apenas *duas* dimensões ($D = 2$), o que já é suficiente para que se tenha idéia dos efeitos da modelagem do espaço de busca. Nas Figuras 4.6, 4.7, 4.8 e 4.9 tem-se a representação gráfica das funções de teste.

A função Parábola (Figura 4.6) apresenta um grau de dificuldade extremamente baixo, de modo que um polinômio quadrático é capaz de realizar uma aproximação perfeita da função. Por outro lado, a função Rosenbrock (Figura 4.7) possui um grande vale, o que

Tabela 4.1: Funções de teste utilizadas nos experimentos de modelagem do espaço de busca.

Função de Teste	Definição
Parábola	$f(x) = \sum_{i=1}^{D} x_i^2$
Rosenbrock	$f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
Rastringin	$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$
Griewangk	$f(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \frac{x_i}{\sqrt{i}} + 1$



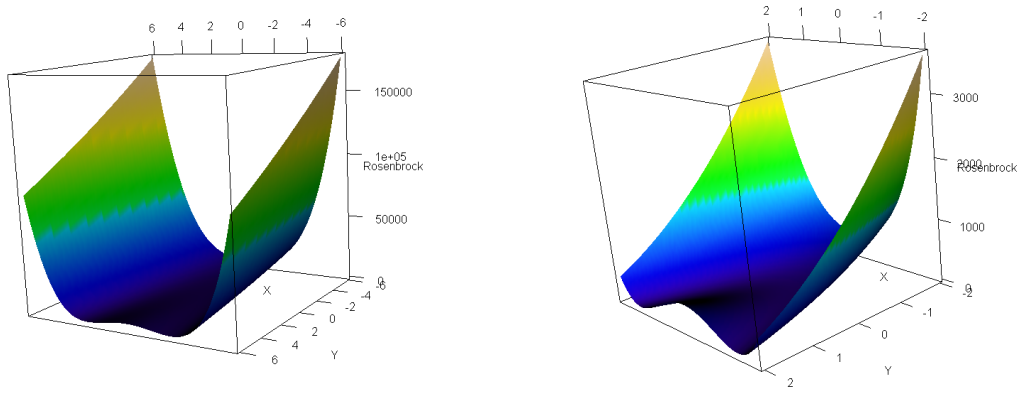
Intervalo de -20 a 20 para x e y.

Figura 4.6: Função Parábola.

direciona o algoritmo de busca rapidamente para um intervalo reduzido na região central. Porém, esta região é praticamente plana, dificultando a busca. Outra característica importante é um pequeno máximo local que se encontra próximo ao ponto nas coordenadas *x=0, y=6* (Figura 4.7a) e *x=0, y=2* (Figura 4.7b).

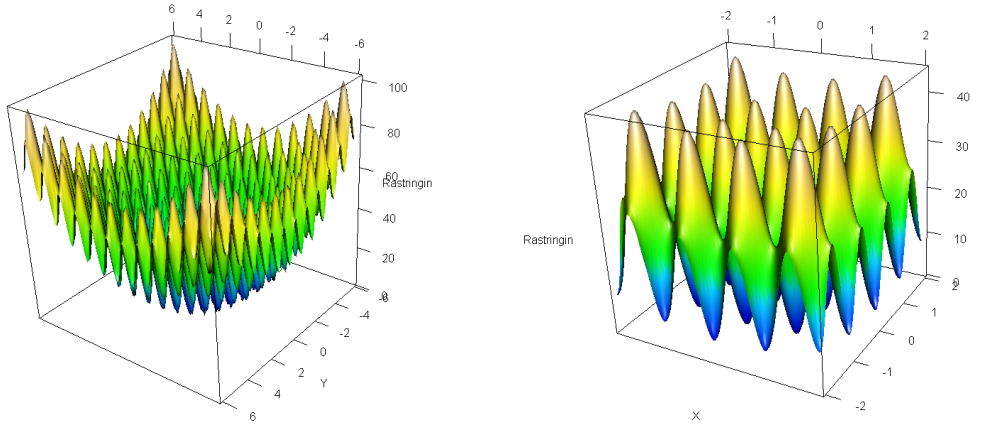
As funções Rastringin e Griewangk (Figuras 4.8 e 4.9) possuem comportamento essencialmente similar ao da função Parábola. Porém, são compostas por uma grande quantidade de ótimos locais, sendo que na função Griewangk os ótimos são mais acentuados e os espaçamentos entre eles são menores, aumentando o grau de dificuldade na busca.

A função Rastringin será usada na demonstração da evolução de uma modelagem em uma dimensão ($D = 1$). Na Figura 4.10 é apresentado um conjunto de imagens referentes à construção de um modelo aproximado por uma SVM. É possível notar que à medida que se aumenta a quantidade de pontos, gerados aleatoriamente dentro do domínio do problema, a qualidade do modelo aumenta significativamente. Ao final, tem-se uma aproximação de alta qualidade. Porém, essa quantidade de pontos (142) é necessária para apenas uma dimensão do problema. Assim, um modelo confiável requereria $142 * D$ pontos. Além



(a) Intervalo de -6 a 6 para x e y. (b) Intervalo de -2 a 2 para x e y.

Figura 4.7: Função Rosenbrock.



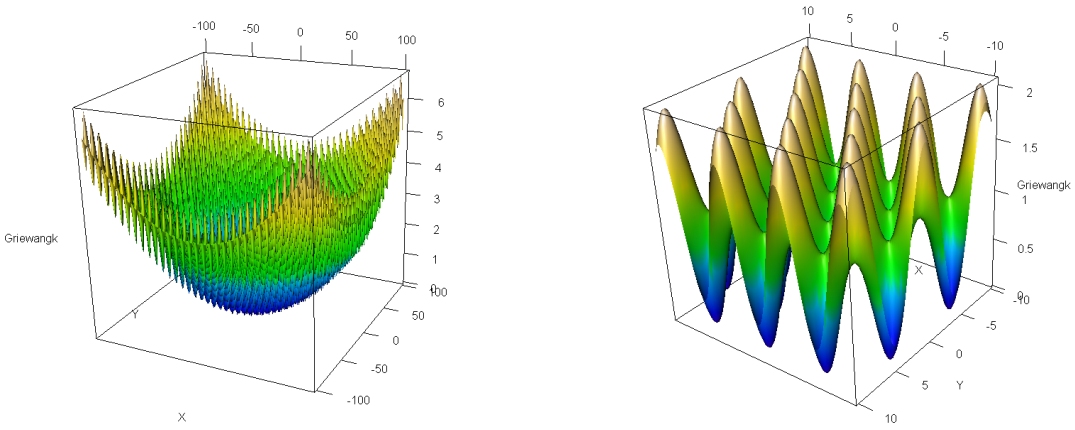
(a) Intervalo de -6 a 6 para x e y. (b) Intervalo de -2 a 2 para x e y.

Figura 4.8: Função Rastringin.

disso, o intervalo utilizado é relativamente pequeno ([-6; 6]). Um intervalo maior exigiria uma quantidade de pontos ainda mais alta para que o espaço de busca possa ser amostrado adequadamente para $D > 1$.

Além disso, o aumento de D multiplicaria esse número de pontos. Essa afirmação fica clara nas Figuras 4.11, 4.12 e 4.13, agora utilizando problemas de duas dimensões ($D = 2$). Para esses casos, os parâmetros (mais detalhes em (Schlkopf & Smola, 2001)) da SVM empregados na modelagem foram determinados empiricamente e são apresentados na Tabela 4.2.

Com a construção de modelos não se espera que esses sejam idênticos à função original, mas que, dentre outras características: 1) apresentem características similares à função original; 2) mantenham o ótimo global na mesma posição; 3) que o ótimo global continue sendo o ótimo global e que 4) possua uma superfície de resposta que permita que esse ótimo seja encontrado.



(a) Intervalo de -100 a 100 para x e y. (b) Intervalo de -10 a 10 para x e y.

Figura 4.9: Função Griewangk.

Tabela 4.2: Parâmetros da SVM na criação dos modelos.

Parâmetro	Valor
Tamanho da Amostra	$N = 100 * D = 200$
Kernel	Rede Neural RBF (Haykin, 1999)
Tipo	Regressão Epsilon
Custo	10
Gamma	50
Epsilon	$1e-3$

Pelas Figuras apresentadas anteriormente, pode-se perceber que os modelos da função de Rosenbrock preservaram parte significativa das características originais, como o vale e a região quase plana. Porém, a superfície obtida pelo modelo no intervalo original ([-6; 6], Figura 4.11a) não ficou tão suave, apresentando maior número de ótimos locais. Além disso, a protuberância não foi mapeada no intervalo original, aparecendo apenas quando o modelo foi gerado para um intervalo reduzido (Figura 4.11b). Esse tipo de efeito pode direcionar o algoritmo de busca a uma região distante do ótimo global.

Para a função Rastringin, o modelo no intervalo menor possui alta qualidade (Figura 4.12a) em comparação com a superfície real mostrada na Figura 4.8. Por outro lado, para o intervalo maior o modelo aproximado é de qualidade consideravelmente inferior.

Para a função Griewangk, a diferença do tamanho do intervalo para o qual o modelo é construído é drástica. Para o intervalo menor (Figura 4.13b), o modelo parece perfeito; entretanto, para o intervalo maior (Figura 4.13a), há quase nenhuma relação entre a superfície do modelo e a superfície original (Figura 4.9).

Com relação a esses testes e resultados, é importante comentar os seguintes aspectos:

1. O número de dimensões adotados para os testes foi $D = 2$. Na literatura, é comum avaliar metaheurísticas utilizando essas mesmas funções de teste, porém, com valores

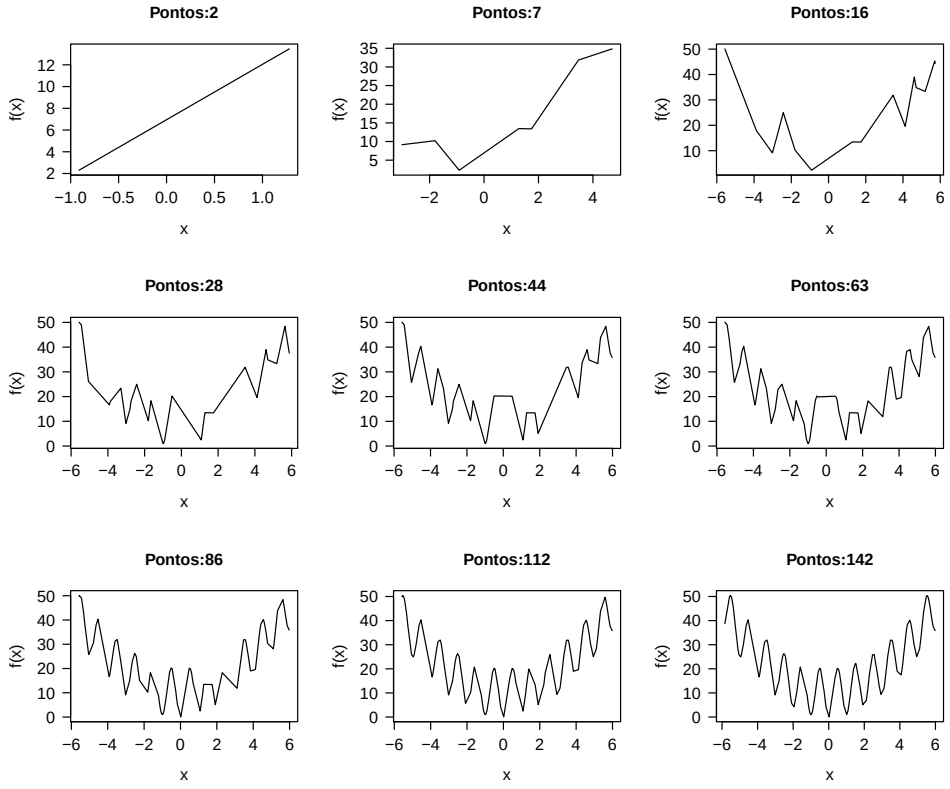
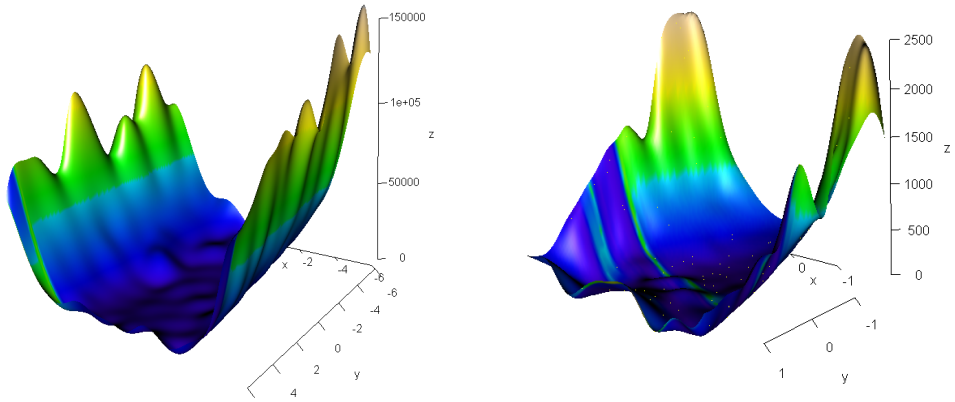


Figura 4.10: Modelos aproximados para a função Rastringin, onde o termo **Pontos** refere-se ao tamanho da amostra utilizada na construção dos modelos.

maiores de D como 10, 30, 50 e 100;

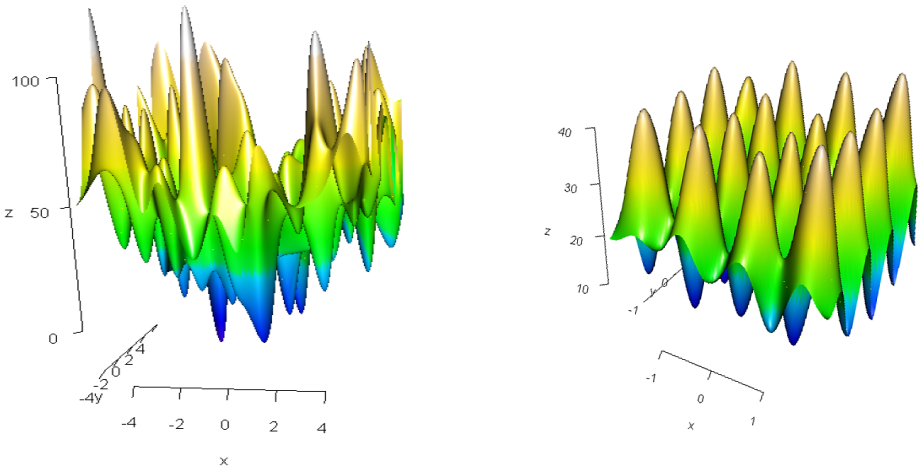
2. A quantidade de pontos utilizada para a modelagem foi $100 * D$. Supondo um problema com $D = 10$, poderia-se supor que seriam necessários 1000 pontos para se obter modelos com qualidade similar à dos modelos obtidos para $D = 2$. Esse aumento está relacionado ao quanto se necessita cobrir do espaço de busca para se construir um modelo de alta qualidade. Por exemplo, supondo um intervalo contínuo de -6 a 6, coberto com pontos espaçados de 1,0, para uma função bidimensional (como ilustra a Figura 4.2), haveria uma relação de 13 pontos (-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 e 6) por dimensão. Portanto, em duas dimensões seriam necessários $13^D = 13^2 = 196$ pontos;

3. Supondo $D = 10$ no mesmo intervalo, seriam necessários $13^{10} = 137.858.491.849$ pontos para se gerar uma malha linear (ver Seção 4.1) que mantivesse o nível de representatividade do espaço de busca. Porém, em experimentos realizados para a função Rastringin, o fator multiplicativo para se obter um modelo de alta qualidade foi 1000, resultando em $1000 * D = 10.000$ ao invés de $100 * D$ pontos, como sugerido no item anterior. Se o intervalo for de -100 a 100, como na função Griewangk, seriam



(a) Intervalo de -6 a 6 para x e y. (b) Intervalo de -2 a 2 para x e y.

Figura 4.11: Modelos da Função Rosenbrock.



(a) Intervalo de -6 a 6 para x e y. (b) Intervalo de -2 a 2 para x e y.

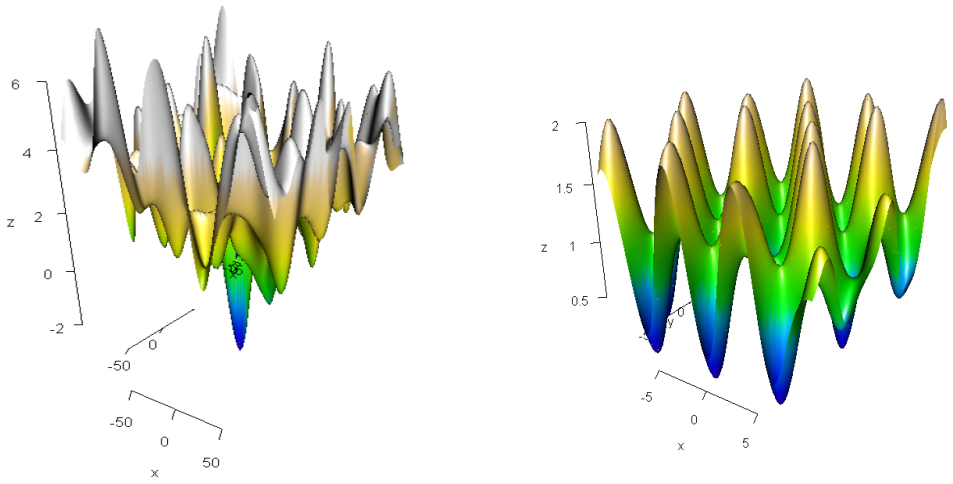
Figura 4.12: Modelo da Função Rastringin.

201 pontos por dimensão ao invés de 13, se mantidas as proporções. Isso pode tornar inviável a construção de modelos de alta precisão;

4. Algumas funções de *benchmark* apresentam superfícies substancialmente mais complexas, difíceis de serem modeladas (ver Figura 4.14 que ilustra algumas dessas funções). Conseqüentemente, necessitam de uma quantidade significativamente maior de pontos por dimensão;

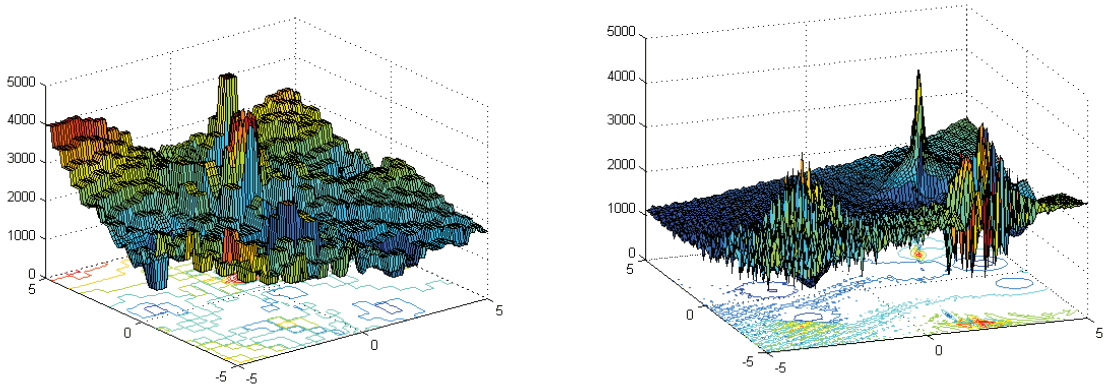
5. O modelo por SVM e os parâmetros utilizados foram adequados para os problemas com D=2. Porém, isso não garante que a SVM e esses mesmos parâmetros possam gerar modelos de alta qualidade para valores maiores de D;

6. A criação do modelo aproximado é, por si só, um problema de otimização no qual



(a) Intervalo de -100 a 100 para x e y. (b) Intervalo de -10 a 10 para x e y.

Figura 4.13: Modelo da Função Griewangk.



(a) f_{21} de (Suganthan et al., 2005). (b) f_{22} de (Suganthan et al., 2005).

Figura 4.14: Exemplos de funções de alta complexidade (Suganthan et al., 2005).

busca-se criar uma representação que se aproxime da superfície de busca real. Desse modo, deseja-se minimizar o erro existente entre os valores calculados/preditos pelo modelo e os valores calculados utilizando-se a função real. . Dada uma função objetivo de grandes complexidade e dimensionalidade, o custo computacional para se construir um modelo pode, também, tornar impraticável esse tipo de abordagem.

Diversas questões precisam ser solucionadas ao se aplicar modelagem de funções juntamente com EAs. Uma delas é definir o nível da aproximação a ser utilizado. Como pôde ser visto nas figuras desta Seção, quanto menor o espaço de busca que precisa ser coberto, melhor tende a ser o modelo aproximado. Portanto, muitos pesquisadores utilizam apenas modelagem local em uma região reduzida, ao invés de global (Liang et al., 1999, 2000).

Outra questão importante é a escolha do algoritmo de construção de modelos. Do

mesmo modo que não existe um algoritmo de otimização que obtenha o melhor desempenho em todos os problemas (NFLT - ver Seção 2.1), não existe um algoritmo de aproximação de função que encontre sempre o modelo mais preciso em relação à função original. Portanto, a escolha depende, em certo grau, do problema em questão.

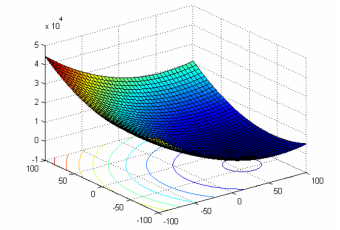
Após escolhido o algoritmo de aproximação, deve-se encontrar os parâmetros que possibilitam a construção de um modelo que se ajuste bem aos dados amostrais e que também seja capaz de apresentar um erro significativamente baixo ao calcular o valor aproximado de um novo ponto. Em outras palavras, o comportamento da superfície do modelo deve ser consideravelmente próximo ao comportamento da verdadeira superfície de resposta. É importante lembrar que devido à grande dimensionalidade do espaço de busca, em geral, é de grande dificuldade a obtenção de uma função de aproximação global com erro insignificante.

Apesar dessas questões, diversos pesquisadores têm apresentado trabalhos com resultados promissores. Em geral, são utilizados problemas mais bem condicionados, com poucos ótimos locais, mas cujo cálculo da função objetivo requer um alto tempo computacional, como em problemas de engenharia (Ong et al., 2004; Nair & Keane, 2001).

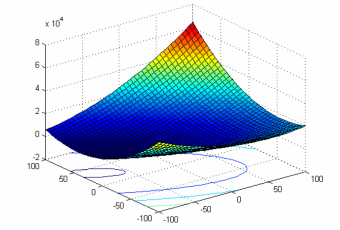
4.5 Redução do Espaço de Busca

Como apresentado na Seção 4.1, uma inicialização aleatória da população dentro dos limites da função objetivo é o processo mais comumente adotado em algoritmos de otimização global. Durante o processo de otimização, algoritmos de otimização global como o CHA e ECS (ver Seção 4.3) são capazes de identificar regiões promissoras utilizando análise de agrupamento de soluções, e intensificar a busca dentro delas, melhorando a eficiência do algoritmo. Nesse estudo, propõe-se encontrar regiões promissoras *antes* do processo de otimização. Essa etapa pode ser realizada, por exemplo, por um Algoritmo de Redução de Espaço de Busca (SRA) (Srinivas & Patnaik, 1991; Chen & Smith, 1999; Guo & Matta, 2003). Pode-se dizer que um SRA é uma técnica para determinar uma região promissora antes da aplicação de um algoritmo de otimização, em vez de determinar tal região durante o processo de otimização global.

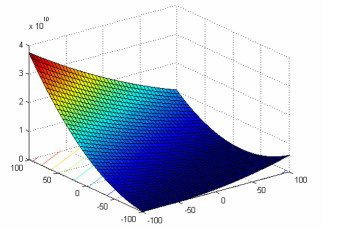
Com o objetivo de avaliar a validade e a viabilidade de SRAs, foi realizado um estudo, de acordo com a proposta de avaliação de algoritmos de otimização global apresentada na Seção 3.2, para verificar que nível de redução no espaço de busca é necessário para melhorar a eficácia dos seguintes algoritmos: GA, PSO e DE (ver Seção 2.1). Para tal, foram efetuadas execuções desses algoritmos inicializados em regiões reduzidas que continham o ótimo global para um conjunto de 12 funções de *benchmark* contínuas, propostas em (Suganthan et al., 2005), cujas representações em 3D podem ser vistas na Figura 4.15. Os testes foram feitos com dimensão $D = 10$ e um limite de $D * 1000$ avaliações.



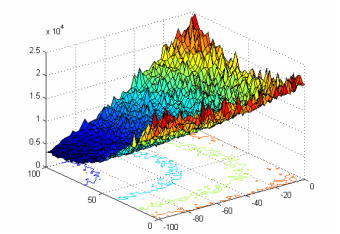
f_1: *Shifted Sphere Function*



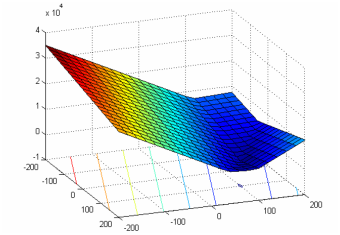
f_2: *Shifted Schwefel's Problem 1.2*



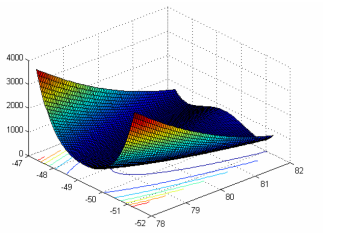
f_3: *Shifted Rotated High Conditioned Elliptic Function*



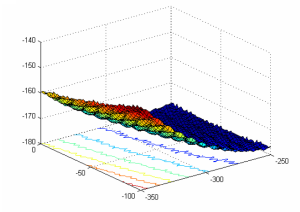
f_4: *Shifted Schwefel's Problem 1.2 with Noise in Fitness*



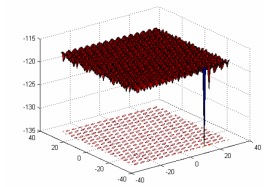
f_5: *Schwefel's Problem 2.6 with Global Optimum on Bounds*



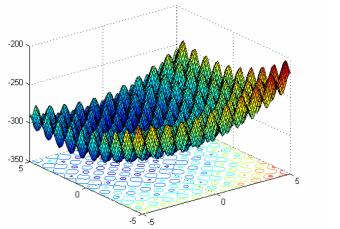
f_6: *Shifted Rosenbrock's Function*



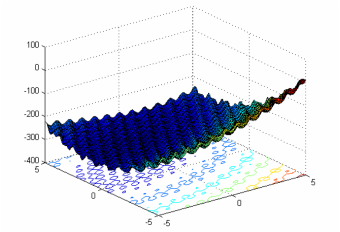
f_7: *Shifted Rotated Griewank's Function without Bounds*



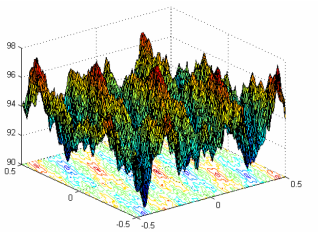
f_8: *Shifted Rotated Ackley's Function with Global Optimum on Bounds*



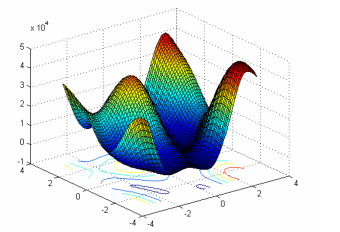
f_9: *Shifted Rastrigin's Function*



f_{10}: *Shifted Rotated Rastrigin's Function*



f_{11}: *Shifted Rotated Weierstrass Function*



f_{12}: *Schwefel's Problem 2.13*

Figura 4.15: Gráficos em 3D das funções de *benchmark* utilizadas nesse estudo, extraídos de (Suganthan et al., 2005).

Como busca-se comparar a eficácia, a única métrica de avaliação utilizada na comparação foi a porcentagem de vezes que o ótimo global foi encontrado (TS, Taxa de Sucesso, ver Seção 3.1) com um erro máximo permissível de $\varepsilon = 10^{-8}$ em relação ao valor do ótimo global.

Para cada teste (nível de redução) de cada função de *benchmark*, foram executadas 100 repetições. Em cada nível, o espaço de busca foi reduzido igualmente em todas as dimensões, sempre mantendo o ótimo global no interior da região. É importante informar que a redução é feita em todas as dimensões. Portanto, a redução do volume é exponencial.

Com a redução do espaço de busca em direção ao ótimo global, a população inicial é gerada dentro de uma região cada vez mais promissora. A Tabela 4.3 sintetiza os efeitos dos diferentes níveis de redução na TS de cada metaheurística, com um resumo mostrando apenas a quantidade de funções nos quais as metaheurísticas tiveram pelo menos 1% de sucesso. Por exemplo, o GA sem qualquer redução do espaço de busca (0%) conseguiu encontrar o ótimo global em 4 das 12 funções. Apenas quando o nível atingiu 95% de redução em cada uma das dimensões, o GA atingiu o ótimo global em outra função. O objetivo dessa tabela é mostrar que, com a redução, é possível fazer com que o algoritmo de otimização encontre o ótimo. Mais detalhes sobre quais funções tiveram melhor resultado e como foi a melhoria obtida nos valores de TS em cada função, para cada algoritmo testado, podem ser encontrados em (Melo & Delbem, 2008b).

Tabela 4.3: Resumo da análise de redução. Quantidade de funções de *benchmark* que o algoritmo de otimização (GA, PSO ou DE) encontrou o ótimo global.

Nível de redução por dimensão	GA	PSO	DE
0%	4	4	6
50%	4	4	6
70%	4	4	6
80%	4	4	6
90%	4	4	6
95%	5	5	8

É possível ver que, mesmo com as reduções de até 95% no intervalo de cada uma das dimensões (reduzindo de [-100; 100] para [-5; 5], por exemplo), as metaheurísticas não foram capazes de encontrar o ótimo global em todas as funções de teste. Portanto, foi possível concluir que, em problemas de alta complexidade e com grande número de variáveis, não pode ser esperado um desempenho adequado das metaheurísticas testadas, conforme o NFLT, apresentado na Seção 2.1.

Por outro lado, pode-se perceber que a quantidade de funções nas quais o ótimo global foi encontrado aumentou à medida que o espaço de busca foi reduzido. Além disso, como apresentado em (Melo & Delbem, 2008b), nas funções em que o ótimo global foi encontrado, a redução no espaço de busca possibilitou um aumento na TS. Como exemplo

tem-se o a DE na função f_6 que subiu de 63% para 93% e na função f_{12} que subiu de 53% para 90%. Na Tabela 4.4, tem-se os resultados para a DE pois, como apresentado na Tabela 4.3, foi a metaheurística mais beneficiada pela redução.

Tabela 4.4: TS usando DE.

Redução	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
0%	100%	93%	0%	83%	100%	63%	0%	0%	0%	0%	0%	53%
50%	100%	97%	0%	100%	100%	67%	0%	0%	0%	0%	0%	53%
70%	100%	100%	0%	100%	100%	80%	0%	0%	0%	0%	0%	67%
80%	100%	100%	0%	100%	100%	93%	0%	0%	0%	0%	0%	67%
90%	100%	100%	0%	100%	100%	73%	0%	0%	0%	0%	0%	70%
95%	100%	100%	0%	100%	100%	93%	0%	0%	100%	30%	0%	90%

A partir desses resultados, tem-se duas conclusões importantes. Primeiro, alguns algoritmos de otimização podem não ser capazes de encontrar o ótimo global mesmo com uma população sendo iniciada bem próxima a ele. Portanto, o principal entrave, em vários problemas, não está relacionado necessariamente à eficiência do SRA utilizado, mas à ineficiência do algoritmo de otimização.

Por outro lado, se o algoritmo de otimização for capaz de encontrar o ótimo global de uma determinada função objetivo, para espaços de busca reduzidos, existe uma grande probabilidade dele ser beneficiado por SRAs quando aplicado a um espaço de busca significativamente maior. Logo, um SRA eficaz pode ser substancialmente útil, reduzindo o número de avaliações da função objetivo necessários para se atingir o ótimo global e aumentando a TS.

4.6 Análise de Tendência da Amostra

Seguindo na direção contrária à dos métodos de otimização que utilizam construção de modelos de aproximação da função objetivo mais precisos e, por sua vez, mais *complexos* com o intuito de facilitar a determinação de regiões promissoras (ver Seções 4.3 e 4.4), foi proposto no desenvolvimento desta Tese, um algoritmo baseado em um método *simples* de Planejamento de Experimentos (Melo et al., 2007a) (Regressão Linear Múltipla (Montgomery, 1997)) .

A partir dos limites do espaço de busca, o algoritmo elimina iterativamente regiões próximas à borda com uma baixa probabilidade de conter o ótimo global. O método termina quando não puder garantir, com certa confiança, uma redução adicional. A abordagem proposta, chamado Algoritmo de Otimização de Domínio (DOA, do inglês *Domain Optimization Algorithm*) pertence à classe de SRAs.

O DOA usa um modelo simples que tenta garantir que o ótimo global está na região determinada como promissora. Em geral, algoritmos como EDAs (ver Seção 2.3.4) tentam

orientar a busca modelando a superfície de resposta da função objetivo por meio de modelos complexos. Isso é exatamente o oposto do funcionamento do DOA, que foi desenvolvido para se verificar o máximo de aumento de desempenho que um modelo simples pode oferecer a um método de otimização global. Descrições mais detalhadas do DOA e das técnicas de Planejamento de Experimentos utilizadas podem ser vistas em (Melo et al., 2007b,a).

Foi realizado um estudo com três metaheurísticas (GA, PSO e DE) em 12 funções de *benchmark* (ver Figura 4.15). O DOA é utilizado para determinar a região promissora em que será gerada a população inicial do GA, PSO e DE, além de fornecer os indivíduos da população inicial. As combinações de GA, PSO e DE com DOA foram denominadas, respectivamente: DOA+GA, DOA+PSO e DOA+DE. O objetivo foi verificar se a inicialização da metaheurísticas dentro de uma região encontrada pelo DOA seria capaz de aumentar a TS e/ou ao menos se as soluções finais seriam mais próximas do ótimo global. Foram obtidos os seguintes resultados (Melo et al., 2007a):

- **GA x (DOA+GA)**: de 12 funções de *benchmark*, o GA obteve *fitness* médio melhor em 6 e DOA+GA em 6. Contudo, a análise estatística acusou apenas uma diferença significativa nos resultados, sendo esta em favor do DOA+GA. Portanto, foi possível concluir que a combinação DOA+GA não foi capaz de melhorar significativamente o desempenho do GA. Porém, uma característica importante detectada nos resultados foi o fato de o GA ter uma TS=0. Isso é um indicativo de que o DOA não foi capaz de melhorar o desempenho do GA pelo fato de o GA não ter sido capaz de encontrar o ótimo global, nem mesmo para espaços de busca relativamente pequenos (ver Seção 4.5);

- **PSO x (DOA+PSO)**: o PSO obteve melhores valores de *fitness* médio em 5 das 12 funções de teste e o DOA+PSO em 7. De acordo com a análise estatística, o DOA melhorou o desempenho do PSO em 4 delas, perdeu em 2 e houve empate nas demais. A TS de ambos algoritmos foi igual, sendo que o ótimo global foi encontrado para 3 das funções;

- **DE x (DOA+DE)**: esse experimento mostrou que o DOA gera uma diferença significativa nos resultados obtidos pela DE. O DOA+DE foi melhor (obteve menor valor de *fitness* de médio) que a DE em todas as funções de teste. De acordo com o teste estatístico, houve igualdade em apenas 3 das funções. Além disso, a TS do DOA+DE foi superior em 3 dos 12 testes (ver Tabela 4.5).

Como a DE foi o algoritmo mais beneficiado pela aplicação do DOA, os resultados desse experimento são apresentados na Tabela 4.5. De acordo com os resultados, pode-se

concluir que a SRA proposta para determinar regiões promissoras é relevante. Ao analisar os resultados, foi possível perceber que, quando houve melhora da TS devido à combinação, o ótimo global encontrava-se próximo ao centro da região promissora determinada pelo DOA. Por outro lado, quando não houve melhora, o ótimo global estava fora e distante da região promissora. Provavelmente, a amostragem para a redução do espaço de busca não ocorreu adequadamente, direcionado o DOA a ótimos locais.

De modo a solucionar esse problema, seria interessante o desenvolvimento de outras técnicas capazes de manter o ótimo global dentro da região promissora. Uma estratégia potencial seria subdividir o espaço atual de busca em dois, toda vez que o nível de confiança para mais uma redução for alto. Com isso, geraria-se duas ou mais regiões promissoras, aumentando a probabilidade de pelo menos uma das regiões conter o ótimo global.

Tabela 4.5: DE padrão versus DOA+DE. O^* é o valor do ótimo global, μ é a média dos melhores valores encontrados nas 30 repetições, σ é o desvio-padrão dos melhores valores encontrados nas 30 repetições, TS é a Taxa de Sucesso.

	Método	***O****	μ	σ	**TS**
f1	DE	-450	-3,653906E+02	3,255174E+02	83,33
	DOA+DE		**-4,500000E+02**	1,825740E-07	96,67
f2	DE	-450	-3,206112E+02	5,021913E+02	80,00
	DOA+DE		**-4,481400E+02**	1,018766E+01	93,33
f3	DE	-450	1,075683E+05	4,969392E+05	0,00
	DOA+DE		**-4,499992E+02**	1,415220E-03	0,00
f4	DE	-450	-3,655265E+02	3,238781E+02	73,33
	DOA+DE		**-4,481195E+02**	1,030014E+01	66,67
f5	DE	-310	2,400386E+02	1,293722E+03	83,33
	DOA+DE		**-4,733649E+01**	9,995980E+02	93,33
f6	DE	390	2,627200E+07	7,922668E+07	0,00
	DOA+DE		**3,917685E+02**	3,159689E+00	0,00
f7	DE	-180	-1,697347E+02	3,698725E+01	0,00
	DOA+DE		**-1,793692E+02**	9,786190E-02	0,00
f8	DE	-140	-1,196546E+02	7,576253E-02	0,00
	DOA+DE		**-1,197189E+02**	8,282364E-02	0,00
f9	DE	-330	-3,055507E+02	5,077101E+00	0,00
	DOA+DE		**-3,172557E+02**	4,670964E+00	0,00
f10	DE	-330	-3,040944E+02	4,378923E+00	0,00
	DOA+DE		-3,059390E+02	3,669547E+00	0,00
f11	DE	90	9,707722E+01	2,562276E+00	0,00
	DOA+DE		**9,490644E+01**	2,031225E+00	0,00
f12	DE	-460	1,798956E+03	4,973305E+03	0,00
	DOA+DE		**-9,900257E+0**	8,309911E+02	0,00

4.7 Considerações Finais

Neste Capítulo foram apresentados conceitos e métodos relacionados à identificação de regiões promissoras em otimização global. A concentração de indivíduos em uma região é interpretada por alguns algoritmos de otimização global como sendo uma região promissora. Porém, essa detecção ocorre durante o processo de otimização. As heurísticas de detecção de regiões promissoras utilizam, em geral, critérios como a perda de diversidade da população de soluções. Algoritmos híbridos realizam uma exploração global até que essas regiões sejam detectadas (ver Seção 4.3). A partir de então, adota-se uma busca intensiva na região. Em problemas multimodais, é importante que se tenha cautela para se evitar uma verificação exaustiva de um ótimo local, aumentando significativamente a quantidade de avaliações da função objetivo.

Para prever regiões promissoras evitando excesso de avaliações, alguns algoritmos modelam a superfície de resposta da função objetivo a partir de uma amostra. Como apresentado na Seção 4.4, uma modelagem adequada pode requerer uma grande quantidade de pontos e tempo computacional. Além disso, uma questão importante é o nível da aproximação a ser utilizado. Nas figuras da Seção 4.4, pode ser verificado que quanto menor o espaço de busca a ser coberto, melhor tende a ser o modelo aproximado. Nesse sentido, muitos pesquisadores utilizam apenas modelagem local em uma região reduzida, ao invés de uma modelagem global. Por esses motivos, técnicas menos custosas são de interesse para detecção de regiões promissoras.

Neste Capítulo foi apresentado um estudo realizado para avaliar, primeiramente, se uma abordagem de SRA é válida, no sentido de que a redução pode aumentar significativamente a TS de metaheurísticas (Seção 4.5) para funções de *benchmark* complexas. De acordo com os resultados, um SRA beneficia uma metaheurística se essa for capaz de encontrar o ótimo global quando iniciada em um espaço de busca reduzido, relativamente próximo ao ótimo global. Contudo, se mesmo em um espaço reduzido a metaheurística falha, o SRA não é capaz de prover aumento significativo de desempenho.

Como o primeiro estudo mostrou que a idéia é válida, um SRA, denominado DOA, foi proposto e avaliado para as mesmas funções de *benchmark* do estudo da validade da redução (Seção 4.6). Nesse estudo, houve comparação entre os valores das melhores soluções obtidas em cada repetição. Foram constatadas melhoras devido ao DOA, sendo que a DE foi a metaheurística mais beneficiada. Nas funções em que não houve melhora, o ótimo global não estava dentro nem próximo o suficiente da região promissora determinada pelo DOA. Buscando reduzir esse problema, outras técnicas foram pesquisadas ainda neste trabalho, as quais são apresentadas no Capítulo 5.

Com base nos resultados dos estudos das Seções 4.5 e 4.6, a DE foi escolhida como

metaheurística para os estudos posteriores relativos a este trabalho.

5

Arquitetura de Amostragem Inteligente

Neste Capítulo é apresentada uma técnica mais complexa do que o DOA (ver Seção 4.6). O DOA é aplicado com a metodologia de Planejamento de Experimentos para guiar o processo de otimização na direção em que o *fitness* tende a ser melhorado. Por outro lado, a técnica proposta neste Capítulo emprega duas técnicas de Aprendizagem de Máquina (ML) com o objetivo de: 1) decidir se as novas soluções são ou não de alta qualidade para a resolução do problema e devem ser avaliadas pela função objetivo; 2) determinar as regiões promissoras a partir das soluções promissoras finais.

5.1 Linhas Gerais da Amostragem Inteligente

A combinação de diferentes abordagens tem sido estudada para melhorar o desempenho de metaheurísticas ou outros algoritmos de otimização global aplicados em problemas de alta complexidade. Em geral, o processo de otimização global possui duas fases. A primeira fase (exploração global, do inglês *exploration*) é usada para encontrar regiões promissoras no espaço de busca nas quais podem ser encontradas soluções de alta qualidade. A segunda fase (exploração local, do inglês *exploitation*) tem como objetivo refinar as soluções de alta qualidade, das regiões promissoras encontradas, para extrair o máximo da região e encontrar o seu ótimo local, na expectativa de que esse seja o ótimo global.

O esforço computacional utilizado em cada fase depende de alguns fatores:

1. A superfície do problema: em um problema bem condicionado, com apenas um ótimo global (unimodal), com pouco ou nenhum ruído, sem platôs, o algoritmo de otimização tende a passar pouco tempo em uma exploração global, visto que será

possível identificar uma tendência na superfície que induza o algoritmo para região do ótimo global. Como exemplo, pode-se citar a função Parábola da Figura 4.6, apresentada no Capítulo 4. Por outro lado, problemas mais complexos (com função objetivo multimodal, com alto nível de ruído e discreta) a exploração global requer mais tempo;

2. O algoritmo de otimização global: algumas técnicas privilegiam uma exploração global à uma local. Isso significa que possuem boa capacidade de encontrar regiões promissoras mas demandam muita computação ou não conseguem encontrar a melhor solução dessas regiões. Apesar disso, essas abordagens são úteis em problemas nos quais se busca encontrar soluções de alta qualidade, não necessariamente a melhor delas;

3. Os parâmetros do algoritmo de otimização global: encontrar ótimo global de um problema pode ser apenas uma questão de ajuste dos parâmetros de configuração do algoritmo de otimização. Muitas vezes, pequenas modificações sobre soluções já encontradas produzem uma exploração local adequada, possibilitando a determinação do ótimo local. Por outro lado, quanto menor o tamanho da modificação, maior o tempo gasto necessário para se encontrar o ótimo local. Se for uma região que não contenha o ótimo global e esse for desejado, a exploração local ocorrerá em vão. Por outro lado, uma melhor exploração global poderia ser realizada se o parâmetro de modificação fosse maior. Esse conflito entre os valores de um parâmetro que são mais adequados para uma busca local e global indicam que o uso de uma mesma combinação de Algoritmo+Parâmetros para dois problemas distintos não deve ser adequada, principalmente para problemas mais difíceis.

Nesse sentido, esse Capítulo propõe realizar a 1^a fase de forma independente da 2^a e com maior sucesso. O algoritmo de busca por regiões promissoras para a primeira fase foi denominado arquitetura de Amostragem Inteligente (SS, do inglês *Smart Sampling*). Essa técnica explora o espaço de busca dentro e em volta de regiões com soluções de alta qualidade. Essas regiões são reduzidas a cada iteração do SS, de um modo semelhante ao da estratégia proposta em (Melo et al., 2007a). O termo arquitetura deve-se ao fato de que módulos do SS podem ser trocados por outras técnicas que realizem a mesma tarefa de forma similar e, eventualmente, de maneira mais eficiente.

A motivação para o desenvolvimento do SS é a construção de uma técnica simples e computacionalmente eficiente ao invés de utilizar complexos modelos probabilísticos que, em geral, demandam muitos cálculos. A idéia principal do SS é realizar a reamostragem apenas em áreas consideradas regiões promissoras. A arquitetura proposta

baseia-se em modelos simples de Aprendizagem de Máquina (ML, do inglês *Machine Learning*) (Mitchell, 1997), que aprendem o comportamento da superfície do problema e determinam se novos pontos pertencem ou não a regiões promissoras.

A ML pode ser definida como uma área de pesquisa de métodos computacionais relacionados à aquisição automática de novos conhecimentos, novas habilidades e novas maneiras de organizar o conhecimento previamente existente (Mitchell, 1997). A classificação é uma técnica que consiste em aprender características de um conjunto de exemplos previamente rotulado e gerar um modelo capaz de rotular corretamente um novo conjunto de instâncias desconhecido ou não visto durante o processo de aprendizado. Esse tipo de técnica tem sido usada em descoberta de fraudes, mineração de dados, reconhecimento de padrões, descoberta de fármacos, entre outros (Mitchell, 1997).

Nesse sentido, uma técnica de ML pode ser utilizada para decidir se novas soluções podem ser consideradas promissoras ou não, sem que haja avaliação da função objetivo para essas novas soluções. Essa etapa é realizada sem uso de derivadas, gradientes ou estimadores probabilísticos. As novas soluções que forem classificadas como promissoras pela técnica de ML serão avaliadas pela função objetivo e adicionadas à população atual. As novas soluções classificadas como não-promissoras serão descartadas, evitando esforço computacional desnecessário.

Dada a simplicidade do modelo de classificação utilizado no SS, não é possível gerar novas soluções diretamente a partir dele. A geração de novas soluções (reamostragem) ocorre a partir das soluções definidas como de alta qualidade pelo modelo adotado utilizando um procedimento modificador (ver Seção 2.1).

O processo de modelagem por ML e reamostragem é repetido iterativamente identificando regiões cada vez menores, com soluções de qualidade cada vez maior.

Ao final desse processo iterativo, outra técnica de ML é aplicada para dividir o espaço de busca, identificando as regiões promissoras que contém alta concentração dessas soluções de alta qualidade.

As técnicas de ML do SS podem ser substituídas por outras técnicas que produzam o mesmo tipo de resultado, sem alteração de outras partes do código. Isso torna a arquitetura bastante flexível. É importante esclarecer, também, que o objetivo do SS não é realizar o processo de otimização completo (primeira e segunda fases) e sim encontrar uma ou mais regiões promissoras para que um algoritmo de otimização propriamente dito continue o processo. O SS pode ser usado junto com qualquer outra metaheurística populacional sem modificação alguma da metaheurística, em contraste com abordagens desenvolvidas especificamente para determinados algoritmos como *niching* e *speciation* (Deb & Goldberg, 1989; Bäck et al., 1997) e algoritmos híbridos que identificam essas regiões promissoras entre as iterações de um método de otimização global (Jelasity et al., 2001; Chelouah & Siarry, 2003; Oliveira, 2004).

De modo a facilitar o entendimento, é adequado introduzir conceitos básicos da área de ML, conforme apresentado na Seção 5.2.

5.2 Aprendizado de Máquina Supervisionado

No problema mais comum de aprendizado supervisionado, os dados de entrada consistem de um conjunto de exemplos EX, com n exemplos E_i, $i = 1, ..., n$, todos de tamanho D, escolhidos de um domínio X com uma distribuição Dt fixa, desconhecida e arbitrária, da forma $\{(x_1, y_1), ..., (x_n, y_n)\}$ para alguma função desconhecida $y = f(x)$ em que:

- x_i é um vetor no formato $(x_{i1}, x_{i2}, ..., x_{iD})$ com valores discretos ou contínuos;
- x_{ij} refere-se ao valor do atributo X_j para o exemplo E_i, como mostrado na Tabela 5.1;

Tabela 5.1: Conjunto de exemplos no formato de tabela atributo-valor.

	X_1	X_2	...	X_D	Y
E_1	x_{11}	x_{12}	...	x_{1D}	y_1
E_2	x_{21}	x_{22}	...	x_{2D}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_n	x_{n1}	x_{n2}	...	x_{nD}	y_n

- os valores y_i referem-se ao valor do atributo Y de cada exemplo E_i, freqüentemente denominado classe.

Os valores de y são tipicamente pertencentes a um conjunto discreto de classes C_v, $v = 1, ..., n_{Cl}$, i.e , $y \in \{C_1, ..., C_{n_{Cl}}\}$, tal que n_{Cl} é a quantidade de classes distintas. A classe é o conceito (meta) que descreve o fenômeno de interesse. Por exemplo, em uma base de dados médica, o conceito classe pode ser *doente* ou *não-doente*, ou então em uma base de mineração de textos o conceito classe pode conter mais divisões como: *política*, *saúde*, *educação* ou *esportes*.

Um classificador h é induzido por um algoritmo de aprendizado que recebe um conjunto S de exemplos já rotulados como entrada. Tal classificador consiste da hipótese feita sobre a verdadeira (mas desconhecida) função f. Quando uma nova instância x é apresentada ao classificador, ele é capaz de predizer a classe correspondente y de acordo com o conhecimento aprendido.

Como o objetivo é desenvolver um SS simples e computacionalmente eficiente, foram investigados dois dos classificadores mais simples existentes, apresentados nas Seções 5.2.1 e 5.2.2.

5.2.1 K-Vizinhos mais próximos

Classificadores baseados em instância, tais como o algoritmo dos K-vizinhos mais próximos (KNN, do inglês *K-Nearest Neighbors*), estão entre os mais simples e computacionalmente eficientes algoritmos de ML. Na etapa de aprendizagem (treinamento), o KNN pode memorizar todos os exemplos de treino em uma única iteração. Com isso, sua complexidade de tempo seria $\Theta(n) = \Theta(1)$, tal que n é a quantidade de exemplos. Por outro lado, na etapa de classificação, pode ser necessário comparar uma nova instância com todos os exemplos de treino. Portanto, a complexidade de tempo nesse caso é dada por $\Theta(n * D)$, tal que D é a quantidade de características de cada exemplo.

Técnicas como o KNN classificam as instâncias desconhecidas com relação às instâncias conhecidas, segundo alguma função de distância/similaridade. Em outras palavras, duas instâncias próximas, baseadas em uma função de distância apropriada, tendem a pertencer a mesma classe, enquanto que duas instâncias distantes tendem a pertencer à classes diferentes.

Um objeto é classificado pelo voto majoritário dos seus vizinhos, com o objeto sendo atribuído à classe mais comum dentre seus K vizinhos mais próximos. O valor de K é um inteiro positivo, tipicamente pequeno. Se $K = 1$ então o objeto é simplesmente atribuído a classe de seu vizinho mais próximo. Os vizinhos são do conjunto de exemplos (EX), representado por vetores de posição em um espaço de características multidimensional.

É comum o uso da distância Euclidiana para calcular a proximidade entre os elementos. Em um problema de otimização no qual as soluções são pontos em uma superfície de resposta, essa função de distância é a mais indicada.

Na Figura 5.1 tem-se um exemplo de classificação utilizando essa técnica. A amostra de teste (*círculo* - ●) deve ser classificada como *quadrado* (■) ou *triângulo* (▲). Se $K = 3$, a amostra é classificada como da classe dos triângulos, pois há 2 triângulos e somente 1 quadrado dentro do círculo mais interno, que representa os K vizinhos mais próximos. Se $K = 5$, a amostra é classificada na classe dos quadrados (3 quadrados contra 2 triângulos dentro do círculo mais externo - tracejado).

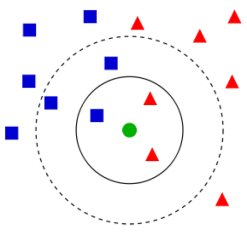


Figura 5.1: Exemplo de classificação utilizando KNN.

5.2.2 Aprendizado baseado em regras

Um classificador baseado em regras (RBL, do inglês *Rule-Based Learner*)(Mitchell, 1997) usa um conjunto de regras no formato SE-ENTÃO para classificar as instâncias. Uma regra SE-ENTÃO é uma expressão lógica da forma:

- SE *condição* ENTÃO *conclusão*.

Ao contrário de outros métodos de aprendizagem que utilizam modelos matemáticos ou probabilísticos de mais difícil compreensão, regras no formato SE-ENTÃO, com uso de operadores lógicos E e OU dentro das condições, podem ser facilmente entendidas por seres humanos. Como exemplo dessa técnica, suponha um conjunto de dados de condições climáticas, cujo processo de classificação aconselha ou não a prática de esportes, no caso, Tênis. Um possível conjunto de regras seria:

```
R1: SE (clima = Ensolarado) E (Vento = Fraco) ENTÃO (Jogar_tenis = sim)

R2: SE (clima = Ensolarado) E (Humidade = Alta) ENTÃO (Jogar_tenis = não)

R3: SE (clima = Ensolarado) E (Humidade = Normal) ENTÃO (Jogar_tenis = sim)

R4: SE (clima = Nublado) ENTÃO (Jogar_tenis = sim)
```

A conclusão da regra contém uma classe de predição. No presente caso, a classe é "Jogar_tenis" (rotulada como *sim* ou *não*). Usando esse tipo de estrutura lógica é possível percorrer a seqüência de regras e obter o conjunto de instâncias que dispara cada regra, dividindo-as em diferentes grupos. Em um problema contínuo, esses grupos são regiões na superfície do problema. Para esse tipo de algoritmo, o conjunto de regras determina as regiões em forma de hipercubos. Na Figura 5.2, tem-se uma ilustração de classificação em um problema contínuo e de como as regiões são determinadas e o conjunto de regras gerado para a amostra utilizada. Na Figura 5.3 tem-se o conjunto de regras gerado para essa amostra. A técnica de RBL escolhida para uso no SS foi o RIPPER (*Repeated Incremental Pruning to Produce Error Reduction*)(Cohen, 1995).

Outros tipos de classificadores mais complexos, como ANN e SVM, são capazes de encontrar regiões sem forma definida, podendo possuir inclusive cavidades (regiões não convexas). Como o SS busca ser simples e eficiente, modelos complexos (como ANN e SVM) não foram considerados.

5.3 Amostragem Inteligente

As principais características do SS proposto, apresentado na Figura 5.4, são as seguintes:

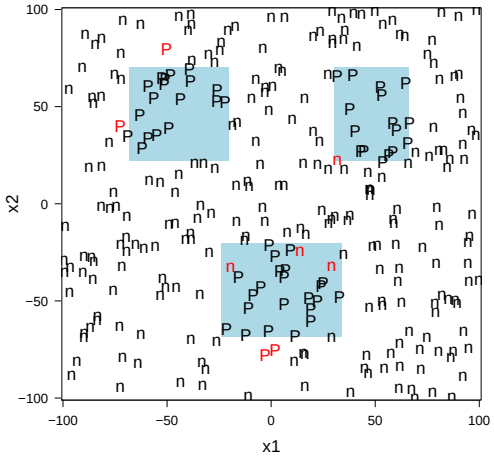


Figura 5.2: Exemplo de regiões (em azul) separadas de acordo com regras geradas para a função Tripod. P são os pontos rotulados como Promissores e N indica os pontos Não promissores. Os pontos em cor vermelha foram classificados na classe errada.

```
R1: x1 >= -72.39 and x1 <= 21.98 and x2 => 20.19 and x2 <= 74.76 => classe=P

R2: x1 >= 33.07 and x1 <= 68.51 and x2 => 21.23 and x2 <= 74.34 => classe=P

R3: x1 >= -24.60 and x1 <= 37.83 and x2 => -71.37 and x2 <= -22.09 => classe=P

R4: => classe=N
```

Figura 5.3: Exemplo de regras geradas pelo RBL para a amostra da Figura 5.2.

1. Dado um subconjunto do intervalo do problema, chamado janela, o algoritmo localiza intervalos menores do espaço de busca, em que haja uma maior concentração de soluções de alta qualidade, utilizando uma busca guiada pelas técnicas de aprendizagem de máquina;

2. Para encontrar melhores soluções, não são utilizadas técnicas de gradiente[1] ou matriz Hessiana[2];

3. Para evitar que a busca de regiões promissoras perca o ótimo global, o SS utiliza uma estratégia mais conservadora, em que explora, também, em torno das regiões promissoras, gerando pontos não apenas dentro da região. Com isso, existe uma chance maior de o SS manter o ótimo global dentro de uma região promissora, ou próximo de uma das regiões encontradas;

4. O SS proposto determina regiões promissoras sem um número pré-especificado de

[1] Vetor composto por derivadas parciais da função para uma determinada solução, indicando a direção e a taxa de crescimento. (Boyd & Vandenberghe, 2004)

[2] Matriz quadrada composta por derivadas parciais de segunda-ordem de uma função multivariada, utilizada para descrever a curvatura local da função. (Boyd & Vandenberghe, 2004)

regiões (*clusters*) como um parâmetro. Assim, o número de regiões promissoras encontradas depende do problema.

Para atingir esses objetivos, o SS emprega duas técnicas de ML: uma capaz de separar soluções promissoras de soluções não-promissoras e a outra capaz de identificar as regiões promissoras finais obtidas para que um algoritmo de otimização global seja executado nessas regiões. A utilização das duas técnicas de ML é devida aos seguintes motivos:

1. Um dos objetivos do SS é utilizar técnicas simples. Por esse motivo, foi o escolhido o KNN (com $K = 1$, o modo mais simples) como classificador. Contudo, apesar de o KNN ser capaz de classificar ele não separa o espaço em regiões. Por esse motivo, outra técnica deve ser aplicada;

2. Outras técnicas de funcionamento mais simples e que possibilitam a separação das regiões são a Árvore de Decisão (Mitchell, 1997) e o RBL (ver Seção 5.2.2). Dentre esses, é mais simples extrair as soluções de cada região a partir do conjunto de regras;

3. O RBL poderia ser empregado no lugar do KNN, mas o RBL possui maior complexidade computacional. Portanto, é melhor utilizar as duas técnicas: KNN seguido de RBL.

O algoritmo básico do SS proposto, dividido em duas fases, é sintetizado na Figura 5.4. Inicialmente, o algoritmo retira uma amostra considerável do espaço de busca para identificar as primeiras regiões que serão exploradas. Como comentado na Seção 4.4, quanto maior o número de dimensões e o intervalo em cada dimensão do problema, maior deve ser a primeira amostra.

O conjunto de exemplos pode ser dividido entre *promissores* e *não promissores*. Desse modo, é possível rotular os exemplos manualmente, sendo 50% para cada classe, por exemplo. Como os exemplos possuem uma rotulação prévia e deseja-se encontrar um modo de identificar novos pontos não rotulados, é mais adequado utilizar técnicas Supervisionadas (com rótulos conhecidos), por isso o uso de classificadores. Caso não existisse uma rotulação prévia dos dados (*promissores* e *não promissores*) e fosse desejado dividir o conjunto de exemplos automaticamente, seria indicado o uso de técnicas Não-Supervisionadas (sem rótulos conhecidos). Nesse caso, poderiam ser usados algoritmos de Agrupamento (*Clustering*).

O funcionamento do SS para funções de teste contínuas (em 2D) Ackley, Alpine, Griewangk, Parábola, Rastringin, Rosenbrock e Tripod (ver definição matemática na

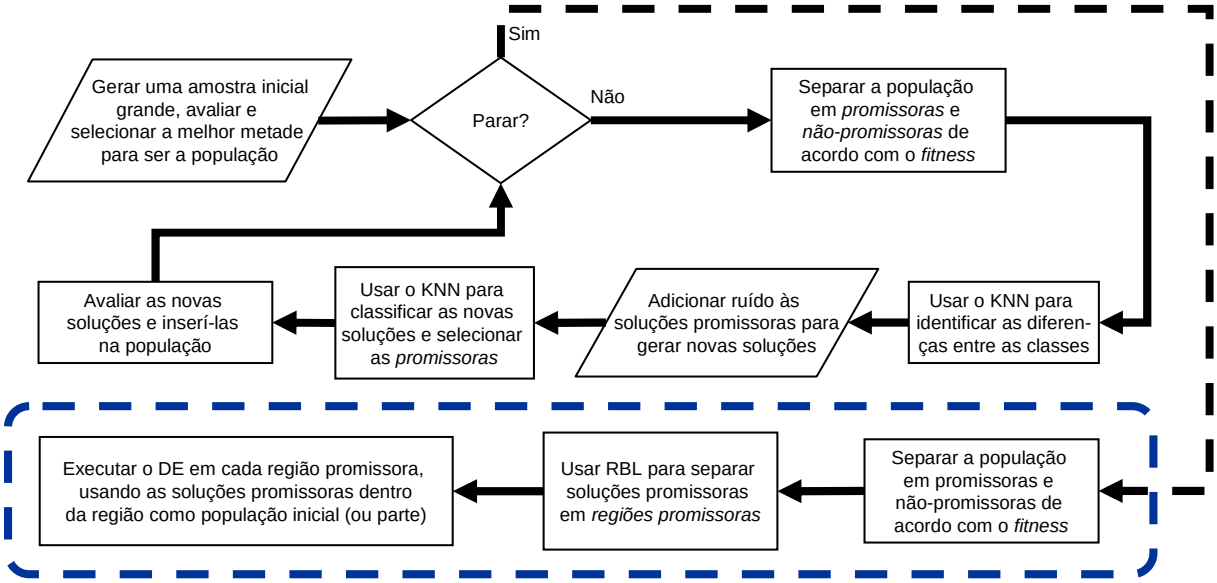


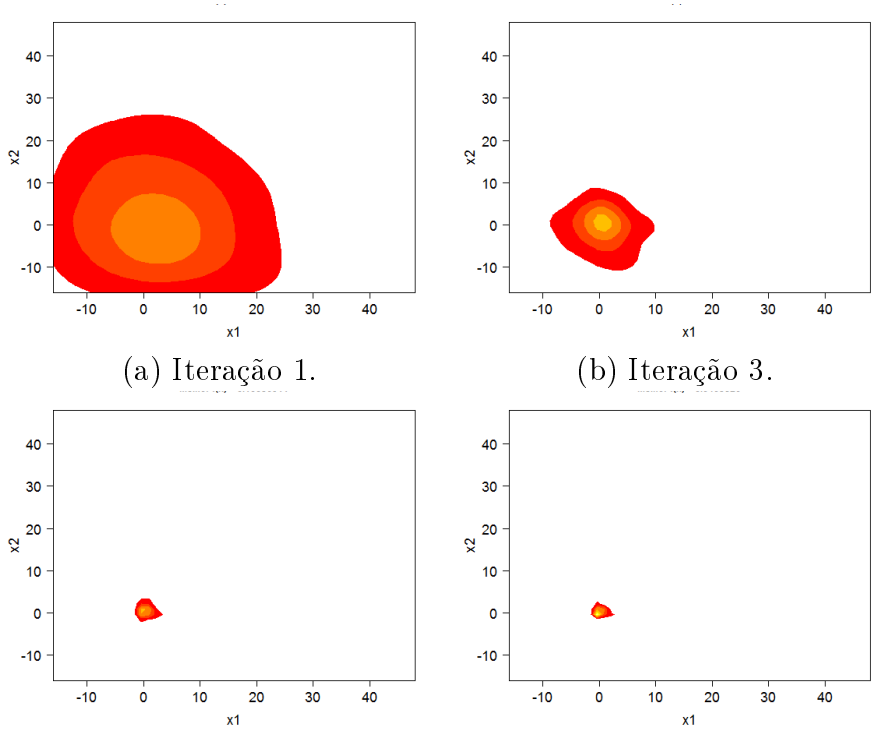
Figura 5.4: Esquema sintetizando o funcionamento do SS proposto.

Seção 8.1) é ilustrado respectivamente nas Figuras 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 com os mesmos parâmetros que serão apresentados no Capítulo de experimentos (ver Seção 8.1.1). Nessas figuras é apresentada a evolução/redução das regiões promissoras ao longo das iterações do SS pra as funções de *benchmark* citadas anteriormente, de acordo com a distribuição das soluções de alta qualidade no espaço de busca. Ou seja, seleciona-se as soluções de alta qualidade e aplica-se um algoritmo de Estimação de Densidade de *Kernel*[3] (do inglês, *Kernel Density Estimation*). A graduação das cores, do vermelho para o amarelo, indicam a concentração de soluções de alta qualidade, sendo que quanto mais claro, maior a concentração. Nessas Figuras, é possível perceber que o SS cumpre satisfatoriamente o seu papel. As regiões vão diminuindo e de encontro ao ótimo global. O ótimo está na posição $x^* = (1, 1)$ para a função Rosenbrock e $x^* = (0, 0)$ para as demais.

Quando o SS termina, as regiões promissoras finais devem ser separadas. Essa corresponde à segunda fase do algoritmo (região tracejada na Figura 5.4). Alguns autores propõem o uso de um algoritmo de agrupamento para realizar essa tarefa(Ortigosa et al., 2001; Chelouah & Siarry, 2003; Oliveira, 2004). Porém, há algumas desvantagens nesse tipo de abordagem. Para muitos algoritmos de agrupamento, o resultado do agrupamento é dependente dos elementos representativos[4] de cada *cluster* que são definidos no início do processo, o que pode tornar essa abordagem menos robusta. Além disso, o número de *clusters* é um parâmetro de entrada que deve ser especificado para a maioria dos algo-

[3]Utilizando a função *kde2d* do pacote MASS do R.

[4]Vetores que correspondem, por exemplo, ao centro do *cluster*, gerado de maneira aleatória e atualizado ao longo do processo.



(a) Iteração 1. (b) Iteração 3.

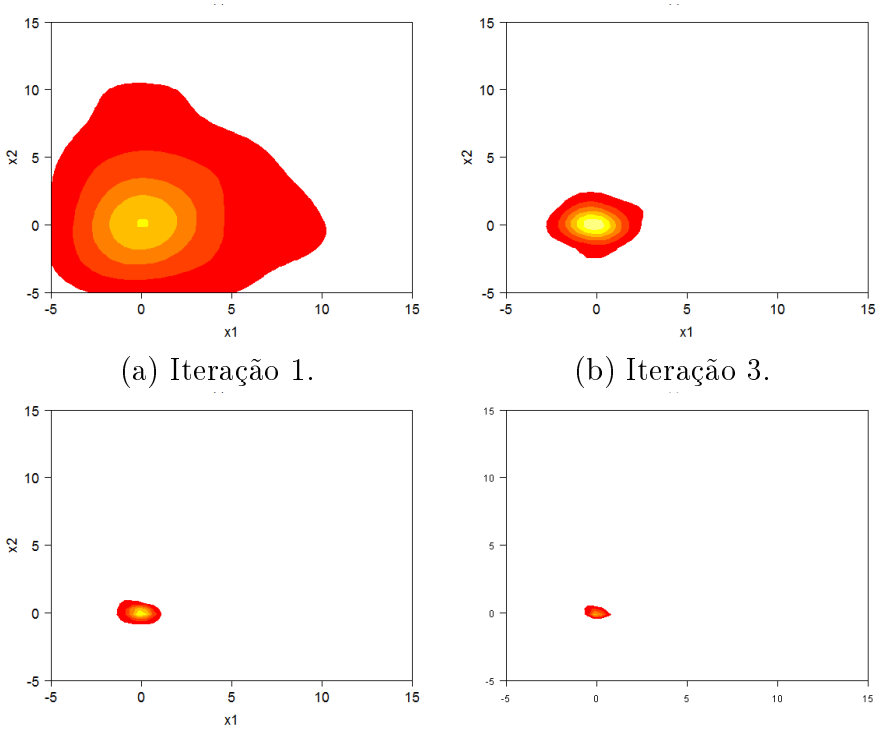
(c) Iteração 5. (d) Iteração 7.

Figura 5.5: Funcionamento do SS na Função Ackley.

ritmos de agrupamento. A obtenção de tal parâmetro não é trivial. Por exemplo, pode ser necessário executar o agrupamento com diversas quantidades de *clusters*, avaliar os resultados de acordo com alguma métrica para decidir configuração mais adequada pra cada conjunto de dados (cada conjunto final de soluções encontradas pelo SS).

Um dos raros algoritmos de agrupamento que não necessita desse parâmetro é o *K-windows* não-supervisionado (Vrahatis et al., 2002). Esse algoritmo foi utilizado com sucesso como um operador em um DE híbrido (Tasoulis et al., 2005). No entanto, os autores realizaram experimentos com esse DE híbrido em somente 4 funções de *benchmark*, bidimensionais, devido a dificuldades no agrupamento de pontos de maior dimensionalidade, causada pelo algoritmo de minimização de distâncias adotado (Alevizos & Tasoulis, 2002).

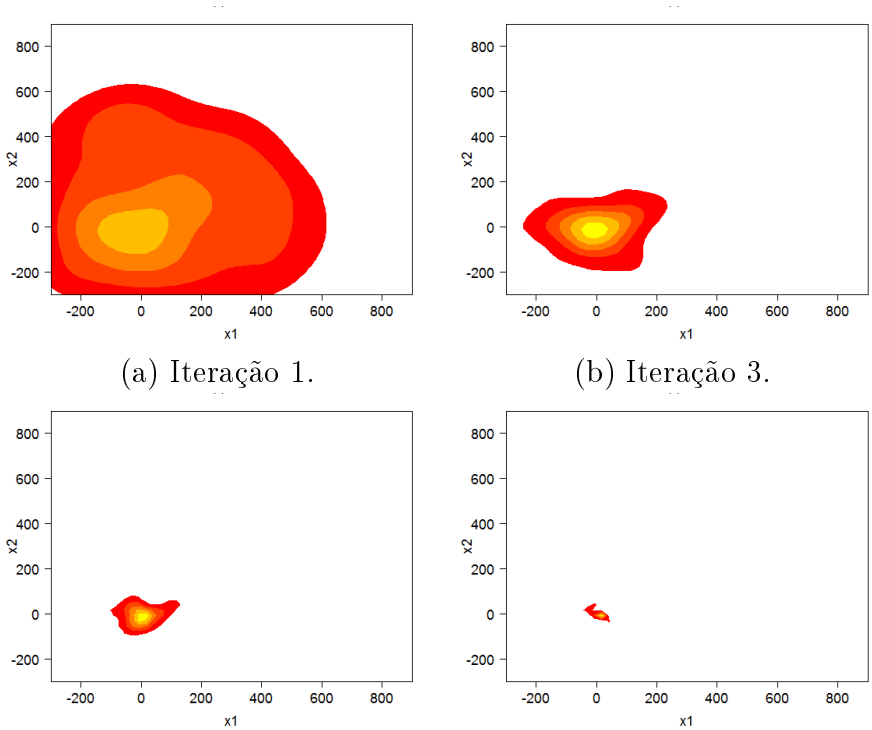
Ao final do processo do SS (usando KNN para separar soluções promissoras das não-promissoras e RBL para extrair as regiões promissoras), obtém-se o conjunto de soluções de alta qualidade separadas por regiões, como apresentado na Figura 5.2.



(a) Iteração 1. (b) Iteração 3.

(c) Iteração 5. (d) Iteração 7.

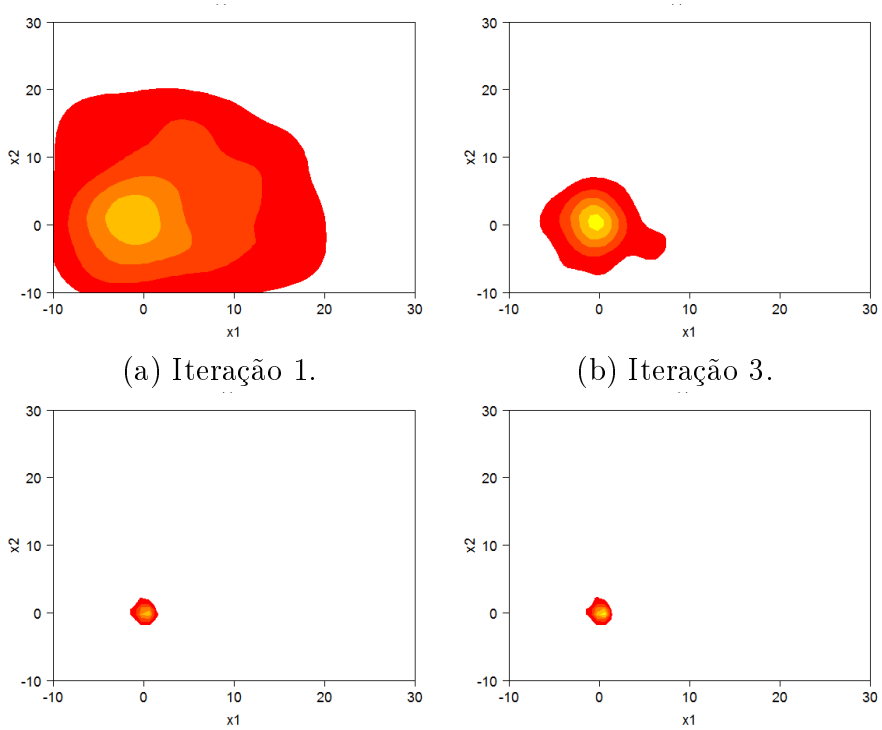
Figura 5.6: Funcionamento do SS na Função Alpine.



(a) Iteração 1. (b) Iteração 3.

(c) Iteração 5. (d) Iteração 11.

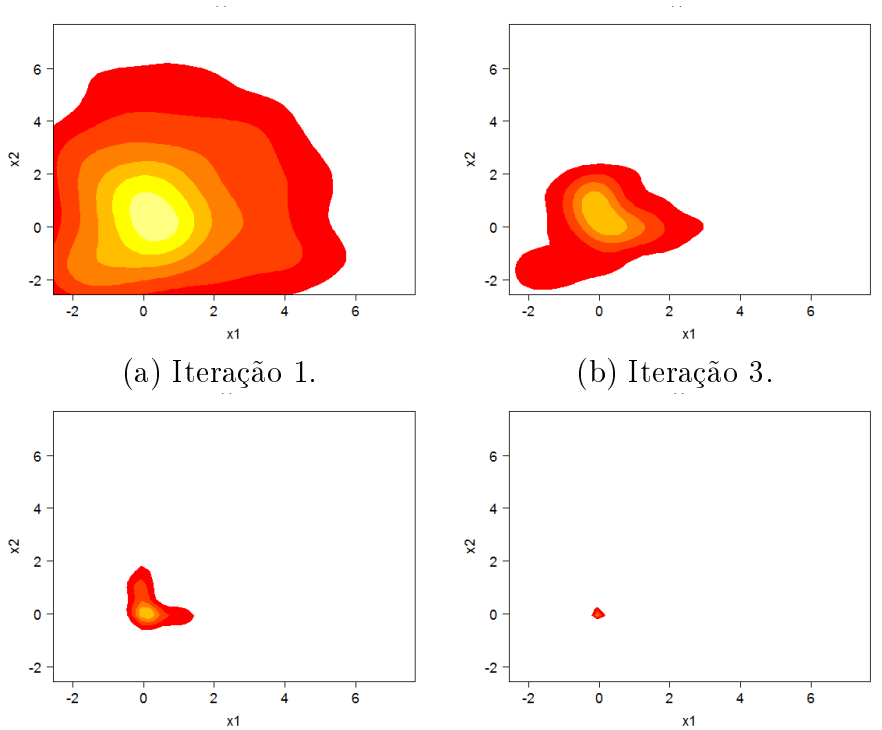
Figura 5.7: Funcionamento do SS na Função Griewangk.



(a) Iteração 1. (b) Iteração 3.

(c) Iteração 5. (d) Iteração 6.

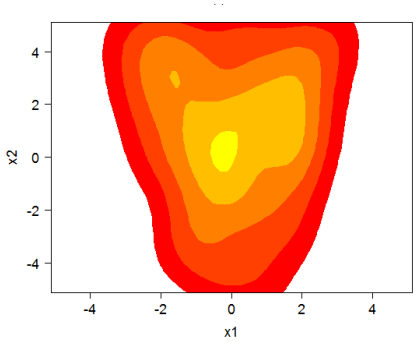
Figura 5.8: Funcionamento do SS na Função Parábola.



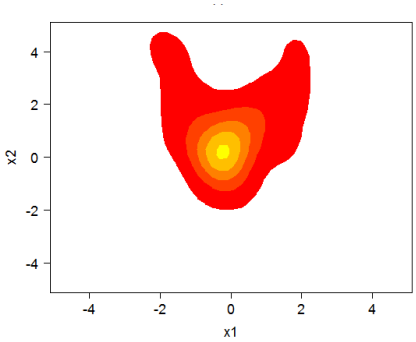
(a) Iteração 1. (b) Iteração 3.

(c) Iteração 5. (d) Iteração 7.

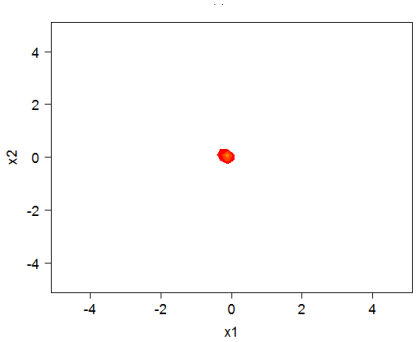
Figura 5.9: Funcionamento do SS na Função Rastringin.



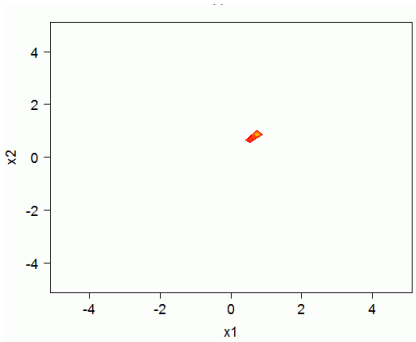
(a) Iteração 1.



(b) Iteração 3.

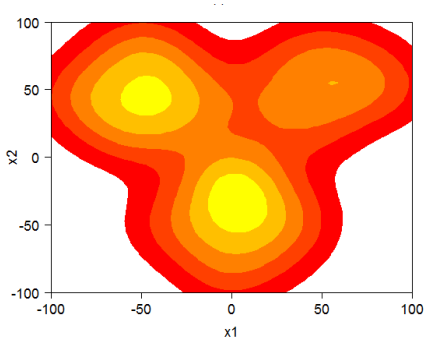


(c) Iteração 7.

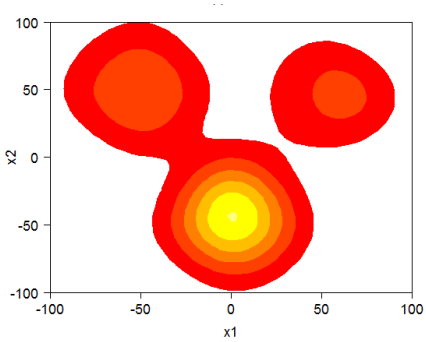


(d) Iteração 13.

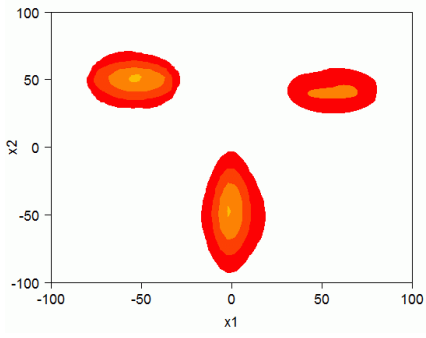
Figura 5.10: Funcionamento do SS na Função Rosenbrock.



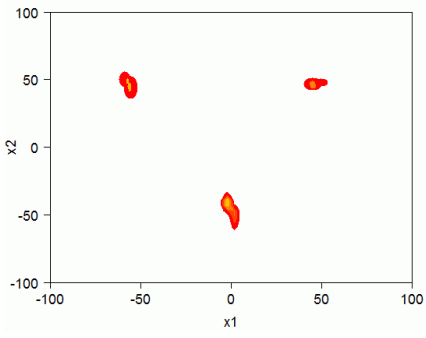
(a) Iteração 1.



(b) Iteração 3.



(c) Iteração 7.



(d) Iteração 10.

Figura 5.11: Funcionamento do SS na Função Tripod.

Detalhes do SS proposto

Para um melhor entendimento do algoritmo proposto, na seqüência apresentam-se detalhes da implementação e funcionamento do SS envolvendo: critérios de parada, operadores para reamostragem de novas soluções. Seguindo o diagrama da Figura 5.4, tem-se:

Gerar população aleatória inicial: nesse passo, é gerada uma amostra aleatória grande, dentro dos intervalos do problema. A melhor metade dessa amostra é selecionada para ser a população do SS;

Parar o SS se algum critério for atingido: os três critérios empregados foram: 1) um número de reamostragens realizadas, 2) uma quantidade de reamostragens sem melhoria no melhor valor encontrado (estagnação) e 3) uma redução excessiva, em pelo menos uma das dimensões do problema, ultrapassando um valor pré-especificado;

Selecionar soluções promissoras: uma fração das melhores soluções é escolhida e rotulada como *promissoras*, sendo que as demais soluções são rotuladas como *não-promissoras*. Se essa fração for muito pequena o classificador será muito restrito, podendo conduzir o SS a um ótimo local rapidamente. Por outro lado, um fração alta provavelmente produzirá uma busca mais exploratória e mais lenta, visto que soluções de qualidade relativamente baixa serão tratadas como promissoras. Para evitar essas situações extremas desfavoráveis, uma fração conservadora de 50% pode ser mais adequada;

Induzir um classificador KNN: um classificador é obtido com base nas soluções rotuladas como promissoras ou não-promissoras no processo de seleção anterior;

Amostrar novas soluções promissoras: gerar soluções usando o operador de reamostragem por inserção de ruído, apresentado no Algoritmo 5.1, ou o de reamostragem em direção à melhor solução já encontrada, ver Algoritmo 5.2, e classificá-las. Selecionar as soluções classificadas como promissoras e descartar as demais. Repetir esse processo até que um número desejado de soluções promissoras seja gerado. Quando terminar, avaliar as novas soluções.

O Algoritmo 5.1 é um operador de reamostragem que gera pontos próximos a todas as soluções de alta qualidade, em qualquer região do espaço de busca, pela adição de um ruído aleatório nessas soluções. Por outro lado, o Algoritmo 5.2 é um operador tendencioso que gera novas soluções de alta qualidade na direção da melhor solução já encontrada, de maneira similar ao operador de deslocamento da PSO (ver Seção 2.3.2), mas com menor poder de exploração global. O operador padrão adotado para o SS é o de reamostragem

por ruído (Algoritmo 5.1).

Algoritmo 5.1 Operador de reamostragem por ruído.

entrada: $SP_{N,D}$: matriz com as Soluções Promissoras da população, sendo N a quantidade de soluções promissoras e D a quantidade de variáveis.
saída: $NS_{N,D}$: conjunto de Novas Soluções

1: **para** $i = 1$ **até** N **faça**
2: **para** $j = 1$ **até** D **faça**
3: $NS[i,j] = SP[i,j] + (SP[i,j] * RAND())$; *//RAND é um número aleatório entre -1 e 1.*
4: Certificar de que $NS[i,j]$ esteja dentro do domínio do problema. Se estiver fora, substituir pelo valor do intervalo mais próximo do problema ou gerar um novo RAND(); *//Ex.: se o intervalo for [-1; 1] e* $NS[i,j] = -1,328$, *modificá-la para* $NS[i,j] = -1$.
5: **fim para**
6: **fim para**

Algoritmo 5.2 Operador de reamostragem em direção à melhor solução já encontrada, adicionando à cada solução uma porcentagem da diferença entre ela e a melhor solução.

entrada: $SP_{N,D}$: matriz com as Soluções Promissoras da população, sendo N a quantidade de soluções promissoras e D a quantidade de variáveis.
MS_D: a melhor solução encontrada até o momento.
saída: NS: conjunto de Novas Soluções

1: **para** $i = 1$ **até** N **faça**
2: **para** $j = 1$ **até** D **faça**
3: $NS[i,j] = SP[i,j] + (PORCENTO() * (MS[j] - SP[i,j]))$; *//PORCENTO é um número aleatório entre 0 e 1.*
4: Certificar de que $NS[i,j]$ esteja dentro do domínio do problema;
5: **fim para**
6: **fim para**

É importante observar que operadores de modificação mais complexos tendem a gerar melhores resultados. A seguir são registrados outros operadores para reamostragem que poderiam ser estudados mais profundamente em trabalhos futuros.

1. Baseado na probabilidade marginal de cada variável, considerando variáveis sem correlação (ver Seção 6.2.1);

2. Utilizando estimação de distribuição de variáveis conjuntas por meio de distribuição Normal Multivariada (Choi et al., 2004; Thissen et al., 2005);

3. Baseado na tendência de cada variável (a direção da variação do valor da variável que mais aproxima do ótimo global) por meio de um modelo de regressão linear múltipla (Montgomery, 1997) e gerar indivíduos na direção da tendência;

4. Baseado no operador de recombinação da DE.

Avaliar e inserir as novas soluções na população: as novas soluções devem ser avaliadas pela função objetivo e inseridas na população. Por fim, as soluções de qualidade mais baixa devem ser removidas para manter o tamanho da população constante;

Usar RBL para separar as soluções finais soluções de alta qualidade em regiões promissoras: o RBL determina uma ou mais regiões contendo as soluções promissoras finais. Para cada região promissora encontrada, usar as soluções promissoras dentro delas como população inicial para um algoritmo de otimização global. Para extrair as soluções de cada conjunto de regras é utilizado o Algoritmo 5.3.

Algoritmo 5.3 *Parser* das regras do RBL.

entrada: *Sol:* o conjunto final de soluções encontradas pelo SS
R: o conjunto de n_R regras geradas pelo RBL
saída: *RP:* um vetor de regiões promissoras, de tamanho n_R, contendo o conjunto de soluções de cada região;

1: **para** i=1 **até** n_R **faça**
2: Retirar o conjunto de soluções em *Sol* que satisfaça a regra $R[i]$; *//Portanto o conjunto será menor na próxima iteração;*
3: Inserir em $RP[i]$ o conjunto de soluções da região definida pela regra $R[i]$;
4: **fim para**

Um dos pontos principais para uma alta eficácia do SS é o operador de reamostragem. O operador adotado é consideravelmente simples e não apresenta qualquer característica inteligente (utilizando ML, por exemplo). Outros operadores mais elaborados poderiam a ser investigados. Por exemplo, poderia-se implementar um SS com operadores de reamostragem distintos (ver itens 1 a 4 descritos acima), que seriam executados de acordo com certa probabilidade (taxa de aplicação) ao longo do processo de busca por regiões promissoras.

5.4 Considerações Finais

Neste Capítulo foi apresentada a arquitetura de determinação de regiões promissoras denominada Amostragem Inteligente (SS). Essa técnica realiza a exploração do espaço de

busca dentro e em torno de regiões com maior concentração de soluções de alta qualidade. Essas regiões são reduzidas a cada iteração do SS, encontrando soluções cada vez melhores em regiões cada vez mais distantes entre si. O processo utilizado para encontrar essas regiões utiliza técnicas de ML, sem qualquer outra informação heurística sobre o comportamento do problema, uso de derivadas parciais ou outra análise estatística. No final do processo, o SS retorna um ou mais conjuntos de soluções divididos em regiões promissoras. A partir desse ponto, uma metaheurística qualquer pode ser iniciada para cada região (usando inclusive as soluções promissoras encontradas na última iteração do SS), para continuar a busca pelo ótimo global.

O SS é classificado como uma arquitetura pois módulos desse algoritmo podem ser facilmente substituídos por outras técnicas que realizem a mesma tarefa de forma similar. Essa característica torna o SS adequado para aumentar o desempenho de uma diversidade de metaheurísticas.

6

Algoritmos Genéticos com Construção de Modelos Probabilísticos

Conforme apresentado no Capítulo 2, os EAs têm produzido soluções relevantes em diversas áreas da Engenharia e Ciências como transporte (Mauttone & Urquhart, 2009), aprendizagem de máquina (Zhu et al., 2005) e problemas industriais (Michalewicz et al., 1996). No entanto, ainda existem problemas desafiadores para os EAs e outras metaheurísticas populacionais, conforme apresentado neste Capítulo. Uma das frentes de pesquisa mais forte é o desenvolvimento de EAs utilizando modelos probabilísticos de distribuições (ver Seção (2.3.4)). Com base em modelos pode-se gerar soluções que possuam as características das soluções de alta qualidade, em vez de usar procedimentos modificadores probabilísticos (ver Seção 2.1) como ocorre nos EAs (GA, PSO e DE, ver Seção 2.3). Além de evitar a elaboração de procedimentos modificadores, o uso de um modelo probabilístico não requer os parâmetros referentes às taxas de aplicação desses procedimentos pelo algoritmo. Mais ainda, pode-se estabelecer a confiança com que certo modelo representa os dados, possibilitando orientar o algoritmo na direção dos melhores modelos.

Além dessas vantagens do uso de modelos probabilísticos, a linha de pesquisa de EDAs, também denominada Algoritmos Genéticos com Construção de Modelos Probabilísticos (PMBGAs, do inglês *Probabilistic Model Building Genetic Algorithms*) explora o aspecto teórico da preservação dos Blocos de Construção (BBs, do inglês *Building Blocks*, ver Seção 2.3.4) (Goldberg, 2002) em tais modelos. Dessa maneira, pode-se dizer que a fundamentação teórica dos PMBGAs não difere dos demais EDAs[1], como um dos algoritmos

[1]EDA é o nome adotado pela maior parte da comunidade acadêmica. PMBGA é o nome utilizado por

mais simples dessa classe, denominado PBIL (do inglês, *Population Based Incremental Learning*) (Baluja, 1994). Tais algoritmos foram propostos para trabalhar, inicialmente, com problemas binários como os apresentados na Seção 6.1. É importante observar que, atualmente, existem implementações capazes de lidar com problemas contínuos (Ding et al., 2008), mas que não são o foco desta Tese na parte referente a PMBGAs, uma vez que a teoria e a experimentação nessa área têm sido construídas no contexto de variáveis discretas.

Na Seção 6.1 é apresentado o conceito de funções deceptivas e as relações dessas com construção de BBs. Na Seção 6.2, as classes de algoritmos capazes de descobrir explorar adequadamente os BBs são brevemente introduzidas. Na Seção 6.3, alguns dos principais PMBGAs são explicados.

6.1 Funções Deceptivas e Blocos de Construção

Considere o seguinte exemplo. Suponha que uma determinada função de aptidão f aplicada sobre uma cadeia c de l *bits* tenha a seguinte definição:

$$f(c) = \begin{cases} soma(c), & se\, soma(c) > 0; \\ l+1, & se\, soma(c) = 0. \end{cases} \tag{6.1}$$

Observe que a função f (ver Figura 6.1) premia qualquer algoritmo de busca pela adição de 1s à cadeia. No entanto, o melhor resultado possível é uma cadeia composta somente por 0s. Essa função é chamada de *deceptiva* do tipo "agulha-no-palheiro" (*needle-in-a-haystack*). Assim, não se espera que alguma metaheurística resolva esse problema, com l grande (acima de centenas), de maneira eficiente. Além disso, não há muito interesse prático em problemas desse tipo visto que são pouco encontrados em problemas do mundo real. Em geral, mesmo existindo um grande número de variáveis, a interação costuma ser entre poucas delas e de pequena ordem (2 ou 3)(Goldberg, 2002).

Considere agora uma função f que seja deceptiva por partes (Harik, 1999). Essa é uma função aditiva dada por $f(c) = \sum_{i=1}^{m} fdi$ sobre uma cadeia c de l *bits* que trabalhe agrupando k *bits* em m funções deceptivas fdi, tal que l=k*m, como ilustrado na Figura 6.2. O valor ótimo de um subproblema (fdi) é dado por uma cadeia de k *bits* consecutivos iguais a 0, portanto, o ótimo global de f é a cadeia composta por l 0s. Por outro lado, as armadilhas (ótimos locais, ver Capítulo 1) ocorrem para conjuntos de k *bits* consecutivos iguais a 1. A solução ótima desse problema é encontrada quando todos os subproblemas possuem o seu valor ótimo, nesse caso, a solução ótima de c é uma cadeia formada apenas por 0s em todos os subproblemas.

David Goldberg e seu grupo de pesquisa.

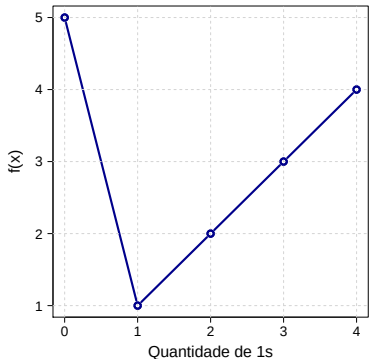


Figura 6.1: Exemplo de função deceptiva da Equação 6.1, para $l = 4$, também conhecida como *Trap*-4 ou armadilha de 4 bits.

BBs em problemas de otimização são variáveis com alta correlação e/ou dependência (ver Seção 2.3.4). Soluções que mantém essas correlações (ou parte delas) tendem a ter valores de aptidão relativamente melhores. Considera-se que indivíduos bem avaliados são formados pela justaposição de BBs supostamente bem avaliados.

Alguns tipos usuais de funções aditivas presentes na literatura da área (Goldberg, 2002) envolvem:

- Problemas separáveis que contém subproblemas não sobrepostos, em que os BBs correspondem a *bits* contíguos (ver Figura 6.2);

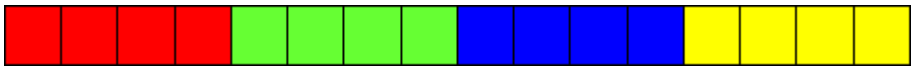


Figura 6.2: BBs contíguos e não sobrepostos.

- Problemas separáveis que contém subproblemas não sobrepostos. Os *bits* de um BB podem estar embaralhados tornando o problema mais difícil (ver Figura 6.3);

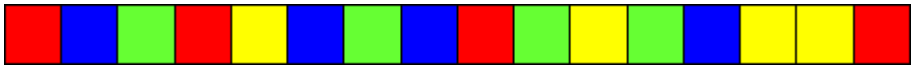


Figura 6.3: BBs embaralhados e não sobrepostos.

- Pode haver sobreposição em *bits* (um mesmo *bit* é utilizado em dois ou mais subproblemas) entre subproblemas vizinhos, com subproblemas definidos em blocos contíguos (ver Figura 6.4);

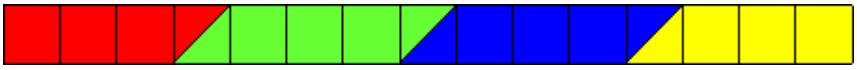


Figura 6.4: BBs contíguos e sobrepostos.

- Novamente, os *bits* podem estar embaralhados gerando um problema ainda mais difícil (ver Figura 6.5).

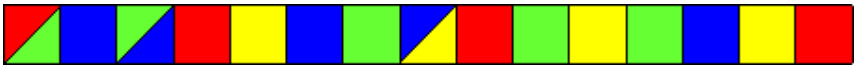


Figura 6.5: BBs embaralhados e sobrepostos.

Um GA convencional poderia até ser capaz de encontrar a solução ótima para alguns dos subproblemas. No entanto, o operador de recombinação (*crossover*) teria grande probabilidade de destruir tais soluções parciais, pois tal operador não usa nenhuma informação sobre possíveis correlações na escolha do ponto de corte (ver Seção 2.5). Isso significa que BBs que tenham sido gerados com valores corretos podem ser destruídos. Para tentar evitar esse problema, deve-se descobrir quais as variáveis que interagem entre si e que precisam ser analisadas em conjunto. Algoritmos capazes de descobrir e utilizar essa interação são conhecidos, dentre outros nomes, como Algoritmos Genéticos Competentes (Goldberg, 2002).

6.2 Principais PMBGAs

A etapa mais complexa nos PMBGAs é a construção de um modelo probabilístico, que não é uma tarefa trivial. Modelos mais complexos são mais poderosos e precisos, mas demandam grande esforço computacional. O oposto ocorre com modelos mais simples. Em geral, existe um compromisso entre a precisão do modelo construído e a eficiência do processo de estimação (Pelikan et al., 2002). PMBGAs utilizam o conceito de distribuição de probabilidade conjunta (JPD, do inglês *Joint Probability Distribution*). A JPD é a distribuição da probabilidade de todas as combinações possíveis de duas ou mais variáveis de uma amostra. Como essa JPD é difícil de ser obtida em problemas de larga-escala, utiliza-se simplificações para encontrar uma estimativa dessa distribuição. Os modelos utilizados pelos PMBGAs são categorizados como Univariados, Bivariados ou Multivariados, conforme apresentados nas Seções 6.2.1, 6.2.2 e 6.2.3.

6.2.1 Modelos Univariados

Modelos probabilísticos univariados desconsideram qualquer correlação entre as variáveis. Isso corresponde a ter todos os BBs com tamanho igual a 1. Desse modo, a JPD de uma solução x na amostra, $P(x)$, equivale ao produto das probabilidades marginais de cada variável dessa solução:

$$P(x) = \prod_{i=1}^{D} p(x_i), \tag{6.2}$$

tal que D é a quantidade de variáveis de um indivíduo x da população e $p(x_i)$ é a probabilidade marginal da variável i. Essa probabilidade é calculada como a freqüência de um determinado valor da variável x_i na população, dividido pelo tamanho da população. Um exemplo é apresentado na Figura 2.11 da Seção 2.3.4.

A Figura 6.6 representa graficamente um modelo de probabilidade que não assume dependência entre as variáveis. Nesse modelo, cada x_i é um nó de um grafo representando as variáveis. A ausência de aresta indica que não há correlação entre as variáveis.

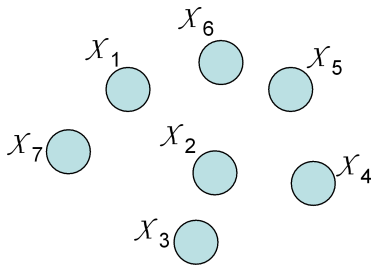


Figura 6.6: Ilustração de um Modelo Univariado.

Devido à simplicidade, as metaheurísticas que utilizam esse modelo são computacionalmente eficientes e apresentam desempenho acima da média em problemas nos quais não existe interação entre variáveis (Harik et al., 1999; Mühlenbein & Paass, 1996). Contudo, falham em problemas mais difíceis, isto é, BBs maiores do que 1. Exemplos de algoritmos univariados são PBIL, UMDA e cGA, ver Seção 6.2.1.

6.2.2 Modelos Bivariados

Os modelos bivariados consideram que pode haver dependências entre pares de variáveis do indivíduo, ou seja, BBs de tamanho igual a 2.

$$P(x) = \prod_{i=1}^{D} p(x_i|x_{j(i)}), \tag{6.3}$$

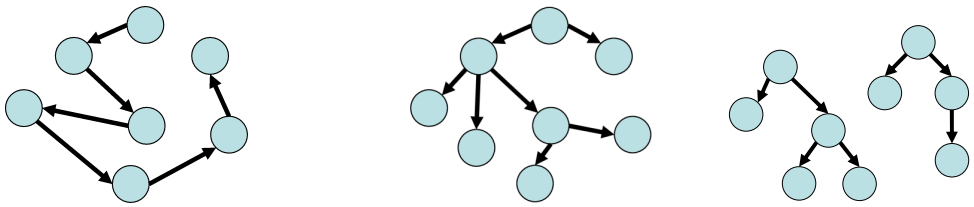
tal que $x_{j(i)}$ é a variável, se existir, da qual x_i depende. Devido a esse fato, o modelo de probabilidade torna-se mais complexo, necessitando de uma rede de probabilidades entre as variáveis para representá-lo (ver Figura 6.7). Tal rede precisa ser aprendida a partir de uma amostra por meio de algum algoritmo. Em geral, esse é um algoritmo de otimização com função objetivo correspondente a um critério que avalia o quanto a rede/modelo representa a amostra. Como a construção do modelo é um problema de otimização, existe um aumento no custo computacional do PMBGA correspondente. Por outro lado, PMBGAs com modelos bivariados possuem desempenho superior ao dos univariados (menor quantidade de avaliações da funções objetivo e maior TS, ver Seção 3.1) para problemas que possuem interação entre pares de variáveis. Como exemplos de PMBGAs com modelos

bivariados podem ser citados (ver Figura 6.7):

a) *Mutual Information Maximization for Input Clustering* (MIMIC) (Bonet et al., 1996): utiliza uma estrutura de cadeia de distribuição de probabilidade, partindo de uma variável e conectando-se a uma outra, que se conecta a outra e assim por diante, formando um grafo acíclico orientado com graus de entrada e saída menores ou iguais a 1. O objetivo é encontrar uma ordem nas conexões que se aproxime ao máximo a JPD, maximizando o critério de informação mútua. A informação mútua é uma medida que mensura a dependência mútua de duas variáveis. A construção da cadeia é feita por um algoritmo guloso, não garantindo o melhor modelo baseado em estrutura de cadeia.

b) *Combining Optimizers with Mutual Information Trees* (COMIT) (Baluja & Davies, 1997): utiliza uma árvore com grau de entrada menor ou igual a 1[2] para encontrar os relacionamentos entre pares de variáveis. Portanto, possui um custo computacional maior do que o MIMIC. A árvore é construída utilizando o algoritmo *Maximum Weight Spanning Tree* (MWST) (Chow & Liu, 1968) que garante encontrar a melhor árvore dada a maximização da informação mútua como função objetivo. Ao contrário do MIMIC, no qual uma variável i exerce influência apenas sobre uma única variável j, no COMIT é possível identificar se uma mesma variável influencia diversas variáveis.

c) *Bivariate Marginal Distribution Algorithm* (BMDA) (Pelikan & Mühlenbein, 1999): visto como uma extensão do UMDA, sendo mais geral do que o MIMIC e o COMIT, pois trata de problemas lineares (com variáveis independentes), bem como de problemas com interação entre pares de variáveis. Ao contrário do COMIT que usa uma árvore, o BMDA usa uma Floresta (um conjunto de árvores mutuamente independentes). As dependências entre os pares são detectadas por meio do cálculo da estatística de χ^2 de Pearson (Montgomery, 1997) para cada par. A estatística é obtida somando-se razões dadas pelos quadrados das diferenças entre as freqüências observadas e as esperadas, divididos pelas freqüências esperadas. Quando as variáveis são independentes, assume-se que não existe associação entre o par e, nesse caso, o valor da estatística será zero. Se uma dada variável não se relacionar com nenhuma outra, poderá ser tratada isoladamente.



a) Cadeia de Conexões (MIMIC). b) Árvore (COMIT). c) Floresta (BMDA).

Figura 6.7: Ilustração de Modelos Bivariados por grafos.

[2]Significa que um determinado nó de um grafo orientado pode receber uma aresta ou não, no caso de um nó raiz.

6.2.3 Modelos Multivariados

Modelos probabilísticos que considerem a dependência de mais do que duas (grau de entrada maior do que dois) variáveis são chamados de modelo multivariados. Nesse caso, a distribuição pode ser dada por:

$$P(x) = \prod_{i=1}^{D} p(x_i|pa_i), \tag{6.4}$$

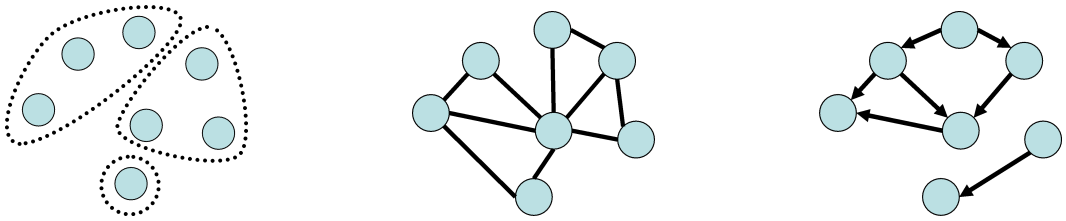
tal que pa_i são instâncias de Pa_i, que é o conjunto de variáveis do qual x_i depende. A construção de uma rede de probabilidades que represente as dependências pode tornar-se consideravelmente mais complexa. Portanto, o tempo computacional para construí-la pode crescer exponencialmente, fazendo com que uma busca exaustiva pela melhor rede, testando todas as possibilidades de conexão, seja impraticável. Desse modo, é comum o uso de heurísticas gulosas, dada a eficiência delas na construção desse tipo modelo. Apesar de não garantirem que a melhor rede seja encontrada, a qualidade dos modelos gerados pela heurística é aceitável. Como exemplos de algoritmos com modelos multivariados, podem ser citados (ver Figura 6.8):

a) *Extended Compact Genetic Algorithm* (ECGA) (Harik, 1999): é uma extensão do cGA (ver Algoritmo 6.1 na Seção 6.3.1). O modelo de probabilidade utilizado no ECGA é chamado Modelo de Produto Marginal (MPM). Esse modelo aceita distribuição marginal univariada, bivariada ou multivariada. Contudo, não aceita probabilidade condicional nem sobreposição de dependências. Isso significa que um grupo de variáveis interage entre si, sem que uma delas possa interagir com variáveis de outro grupo. Para gerar o modelo, é utilizada uma busca gulosa que agrupa variáveis, ou grupos de variáveis, dois a dois, de modo similar a um algoritmo de agrupamento hierárquico (Mitchell, 1997). O agrupamento é aceito caso uma determinada métrica seja melhorada. Mais detalhes são apresentados na Seção 6.3.2;

b) *Factorized Distribution Algorithm* (FDA) (Mühlenbein & Mahnig, 1999): pode ser considerado como uma extensão do BMDA. De acordo com o Teorema de Fatoração é possível dizer quais os esquemas (teoria dos esquemas (Holland, 1975)) necessários para gerar a distribuição da população atual, a partir da distribuição de Boltzmann. Para isso, a distribuição é fatorada em um produto de probabilidades marginais e condicionais. A principal desvantagem é o fato de o FDA necessitar que essa fatoração do problema seja informada pelo usuário na forma de uma decomposição da função, o que nem sempre está disponível em problemas do mundo real;

c) *Bayesian Optimization Algorithm* (BOA) (Pelikan et al., 1999): codifica a distribuição de probabilidade conjunta do problema utilizando uma Rede Bayesiana. Diferentemente do FDA, que necessita que o modelo seja construído manualmente e apresen-

tado ao algoritmo de otimização, no BOA original a rede é construída usando um algoritmo guloso que adiciona arestas conectando variáveis. Caso uma determinada métrica seja melhorada, a aresta é mantida. Os dados utilizados para a construção da rede são um subconjunto de soluções promissoras, extraído de uma amostra (soluções) do problema. Mais detalhes são apresentados na Seção 6.3.3.



a) Produtos Marginais (ECGA). b) Fatoração Triangular (FDA). c) Redes Bayesianas (LFDA, BOA, EBNA).

Figura 6.8: Ilustração de Modelos Multivariados por grafos.

6.3 Algoritmos Genéticos Competentes

Pesquisas que identificaram e avaliaram dificuldades dos GAs de 1^a geração em preservar BBs resultaram nos GAs de 2^a geração. Esses algoritmos, também chamados de GAs competentes, são PMBGAs capazes de resolver problemas complexos rapidamente, confiavelmente e com precisão (Goldberg, 2002). Nos últimos anos, tem havido um progresso considerável no estudo e desenvolvimento de vários GAs competentes (Goldberg et al., 1993; Harik, 1999; Mühlenbein & Mahnig, 1999; Pelikan et al., 2000).

Embora o princípio utilizado para identificar e preservar BBs em cada um desses algoritmos seja consideravelmente diferente, todos produziram aumento significativo do desempenho de GAs para problemas complexos. Isso ocorre porque, ao evitar a quebra de BBs, aumenta-se a faixa de valores dos parâmetros para os quais o GA tem sucesso.

Dentro do espaço de possíveis configurações dos parâmetros de um GA, existe uma região (*sweet spot*) que contém configurações que possibilitam ao GA resolver um determinado problema de maneira eficiente. Isso significa que, caso os parâmetros não estejam contidos nesse *sweet spot*, o GA não será capaz de resolver o problema de maneira eficiente. A região de sucesso do *sweet spot* pode ser representada pela região de Sucesso no mapa de controle, um gráfico de Pressão de Seleção (Ps, que corresponde à quantidade - em porcentagem do tamanho da população - de cópias do melhor indivíduo que são feitas para a geração seguinte) x Probabilidade de *Crossover* (Pc), conforme mostrado na Figura 6.9.

O processo de inovação ocorre pela mistura adequada de informações existentes em diferentes indivíduos da população. Em GAs, esse processo é realizado pelo operador de

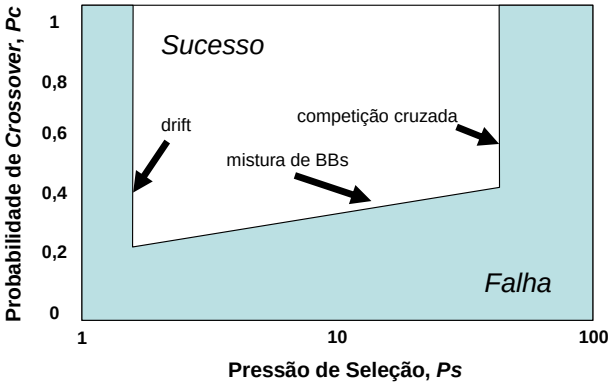


Figura 6.9: Mapa de controle ((Goldberg, 2002)).

crossover. Como comentado em (Goldberg, 2002), existe um limite entre a convergência prematura e a inovação a cada iteração do GA, dado pela combinação Pc e Ps .

Do lado esquerdo dessa figura tem-se o limite de *drift*. Quando a Ps é pequena o suficiente, variações estocásticas na população sobressaem mais do que a própria Ps a favor ou contra o aumento de instâncias de uma variável na população. Em tais circunstâncias, o valor de uma variável, ou um indivíduo inteiro, podem dominar a população, devido simplesmente à natureza estocástica do funcionamento do GA.

Do lado direito existe a região de falha dominada pela *competição cruzada*. Quando a Ps é alta, cópias dos melhores indivíduos propagam-se consideravelmente na população, reduzindo a diversidade e, conseqüentemente, a chance de inovação. A competição cruzada ocorre porque BBs diferentes podem ter influência similar no valor do *fitness* e o GA não consegue realizar uma mistura adequada com as melhores instâncias de cada BB (Goldberg, 2002).

O aumento na escala e dificuldade do problema encolhe rapidamente esse *sweet spot*, tornando quase impossível para um GA de 1^a geração encontrar a solução ótima do problema. Para superar essa dificuldade, GAs competentes incorporam reconhecimento e mistura eficientes de BBs, de modo que problemas difíceis possam ser resolvidos em tempo polinomial (freqüentemente em tempo sub-quadrático) (Goldberg, 2002), ao contrário do tempo exponencial que pode ser utilizado por GAs de primeira geração(Goldberg, 2002).

A seguir, são apresentados três PMBGAs. O primeiro deles (cGA, ver Seção 6.3.1) desconsidera qualquer interação entre as variáveis (posições da cadeia que representa o indivíduo). O segundo (ECGA, Seção 6.3.2) considera que as variáveis podem estar relacionadas em BBs, porém desconsidera a possibilidade de sobreposição entre esses BBs. O terceiro (BOA, Seção 6.3.3) assume possibilidades de iteração mais complexas, representadas por Redes Bayesianas (Heckerman et al., 1994), sendo capaz de lidar com sobreposição. Claramente o terceiro algoritmo é mais poderoso que os anteriores. No entanto, esse possui um custo computacional maior. Por modelarem mais adequadamente

a interação entre variáveis, os dois últimos são considerados GAs Competentes, conforme argumentado em (Goldberg, 2002).

6.3.1 Algoritmo Genético Compacto

A maneira mais simples de estimar a distribuição de uma população é assumir que todas as variáveis são independentes. É nesse princípio que é baseado o Algoritmo Genético Compacto (cGA, do inglês *compact Genetic Algorithm*) (Harik et al., 1999).

A população de indivíduos do cGA não é representada explicitamente. Em seu lugar, é utilizado um vetor de probabilidades p de tamanho l, o mesmo tamanho dos indivíduos. Cada posição desse vetor indica a probabilidade do respectivo *bit* no indivíduo ser igual a 1. Esse vetor representa um modelo probabilístico da população que é utilizado pelo cGA no processo de otimização. Por não utilizar uma população explícita, o cGA é um algoritmo eficiente em termos de consumo de memória. Apesar de sua estrutura simplificada e a ausência de uma população, é possível demonstrar que o cGA pode ser comportar de maneira similar ao GA clássico (Jong, 1975; Harik et al., 1999).

Algoritmo 6.1 Pseudocódigo do cGA.

1: para $i = 1$ **até** l **faça** // Inicializar o Vetor de Probabilidades
2: $p[i] = 0.5$
3: fim para
4: Gerar dois indivíduos A e B com base no vetor p;
5: Determinar o *Melhor* e o *Pior* entre A e B;
6: para $i = 1$ **até** l **faça** // Atualizar o Vetor de Probabilidades influenciado pelo *Melhor*
7: **se** $Melhor[i] \neq Pior[i]$
8: **se** $Melhor[i] = 1$
9: $p[i] = p[i] + 1/N$;
10: **senão**
11: $p[i] = p[i] - 1/N$;
12: fim para
13: para i=0 **até** l **faça** // Verificar o critério de convergência
14: **se** $(p[i] \neq 0$ && $p[i] \neq 1)$
15: **ir para** o Passo 4;
16: fim para

O pseudocódigo do cGA é mostrado no Algoritmo 6.1, no qual n corresponde ao tamanho da população que teria um GA para produzir o mesmo efeito que o cGA. O procedimento "gerar" consiste em preencher uma cadeia de tamanho l com *bits* aleatórios, de acordo com o vetor de probabilidades p. Em cada posição i da cadeia, a probabilidade do *bit* gerado ser 1 é dada por $p[i]$. O procedimento "melhor" retorna aquele entre dois indivíduos que tiver melhor valor na função de avaliação e o procedimento "pior" retorna

aquele que tem pior valor. O modelo probabilístico é atualizado de modo a aumentar a probabilidade de aumentar a freqüência dos genes do melhor indivíduo.

Além do cGA, outros PMBGAs trabalham assumindo que todas as variáveis são independentes. Os principais algoritmos dessa classe são o PBIL (Baluja, 1994; Kvasnicka et al., 1996) e o UMDA (Mühlenbein & Paass, 1996).

6.3.2 Algoritmo Genético Compacto Estendido

O desempenho do cGA é similar ao GA clássico, pois ele é capaz de resolver os mesmos problemas apresentando desempenho semelhante. Dado que um GA típico não é um EA eficiente para solução de problemas que envolvam relações complexas entre variáveis, o mesmo vale para o cGA. Por esse motivo, foi desenvolvido o Algoritmo Genético Compacto Estendido (ECGA, do inglês *Extended Compact Genetic Algorithm*) (Harik, 1999), com o qual procura-se obter melhor desempenho nesse tipo de problema por meio do aprendizado das relações entre variáveis, processo conhecido como *linkage learning* (Harik, 1999; Harik & Goldberg, 2000).

A partir desse conhecimento, é possível usar operadores reprodutivos denominados competentes (Goldberg, 2002), capazes de transmitir às gerações seguintes BBs de alta qualidade que foram encontrados durante o processo de otimização. Com isso, o ECGA é capaz de resolver problemas deceptivos (ver Seção 6.1) de maneira muito mais eficiente do que o cGA.

O ECGA é baseado na idéia de que o processo de *linkage learning* é equivalente a encontrar uma boa distribuição de probabilidade. Para tal, o ECGA constrói uma função adequada de distribuição de probabilidade, um modelo que assume que há a possibilidade de dependência entre as variáveis, agrupando-as como mostrado na Figura 6.8a.

O princípio adotado para a criação desse modelo é que as distribuições mais simples são preferidas em relação às mais complexas. O conceito de simplicidade é definido em termos da complexidade de representação do conjunto de soluções de uma população do ECGA. Contudo, é necessário um compromisso entre simplicidade e precisão do modelo. Motivada por esse compromisso, é formulada a seguinte hipótese sobre a natureza das distribuições (Harik, 1999): "boas distribuições são aquelas para as quais a representação da distribuição usando a codificação atual, juntamente com a representação da população comprimida sob esse critério, é mínima". Esse é um *Minimum Description Length bias* sobre o modelo. Esse *bias* tem dois efeitos:

1. Penaliza diretamente modelos complexos, induzindo a minimização do tamanho do modelo;

2. Penaliza distribuições sem precisão, pois essas não possibilitam uma compressão adequada da população, de acordo com a Teoria da Informação (Harik, 1999).

Tabela 6.1: Exemplo da representação de um MPM.

[0]	$P_{[0]}$	[1, 3]	$P_{[1,3]}$	[2]	$P_{[2]}$
0	0,8	0 0	0,4	0	0,5
1	0,2	0 1	0	1	0,5
		1 0	0		
		1 1	0,6		

Tabela 6.2: Exemplo da representação de um MPM usando *1 bit* para armazenar as duas possíveis instâncias da partição [0, 1, 2, 3, 4].

[0, 1, 2, 3, 4]	$P_{[0,1,2,3,4]}$
0	i/n
1	$(n\text{-}i)/n$

O ECGA trabalha com o produto de distribuições marginais sobre uma partição de genes (MPM, do inglês *Marginal Product Models*). Diferentemente do cGA e do PBIL, que utilizam modelos univariados, o modelos MPM podem representar as probabilidades a respeito de mais de um gene em um mesmo momento por meio do agrupamento deles. Colchetes são utilizados para delimitar os grupos. No exemplo da Tabela 6.1, a posição do gene no cromossomo original (sem agrupamento) é apresentada entre colchetes. Na coluna $P_{[]}$, é apresentada a probabilidade da partição correspondente. Por exemplo, na partição do gene *0* ([0]), o gene tem *80%* de probabilidade de ser 0 e *20%* de probabilidade de ser 1.

As partições escolhidas foram [0], [1,3], [2]. Isso indica que o MPM representa os genes 0 e 2 independentemente. Por outro lado, representa os genes 1 e 3 em conjunto. De acordo com a distribuição de probabilidades do exemplo, uma cadeia não pode ser gerada com o último gene ([3]) com valor diferente do segundo gene ([1]), visto que as probabilidades da partição [1, 3] para as combinações (0 1) e (1 0) são iguais a 0. Portanto, são cadeias válidas para esse distribuição: 0101, 1101, 0000, 1111, etc. As cadeias 0100, 0011, etc, são cadeias inválidas.

Considere uma população com n indivíduos, na qual i indivíduos possuem representação 00000 e o restante, 11111. Para armazenar essa população seriam necessários $5n$ *bits*. Uma MPM que usasse a partição [0, 1, 2, 3, 4] poderia representar a probabilidade de cada uma das 32 combinações de bits ($2^5 bits$) independentemente do tamanho de n. No entanto, detectando que somente 00000 e 11111 teriam probabilidade maior do que 0, pode-se representar a cadeia 00000 somente como o *bit* 0 e a cadeia 11111 somente como o *bit* 1. Com isso, é possível representar a população usando apenas n *bits*, equivalente a 1 *bit* por indivíduo, resultando no MPM mostrado na Tabela 6.2.

A quantidade média de *bits* necessária para representar cadeias retiradas aleatoriamente dessa distribuição pode ser indicada pela medida da entropia da distribuição que é

adotada como critério de Complexidade da População Comprimida (C_p, do inglês *Compressed Population Complexity*):

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} log_2(p_{ij}), \tag{6.5}$$

tal que p_{ij} é a freqüência da j-ésima instância de genes dos genes que pertencem à partição i. Em outras palavras, $p_{ij} = n_{ij}/n$, sendo que n_{ij} é a quantidade de cromossomos na população (após a seleção) possuindo a instância de *bits* $j \in [1, 2^{k_i}]$ para a partição i, tal que k_i é o tamanho da partição. Ajustando as probabilidades do MPM e da população, o valor de C_p pode ser usado para estimar a compressão da população sob a distribuição. Desse modo, minimizando C_p, o melhor MPM é definido como o modelo que consegue a maior compressão da população. Assim, como não é possível gerar uma compressão alta utilizando um modelo incorreto, esse modelo não é escolhido para representar a distribuição da amostra.

A utilização de um MPM para modelar a distribuição da população de indivíduos consiste em duas fases:

1. Escolher o conjunto de particionamentos dos genes;

2. Ajustar o MPM contando a freqüência de cada um dos padrões de cada partição na população.

A eficiência de uma dada modelagem é dada pelo tamanho da população comprimida somado com o tamanho da representação do modelo propriamente dito. A qualidade do MPM é avaliada pelo critério da complexidade combinada (C_c, do inglês *Combined Complexity Criterion*) (Harik, 1999). De acordo com esse critério, a soma da complexidade do modelo (C_m, do inglês *Model Complexity*) com a C_p deve ser mínima. Mais detalhes teóricos sobre a utilização desse critério são encontrados no trabalho original (Harik, 1999).

O ECGA é sintetizado no Algoritmo 6.2. É importante observar que o MPM é construído sobre os indivíduos selecionados, não sobre a população inteira. Os indivíduos são selecionados de acordo com seus valores de aptidão, privilegiando os melhores para que propaguem sua carga genética para a próxima geração.

Algoritmo 6.2 Algoritmo do ECGA.

1: Gerar uma população aleatória;
2: Realizar seleção sobre essa população;
3: Construir um modelo (MPM) da população usando uma busca gulosa;
4: Se a população convergiu (todos são iguais), Finalizar;
5: Gerar uma nova população usando o modelo;
6: Voltar para o Passo 2.

A criação do MPM é realizada utilizando uma busca gulosa como no Algoritmo 6.3. Esse MPM será o modelo usado na próxima geração do ECGA. O processo de busca pela MPM é repetido a cada geração, construindo um novo modelo para que corresponda à população atual adequadamente.

Algoritmo 6.3 Algoritmo para criação do MPM.

1: Iniciar o algoritmo assumindo que todas as variáveis são independentes.;
2: Unir cada duas partições, testando todas as possibilidades, e verificando se há melhora no C_C;
3: Manter a união que obteve o menor C_C;
4: Ir para o Passo 2 enquanto for possível melhorar o valor do C_C.

De modo a superar o problema da quebra de BBs por operadores clássicos de recombinação, o ECGA utiliza as relações entre variáveis no operador de recombinação. Esse operador emprega informações obtidas no processo de *linkage learning* para combinar as soluções parciais dos subproblemas de k *bits*, a partir de um conjunto de soluções promissoras de mais alta qualidade, em uma solução completa para o problema de l *bits*. Para cada BB encontrado (partição do MPM), uma das possíveis cadeias, de acordo com as probabilidades do modelo, é selecionada aleatoriamente. As cadeias são unidas formando uma nova solução. Assim, a nova população tende a seguir a distribuição de probabilidade dos melhores indivíduos que foram selecionados para a criação do modelo.

Mais detalhes sobre o ECGA e discussões sobre os resultados obtidos por esse algoritmo podem ser encontrados em (Harik, 1999).

6.3.3 Algoritmo de Otimização Bayesiana

A idéia principal do Algoritmo de Otimização Bayesiana (BOA, do inglês *Bayesian Optimization Algorithm*) (Pelikan et al., 1999) é a utilização de redes Bayesianas acíclicas (como mostrado na Figura 6.8c) para representação das correlações entre variáveis. As variáveis são representadas como nós na rede e a existência de uma aresta entre dois nós representa que uma dessas variáveis está correlacionada à outra e, portanto, que devem estar conectadas. A não existência de uma aresta indica independência entre elas, portanto, podem ser combinadas por operadores modificadores de maneira independente uma da outra.

O procedimento do BOA é bem similar ao do ECGA. As diferenças principais são a construção do modelo e a amostragem a partir deste. O BOA começa como um ECGA, com uma população inicial aleatória e um processo de seleção. Em seguida, dados os indivíduos selecionados, o BOA constrói uma rede Bayesiana que represente as interações entre as variáveis com uma precisão adequada, em vez do MPM usado pelo ECGA. Para isso, são necessários uma métrica que informe a qualidade da rede e uma técnica de

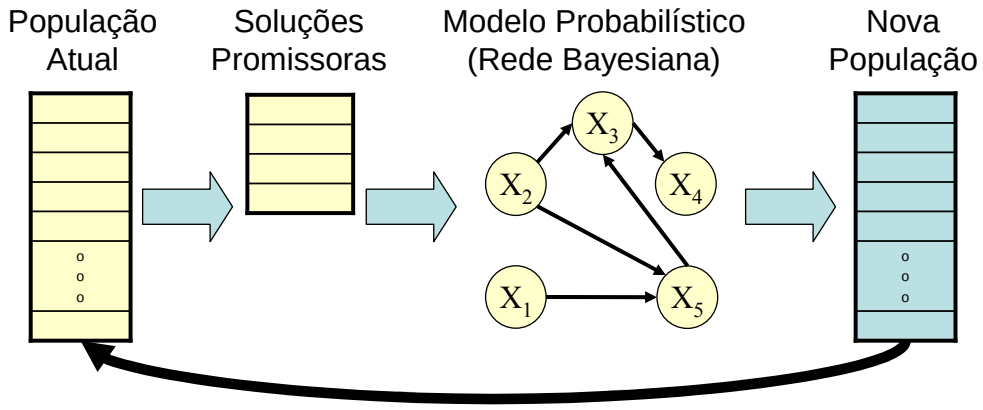


Figura 6.10: Ilustração do funcionamento do BOA.

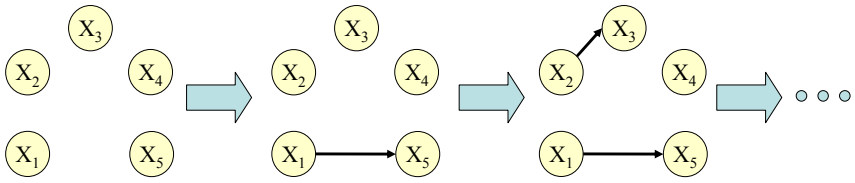


Figura 6.11: Ilustração da construção de uma Rede Bayesiana usando um algoritmo guloso.

busca para construção de redes. O BOA original usa uma simplificação da métrica BD (do inglês *Bayesian Dirichlet*) denominada K2, definida em (Pelikan et al., 1999). Essa métrica combina o conhecimento *a priori* sobre um problema e informações estatísticas de um determinado conjunto de dados. O objetivo é mensurar o quão próxima uma amostra gerada a partir de uma rede Bayesiana é do conjunto de dados original que foi utilizado na construção da rede. Uma ilustração do algoritmo do BOA é apresentada na Figura 6.10.

O método de busca gulosa para a criação da rede consiste em assumir inicialmente a existência de nenhuma aresta (um modelo em que todas as variáveis são independentes) e adicionar novas arestas sistematicamente uma por vez, mantendo a que resultar em maior melhora na rede segundo a métrica escolhida (ver Figura 6.11).

Uma vez construída a estrutura da rede, torna-se possível calcular as probabilidades condicionais de cada variável (ver Figura 6.12). Essas probabilidades serão usadas para amostrar novos indivíduos na população, como no caso do ECGA. O processo de construção do modelo e amostragem a cada geração é repetido como nos outros PMBGAs.

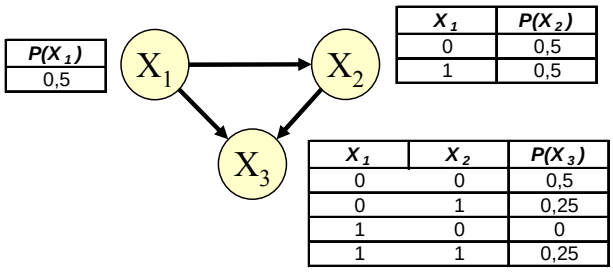


Figura 6.12: Exemplo de Tabelas de Probabilidade Condicional (CPT, do inglês *Conditional Probability Table*) para uma Rede Bayesiana com três variáveis. As tabelas indicam a probabilidade da variável em questão (P(X_2), por exemplo) ser 0 dado o valor da variável da qual ela depende.

6.3.4 Representações $\chi - árias$

É importante observar que os três PMBGAs descritos anteriormente baseiam-se na representação de um indivíduo como uma cadeia binária. No entanto, todos esses algoritmos possuem extensões $\chi - árias$, isto é, extensões que permitem que esses algoritmos lidem com uma cadeia de variáveis enumeráveis (números inteiros, caracteres ou outros tipos definidos pelo usuário). Optou-se por descrever apenas as versões binárias por simplicidade e facilidade de compreensão.

6.4 Considerações Finais

Neste Capítulo foram introduzidas questões relacionadas a dificuldades envolvendo GAs clássicos e novos ramos de desenvolvimento de EAs capazes de evitar tais dificuldades.

Os algoritmos propostos são capazes de modelar o problema e identificar interações entre variáveis (BBs). Com esse conhecimento, pode-se explorar o espaço de busca, isto é, gerar novas soluções sem destruir BBs, preservando combinações de valores de variáveis importantes na construção de soluções de melhor qualidade. Assim, ao invés de operadores clássicos de recombinação, que podem quebrar os BBs, utilizam-se operadores que mantém a distribuição dos dados de cada BBs. Esse procedimento permite que indivíduos com BBs bem avaliados consigam propagar essa informação para as populações seguintes, evitando que seja perdida e que somente possa ser recuperada por mutação.

Dentre os algoritmos brevemente apresentados, o BOA tem conseguido resolver um universo maior de problemas, por ser capaz de encontrar relações entre subproblemas sobrepostos (ver Figuras 6.4 e 6.5).

Apesar de serem eficientes na resolução de problemas com interação entre variáveis, esses algoritmos possuem uma desvantagem em relação aos GAs. O tamanho da população necessário para se realizar a identificação correta dos BBs, construindo um modelo adequado, é grande, bem como a quantidade de avaliações da função objetivo. Por esse

motivo, diversos estudos procuram reduzir a quantidade de avaliações requerida por esses algoritmos. Uma maneira de obter essa redução é encontrar mais rapidamente regiões promissoras no espaço de busca, como proposto neste trabalho. Duas propostas com esse fim são apresentadas no Capítulo 7.

7

Técnicas de Melhoria de Eficiência para GAs competentes

Uma característica fundamental em GAs competentes é a habilidade para identificar BBs (ver Seção 6.2.3). Tal informação pode ser usada para recombinação de indivíduos evitando quebra de BBs, possibilitando que GAs resolvam problemas relativamente complexos (com vários subconjuntos de variáveis que interagem entre si) usando uma quantidade polinomial de avaliações da função objetivo. Vários EDAs (ver Seção 2.3.4) realizam a identificação de BBs por meio da construção de um modelo probabilístico a partir de um conjunto de soluções promissoras extraído da população de indivíduos. Em geral, EDAs com princípios similares aos de GAs, mas que sejam capazes de identificar BBs e solucionar problemas difíceis de maneira rápida, precisa e confiável podem ser classificados como GAs competentes. Logo, esses algoritmos apresentam uma alta taxa de sucesso na obtenção do ótimo global com quantidade de avaliações relativamente baixa.

A qualidade do modelo probabilístico é crucial para o desempenho dos GAs competentes. Um modelo probabilístico de melhor qualidade resulta em um melhor aproveitamento do material genético da população. Uma população de tamanho grande provê mais informações para a construção do modelo, aumentando sua qualidade. Uma amostra grande também reduz a probabilidade de tomadas de decisão erradas no processo de seleção, evitando que indivíduos com melhores instâncias de BBs não sejam selecionados. Porém, se o tamanho da população for demasiadamente grande, o custo computacional para avaliá-la fica extremamente alto. Portanto, é importante a determinação de um tamanho de população adequado, que corresponde a um equilíbrio entre Qualidade e Custo Computacional.

Diversos trabalhos mostraram que o ECGA é capaz de resolver problemas de otimização aditivos-separáveis (com BBs que não se sobrepõem, ver Seção 6.1) utilizando um número de avaliações sub-quadrático em relação à quantidade de BBs (m) (Goldberg, 2002; Sastry & Goldberg, 2004; Lima et al., 2005). Contudo, para problemas consideravelmente grandes e complexos (milhares de BBs e grandes valores de k - tamanho do BB), mesmo um comportamento sub-quadrático ou logarítmico do tempo em relação a m pode não ser suficiente para garantir que o ECGA encontre o ótimo global em tempo aceitável na prática. Para superar essa desvantagem, têm sido desenvolvidas Técnicas de Melhoria de Eficiência (EETs, do inglês *Efficiency-Enhancement Techniques*) para tornar o ECGA e outros EAs aplicáveis a problemas extremamente grandes e complexos (Sastry, 2001; Srivastava, 2002; Sastry et al., 2004; Pelikan et al., 2008; Duque et al., 2008a).

As Seções deste Capítulo estão organizadas da seguinte maneira: na Seção 7.1 são brevemente apresentados alguns trabalhos em técnicas para melhorar GAs competentes. Na Seção 7.2 é comentada a importância do modelo inicial em um EDA. Nas Seções seguintes são apresentadas as duas técnicas de melhoria de eficiência para GAs competentes propostas neste trabalho: o gerenciamento do tamanho da população (ver Seção 7.3) a análise de tendência na população inicial (ver Seção 7.4).

7.1 Técnicas de Melhoria de Eficiência

GAs competentes podem resolver, em um tempo polinomial tratável, problemas que são intratáveis para métodos de otimização tradicionais e para EAs comuns que não empregam a construção de um modelo probabilístico. As EETs vão mais além, transformando problemas tratáveis em problemas práticos, isto é, em problemas para os quais o GA competente encontraria soluções rapidamente, mesmo envolvendo larga-escala. O ponto principal das EETs é o fato de serem genéricas, podendo ser empregadas em qualquer EDA. Essa flexibilidade é importante pois estabelece que uma EET não é apenas um ajuste no método de otimização e sim uma estratégia para melhoria do desempenho. Algumas categorias de EETs já desenvolvidas são apresentadas a seguir:

1. Continuidade no Tempo (Srivastava, 2002): esses métodos têm como objetivo a maximização da qualidade da solução em um determinado tempo limite de execução ou a minimização do tempo de execução para se obter uma solução em uma determinada qualidade, buscando aumentar o processamento serial (uma única população grande) em relação ao processamento paralelo (populações pequenas durante várias épocas);

2. Relaxamento da Avaliação (Sastry, 2001): substitui uma avaliação de função objetivo precisa e custosa por uma avaliação menos precisa, porém menos custosa,

similarmente ao apresentado na Seção 4.4;

3. Paralelização (Cantú-Paz, 2000): os processos do EDA são divididos e executados por vários elementos processadores em paralelo;

4. Hibridização (Yen & Lee, 1997): o EDA é auxiliado por um procedimento de busca local de modo a produzir soluções de qualidade superior ou reduzir a quantidade de recursos computacionais disponíveis.

Alguns EETs, como Construção Esporádica do Modelo (SMB, do inglês *Sporadic Model Building*) (Pelikan et al., 2008), não se encaixam em qualquer dessas categorias apresentadas acima. Outra EET (Duque et al., 2008b) utiliza compactação dos bits à media que os ótimos dos BBs são identificados, reduzindo um BB de valor ótimo igual a 00000 para apenas 0, ocasionando redução de espaço e possibilitando trabalhar com problemas maiores.

A maioria do EETs produz um efeito de aumento de desempenho constante sobre os recursos, ao invés da redução da complexidade do GA. Isso significa que um método de otimização de complexidade quadrática continuará sendo quadrático, mesmo sendo executado em metade do tempo após o uso de uma EET, diferentemente de uma mudança na própria estrutura do algoritmo, como a eliminação de uma das etapas na construção do MPM (Duque et al., 2008a). Porém, ao serem aplicadas em conjunto, essas técnicas normalmente apresentam efeito multiplicativo ou até mesmo super-multiplicativo. Assim, quando EETs são usadas em conjunto, o *speedup* global é a multiplicação dos *speedups* individuais ou, em alguns casos, maior do que a multiplicação dos *speedups* individuais.

Um exemplo de *speedup* multiplicativo é a utilização de Relaxamento na Avaliação, Hibridização e Paralelização juntos. Suponha que o Relaxamento na Avaliação resulte em um *speedup* de 4 vezes, Hibridização em *speedup* de 2 vezes e Paralelização em *speedup* de 50 vezes com 50 processadores. O *speedup* total seria 400 vezes. Esse ganho seria equivalente ao uso de 400 processadores somente com Paralelização, supondo um *speedup* linear. Um exemplo de *speedup* super-multiplicativo é a utilização de Continuidade no Tempo e Hibridização. A técnica de Continuidade no Tempo para instâncias consideravelmente grandes pode requerer o uso de memória de *swap* para lidar com uma única população grande. Por outro lado, a Hibridização pode reduzir o tamanho da população de modo que essa caiba na memória física de um computador, eliminando a necessidade de uso do *swap*. O *speedup* proporcionado pelas duas EETs não apenas multiplica o desempenho, mas atinge um desempenho muitas vezes superior, dada a maior velocidade da memória física quando comparada com a do disco rígido.

7.2 A importância do modelo inicial

Nesta Seção é apresentado um estudo sobre a importância do modelo probabilístico (MPM, ver Seção 6.3.2) gerado a partir da população inicial no processo de otimização de um EDA. Para problemas do tipo função aditiva, encontrar uma região promissora equivale a encontrar o modelo que particiona o cromossomo em BBs corretos. Usando o modelo correto, os BBs não serão quebrados, logo a busca ocorre em um espaço de busca menor (espaço dos BBs), otimizando os subproblemas já identificados.

Algoritmo 7.1 Análise da importância do modelo inicial do ECGA.
1: Iniciar o ECGA;
2: Gerar população inicial aleatoriamente;
3: Gerar o modelo probabilístico (MPM) correto para o problema;
4: Adicionar ruído comum nível N ao MPM correto;
5: Continuar com a evolução padrão do ECGA.

O objetivo desse estudo é verificar até que ponto o uso de um modelo mais/menos preciso reduz/aumenta as chances do ECGA convergir para o ótimo global. Para isso, foi adotado o procedimento descrito no Algoritmo 7.1. Esse algoritmo troca o MPM da 1^a geração, que seria construído pelo ECGA, pelo MPM correto (pré-conhecido) com a adição de ruído ao modelo em um nível controlado. Nas demais gerações, o ECGA constrói o MPM normalmente. A alteração do MPM por ruído pode ser ilustrada utilizando uma função deceptiva com $k = 4$ e $m = 4$, com BBs compostos por bits consecutivos. Suponha que o MPM correto seja dado por:

[0 1 2 3] [4 5 6 7] [8 9 10 11] [12 13 14 15].

Com um nível de ruído (N, do inglês *Noise*) de 10%, cerca de 10% das posições são permutadas aleatoriamente, gerando um MPM como:

[0 1 12 3] [4 5 6 15] [8 9 10 11] [2 13 14 7].

O modelo teórico de tamanho populacional desenvolvido para PMBGAs(Pelikan et al., 2000) estima que a solução correta de um problema deceptivo binário com alta TS requer uma população de tamanho mínimo n dado pela equação:

$$n = 2^k.m^{DG}.log_2(m), \tag{7.1}$$

tal que DG é um valor próximo de 1,2 e tende a 1 quando $k * m$ tende ao infinito. Sastry (Sastry, 2001) utilizou o método da bisseção para determinar o tamanho mínimo da população de modo que o ECGA convergisse em 30 repetições consecutivas, considerando

satisfatória a convergência da população para uma solução com m-1 subproblemas solucionados corretamente. O valor de $DG = 1,2$ pode gerar populações com praticamente o dobro tamanho de $DG = 1$, tendendo a aumentar para casos maiores. Por isso, EETs têm sido desenvolvidas de modo a reduzir o valor de DG.

A influência do uso do MPM correto na 1^a geração, sobre o tamanho mínimo da população necessário para o ECGA ter sucesso, pode ser avaliada usando n menor do que o recomendado pela Equação 7.1. Com isso, é possível verificar se um MPM inicial de alta qualidade pode aumentar o desempenho do ECGA. Para isso, compara-se a TS do ECGA padrão com uma população pequena, o que deve resultar na construção de um MPM de baixa qualidade, com a TS do ECGA inicializado com um MPM de alta qualidade (correto ou com um certo nível de ruído), construído manualmente.

Esse estudo foi realizado para a função deceptiva da Equação 8.2 utilizando o ECGA. Foram realizados testes com m=2, 3, 4, 5, 10, 15, 20 e 30. Para essas configurações, os tamanhos da população, de acordo com o modelo teórico de tamanho populacional, seriam n=36, 94, 168, 256, 842, 1612, 2518 e 4650. Após análises preliminares da execução do Algoritmo 7.1, foi possível verificar que essa população poderia ser reduzida sem perda considerável na TS se apenas o MPM da 1^a geração fosse melhor. Os tamanhos reduzidos seguiram um padrão próximo a $n = (k*m)^{1,6}$, que é cerca de metade do tamanho previsto no modelo teórico. Para cada nível de ruído (que é uma maneira de controlar a qualidade do MPM da 1^a geração) em cada problema com m BBs, foram realizadas 100 repetições do ECGA. Os resultados são apresentados na Tabela 7.1.

Tabela 7.1: TS, em 100 execuções, do ECGA padrão (ECGA) em relação ao ECGA com N níveis de ruído.

m	n	ECGA	N=20%	N=15%	N=10%	N=5%	N=0%
2	28	30%	36%	41%	58%	56%	60%
3	52	26%	33%	51%	54%	56%	54%
4	84	14%	37%	45%	45%	57%	65%
5	120	15%	32%	33%	44%	73%	74%
10	366	5%	12%	15%	29%	41%	91%
15	700	0%	36%	47%	71%	86%	97%
20	1110	0%	42%	63%	72%	90%	94%
30	2122	0%	37%	51%	75%	91%	99%

De acordo com os resultados, é possível verificar que a influência do modelo inicial é expressiva, sendo possível concluir que o mesmo direciona o algoritmo a explorar regiões promissoras do espaço de busca. Enquanto a TS do ECGA padrão diminui conforme aumenta a quantidade de subproblemas (m), a TS do ECGA utilizando MPMs mais próximos do correto aumentam juntamente com m. Para $m > 10$, o ECGA padrão não foi capaz de encontrar o ótimo global em qualquer das 100 repetições. Para o mesmo m,

o ECGA com $N = 20\%$ obteve aproximadamente $TS = 40\%$ e com o modelo correto ($N = 0\%$) atingiu quase $TS = 100\%$. Portanto, pode-se concluir que a apresentação de um MPM inicial de alta qualidade é capaz de aumentar substancialmente a eficiência do desempenho do ECGA para o problema deceptivo testado, mesmo com uma população de tamanho reduzido a praticamente metade do valor teórico estimado.

A partir das conclusões obtidas nesse experimento, foram propostas duas EETs para encontrar MPMs iniciais de alta qualidade com populações menores, capazes de orientar o ECGA para regiões promissoras sem aumento no custo computacional:

1. Gerenciamento do Tamanho da População (ver Seção 7.3);
2. Análise de Tendência na População Inicial (ver Seção 7.4).

7.3 Gerenciamento do Tamanho da População

Nesta Seção é proposto o Gerenciamento do Tamanho da População (PSM, do inglês *Population Size Management*). O PSM é uma EET que consiste em reduzir o tamanho de população depois da primeira geração para usar os recursos computacionais disponíveis de maneira mais eficiente. O PSM pode ser estendido para realizar reduções sucessivas do tamanho da população em gerações subseqüentes, melhorando ainda mais a eficiência do algoritmo.

Para entender o princípio do PSM, é importante entender como EDAs processam as informações de uma população e porque precisam de populações grandes. De acordo com (Harik et al., 1999), há dois modelos para se decidir o tamanho da população:

1. o primeiro modelo é chamado de Ruína do jogador (do inglês, *Gambler's ruin*) e
2. o segundo modelo é chamado de geracional.

O primeiro modelo é menos conservador e resulta em uma melhor correspondência com resultados experimentais obtidos por um GA clássico. Porém, para EDAs multivariados, o segundo modelo permite estimar tamanhos de população mais próximos dos ideais obtidos em análises experimentais.

Esse modelo é formulado buscando determinar o tamanho de população mínimo para que não sejam tomadas decisões erradas no processo de seleção sobre a população inicial, possibilitando a construção de MPMs de alta qualidade. Desse modo, o modelo de geração é mais apropriado para EDAs, pois as decisões tomadas na primeira geração influenciam consideravelmente a busca nas gerações subseqüentes (ver Seção 7.2).

O ponto chave do PSM é a geração de um modelo de alta qualidade na primeira geração, usando um tamanho de população relativamente grande, obtido a partir do

modelo de tamanho populacional (ver Seção 7.2). Como mostrado na Tabela 7.1, o ECGA constrói modelos de qualidade consideravelmente baixa se utilizadas populações pequenas. A partir de um modelo de alta qualidade, que pode ser obtido a partir de uma população grande, o ECGA restringe a pesquisa subseqüente a uma região mais promissora. As justificativas para o sucesso do ECGA usando PSM (ECGA+PSM) podem ser sintetizadas da seguinte maneira:

1. O ECGA constrói um modelo do problema baseado em um subconjunto de soluções de promissoras (soluções de alta qualidade em relação às demais soluções já obtidas até o momento), extraído do conjunto de soluções (população);

2. A população da geração seguinte é obtida a partir desse modelo (processo de amostragem). Novos indivíduos são gerados pela concatenação de instâncias dos BBs das soluções promissoras, priorizando aquelas de maior freqüência;

3. Como o conjunto de soluções promissoras é de alta qualidade, a combinação das instâncias de seus BBs tende a gerar soluções de alta qualidade;

4. A probabilidade da obtenção de soluções de alta qualidade está diretamente relacionada com o tamanho da amostra, como foi apresentado na Seção 7.2;

5. Assim, é importante que se tenha uma grande quantidade de instâncias de BBs com soluções de alta qualidade. Se houver uma única instância que seja a solução ótima de um subproblema, a probabilidade dela não ser selecionada é significativa, dificultando sua propagação para a geração seguinte. Esse comportamento pode ser facilmente verificado no ECGA, uma vez que ele não possui elitismo;

6. Além disso, se o tamanho da população for pequeno, a amostragem do espaço de busca terá sido menor e o conjunto de soluções selecionadas para a construção do MPM terá uma maior chance de conter soluções de baixa qualidade. Com isso, o MPM gerado será prejudicado, encontrando BBs incorretos, como mostrado no estudo anterior da Seção 7.2;

Uma vez em uma região promissora, identificada por um MPM de alta qualidade, o ECGA+PSM continuará a busca naquela região. Porém, o espaço de busca foi reduzido e várias decisões importantes foram tomadas corretamente na primeira geração. Essa característica importante viabiliza o uso de uma população de tamanho reduzido para a segunda geração, como apresentado na Figura 7.1. O tamanho reduzido foi encontrado experimentalmente utilizando o algoritmo de bisseção. Esse algoritmo reduz o tamanho da população para a segunda geração, a partir do tamanho da população inicial, enquanto for

possível atingir uma TS de 100% em r execuções ($r = 30$) em um determinado problema de *benchmark* (função deceptiva).

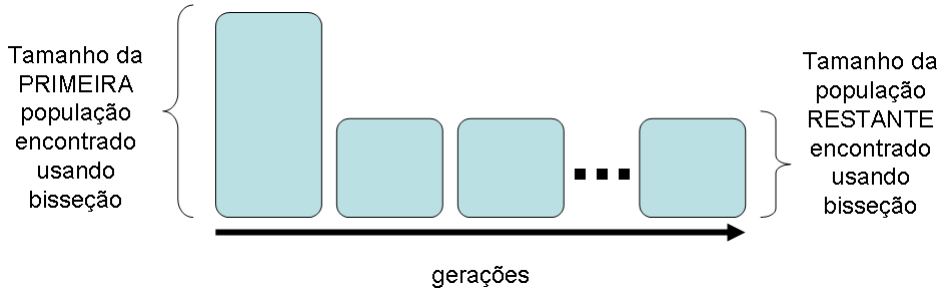


Figura 7.1: Tamanho da população ao longo das gerações (ECGA+PSM).

Argumento similar sobre a qualidade do modelo da 1^a para a 2^a geração pode ser usado para o modelo da 2^a para a 3^a geração e assim sucessivamente, gerando um VPSM (do inglês, *Variable PSM*). Essa idéia é ilustrada na Figura 7.2.

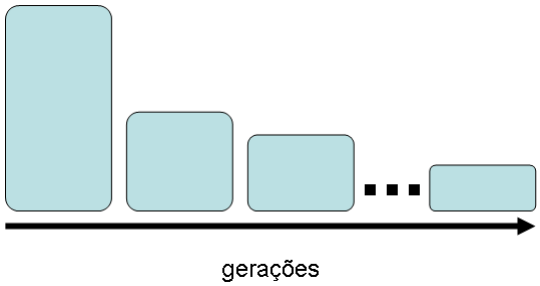


Figura 7.2: Tamanho da população ao longo das gerações (ECGA+VPSM).

Em princípio, o PSM pode ser aplicado a qualquer EDA, como ECGA e BOA. O procedimento básico de um ECGA+PSM é apresentado no Algoritmo 7.2. As diferenças entre o ECGA e o ECGA+PSM estão em negrito. Os resultados de experimentos com o do uso dessa EET são apresentados no Capítulo 8.

Algoritmo 7.2 ECGA+PSM.

Fase 1

1: Inicialização: a população é geralmente inicializada com indivíduos aleatórios - **usar um tamanho população maior do que o modelo teórico**;

2: Avaliação do valor de *fitness* dos indivíduos;

3: Seleção das soluções promissoras: o ECGA geralmente emprega seleção por torneio;

4: Construção do modelo probabilístico (MPM) utilizando uma busca heurística gulosa;

Fase 2

5: Geração de novos indivíduos: os novos indivíduos são criados por amostragem a partir do modelo probabilístico - **agora, usar um tamanho de população reduzida;**

6: Substituição dos pais pelos filhos;

7: Repetir os Passos 2 a 6 até que um critério de parada seja satisfeito.

7.4 População Inicial Tendenciosa

Nesta Seção é proposta outra EET capaz gerar um modelo de alta qualidade na primeira geração por meio de uma tendência na população inicial. Essa técnica foi chamada de BIP (do inglês, *Biased Initial Population*). Para uma metaheurística populacional padrão, como não se sabe a distribuição mais adequada de valores dos genes, utiliza-se uma distribuição uniforme com probabilidades idênticas para cada valor do gene. Em um problema binário, a probabilidade de um gene ser 1 na 1^a geração é de 50%, a mesma probabilidade de ser 0. Caso fossem três valores por variável ao invés de dois, as probabilidades seriam cerca de 33,33% para cada valor e assim por diante para variáveis de maior aridade. O BIP emprega probabilidades diferentes de 50% (no caso de variáveis binárias) na 1^a geração para escolher os valores de cada gene na construção do modelo inicial.

Foi realizado um estudo que se assemelha ao do apresentado na Seção 7.2, no sentido de avalia a influência da tendência no tamanho da população do ECGA. Porém, o estudo agora está relacionado à geração de uma população inicial com distribuição de valores de maneira heurística. Ao invés de níveis de ruído (N), foram estudados níveis de tendência (B, do inglês *Bias*). De modo similar ao estudo do ruído, o estudo de tendência utiliza populações menores, cerca de metade do tamanho estimado pelo modelo teórico.

Os testes utilizam a função deceptiva dada pela Equação 8.2, para subproblemas de tamanho $k = 4$ e $k = 5$. Na Tabela 7.2, pode-se verificar que os resultados obtidos para $k = 4$ por meio da apresentação de populações tendenciosas ao ECGA são semelhantes aos resultados apresentados na Tabela 7.1 por meio de um MPM correto ($N = 0$).

Uma tendência de 10% na população (B40) no sentido do ótimo global, ou seja, 40% de chance de ser 0, aumenta consideravelmente a TS se comparado à inicialização padrão (ECGA = B50) sem tendência (50% de chance de ser 0). No entanto, uma tendência na direção contrária à do ótimo global (B60) leva a um completo fracasso. Mesmo nesse caso, obtém-se uma informação relevante. Apesar da convergência para instâncias de BBs correspondentes a armadilhas, as partições encontradas pelo MPM são próximas às corretas. Esse tipo de informação também pode ser utilizado para o desenvolvimento de uma EET robusta a tendências ruins, isto é, tendências que orientem o algoritmo a ótimos locais.

Utilizando a configuração B30, a TS é de 100% em todos os tamanhos problema. Para B40, o algoritmo também atinge TS=100%, exceto para $m = 15$. Como foram feitas 30 execuções, significa que 1 de 30 não encontrou o ótimo. No entanto, é importante notar o aumento substancial a partir de 0% no ECGA padrão.

Tabela 7.2: TS usando ECGA padrão em relação a configurações do ECGA+BIP (B30, B40 e B60) para k=4.

m	n	B30	B40	ECGA	B60
10	366	100%	100%	5%	0%
15	700	100%	96%	0%	0%
20	1110	100%	100%	0%	0%
30	2122	100%	100%	0%	0%

Em um problema mais complexo, $k = 5$ (ver Tabela 7.3), a configuração B30 garante a eficiência do ECGA. Contudo, diferentemente do experimento com $k = 4$, a eficiência da B40 diminui à medida que m aumenta. Esse resultado representa uma característica a ser melhor estudada em um trabalho futuro. Apesar disso, a TS com B40 é superior à do ECGA padrão.

Tabela 7.3: TS usando ECGA padrão em relação a configurações do ECGA+BIP (B30, B40 e B60) para k=5.

m	n	B30	B40	ECGA	B60
10	522	100%	61%	0%	0%
15	1000	100%	33%	0%	0%
20	1584	100%	16%	0%	0%
30	3032	100%	15%	0%	0%

A partir desses resultados, é possível concluir que a identificação da tendência em problemas deceptivos (que pode advir de conhecimento *a priori* sobre o domínio de um problema) resulta em uma melhoria no desempenho do ECGA, possibilitando trabalhar com populações menores. Na Seção 8.2.3 do Capítulo 8, são apresentados resultados de experimentos para tentar identificar e utilizar a tendência em problemas deceptivos, além de um estudo para identificar qual seria o valor mínimo de tendência necessário para se obter uma melhora significativa em relação ao ECGA padrão.

7.5 Considerações Finais

Neste Capítulo foi apresentado um estudo sobre o modelo inicial, mostrando a importância do ECGA tomar decisões corretas em relação ao agrupamento de BBs, desde a primeira geração. Nesse estudo, foi possível verificar que um modelo de alta qualidade viabiliza o uso de populações menores. Por esse motivo, foram propostas duas EETs: o PSM e a BIP.

Com o PSM procura-se gerar modelos de alta qualidade por meio de uma amostra maior na primeira geração, possibilitando a redução do tamanho da amostra nas gerações

seguintes. Esse efeito ocorre pelo fato de, implicitamente, um modelo melhor na primeira geração direcionar a busca para uma região mais promissora.

Por outro lado, com a BIP tenta-se explorar tendências na população inicial de modo que seja possível identificar qual delas direciona o algoritmo à armadilha e qual direciona a regiões mais promissoras que são mais próximas do ótimo global. Mesmo com tendências ruins (advinda, por exemplo, de conhecimento *a priori* equivocado sobre o domínio de um problema), as partições do MPM são determinadas na primeira geração com um erro baixo, utilizando uma população de tamanho reduzido.

Essas duas EETs são utilizadas em problemas binários e para funções aditivas (ver Seção 6.1). Deve-se observar que, para problemas contínuos, outra técnica de determinação de regiões promissoras foi proposta e apresentada no Capítulo 5.

8

Análise Experimental

Neste Capítulo são apresentados os resultados dos experimentos referentes às técnicas de busca por regiões promissoras e de melhoria de eficiência propostas nos Capítulos 5 e 7. Dois tipos de problemas de otimização global são investigados:

1. Contínuos (Seção 8.1), utilizando a Arquitetura de Amostragem Inteligente (SS, ver Seção 5) proposta e comparando com resultados obtidos por outras técnicas de otimização competitivas da literatura. Observa-se que a técnica de SRA baseada em DOE proposta no Capítulo 4 não é analisada neste Capítulo de experimentos, devido aos resultados superiores obtidos com a técnica SS que possui a mesma finalidade. A metaheurística escolhida para ser utilizada em conjunto com o SS foi a DE. Essa metaheurística apresentou o melhor desempenho dentre as metaheurísticas populacionais testadas nos estudos apresentados nas Seções 4.5 e 4.6. Além disso, a DE não possui nenhum mecanismo para usar informação global diretamente sobre o espaço de busca de modo a guiar a população para áreas promissoras (ver Seção 2.3.3). Por esse motivo, o estudo de algoritmos capazes de encontrar regiões promissoras pode melhorar o desempenho da DE em diversos problemas. Desse modo, o desempenho do SSDE é avaliado da seguinte maneira:

   (a) Comparação com os resultados apresentados para os algoritmos ODE e QODE (ver Seção 8.1.2) em (Rahnamayan et al., 2007), escolhido por se tratarem de novas estratégias que melhoraram significativamente o desempenho da DE;

   (b) Comparação com os resultados apresentados para o algoritmo CGRASP (ver Seção 8.1.3) em (Hirsch et al., 2007), escolhido por ser uma metaheurística

que tem apresentado resultados relevantes, com vários trabalhos publicados recentemente para a resolução de diversos tipos de problemas;

(c) Comparação com os resultados apresentados para os algoritmos ECS e o CHA (ver Seção 8.1.4) em (Oliveira, 2004), escolhidos por serem algoritmos de otimização com detecção de regiões promissoras utilizando *clustering* como algoritmo de aprendizagem de máquina, sendo para essa classe de algoritmos os que têm apresentado os melhores resultados;

2. *Discretos Binários* (Seção 8.2), utilizando o ECGA com duas EETs propostas: 1) Gerenciamento de Tamanho da População (PSM, ver Seção 7.3) e Tendência na População Inicial (BIP, ver Seção 7.4).

   (a) Comparação dos resultados do ECGA+PSM em relação a ECGA em problemas deceptivos;

   (b) Análise da influência de tendência na população inicial (BIP) do ECGA;

Os resultados são avaliados como definidos e justificados na Seção 3.4. O desempenho do SSDE foi avaliado com as métricas $\overline{Aval}_{O*}$, TS e DS, além do teste estatístico não-paramétrico para comparação das médias. Por outro lado, os resultados do ECGA com EETs foram avaliados por meio da $\overline{Aval}_{O*}$, TS e $\overline{Ger}_{O*}$.

8.1 Resultados com o SSDE

Esta Seção está organizada como segue. Na Seção 8.1.1 são apresentadas as configurações do SS e da DE usadas para todos os experimentos. São comparados o DE com o SSDE, e o SSDE com os seguintes algoritmos: 1) ODE e QODE (ver Seção 8.1.2), 2) CGRASP (ver Seção 8.1.3) e 3) ECS e CHA (ver Seção 8.1.4). Para cada conjunto de comparações são apresentadas as funções de *benchmark* utilizadas, os resultados e as conclusões parciais.

8.1.1 Configuração dos Algoritmos

Nesta Seção são apresentadas as configurações do SS e da DE que combinados formam o SSDE. Os parâmetros foram determinados de modo que apresentassem resultados de qualidade aceitável para o maior número de funções de *benchmark*. Contudo, como todo algoritmo de otimização, o ajuste dos parâmetros especificamente para determinada função de *benchmark* pode aumentar consideravelmente o desempenho do algoritmo. Esse tipo de verificação também foi realizada quando necessário, de modo a mostrar que se o SSDE obtiver resultados ruins, um ajuste manual nos parâmetros pode resultar em uma melhora no desempenho. O ideal é que o algoritmo de otimização adapte seus parâmetros

Tabela 8.1: Conjunto de parâmetros do SS.

Parâmetro	Valor
Tamanho da amostra inicial	A melhor metade de $D * 100$ soluções, resultando em $D * 50$ soluções promissoras, D é a quantidade de dimensões do problema
Quantidade de soluções promissoras	A melhor metade da população
Quantidade de novas soluções promissoras geradas a cada iteração	D * 10
Máximo de iterações	100
Máximo de iterações sem melhoria	10
Tamanho mínimo da janela	10% para cada D

Tabela 8.2: Conjunto de parâmetros da DE.

Parâmetro	Valor
Tamanho da população	N = 100
Divisão da população	25% dentro da região promissora e 75% fora
Fator de amplificação diferencial	F = 0,5
Constante da probabilidade de crossover	C = 0,9
Quantidade máxima de avaliações, incluindo as do SS	$MAX_{aval} = 10^6$
Máximo de iterações sem melhoria (estagnação)	300

automaticamente de acordo com o seu comportamento no processo de otimização. Tornar o SS auto-adaptável é uma tarefa a ser investigada futuramente.

O conjunto de parâmetros do SS é sintetizado na Tabela 8.1. O código foi desenvolvido na linguagem R[1]. O conjunto de parâmetros da DE é apresentado na Tabela 8.2, como definido em (Rahnamayan et al., 2008). Foi utilizado o código de DE do pacote DEoptim(Ardia, 2008) disponível no R.

Para os classificadores foi utilizado o Weka[2]. O método KNN empregado foi o Ibk, com modificação do valor do parâmetro K para 1. O método RBL utilizado foi o JRip, com alteração no parâmetro (P) para TRUE, que evita a poda nas regras.

8.1.2 Comparação de SSDE com DE, ODE e QODE

Nesse primeiro experimento, comparam-se os resultados do SSDE em relação a DE original e as variantes de maior desempenho da DE encontradas na literatura, utilizando como base o trabalho de Rahnamayan et al. (Rahnamayan et al., 2007).

Em um recente estudo, Rahnamayan et al. (Rahnamayan et al., 2007) propuseram

[1] www.r-project.org
[2] www.cs.waikato.ac.nz/ml/weka

a *Oppositional Differential Evolution* (ODE) e a *Quasi-Oppositional Differential Evolution* (QODE), baseados no conceito de Aprendizagem Baseada na Oposição (do inglês, *Opposition-Based Learning*, OBL) introduzida por Tizhoosh (Tizhoosh, 2005). Nesse trabalho, os pesquisadores usam pontos quase-opostos na inicialização da população e também na geração de populações novas durante o processo evolutivo. Os pesquisadores concluíram que a QODE supera a DE e a ODE em valor de DS (ver Seção 3.1), mostrando ser uma melhoria importante sobre a DE original.

Funções de *Benchmark*

O conjunto de 15 funções de *benchmark* complexas comumente utilizadas na literatura (ver Tabela 8.3), sendo 7 unimodais e 8 multimodais, é usado na análise experimental do desempenho do SSDE. Na coluna I.D. é apresentado o Intervalo Deslocado da função. Na coluna O^* o valor do ótimo global, que pode variar de acordo com a dimensão do problema (Função Michalewicz).

Tabela 8.3: Definição das funções de *benchmark* utilizadas em (Rahnamayan et al., 2007).

Nome	Definição	I.D.	O^*
First De Jong	$f_1(X) = \sum_{i=1}^{D} x_i^2$	$[-2,56;\ 7,68]^D$	0
Axis parallel hyper-ellipsoid	$f_2(X) = \sum_{i=1}^{D} ix_i^2$	$[-2,56;\ 7,68]^D$	0
Schwefel's Problem 1.2	$f_3(X) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-32,5;\ 97,5]^D$	0
Rastrigin's function	$f_4(X) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i))$	$[-2,56;\ 7,68]^D$	0
Griewangk's function	$f_5(X) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})$	$[-300;\ 900]^D$	0
Sum of different power	$f_6(X) = \sum_{i=1}^{D} \lvert x_i\rvert^{(i+1)}$	$[-0,5;\ 1,5]^D$	0
Ackley's problem	$f_7(X) =$ $-20\exp\left(-0.2\sqrt{\sum_{i=1}^{D} \frac{x_i^2}{D}}\right) - \exp(\sum_{i=1}^{D} \frac{\cos(2\pi x_i)}{D}) + 20 + e$	$[-16;\ 48]^D$	0
Levy function 13	$f_8(X) = sen^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2(1 + sen^2(3\pi x_{i+1}) + (x_D - 1)(1 + sen^2(2\pi x_D))$	$[-10;\ 10]^D$	0
Michalewicz function	$f_9(X) = -\sum_{i=1}^{D} sen(x_i)(sen(ix_i^2/\pi))^{2m}$, sendo que $m = 10$	$[0;\ \pi]^D$	-9,66015; -20,1233
Zakharov function	$f_{10}(X) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0,5ix_i)^2 + (\sum_{i=1}^{D} 0,5ix_i)^4$	$[-5;\ 10]^D$	0
Schwefel's Problem 2.22	$f_{11}(X) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-5;\ 15]^D$	0
Step function	$f_{12}(X) = \sum_{i=1}^{D} (\llcorner x_i + 0,5 \lrcorner)^2$	$[-50;\ 150]^D$	0
Alpine function	$f_{13}(X) = \sum_{i=1}^{D} \lvert x_i sin(x_i) + 0,1x_i\rvert$	$[-5;\ 15]^D$	0
Exponential problem	$f_{14}(X) = exp(0,5\sum_{i=1}^{D} x_i^2)$	$[-0,5;\ 1,5]^D$	1
Salomon problem	$f_{15}(X) = 1 - cos(2\pi \parallel x \parallel) + 0,1 \parallel x \parallel$, sendo que $\parallel x \parallel = \sqrt{\sum_{i=1}^{D} x_i^2}$	$[-50;\ 150]^D$	0

Essas 15 funções são testadas com dois tamanhos diferentes (D e D^*2), com D variando de 10 a 60, dependendo da função de teste. Com isso, tem-se 30 casos de teste. A variação de D para D^*2 é importante para avaliar o desempenho do algoritmo com o aumento da dificuldade do problema. Para cada caso, foram realizadas 100 repetições de cada algoritmo.

Como 13 das 15 funções de *benchmark* possuem o ótimo global no centro do espaço de busca, uma série sucessiva de reduções sem perder o centro do espaço de busca nunca

perderia o ótimo global. Com isso, esses problemas poderia ser resolvidos pelo SS sem muita dificuldade. Por esse motivo, o intervalo do problema é deslocado de modo a torná-lo assimétrico e mais difícil para o SS. O ótimo global é deslocado do centro conforme descrito a seguir. Se os intervalos originais da função forem $-\alpha \leq x_i \leq \alpha$ e $O^* = f(0, ..., 0) = 0$, então os I.D. tornam-se $-\alpha + \frac{\alpha}{2} \leq x_i \leq \alpha + \frac{\alpha}{2}$. É importante observar que a posição e o valor do ótimo global permanecem os mesmos.

Resultados de SSDE x DE

Foi realizada uma comparação entre SSDE e DE com o objetivo de verificar se o algoritmo SS para encontrar regiões promissoras é capaz de melhorar significativamente o desempenho da DE para o conjunto de funções de *benchmark* da Tabela 8.3.

Na Tabela 8.4 são apresentados os resultados da otimização dos 30 casos de teste. A coluna $\overline{Aval}_{O*}$ corresponde à quantidade média de execuções necessárias para se encontrar o ótimo global. Logo, esse índice ignora as execuções que falham, que não encontram o ótimo global. Esse procedimento é adotado para ser usado no cálculo do DS. *Aval* também é usada no cálculo do *p-valor*. Um resultado de *p-valor* $< 0,05$ no teste de Wilcoxon representa que existe uma diferença significativa entre as médias das duas amostras. As medidas $\overline{TS}$ e $\overline{DS}$, mostradas nas últimas duas linhas da tabela, são as respectivas médias da TS e do DS.

É possível verificar que o SSDE alcançou os melhores valores de DS em 27 das 30 funções. Por um lado, os valores de $Aval_{SSDE}$ são substancialmente menores para as funções $f_3, f_4, f_6, f_{10}, f_{12}, f_{13}$, e f_{14} apresentando uma redução de 65% em média para essas funções. Esses resultados indicam que a DE realiza muitas avaliações em regiões não-promissoras, ou fica preso em ótimos locais por muitas gerações. Com o SS, esse comportamento tende a ser evitado, visto que o SS foca em regiões promissoras do espaço de busca, reduzindo a exploração excessiva do espaço de busca e a competição entre soluções de alta qualidade distantes umas das outras, ou em ótimos locais. Dadas duas soluções de alta qualidade distantes entre si, o cruzamento delas tende a gerar indivíduos intermediários, que pertençam a nenhuma das regiões promissoras, resultando em consumo de recursos com avaliações desnecessárias.

Por outro lado, o $Aval_{SSDE}$ é consideravelmente mais alto para funções f_8 e f_9. Nesses casos, as regiões promissoras determinadas pelo SS não continham o ótimo global. Por esse motivo, o SSDE teve que buscar fora das regiões para alcançar o ótimo global, exigindo um montante consideravelmente maior de avaliações.

Com o SSDE foi possível aumentar a $\overline{TS}$ em cerca de 9% e reduzir o $\overline{DS}$ em aproximadamente 63%, o que corresponde a um melhoramento na eficácia e uma redução significativa no custo computacional para resolver as funções de *benchmark* testadas. Como

Tabela 8.4: Comparação entre DE e SSDE. $\overline{Aval}_{O*}$: quantidade média de avaliações da função objetivo até encontrar O^*, TS: taxa de sucesso, DS: desempenho de sucesso (conforme definido na Seção 3.1). $\overline{TS}$ e $\overline{DS}$ na última linha da tabela são, respectivamente, as médias de TS e DS. O melhor DS para cada caso está em negrito. Células com conteúdo igual a "-" correspondem a 0% de TS. Wilcoxon é o teste para comparação das médias de *Aval*. Um *p-valor*$> 0,05$ indica que as médias são iguais.

		DE			**SSDE**			**Wilcoxon**
f	**D**	$\overline{Aval}_{O*}$	**TS**	**DS**	$\overline{Aval}_{O*}$	**TS**	**DS**	**P-Valor**
f_1	30	78210	1	78210	41180	1	**41180**	$< 0,05$
	60	140070	1	140070	83740	1	**83740**	$< 0,05$
f_2	30	86730	1	86730	51980	1	**51980**	$< 0,05$
	60	159300	1	159300	100760	1	**100760**	$< 0,05$
f_3	20	161660	1	161660	86680	1	**86680**	$< 0,05$
	40	746430	1	746430	363080	1	**363080**	$< 0,05$
f_4	10	275430	1	275430	16020	1	**16020**	$< 0,05$
	20	642800	0,1	6428000	37940	1	**37940**	$< 0,05$
f_5	30	103280	1	103280	66080	1	**66080**	$< 0,05$
	60	177250	1	177250	119980	1	**119980**	$< 0,05$
f_6	30	17160	1	17160	7900	1	**7900**	$< 0,05$
	60	28980	1	28980	18020	1	**18020**	$< 0,05$
f_7	30	150670	1	150670	114360	1	**114360**	$< 0,05$
	60	268800	0,7	384000	201000	1	**201000**	$< 0,05$
f_8	30	87160	1	**87160**	94460	1	94460	$< 0,05$
	60	155844,4	0,9	**173160,4**	614500	1	614500	$< 0,05$
f_9	10	183775	0,4	**459437,5**	240400	0,4	601000	$< 0,05$
	20	-	0	-	-	0	-	$> 0,05$
f_{10}	30	359640	1	359640	135190	1	**135190**	$< 0,05$
	60	-	0	-	542070	1	**542070**	$< 0,05$
f_{11}	30	164890	1	164890	115700	1	**115700**	$< 0,05$
	60	287910	1	287910	204900	1	**204900**	$< 0,05$
f_{12}	30	39760	1	39760	10000	1	**10000**	$< 0,05$
	60	75070	1	75070	23740	1	**23740**	$< 0,05$
f_{13}	30	383070	1	383070	100800	1	**100800**	$< 0,05$
	60	403610	1	403610	185420	1	**185420**	$< 0,05$
f_{14}	10	16820	1	16820	5560	1	**5560**	$< 0,05$
	20	39940	1	39940	14580	1	**14580**	$< 0,05$
f_{15}	10	-	0	-	22500	0,2	**112500**	$< 0,05$
	20	-	0	-	52800	0,2	**264000**	$< 0,05$
$\overline{TS}$		0,803			**0,893**			
$\overline{DS}$		380921,27			**144438,0**			

as regiões promissoras encontradas pelo SS contém, em geral, soluções consideravelmente perto do ótimo global, a DE necessita de poucas gerações para alcançá-lo. Esses resultados evidenciam que a técnica de SS proposta para localizar regiões promissoras pode melhorar o desempenho da DE clássica e, possivelmente, outros algoritmos de otimização populacionais.

Para facilitar a comparação dos resultados com outros trabalhos, é apresentada a Tabela 8.5 com mais informações estatísticas.

Tabela 8.5: Estatísticas da *Aval* do SSDE para as funções da Tabela 8.3.

f	D	Mín.	1^o Quartil	Mediana	Média	3^o Quartil	Máx.	Desv.Pad.
f_1	30	38500	39500	41200	41180	41800	44900	2461,1
	60	77300	81800	82100	83740	84000	93500	5984,4
f_2	30	46500	47600	52100	51980	56200	57500	4937,31
	60	90600	100700	103100	100760	103200	106200	6005,25
f_3	20	78200	85400	85700	86680	91900	92200	5749,52
	40	303400	344600	345200	363080	408800	413400	47024,8
f_4	10	14300	15700	15700	16020	16200	18200	1409,61
	20	36400	37300	37700	37940	38400	39900	1312,63
f_5	30	58100	61100	65500	66080	66900	78800	7926,03
	60	112500	117900	118800	119980	125100	125600	5465,07
f_6	30	7300	7900	7900	7900	8200	8200	367,42
	60	15100	16900	17500	18020	19100	21500	2414,95
f_7	30	107200	110100	115000	114360	119200	120300	5668,6
	60	192300	196000	201600	201000	207000	208100	6845,8
f_8	30	91300	93000	93400	94460	94900	99700	3197,34
	60	253600	512700	771800	614500	794950	818100	313404,74
f_9	10	160200	200300	240400	240400	280500	320600	113419,93
	20	0	0	0	0	0	0	0
f_{10}	30	103400	119550	131550	135190	153775	165100	21888,88
	60	350100	418400	515400	542070	680750	797700	157870,43
f_{11}	30	111900	114300	114400	115700	115400	122500	4013,1
	60	177100	207600	213100	204900	213300	213400	15733,56
f_{12}	30	9700	9700	10000	10000	10000	10600	367,42
	60	22300	22900	23500	23740	24700	25300	1244,19
f_{13}	30	93800	94100	99000	100800	106300	110800	7539,56
	60	171500	171900	175200	185420	203700	204800	17253,61
f_{14}	10	2900	3700	6400	5560	7400	7400	2122,03
	20	12300	12400	13400	14580	16800	18000	2644,24
f_{15}	10	12830	19800	21610	22500	24960	30780	3402,78
	20	34290	44830	52220	52800	58530	68800	8301,23

Resultados com SSDE x ODE e QODE

Os resultados do SSDE são comparados aos obtidos em (Rahnamayan et al., 2007). As conclusões são similares às do item anterior. Algumas notas importantes:

1. Os pesquisadores informam $TS = 1$ para f_{15}, até mesmo para a DE original. Em nossos experimentos, o resultado foi $TS = 0$, sendo que o melhor valor obtido foi de $9,99e-2$, mesmo com a DE original. Outros pesquisadores não puderam achar o ótimo global, alcançando resultados com a DE semelhantes aos apresentados neste texto(Noman & Iba, 2008).

2. Como os autores do QODE não apresentaram informações suficientes, não foi possível realizar testes estatísticos para comparar os resultados das médias. Nesse caso, foi realizada apenas uma comparação direta entre os valores de DS.

Como pode ser visto nos resultados apresentados em Tabela 8.6, SSDE alcançou os melhores resultados em 20 das 30 funções, mas obteve desempenho semelhante em termos de $\overline{TS}$ (0,88 para o ODE, 0,86 para o QODE e 0,89 para o SSDE). Os valores de DS para o SSDE são substancialmente melhores nas funções $f_4, f_{10}, f_{12}, f_{13}$ e f_{14}, significando que, mesmo com o conceito de *Opposition-Based* a DE ainda pode desperdiçar muitas avaliações em regiões não-promissoras, em ótimos locais ou na competição entre soluções em ótimos locais distintos, que podem gerar soluções intermediárias de baixa qualidade. Nos casos em que o $Aval_{SSDE}$ foi consideravelmente maior, foi identificado que o ótimo global não estava contido nas regiões encontradas. É importante observar que na função f_{10}, o SSDE foi o único algoritmo capaz de encontrar o ótimo global. Apesar de uma maior TS em f_8 com D=60, o valor de DS é inferior. Isso significa que a TS maior é atingida com o custo de investigar mais o espaço de busca.

O algoritmo SSDE obteve melhor DS em $\frac{2}{3}$ dos testes. Apesar do SSDE ter obtido um valor de $\overline{TS}$ similar aos dos outros dois algoritmos, o $\overline{DS}$ do SSDE foi consideravelmente melhor, com uma redução média de 47% na quantidade de avaliações necessárias para encontrar o ótimo global em relação ao ODE e ao QODE.

Como a arquitetura SS é independente da metaheurística ela pode ser aplicada ao ODE e ao QODE. Essas combinações, SSODE e SSQODE podem ser exploradas em trabalhos futuros.

Tabela 8.6: Comparação entre ODE, QODE e SSDE.

		ODE			**QDE**			**SSDE**		
f	**D**	$\overline{Aval}_{O*}$	**TS**	**DS**	$\overline{Aval}_{O*}$	**TS**	**DS**	$\overline{Aval}_{O*}$	**TS**	**DS**
f_1	30	50844	1	50844	42896	1	42896	41180	1	**41180**
	60	101832	1	101832	94016	1	94016	83740	1	**83740**
f_2	30	56944	1	56944	47072	1	**47072**	51980	1	51980
	60	117756	1	117756	105992	1	105992	100760	1	**100760**
f_3	20	177300	1	177300	116192	1	116192	86680	1	**86680**
	40	834668	1	834668	539608	1	539608	363080	1	**363080**
f_4	10	75278	0,92	81823	181100	1	181100	16020	1	**16020**
	20	421300	0,16	2633125	615280	0,16	3845500	37940	1	**37940**
f_5	30	74717	0,92	81214	100540	0,80	125675	66080	1	**66080**
	60	128340	0,68	188735	115280	0,68	169529	119980	1	**119980**
f_6	30	10152	1	10152	9452	1	9452	7900	1	**7900**
	60	11452	1	**11452**	14667	0,84	17461	18020	1	18020
f_7	30	100280	1	100280	82448	1	**82448**	114360	1	114360
	60	202010	0,96	210427	221850	0,72	308125	201000	1	**201000**
f_8	30	70408	1	70408	50576	1	**50576**	94460	1	94460
	60	121750	0,60	202900	98300	0,40	**245800**	614500	1	614500
f_9	10	213330	0,56	380900	247640	0,48	515900	240400	0,4	601000
	20	253910	0,55	461700	193330	0,68	**284300**	-	0	-
f_{10}	30	369104	1	369104	239832	1	239832	135190	1	**135190**
	60	-	0	-	-	0	-	542070	1	**542070**
f_{11}	30	167580	1	167580	108852	1	**108852**	115700	1	115700
	60	274716	1	274716	183132	1	**183132**	204900	1	204900
f_{12}	30	26400	1	26400	21076	1	21076	10000	1	**10000**
	60	64780	1	64780	64205	1	64205	23740	1	**23740**
f_{13}	30	361884	1	361884	291448	1	291448	100800	1	**100800**
	60	425700	0,96	443438	295084	1	295084	185420	1	**185420**
f_{14}	10	16112	1	16112	13972	1	13972	5560	1	**5560**
	20	31720	1	31720	23776	1	23776	14580	1	**14580**
f_{15}	10	26108	1	26108	18944	1	**18944**	22500	0,2	112500
	20	57888	1	57888	40312	1	**40312**	52800	0,2	264000
$\overline{TS}$		0,88			0,86			**0,89**		
$\overline{DS}$		262489,31			269409,17			**144438,0**		

8.1.3 Comparação de SSDE com CGRASP

Nesse segundo experimento, compara-se os resultados do SSDE contra os da DE utilizando como base o trabalho de CGRASP et al. (Hirsch et al., 2007). O CGRASP é uma metaheurística de busca estocástica com vários reinícios, quando necessário, que utiliza um procedimento guloso para gerar soluções que são repassadas a um algoritmo de melhoria local. O CGRASP consiste de uma série de ciclos do tipo construção-melhoria local, sendo que a saída da construção é usada como entrada do procedimento de melhoria local e a

saída da melhoria local serve como entrada no procedimento de construção.

Funções de *Benchmark*

O conjunto de funções referentes ao artigo (Hirsch et al., 2007), que pode ser visto na Tabela 8.7, corresponde a 8 problemas de teste de otimização globais contínuos encontrados na literatura, sendo que alguns são executados com mais de um valor para D. Como o mínimo global é conhecido para cada uma dessas funções, o erro máximo permitido utilizado em (Hirsch et al., 2007) foi definido como

$$|O* - O| \leq \varepsilon_1 |O*| + \varepsilon_2, \text{sendo que } \varepsilon_1 = 10^{-4} \text{ e } \varepsilon_2 = 10^{-6}.$$

Para cada problema, foram realizadas 100 repetições dos algoritmos. Não foram utilizados intervalos deslocados (I.D.), mas intervalos originais (I.O.). Além disso, não foi definido um limite de avaliações em (Hirsch et al., 2007) como critério de parada. Desse modo, foram adotados os critérios de parada apresentados na Seção 8.1.1: 1) o limite de avaliações ou 2) caso o algoritmo encontre o ótimo global.

Tabela 8.7: Definição das funções de *benchmark* em (Hirsch et al., 2007).

Nome	Definição	I.O.	$O*$
Branin	$f_{16}(X) = (x_2 - \frac{5,1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_1) + 10$	$[-5 \leq x_1 \leq 10]$ $[0 \leq x_2 \leq 15]$	0,397887
Easom	$f_{17}(X) = -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$[-100;\ 100]^D$	-1
Goldstein-Price	$f_{18}(X) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) *$ $(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2;\ 2]^D$	3
Shubert	$f_{19}(X) = (\sum_{i=1}^{D}(i\,cos((i+1)x_1 + i)))(\sum_{i=1}^{D}(i\,cos((i+1)x_2 + i)))$	$[-10;\ 10]^D$	-186,7309
Hartmann-3	$f_{20}(X) = -\sum_{i=1}^{4}\beta_i exp\left[-\sum_{i=1}^{3}A_{i,j}(x_j - P_{i,j})^2\right]$ $\beta = [1,0;\ 1,2;\ 3,0;\ 3,2]$ $A = \begin{bmatrix} 3,0 & 10 & 30 \\ 0,1 & 10 & 35 \\ 3,0 & 10 & 30 \\ 0,1 & 10 & 35 \end{bmatrix}$ $P = \begin{bmatrix} 0,36890 & 0,11700 & 0,26730 \\ 0,46990 & 0,43870 & 0,74700 \\ 0,10910 & 0,87320 & 0,55470 \\ 0,03815 & 0,57430 & 0,88280 \end{bmatrix}$	$[0;\ 1]^3$	- 3,86278
Hartmann-6	$f_{20}(X) = -\sum_{i=1}^{4}\beta_i exp\left[-\sum_{i=1}^{6}B_{i,j}(x_j - Q_{i,j})^2\right]$ $\beta = [1,0;\ 1,2;\ 3,0;\ 3,2]$ $B = \begin{bmatrix} 10,00 & 3,00 & 17,00 & 3,50 & 1,70 & 8,00 \\ 0,05 & 10,00 & 17,00 & 0,10 & 8,00 & 14,00 \\ 3,00 & 3,50 & 1,70 & 10,00 & 17,00 & 8,00 \\ 17,00 & 8,00 & 0,05 & 10,00 & 0,10 & 14,00 \end{bmatrix}$ $Q = \begin{bmatrix} 0,1312 & 0,1696 & 0,5569 & 0,0124 & 0,8283 & 0,5886 \\ 0,2329 & 0,4135 & 0,8307 & 0,3736 & 0,1004 & 0,9991 \\ 0,2348 & 0,1451 & 0,3522 & 0,2883 & 0,3047 & 0,6650 \\ 0,4047 & 0,8828 & 0,8732 & 0,5743 & 0,1091 & 0,0381 \end{bmatrix}$	$[0;\ 1]^6$	- 3,32237
Rosenbrock	$f_{21}(X) = \sum_{i=1}^{D}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	$[-5,12;\ 5,12]^D$	0
Shekel	$f_{22}(X) = -\sum_{j=1}^{m}\left[\sum_{i=1}^{4}(x_i - C_{i,j})^2 + \beta_j\right]^{-1}$ $\beta = [0,1;\ 0,2;\ 0,2;\ 0,4;\ 0,4;\ 0,6;\ 0,3;\ 0,7;\ 0,5;\ 0,5]$ $C = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7,0 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3,6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7,0 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3,6 \end{bmatrix}$	$[0;\ 10]^D$	m5: -10,1532 m7: -10,4029 m10: -10,5364

Tabela 8.8: Comparação entre DE e SSDE para as funções de *benchmark* da Tabela 8.7.

		DE			SSDE			SSDE_{75}			Wilcoxon DE x SSDE_{75}
f	D	$\overline{Aval}_{O*}$	TS	DS	$\overline{Aval}_{O*}$	TS	DS	$\overline{Aval}_{O*}$	TS	DS	P-Valor
f_{16}	2	3540	1	3540	3428	1	3428	2448	1	**2448**	$< 0,05$
f_{17}	2	4950	1	4950	5920	1	5920	2974	1	**2974**	$< 0,05$
f_{18}	2	2520	1	2520	2632	1	2632	1826	1	**1826**	$< 0,05$
f_{19}	2	10890	1	10890	10314	1	10314	6704	1	**6704**	$< 0,05$
f_{20}	3	1860	1	**1860**	2632	1	2632	2398	1	2398	$< 0,05$
	6	-	0	-	55380	0,2	276900	22326,67	0,6	**37211,12**	$< 0,05$
f_{21}	2	4730	1	4730	6114	1	6114	4532	1	**4532**	$< 0,05$
	5	22560	1	**22560**	36075	1	36075	70190	1	70190	$< 0,05$
	10	68440	1	**68440**	82200	1	82200	73690	1	73690	$< 0,05$
$f_{22}, m = 5$	4	7210	1	**7210**	11348	1	11348	28808	0,5	57616	$< 0,05$
$m = 7$	4	6560	1	**6560**	16088	1	16088	20882,5	0,8	26103,12	$< 0,05$
$m = 10$	4	6650	1	**6650**	12176	1	12176	20988	0,5	41976	$< 0,05$
f_{10}	5	10260	1	10260	10745	1	10745	5520	1	**5520**	$< 0,05$
	10	33250	1	33250	29250	1	29250	16480	1	**16480**	$< 0,05$
$\overline{TS}$		0,93			**0,94**			0,88			
$\overline{DS}$		**14109,23**			36130,14			24976,3			

Resultados de SSDE x DE

Foi realizada uma comparação entre SSDE e DE com o objetivo é verificar se o algoritmo SS para encontrar regiões promissoras é capaz de melhorar o desempenho da DE no conjunto de funções de *benchmark* da Tabela 8.7. Os resultados são apresentados na Tabela 8.8. A Tabela 8.9 contém mais informações estatísticas para facilitar a comparação dos resultados com outros trabalhos.

Nesse experimento, o SSDE teve melhor valor de DS do que a DE apenas nas funções f_{19}, f_{20} com D=6 e f_{10} com D=10, valores similares nas funções f_{16}, f_{18} e f_{10} ($D = 5$). Nas demais, a DE mostrou-se consideravelmente superior. A vantagem do SSDE aparece na função f_{20} com D=6, que teve $TS > 0$, enquanto a DE não conseguiu resolver esse problema.

Uma explicação para a superioridade da DE é o fato de os problemas nesse experimento serem de baixa dimensão, enquanto no experimento anterior (ver Seção 8.1.2) eram consideravelmente maiores. Portanto, pode-se supor que o SSDE tem uma degradação mais suave à medida que a dimensão do problema aumenta e que a DE é eficiente em problemas com poucas variáveis.

Em outras palavras, para poucas dimensões o espaço de busca é pequeno de modo que o SS reduz pouco o domínio para a DE. Isso pode ser constatado pelos valores de TS, que também foram similares. Além disso, a análise das regiões encontradas pelo SS mostrou que essas eram de alta qualidade, próximas ao ótimo global e que esse estava contido nelas, não necessitando exploração externa pela DE.

Tabela 8.9: Estatísticas da *Aval* do SSDE para as funções da Tabela 8.7.

f	D	Mín.	1^o Quartil	Mediana	Média	3^o Quartil	Máx.	Desv.Pad.
f_{16}	2	1500	3225	3690	3428	4050	4260	867,29
f_{17}	2	5660	5725	5910	5920	6065	6240	214,17
f_{18}	2	1800	2480	2660	2632	2925	3040	379,03
f_{19}	2	7020	10275	10900	10314	11290	11620	1589,58
f_{20}	3	2090	2500	2650	2632	2877,5	2990	304,22
	6	54760	55070	55380	55380	55690	56000	876,81
f_{21}	2	5320	5890	6100	6114	6400	6600	411,61
	5	29050	31712,5	35175	36075	39125	45250	5557,04
	10	68200	80175	82800	82200	86325	96900	8010,13
$f_{22}, m = 5$	4	9820	11345	11480	11348	11675	12160	727,44
$f_{22}, m = 7$	4	10500	11650	12370	16088	12780	52720	12893,74
$f_{22}, m = 10$	4	11700	11780	11930	12176	12615	12960	490,06
f_{10}	5	9500	10225	10750	10745	11250	11850	731,23
	10	23800	28250	29450	29250	31275	32500	2837,94

Buscando tornar o SSDE mais agressivo, a população inicial utilizada na DE (após o uso do SS) foi reorganizada de forma a gerar 75% das soluções dentro da região promissora e 25% fora, ao invés da divisão inversa (25% - 75%) apresentada na Tabela 8.2. Os resultados dessa nova configuração são apresentados na coluna $SSDE_{75}$.

A melhora considerável atingida pela mudança na divisão da população da DE pode ser explicada pelo fato de que a recombinação de soluções próximas ao ótimo global (dentro da região promissora) com soluções externas distantes do ótimo global tendem a gerar soluções intermediárias. Do mesmo modo, a recombinação de soluções distantes tende a gerar soluções distantes. O fato do SS ter sido eficaz em encontrar uma região reduzida que contém o ótimo global, em 8 das 14 funções de *benchmark* desse experimento, reforça a suposição de que para problemas de baixa dimensão a região promissora obtida pela SS pode não ser uma contribuição suficiente para aumentar o desempenho da DE. Para esses casos, o SSDE deve ser reajustado. Por exemplo, com a função f_{22}, o ótimo global localiza-se em $x* = (4, ..., 4)$, mas o SS encontra um ótimo local próximo a $x = (1, ..., 1)$. Nesse caso, foi verificado experimentalmente que se o grau de exploração do SS for reduzido na função de reamostragem (ver Algoritmo 5.1), a região do ótimo global é encontrada em 100% dos casos. Portanto, uma modificação no operador de reamostragem é capaz de melhorar o desempenho do SS.

Em resumo, usar uma amostra maior dentro da região promissora pode melhorar o desempenho do SSDE para problemas de menor dimensão. Apesar de o valor do DS do $SSDE_{75}$ ter sido superior em 8 das 14 funções testadas, $\overline{TS}$ e $\overline{DS}$ foram inferiores devido ao desempenho na função f_{22}, na qual o SS ficou preso em um ótimo local. Assim como ocorre em qualquer metaheurística, a qualidade do desempenho depende dos valores

escolhidos para os parâmetros. Isso significa que é possível encontrar uma configuração que possibilite ao SS ter um melhor desempenho nessa função. Contudo, encontrar essa configuração não foi o foco deste trabalho.

Resultados com SSDE x CGRASP

Os resultados do SSDE (com a configuração padrão, ver Seção 8.1.1) são agora comparados aos obtidos pelo CGRASP (ver Tabela 8.10). Como o CGRASP possui um procedimento de melhoria local, foi adicionado um algoritmo de busca local ao SSDE para que a comparação seja mais adequada. Assim, logo após a execução do SS, ao invés de iniciar a DE dentro da região promissora, é iniciada uma busca local no melhor indivíduo de cada região encontrada. Se o ótimo global não for encontrado, inicia-se o DE. O valor de $\overline{Aval}_{O*}$ contabiliza todas essas avalições. O algoritmo de busca utilizado é o L-BFGS-B (Zhu et al., 1997). Esse algoritmo é um método de otimização *quasi*-Newton que usa valores da função objetivo e gradientes para construir uma imagem da superfície a ser otimizada, utilizando alguma tendência para seguir em direção ao ótimo local. O L-BFGS-B foi limitado em 50 iterações.

Tabela 8.10: Comparação entre SSDE e CGRASP.

		CGRASP			**SSDE**		
f	**D**	$\overline{Aval}_{O*}$	**TS**	**DS**	$\overline{Aval}_{O*}$	**TS**	**DS**
f_{16}	2	59857	1	59857	540,75	1	**540,75**
f_{17}	2	89630	1	89630	753,33	1	**753,33**
f_{18}	2	29	1	**29**	444,24	1	444,24
f_{19}	2	82363	1	82363	543,03	1	**543,03**
f_{20}	3	20743	1	20743	943,42	1	**943,42**
	6	79685	1	79685	5107,12	1	**5107,12**
f_{21}	2	1158350	1	1158350	566,97	1	**566,97**
	5	6205503	1	6205503	1568,94	1	**1568,94**
	10	20282529	0,99	20487403,03	4256,15	1	**4256,15**
$f_{22}, m = 5$	4	5545982	1	5545982	7598,11	0,79	**9617,86**
$m = 7$	4	4052800	1	4052800	3791,65	0,7	**5416,64**
$m = 10$	4	4701358	1	4701358	4936	0,76	**6494,74**
f_{10}	5	959	1	**959**	1109,27	1	1109,27
	10	3607653	1	3607653	2952,18	1	**2952,18**
$\overline{TS}$		**0,99**			0,95		
$\overline{DS}$		3292308,22			**2879,62**		

Na comparação direta entre SSDE e CGRASP existe uma larga vantagem em favor do SSDE, com relação ao DS, em 12 das 14 funções testadas. Tal vantagem está presente mesmo nos testes com a função f_{22}, apesar de a TS do SSDE ter sido menor do que 0,8. Nessa função, o CGRASP consegue 100% de sucesso, mas ao custo de mais de 4.000.000

de avaliações. O SSDE, por outro lado, necessita de menos de 10.000. Portanto, se o SSDE falhar pelo critério de limite de avaliações (10^6, ver Tabela 8.2), é mais interessante executá-lo novamente e ter em torno de 1.010.000 avaliações (sendo 10^6 da execução falha mais 10.000 da execução seguinte com sucesso) do que executar o CGRASP uma única vez. Todavia, é importante lembrar que existe o limite de 300 iterações da DE caso não haja melhoria no valor do melhor ponto já encontrado. Nesse caso, 300 iterações de uma população de tamanho 100 é igual a 30.000 avaliações. Desse modo, é pouco provável que o limite de avaliações seja atingido e, de fato, isso não ocorreu nos experimentos.

O CGRASP apresentou melhor valor de DS nas funções f_{18} e f_{10}, sendo para a primeira a diferença foi consideravelmente grande. Por outro lado, para a segunda, a diferença foi menor. Nesse caso, uma redução no tamanho da população do SSDE de 100 para 50 indivíduos reduziu a DS para 780, mantendo $TS = 1$.

Como o SS é independente da técnica de otimização empregada para otimizar as regiões promissoras empregadas, o SS poderia ser facilmente aplicado ao CGRASP em um trabalho futuro.

8.1.4 Comparação de SSDE com ECS e CHA

O ECS (do inglês, *Evolutionary Clustering Search*) (Oliveira, 2004) procura detectar regiões promissoras por meio do enquadramento dessas por *clusters*. É um algoritmo de otimização global avançado, composto por: 1) um algoritmo evolutivo; 2) um agrupador iterativo; 3) um analisador de agrupamentos e 4) um algoritmo de otimização local.

No CHA (do inglês, *Continuous Hybrid Algorithm*) (Chelouah & Siarry, 2003), o processo evolutivo ocorre normalmente, sem intervenção de otimizadores locais, até ser detectada uma região promissora. A identificação da região promissora é baseada na perda de diversidade populacional, que é obtida verificando se a maior distância entre o melhor e os demais indivíduos da população é menor que um certo raio. Em seguida, o domínio de busca é reduzido em torno do melhor indivíduo, e uma busca local é iniciada.

Uma característica importante de ambos algoritmos (ECS e CHA) é o fato de utilizarem algoritmos de busca local (*Hooke and Jeeves direct search(Hooke & Jeeves, 1961)* no ECS e *Nelder and Mead Simplex(Nelder & Mead, 1965)* no CHA) ao encontrarem regiões promissoras. Com isso, existe uma grande aceleração no processo de descoberta do ótimo local, que pode ser o global. Por outro lado, o SS é uma técnica para encontrar regiões promissoras, não um algoritmo de otimização global. Combinado com o DE, o SSDE torna-se um algoritmo de busca. Porém, a DE não utiliza busca local. Para uma comparação mais adequada entre SSDE, ECS e CHA, foi utilizado o mesmo procedimento da comparação com o CGRASP, ou seja, a execução do algoritmo L-BFGS-B no melhor indivíduo de cada região encontrada pelo SS. Caso o ótimo global não seja encontrado pelo L-BFGS-B, inicia-se a DE em cada região promissora obtida pelo SS.

Funções de *Benchmark*

O conjunto de 11 funções utilizada por Chelouah e Siarry (Chelouah & Siarry, 2003) e posteriormente por Oliveira (Oliveira, 2004) foi testado também com a DE e o SSDE. As funções (ver Tabela 8.11) são f_{17}, f_{18}, f_{20}, f_{21}, f_{22}, f_{10}, f_5, f_9, f_4, f_{11}, definidas previamente nas Tabelas 8.3 e 8.7, com D variando de 2 a 100 dependendo da função de teste.

Tabela 8.11: Definição das funções de *benchmark* em (Oliveira, 2004).

Função	I.O.	$O*$
f_{17}	$[-100; 100]^D$	-1
f_{18}	$[-2; 2]^D$	3
f_{20}	$[0; 1]^6$	-3.,322
f_{21}	$[-5, 12; 5, 12]^D$	0
f_{22}	$[-10; 10]^4$	-10,536
f_{10}	$[-5; 10]^D$	0
f_5	$[-600; 600]^D$	0
f_9	$[0; \pi]^D$	D=5: -4,687
		D=10: -9,660
f_4	$[-5, 12; 5, 12]^D$	0
f_{11}	$[-500; 500]^D$	0

Como o mínimo global é conhecido para cada uma dessas funções, o erro máximo permitido utilizado em (Chelouah & Siarry, 2003) foi:

$$|O * -O| \leq \varepsilon_1, \text{tal que } \varepsilon_1 = 10^{-4}.$$

Para cada função de teste, foram executadas 20 repetições dos algoritmos. Em (Oliveira, 2004), foi definido um limite de 100.000 avaliações da função objetivo para cada repetição, independente de D. Portanto, os critérios de parada adotados foram o limite de avaliações ou caso o algoritmo encontre o ótimo global.

No ECS foram utilizadas populações de tamanho 10, 30 ou 100, dependendo do problema. Contudo, o tamanho não foi informado para cada problema. Nesse caso, a DE e o SSDE utilizam população de tamanho determinado por $n = min(100, D * 10)$, de acordo com o problema.

Resultados com DE x SSDE

Os resultados desse experimento são apresentados na Tabela 8.12. Os resultados detalhados, para comparação com outros trabalhos, são apresentados na Tabela 8.13.

Dos 18 casos de teste (ver Tabela 8.11), a DE foi capaz de encontrar o ótimo global em 8 e o SSDE em 11. Pode-se verificar que ambos algoritmos falharam em todos os

problemas com $D > 10$. Como esses algoritmos não empregam busca local, o limite de 100.000 avaliações não é suficiente para que o ótimo global seja encontrado. Suganthan (Suganthan et al., 2005) sugere um limite de $Max_{Aval} = D * 10000$ para metaheurísticas populacionais sem busca local. Nesse sentido, nas funções f_4 e f_9, o limite de 100.000 deveria ser suficiente ($D \leq 10$), mesmo assim a DE obteve $TS = 0$. O SSDE, por sua vez, obteve $TS = 1$ em f_4, indicando que o uso do SS permitiu uma utilização eficiente dos recursos computacionais (avaliações da função objetivo) na exploração apenas de regiões promissoras. Esse mesmo efeito ocorreu também para as funções f_{10} ($D = 50$), f_5 ($D = 50$) e f_4 ($D = 20$).

Tabela 8.12: Comparação entre DE e SSDE para as funções de (Oliveira, 2004).

		DE			**SSDE**			**Wilcoxon DE x SSDE**
f	**D**	$\overline{Aval_{O*}}$	**TS**	**DS**	$\overline{Aval_{O*}}$	**TS**	**DS**	**P-Valor**
f_{17}	2	1164	1	1164	1176,75	1	1176,75	$> 0,05$
f_{18}	2	619	1	**619**	808,35	1	808,35	$< 0,05$
f_{20}	6	5040	0,1	50400	33818,78	0,9	**37576,42**	$< 0,05$
f_{21}	5	10806,25	0,4	**27015,62**	-	-	-	$< 0,05$
	10	60715	1	60715	59605,7	1	59605,7	$> 0,05$
	50	-	-	-	-	-	-	-
	100	-	-	-	-	-	-	-
f_{22}	4	3068	1	**3068**	11921,10	1	11921,10	$< 0,05$
f_{10}	10	25320	1	25320	6812,84	1	**6812,84**	$< 0,05$
	50	-	-	-	66140,75	0,33	**198422,25**	$< 0,05$
f_5	50	-	-	-	51423	0,9	**57136,67**	$< 0,05$
	100	-	-	-	-	-	-	-
f_9	5	6105,89	0,85	7183,39	7292	1	**7292**	$< 0,05$
	10	-	-	-	-	-	-	-
f_4	10	-	-	-	11739	1	**11739**	$< 0,05$
	20	-	-	-	29418,6	1	**29418,6**	$< 0,05$
f_{11}	20	-	-	-	-	-	-	-
	30	-	-	-	-	-	-	-
$\overline{TS}$		0,35			**0,56**			
$\overline{DS}$		**9749,17**			23439,43			

Na função f_{20}, apesar de o número de avaliações da DE ser consideravelmente inferior ao do SSDE, a TS foi, também, inferior. Por esse motivo, seu valor de DS é superior, o que faz com que o DS do SSDE seja melhor. Nas funções f_{17} e f_{21} ($D = 10$), o teste de comparação de médias apresentou igualdade entre os resultados da DE e SSDE. A função f_{18} possui apenas duas dimensões. Nesse caso, o maior custo em termos de avaliações da função objetivo foi causado pelo SS, pois apenas na primeira amostragem ele realiza $D * 100$ avaliações e, para cada iteração posterior, $D * 10$. Desse modo, é de se esperar

Tabela 8.13: Estatísticas da *Aval* do SSDE para as funções da Tabela 8.11.

f	D	Mín.	1^o Quartil	Mediana	Média	3^o Quartil	Máx.	Desv.Pad.
f_{17}	2	965	1104,5	1171,5	1176,75	1257,25	1403	119,22
f_{18}	2	651	797	799	808,35	844,75	902	60,65
f_{20}	6	3762	26692,25	27150,5	33818,78	49386,25	96819	24213,49
f_{21}	5	0	0	0	0	0	0	0
	10	55312	56577	60010,5	59605,7	62257,75	64200	3285,19
	50	0	0	0	0	0	0	0
	100	0	0	0	0	0	0	0
f_{22}	4	2932	3824,5	4636	11921,11	19298	33988	9333,32
f_{10}	10	3501	4706,5	6732	6812,84	8091	10465	2148,34
	50	42417	52734	66778,5	66140,75	80185,25	88589	20756,58
f_5	50	45244	49329,5	51397,5	51423	53999,5	56551	3345,61
	100	0	0	0	0	0	0	0
f_9	5	6453	6796,75	7247,5	7292	7732	8497	658,83
	10	0	0	0	0	0	0	0
f_4	10	10369	11048,5	11878	11739	12108,75	13599	994,45
	20	24287	25525,75	26104	29418,6	29915,75	45301	6715,16
f_{11}	20	0	0	0	0	0	0	0
	30	0	0	0	0	0	0	0

que o SSDE apresente maior número de avaliações em problemas de baixas dimensão e complexidade.

Em resumo, a DE teve os seguintes valores de média obtidas pela DE foram $\overline{TS} = 0,35$ e $\overline{DS} = 9749,17$; enquanto que o SSDE obteve $\overline{TS} = 0,56$ e $\overline{DS} = 23439,43$. O valor maior de $\overline{DS}$ do algoritmo SSDE ocorreu pelo fato de que, em algumas das execuções, as regiões promissoras determinadas pelo SS não continham o ótimo global, fazendo com que a DE buscasse fora da região. Por esse motivo, é importante continuar investigando maneiras de melhorar a determinação de regiões promissoras pelo SS, possivelmente por meio de um melhor operador de reamostragem.

Resultados com SSDE x ECS e CHA

Na Tabela 8.14 são apresentados os resultados do ECS e do CHA para comparação com o SSDE, o qual apresenta resultados competitivos com os outros dois algoritmos, obtendo DS superior em 6 dos 18 casos de teste. É importante lembrar que, para essa comparação, o SSDE foi dotado de um algoritmo de busca local.

Para ser mais competitivo, foi utilizado no SSDE uma combinação dos operadores de reamostragem apresentados na Seção 5.3. O operador de ruído (ver Algoritmo 5.1) possibilita uma maior exploração global, enquanto o de deslocamento (ver Algoritmo 5.2) efetua uma melhor exploração local. A aplicação desses operadores é alternada a cada

Tabela 8.14: Comparação entre SSDE, ECS e CHA.

		SSDE			ECS			CHA		
f	D	$\overline{Aval}_{O*}$	TS	DS	$\overline{Aval}_{O*}$	TS	DS	$\overline{Aval}_{O*}$	TS	DS
f_{17}	2	587,5	1	**587,5**	593,5	1	593,5	952	1	952
f_{18}	2	407,75	1	407,75	345,4	1	**345,4**	259	1	259
f_{20}	6	2235,75	1	2235,75	633,9	1	**633,9**	930	1	930
f_{21}	5	1464,55	1	**1464,55**	2561,7	1	2561,7	3290	1	3290
	10	3808,4	1	**3808,4**	8979,5	1	8979,5	14563	0,83	17545,78
	50	37077,6	1	37077,6	32780,6	1	**32780,6**	55356	0,79	70070,89
	100	-	-	-	85821	0,8	**107276,25**	124302	0,72	172641,66
f_{22}	4	1683,579	0,95	1772,188	506,8	0,75	**675,73**	635	0,85	747,06
f_{10}	10	2801,75	1	2801,75	2328,6	1	**2328,6**	4291	1	4291
	50	19496,6	1	**19496,6**	-	-	-	75520	1	75520
f_5	50	5205	1	5205	5024,55	1	**5024,55**	-	-	-
	100	-	-	-	24344,45	1	**24344,45**	-	-	-
f_9	5	8855,95	1	**8855,95**	12869,55	1	12869,55	-	-	-
	10	-	-	-	37671,923	0,65	**57956,80**	-	-	-
f_4	10	9211,857	0,35	**26319,592**	26379,95	1	26379,95	-	-	-
	20	-	-	-	71952,667	0,9	**79947,41**	-	-	-
f_{11}	20	4074,3	1	**4074,3**	39987,95	1	39987,95	-	-	-
	30	-	-	-	90853,429	0,7	**129790,61**	-	-	-
$\overline{TS}$		0,68			**0,87**			0,51		
$\overline{DS}$		**6339,27**			29736,85			19235,97		

vez que o operador é chamado.

O custo do SS em termos de quantidade de avaliações da função objetivo é um elemento importante na comparação com algoritmos que não realizam essa etapa inicial de exploração do espaço de busca para determinação de regiões promissoras, como o ECS e o CHA. Nas funções de baixa dimensão (f_{17}, f_{18} e f_{20}, por exemplo), uma parte significativa das avaliações do SSDE são realizadas pelo SS. Por outro lado, em funções de alta dimensão, a quantidade de avaliações realizadas pelo SS tende a ser uma pequena fração do total de avaliações do SSDE. Quando o SS é eficaz em encontrar uma região promissora próxima ao ótimo global, a quantidade de avaliações requeridas pelo DE é reduzida consideravelmente, como nas funções f_{21}, f_9 e f_{11} (ver Tabela 8.14). É interessante que o tamanho da amostra seja determinado automaticamente pelo SS por meio de algum critério, ao invés de ser determinado pelo usuário. Estratégias com esse foco serão investigadas em estudos futuros.

Pelos resultados apresentados na Tabela 8.14, pode-se verificar que o processo de determinação de regiões promissoras, antes do processo de otimização global propriamente dito, é promissor. O SSDE é uma combinação simples, e o SS não possui procedimentos modificadores complexos nem heurísticos, possibilitando pesquisas futuras no refinamento

Tabela 8.15: Estatísticas da *Aval* do SSDE+LS para as funções de (Oliveira, 2004).

F	D	Mín.	1^o Quartil	Mediana	Média	3^o Quartil	Máx.	Desv.Pad.
f_{17}	2	335	362,5	410	587,5	605	1445	368,73
f_{18}	2	365	388,75	402,5	407,75	422,5	470	27,79
f_{20}	6	1729	2036,25	2142,5	2235,75	2368,75	2976	371,08
f_{21}	5	1242	1360	1436	1464,55	1586	1681	144,06
	10	2693	3480	3567	3808,4	3967	7155	865,59
	50	35704	36209	36714	37077,6	37825	38936	1302,47
	100	0	0	0	0	0	0	0
f_{22}	4	712	747,5	752	1683,58	792	11523	2622,43
f_{10}	10	2510	2748	2810	2801,75	2898	3031	162,72
	50	15838	16136	21035	19496,6	21535	22939	3280,66
f_5	50	5005	5005	5005	5205	5505	5505	273,86
	100	6005	6005	6505	6305	6505	6505	273,86
f_9	5	1887	2369,25	5940,5	8855,95	10864,5	35792	9112,79
	10	0	0	0	0	0	0	0
f_4	10	2247	2297	2347	9211,86	2597	50101	18031,04
	20	0	0	0	0	0	0	0
f_{11}	20	3046	3605	3846	4074,3	4443	5543	671,51
	30	0	0	0	0	0	0	0

do algoritmo. Apesar dessa simplicidade, os resultados são bastante relevantes, tornando essa abordagem competitiva com dois algoritmos de alta qualidade.

Com relação aos problemas que o SSDE não foi capaz de resolver, foi detectado que os parâmetros da DE podiam ser alterados para melhorar seu desempenho nesses problemas, apenas modificando o operador que mantém a diversidade populacional (Constante da probabilidade de *crossover*) de $C = 0,9$ para $C = 0,3$, determinado empiricamente. Os novos resultados para essas funções são apresentados na Tabela 8.16. Com essa nova configuração, o SSDE conseguiu ser superior em todas as funções de teste. Nas funções f_5 e f_4, o valor de DS é consideravelmente menor para o SSDE. Além disso, nas funções f_{21} e f_{11} o valor de DS do ECS ultrapassa o limite de 100.000 avaliações. Por outro lado, o SSDE mantém um DS consideravelmente inferior ao limite. Outro ponto importante é o valor da $\overline{TS}$. O SSDE alcançou uma média de 0,97, enquanto que o ECS obteve 0,81.

Para a função f_4, é possível perceber que o SSDE precisou de mais avaliações para resolver o problema menor, com D=10 (ver Tabela 8.14). Isso ocorreu pelo fato de que, nesse problema, o SS executou mais iterações e determinou mais e menores regiões promissoras. Esse efeito sugere que o critério de parada do SS pode ser melhorado. Como a DE precisou procurar em mais de uma região, a quantidade de avaliações foi maior.

Do mesmo modo como concluído nos experimentos das Seções anteriores, o SS pode ser facilmente utilizado em conjunto com o ECS, ou o CHA, para avaliação do desempenho

em trabalhos futuros.

A partir da próxima Seção, são apresentados resultados referentes ao uso das EETs propostas neste trabalho para uso com o ECGA em problemas discretos (binários) de funções deceptivas (ver Seção 6.1).

Tabela 8.16: Comparação entre SSDE+LS (C=0,3), ECS e CHA.

		SSDE			**ECS**			**CHA**		
F	**D**	**Aval**	**TS**	**DS**	**Aval**	**TS**	**DS**	**Aval**	**TS**	**DS**
f_{21}	100	64820	1	**64820**	85821	0,8	107276,25	124302	0,72	172641,66
f_5	100	6005	1	**6005**	24344,45	1	24344,45	-	-	-
f_9	10	26209,45	1	**26209,45**	37671,923	0,65	57956,80	-	-	-
f_4	20	3892,67	1	**3892,67**	71952,667	0,9	79947,41	-	-	-
f_{11}	30	43875,71	0,85	**51618,48**	90853,429	0,7	129790,61	-	-	-
$\overline{TS}$		**0,97**			0,81			0,72		
$\overline{DS}$		**30429**			79863,10			172641,66		

Tabela 8.17: Estatísticas da *Aval* do SSDE+LS com C=0,3 para as funções de (Oliveira, 2004).

f	**D**	**Mín.**	**1^o Quartil**	**Mediana**	**Média**	**3^o Quartil**	**Máx.**	**Desv.Pad.**
f_{21}	100	54647	63488,5	65297,5	64820	67281,75	74742	5929,34
f_5	100	5505	6005	6005	6005	6005	6505	353,55
f_9	10	20039	22558,75	24078,5	26209,45	27110	58924	8136,26
f_4	20	3369	3470,25	3830,5	3829,67	4010	4528	337,79
f_{11}	30	5027	42508	47838	43875,71	53150	97785	25384,75

8.2 Resultados com EETs para o ECGA

Esta Seção está organizada como segue. Na Seção 8.2.1 são apresentados os problemas utilizados para avaliar as propostas, bem como a configuração do ECGA para os experimentos. Na Seção 8.2.2 são apresentados os resultados e comentários sobre os experimentos realizados comparando o ECGA com o ECGA+PSM (descrito na Seção 7.3). Na Seção 8.2.3 são apresentados os resultados e comentários sobre os experimentos realizados comparando o ECGA com o ECGA+BIP (proposto na Seção 7.4).

8.2.1 Problemas de Teste e Configuração do ECGA

As funções deceptivas descritas a seguir foram escolhidas entre as mais comumente usadas na literatura de GAs competentes (Goldberg, 2002) dada sua dificuldade.

$$f_{trap}(u) = \begin{cases} 1, & se\, u = k \\ 1 - d - u * \frac{1-d}{k-1}, & caso\, contrário, \end{cases} \tag{8.1}$$

tal que u é a quantidade de 1s na *string*, k o tamanho da armadilha e d ($d = \frac{1}{k}$) o sinal do *fitness*, a distância entre o ótimo global e o ótimo da decepção;

$$f_{deception}(x) = \sum_{i=0}^{m-1} f_{trap(subx_{k*i} + subx_{k*i+1} + ... + subx_{k*i+k-1})}, \tag{8.2}$$

tal que x é uma *string* e *subx* um subconjunto de bits de x. Assim, essa equação é uma função aditiva (ver Seção 6.1) que soma os valores calculados nas armadilhas (*subx*); e

$$f_{decnoise}(x) = f_{deception}(x) + G(0, \sigma_N^2), \tag{8.3}$$

tal que $G(0, \sigma_N^2)$ é uma função de ruído que segue uma distribuição Gaussiana com média 0 e variância σ_N^2.

Os resultados dos testes estão organizados em tabelas apresentando as seguintes informações:

1. Tamanho do problema (PS, do inglês *Problem Size*): o tamanho do problema no qual o algoritmo foi testado, $PS = k * m$;

2. Tamanho de população usando ECGA (corresponde à coluna ECGA nas tabelas): o tamanho da população ao usar o ECGA original;

3. Tamanho das populações usadas no ECGA+PSM (mostrado nas tabelas como PSM): o tamanho da população ao usar o PSM; PSM_F é o tamanho da Primeira (*First*) população, usado para construir o modelo de alta qualidade. PSM_R é o tamanho das populações restantes;

4. Razão do tamanho da população (RPOPS, do inglês *Ratio of POPulation Size*): $\frac{PSM_R}{ECGA}$. Por exemplo, $RPOPS = 0,4$ significa que o tamanho da população reduzida é 40% do tamanho da população original;

5. Razão de avaliações de função (RFE, do inglês *Ratio of Function Evaluations*): o número de avaliações de função usando PSM dividido pelo número de avaliações de função ao não usar PSM.

Os tamanhos de população foram determinados pelo método de bisseção (Sastry, 2001) com base em 30 execuções do algoritmo. O tamanho médio encontrado pelo método de bisseção é usado como tamanho da população do ECGA e também como tamanho da primeira população (PSM_F) do ECGA+PSM. Para o PSM_R, é usado o tamanho máximo encontrado pelo método da bisseção.

8.2.2 Resultados de ECGA x ECGA+PSM

Na Tabela 8.18 são apresentados resultados para $k = 4$, usando a função deceptiva da Equação 8.2 (lembrando que k é o tamanho da armadilha da função deceptiva). As colunas RPOPS e RFE mostram reduções no tamanho da população e no número de avaliações de função maior que 50%. Assim, o PSM possibilita pelo menos dobrar o desempenho computacional mantendo a mesma qualidade das soluções e a escalabilidade. Outro ponto interessante é que, para valores de PS maiores que 96, RPOPS e RFE diminuem, indicando que, para problemas maiores, a população necessária para resolvê-los pode ser ainda menor.

Tabela 8.18: Resultados para k=4 sem ruído.

PS	ECGA	PSM_F	PSM_R	RPOPS	RFE
32	652	736	304	0,46626	0,44165
48	1192	1344	604	0,50671	0,49336
64	1768	2112	822	0,46493	0,43631
80	2544	2944	1080	0,42453	0,40427
96	3376	3584	1735	0,51392	0,49193
112	4144	4352	2056	0,49614	0,48152
128	5090	5120	2411	0,47367	0,46725
144	5858	6304	2759	0,47098	0,45283
160	6751	7488	2986	0,44230	0,42322
176	7680	8608	3371	0,43893	0,41811
192	8801	9344	3856	0,43813	0,42724

Na Figura 8.1 pode-se notar a melhoria no desempenho usando o ECGA+PSM. O tamanho de população apresentava comportamento $\Theta(m^{1,19}log_2(m))$ (m é a quantidade de armadilhas), mas com o PSM passou a ser consideravelmente menor, $\Theta(m^{0,97}log_2(m))$, com DG próximo do teórico ($DG = 1$). Em outras palavras, o ECGA+PSM experimentalmente aproximou-se mais do modelo geracional de tamanho populacional descrito na Seção 7.2. Uma melhora similar pode ser verificada na quantidade de avaliações, que foi reduzida para cerca de 45%, como pode ser visto na coluna RFE da Tabela 8.18.

Para a função deceptiva da Equação 8.3 com $k = 4$ e ruído, os resultados (ver Tabela 8.19) mostram redução no tamanho da população acima de 50% e próximo a 60% na quantidade de avaliações. A presença de ruído tende a tornar o problema relati-

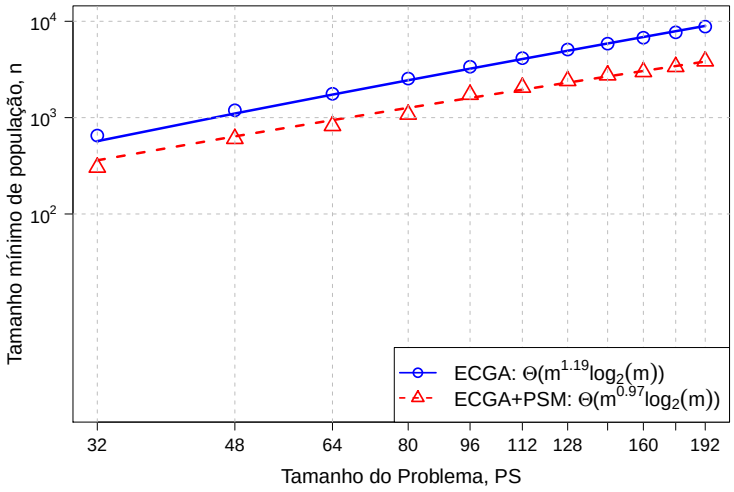


Figura 8.1: Tamanho da População para *k=4*: ECGA versus ECGA com PSM.

vamente mais difícil do que o problema de Equação 8.2, pelo fato de a armadilha ter valor de *fitness* diferente para cada indivíduo. Isso significa que, como o ruído é aleatório, dois indivíduos iguais podem ter valor de *fitness* diferentes. Apesar disso, o PSM provê uma melhoria até maior do que no problema sem ruído.

Tabela 8.19: Resultados para *k=4* com ruído.

PS	ECGA	PSM_F	PSM_R	RPOPS	RFE
16	352	348	269	0,76420	0,75608
32	1046	1152	562	0,53728	0,49073
48	2022	2240	906	0,44807	0,39681
64	3053	3264	1443	0,47265	0,42321
80	4380	4608	1902	0,43425	0,38259
96	5615	5888	2643	0,47070	0,43573

A Figura 8.2 ilustra a redução considerável no tamanho de população e no número de avaliações, como apresentado na Tabela 8.19. As necessidades populacionais do ECGA+PSM crescem mais lentamente, o que proporciona uma melhoria perceptível na eficiência do algoritmo para o problema com ruído testado, bem como em termos de escalabilidade.

A Tabela 8.20 contém os resultados para testes sem ruído para *k =5*. As reduções em termos de RPOPS e RFE são de aproximadamente 50% para PS igual a 15, 30, 60 e 75. O benefício foi menor para PS 45 e 90, com uma redução de pouco mais de 30% para RPOPS e RFE. Mesmo esse resultado mostra uma melhora considerável sobre o ECGA padrão.

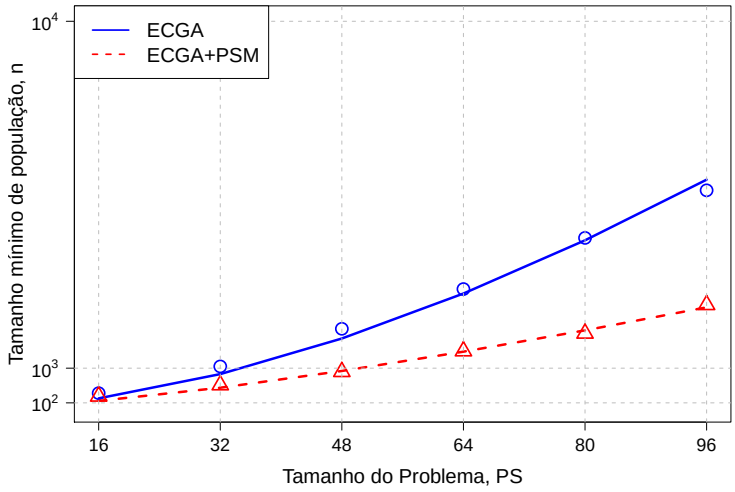


Figura 8.2: Tamanho da População para k=4 com ruído: ECGA x ECGA+PSM.

Além disso, as reduções no tamanho da população, variando de $\Theta(m^{1,33}log_2(m))$ para $\Theta(m^{1,11}log_2(m))$ (Figura 8.3), bem como no número de avaliações de 49% para 67% (ver coluna RFE na Tabela 8.20) são também importantes evidências do efeito da estratégia de PSM.

Tabela 8.20: Resultados para k=5 sem ruído.

PS	ECGA	PSM_F	PSM_R	RPOPS	RFE
15	340	380	155	0,45588	0,49901
30	1080	1180	520	0,48148	0,48694
45	1996	2130	1413	0,70792	0,69431
60	3182	3500	1800	0,56568	0,56311
75	4529	5000	2310	0,51005	0,50335
90	5781	6020	3907	0,67583	0,67431

Na Figura 8.3 é possível comparar a escalabilidade do ECGA com o ECGA+PSM para testes sem ruído utilizando $k = 5$. Os resultados indicam que a contribuição do PSM pode ser ainda mais significativa para problemas de maior escala, uma vez que o expoente $DG = 1,11$ ficou mais próximo do teórico ($DG = 1$).

O uso do PSM reduz o tempo total da execução do ECGA, mostrando que é possível construir modelos precisos com uma quantidade total de indivíduos relativamente pequena. Como a população pode ser menor, o número de avaliações também será inferior, assim como o consumo de memória.

A maior aproximação entre os modelos teóricos para tamanho de população e os resultados experimentais indicam a consistência de modelos já disponíveis na literatura (Harik et al., 1999; Pelikan et al., 2000; Yu et al., 2007), reforçando a importância do mesmo. O ECGA+PSM possibilitou isso devido ao fato de basear-se na qualidade do primeiro modelo, enfatizando a importância de uma amostra de tamanho grande o suficiente na 1^a

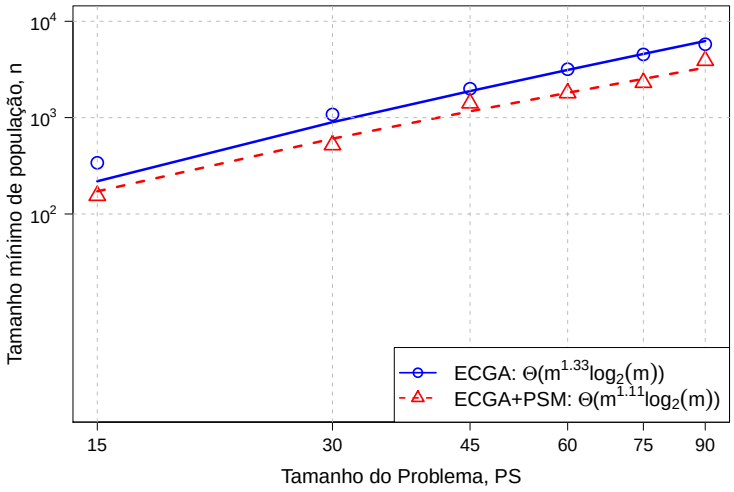


Figura 8.3: Tamanho da População para k=5: ECGA x ECGA+PSM.

geração, podendo ser reduzida nas gerações seguintes sem perda considerável na qualidade do modelo. Como o modelo teórico geracional, na verdade, fornece somente o tamanho para a 1^a população, o ECGA+PSM, naturalmente, aproxima-se mais dos valores previstos pelo modelo.

8.2.3 Resultados de ECGA x ECGA+BIP

Nesta Seção são apresentados os resultados da EET denominada BIP (do inglês, *Biased Initial Population*), introduzida na Seção 7.4. Com essa EET, espera-se que o ECGA seja capaz de encontrar um melhor modelo inicial utilizando uma população de tamanho menor do que o requerido usualmente (ver Seção 7.2).

A função deceptiva utilizada nesse estudo é apresentada na Equação 8.2, na qual o ótimo global é composto apenas por 1s. Para a análise das estratégias propostas, foram utilizados subproblemas de tamanho $k = 4$ (ver Seção 6.1). O valor de tendência utilizado foi de 10% para todos os testes, pois a configuração B40, apresentada na Seção 7.4, foi suficiente para apresentar resultados melhores do que o ECGA padrão. O tamanho da população inicial (n) também é o mesmo apresentado na Seção 7.4, ou seja, é uma população de tamanho reduzido. Como exemplo, será utilizado o resultado de um problema com m=30 (sendo que m é a quantidade de subproblemas concatenados), e população de tamanho n=2122.

Usando BIP, a probabilidade dos possíveis valores de uma variável é modificada, isto é, há uma tendência para algum dos valores possíveis para a variável. No caso do problema binário, a probabilidade tende para 1 ou para 0. Isso significa que se pode gerar uma tendência que induza o algoritmo de otimização para o ótimo global ou para armadilha (ver Seção 6.1). Como não se sabe, de antemão, qual tendência orienta o algoritmo para o ótimo global, uma primeira estratégia de uso dos efeitos de BIP como uma EET foi

investigada para aumento da TS do ECGA com populações pequenas.

Para o ECGA+BIP (ver Figura 8.4) tem-se o seguinte procedimento: metade da população é inicializada com uma tendência em direção a 0 e a outra metade com uma tendência em direção a 1. A partir disso, ocorre a evolução padrão do ECGA (ver Algoritmo 6.2). Ao final da execução, foi possível verificar os seguintes comportamentos:

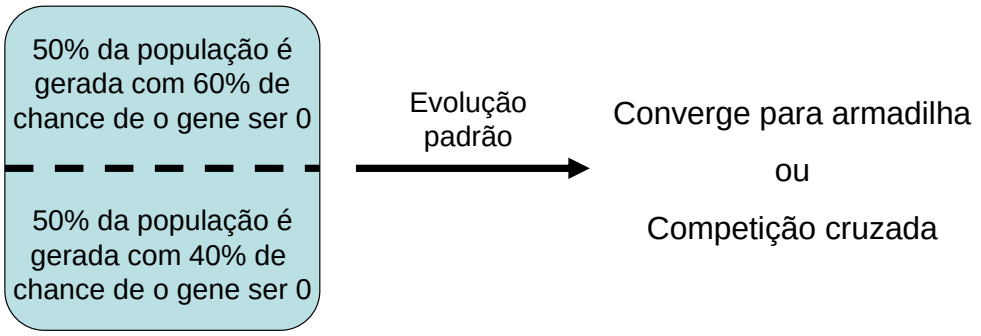


Figura 8.4: Comportamento do ECGA+BIP.

1. O ECGA+BIP convergiu para as armadilhas da função deceptiva na grande maioria dos casos. Isso ocorre devido ao fato de que soluções com subproblemas que convergiram para armadilhas (ótimo locais) possuem, majoritariamente, valor de função objetivo maior do que soluções com subproblemas que convergiram para o seu ótimo global. Essas últimas, para serem melhores, precisam conter muitos subproblemas resolvidos corretamente;

2. Houve competição cruzada entre os BBs (ver Seção 6.3). O ECGA+BIP gerou indivíduos combinando instâncias de BBs que convergiram para a armadilha (ótimo local do subproblema) e instâncias que convergiram para o ótimo global (do subproblema). Com isso, o ECGA+BIP não conseguiu decidir qual tendência seguir, e as soluções ficaram oscilando ao longo das gerações, terminando na convergência de um indivíduo que contém instâncias de armadilhas com instâncias corretas.

É importante informar que os resultados foram similares, independentemente do valor de m. Apesar desses resultados, o comportamento da tendência na população inicial (apresentados na Seção 7.4), pode ser melhor explorado. Assim, outros dois estudos, mostrados na seqüência, foram realizados:

1. Uma análise do comportamento do ECGA+BIP em relação à quantidade de gerações até a convergência;

2. Um estudo buscando identificar o valor de tendência mínimo necessário para que haja um ganho significativo em relação ao ECGA padrão.

Análise de quantidade de gerações de acordo com a tendência

Neste experimento são apresentados os resultados da análise da quantidade de gerações necessárias para a convergência do ECGA+BIP. A análise foi realizada para a função da Equação 8.2 com $k = 4$, para m=8, 12, 16, 20, 24, 28, 32, 36, 40, 44 e 48. Os tamanhos de população correspondentes n=652, 1192, 1768, 2544, 3376, 4144, 5090, 5858, 6752, 7680 e 8802 foram encontrados pelo método da bisseção (ver Seção 7.2).

Dado que o tamanho da população é o mesmo para um determinado problema, independente da tendência, a comparação por quantidade de gerações é plausível. Como apresentado na Seção 3.1, comparar quantidade de gerações não é adequado quando os algoritmos possuem populações de tamanhos distintos. Caso necessário, a quantidade de avaliações nesse experimento pode ser calculada como $Aval = Qtd_{Ger} * n$.

Nas Tabelas 8.21 e 8.22 são apresentadas as porcentagens na redução da quantidade de gerações para a convergência do algoritmo, dadas a probabilidade do gene ser igual a 1 e o tamanho do problema. É possível verificar que existe pouca influência do tamanho do problema em relação à porcentagem de redução. Poderia-se esperar que problemas maiores seriam mais difíceis e que seria necessário uma tendência maior para se obter uma porcentagem de redução similar à de um problema menor. Seria como supor que, se o tamanho do problema dobrar, será necessário dobrar o valor da tendência para se obter a mesma redução. Contudo, os resultados mostram que essa suposição não é válida para o problema testado. Mais do que isso, o comportamento da redução dada a tendência é similar para todos os tamanhos de problema do experimento. Esse comportamento é mais facilmente visualizado nas Figuras 8.5, 8.6 e 8.7.

Tabela 8.21: Porcentagem de redução na quantidade de gerações, m variando de 8 a 28.

Prob.gene=1	**8**	**12**	**16**	**20**	**24**	**28**
50%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
45%	15,34%	14,09%	15,49%	14,59%	9,72%	12,62%
40%	20,63%	18,18%	20,99%	24,20%	16,67%	23,03%
35%	29,63%	29,09%	29,25%	26,33%	27,08%	25,24%
30%	36,51%	31,82%	33,57%	35,94%	28,82%	33,75%
25%	39,15%	42,73%	41,04%	39,15%	37,50%	36,91%
20%	52,38%	45,45%	47,33%	46,62%	45,83%	43,22%
15%	52,38%	59,09%	52,83%	57,30%	47,92%	52,68%
10%	68,25%	59,09%	64,62%	62,99%	58,33%	62,15%

Analisando-se os gráficos, é possível notar uma queda com comportamento próximo a linear, independente do tamanho do problema, como dito anteriormente. Essa conclusão sugere que existe uma possibilidade de aumento de desempenho próximo a 60%, se for utilizada uma tendência de 90%. A questão passa a ser, então, a identificação e exploração dessa tendência.

Tabela 8.22: Porcentagem de redução na quantidade de gerações, m variando de 32 a 48.

Prob.gene=1	32	36	40	44	48
50%	0,00%	0,00%	0,00%	0,00%	0,00%
45%	9,94%	12,93%	11,02%	12,99%	10,50%
40%	18,67%	21,26%	18,77%	21,82%	17,48%
35%	27,71%	23,28%	26,89%	26,23%	24,98%
30%	31,93%	31,03%	34,20%	29,87%	32,48%
25%	36,75%	39,66%	35,28%	37,66%	39,87%
20%	45,78%	40,80%	43,14%	45,45%	43,29%
15%	54,82%	48,28%	51,26%	53,25%	49,24%
10%	63,86%	56,90%	59,38%	61,04%	62,49%

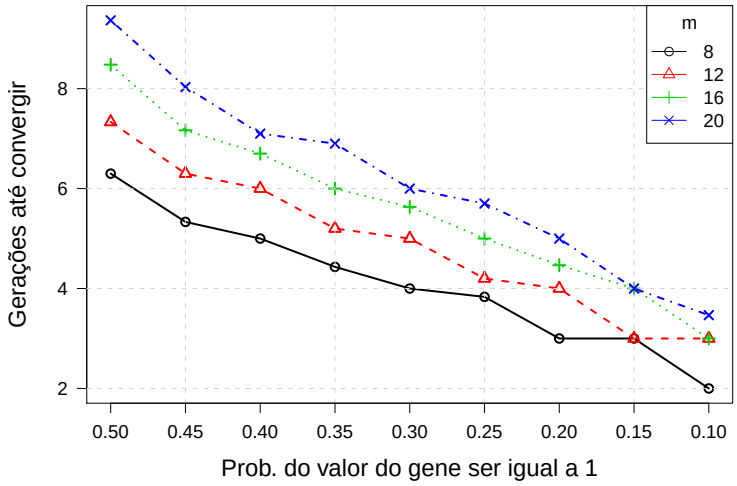


Figura 8.5: Gerações até convergir, para m=8, 12, 16 e 20.

Dado que o número de gerações diminui de forma aproximadamente linear com o valor da tendência (BIP), é possível supor que uma redução no tamanho da população utilizando a combinação ECGA+BIP, de modo similar ao proposto na técnica ECGA+PSM, produziria uma diminuição significativa da quantidade total de avaliações da função objetivo. Nesse caso, a comparação utilizando o número de gerações deixa de ser plausível. Como ainda não foi possível desenvolver uma técnica para identificar o valor da tendência de maneira eficaz, essa combinação (ECGA+BIP) poderá ser explorada novamente em um trabalho futuro. Além disso, devido ao fato de que o BIP utiliza uma população pequena logo na 1^a geração, uma combinação ECGA+BIP+PSM possivelmente não seria adequada, visto que o PSM necessita de uma grande população inicial.

Por fim, deve-se observar também que esses resultados mostram, também, como conhecimento *a priori* sobre o domínio do problema podem acelerar significativamente a convergência do ECGA.

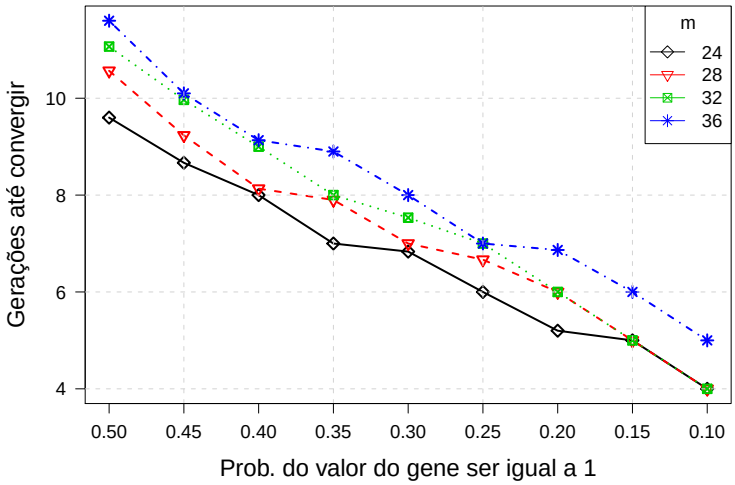


Figura 8.6: Gerações até convergir, para m=24, 28, 32 e 36.

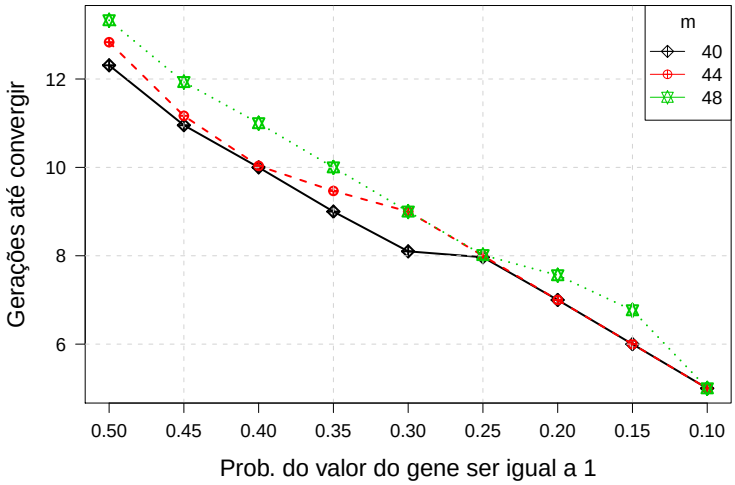


Figura 8.7: Gerações até convergir, para m=40, 44 e 48.

Valor de tendência mínimo

No estudo apresentado na Seção 7.4, duas configurações de tendência na população em direção ao ótimo global foram testadas: B40 (40% de probabilidade de o gene ser 1) e B30 (30% de probabilidade de o gene ser 1). No entanto, é importante identificar qual o valor de tendência mínimo necessário para se obter um aumento significativo da TS, em relação à inicialização imparcial. Para determinar o valor de tendência mínimo, foi utilizado o procedimento descrito no Algoritmo 8.1.

Os resultados dos valores de tendência mínimos são apresentados na Tabela 8.23. Foram necessários pequenos valores de tendência nos problemas $k = 3$ e $k = 4$ para obter um ganho significativo na TS. O valor de tendência mínimo (MB) foi inferior a 5%. Para $k = 5$, o valor do MB foi um pouco maior, cerca de 10%, devido à natureza do problema: valores maiores de k requerem maiores tamanhos de população, como o próprio

Algoritmo 8.1 Procedimento para determinar o valor de tendência mínimo, com incrementos de 1%.

1: Obter a TS do ECGA sem tendência em 100 repetições;

2: Aumentar o valor de tendência em 1% em direção ao ótimo global;

3: Calcular a TS do ECGA+BIP em 100 repetições;

4: Executar um teste de proporção (ver Seção 3.3.3) com $\alpha = 5\%$ para comparar a TS do ECGA com a do ECGA+BIP;

5: Se o teste de hipótese acusar igualdade das proporções, ir para o Passo 2;

6: Retornar o valor de tendência mínimo encontrado.

modelo geracional de tamanho populacional sugere (ver Seção 7.2). Assim, é aceitável a necessidade de um tendência maior.

Tabela 8.23: Tendência mínima (MB, do inglês *Minimum Bias*) para obter uma TS significativamente superior à do ECGA padrão e o valor da melhora obtida ($TS_{Dif} = TS_{ECGA+BIP} - TS_{ECGA}$).

	k=3		k=4		k=5	
m	MB	TS_{Dif}	MB	TS_{Dif}	MB	TS_{Dif}
10	2%	15%	4%	9%	1%	8%
15	4%	4%	3%	9%	1%	8%
20	2%	16%	4%	8%	2%	7%
30	3%	12%	4%	7%	4%	7%

De acordo com os resultados, um pequeno valor de tendência na população inicial representa aumento na TS, pois direciona a população a uma região mais promissora do que uma inicialização aleatória sem informação *a priori*. A possibilidade de uso dessa tendência pode proporcionar uma melhoria considerável na eficiência do processo de otimização, pois possibilita lidar com menores tamanhos de população e uma convergência mais rápida. Assim, essa característica é importante, de modo que seria interessante mais pesquisas para desenvolver algoritmos capazes de identificar e explorar essa tendência de maneira eficiente.

Outra característica importante que deve ser comentada é o fato de a tendência para o ótimo ou para a armadilha resultarem em MPMs similares. Isso significa que os BBs encontrados para os problemas testados foram tão corretos com uma tendência boa quanto com uma tendência ruim. Além disso, é possível obter MPMs de qualidade adequada com populações um pouco menores. Essa informação poderia ser empregada no desenvolvimento de novas EETs.

8.3 Considerações Finais

Neste Capítulo foram apresentados resultados da análise experimental de técnicas de busca por regiões promissoras em problemas contínuos e binários. Para problemas contínuos, foi proposta a Arquitetura de Amostragem Inteligente (SS) combinada com a metaheurística DE. Para técnica combinada, denominada SSDE, foram realizados diversos experimentos, totalizando 62 casos de teste como segue.

1. 30 casos de teste, comparando com os resultados obtidos pelas técnicas ODE e QODE: o SSDE obteve resultado superior em 27 dos 30 casos;

2. 14 casos de teste, comparando com os resultados obtidos pela metaheurística CGRASP: o SSDE obteve resultado superior em 12 dos 14 casos;

3. 18 casos de teste, comparando com os resultados obtidos pelas técnicas ECS e CHA: o SSDE obteve resultado superior em 12 dos 18 casos.

Em algumas das comparações, uma modificação distinta nos parâmetros foi realizada para demonstrar o efeito no desempenho. No primeiro caso (comparação com ODE e QODE), o ganho no valor da DS foi obtido com uma modificação no grau de exploração do operador de reamostragem. Na comparação com o CGRASP, a alteração foi feita na porcentagem de soluções geradas dentro da região promissora encontrada. No último experimento, a melhoria foi obtida com a utilização de dois operadores de reamostragem, um para maior exploração global e outro para maior exploração local. O SS mostrou-se um algoritmo simples, eficiente e eficaz, tornando possível aumentar a eficiência de técnicas de otimização global para problemas multimodais de larga-escala.

Para problemas binários, foram testadas duas técnicas de melhoria de eficiência para o algoritmo ECGA:

1. Gerenciamento de Tamanho de População (PSM): com essa técnica, foi possível reduzir a população em cerca de 50%, bem como a quantidade de avaliações da função objetivo. Apesar disso, a TS manteve-se em 100%. Esse comportamento foi possível pela criação de um modelo de alta qualidade do problema na primeira geração, proporcionando uma orientação do algoritmo para regiões mais promissoras.

2. Análise da Tendência na População Inicial (BIP): com a estratégia utilizada para explorar a tendência foram realizados dois estudos e, a partir desses, foi possível concluir que:

   (a) A quantidade de gerações reduz em proporção praticamente linear com o aumento da tendência;

(b) O valor mínimo de tendência para que se tenha um aumento estatisticamente significativo no valor da TS é relativamente baixo. Portanto, apesar de a estratégia testada não ter resultados conclusivos, os efeitos gerados por uma pequena tendência são interessantes e importantes para serem investigados mais profundamente em trabalhos futuros. Deve-se observar que conhecimento *a priori* embutido na solução de um problema é um tipo de tendência, cujo efeito pode ser melhor determinado com tal tipo de estudo.

9

Conclusões e Desenvolvimentos Futuros

Neste trabalho, diferentes técnicas foram propostas buscando aumentar o desempenho de métodos de otimização global em problemas contínuos e discretos (binários). Tais técnicas seguem uma abordagem conhecida como busca por regiões promissoras no espaço de soluções do problema, de modo a possibilitar uma melhor exploração nos locais com maior probabilidade de conter o ótimo global, evitando a exploração desnecessária em regiões de baixa qualidade. A busca por regiões promissoras pode ser realizada de maneira explícita, como na proposta do DOA e do SS para problemas contínuos, ou implícita, como na abordagem PSM, que viabiliza melhores modelos probabilísticos que orientam o algoritmo de otimização para regiões mais promissoras do espaço de busca, em regiões reduzidas.

Os processos de busca por regiões promissoras propostos diferem de outras abordagens da literatura por possuírem um fim específico e uma independência da metaheurística de otimização global utilizada, uma vez que tais processos podem ser utilizados como uma etapa que antecede um algoritmo de otimização global. Na literatura em geral, as regiões promissoras são identificadas durante o processo exploratório do algoritmo de otimização.

As propostas do DOA e SS mostram que o ganho de desempenho pode ser obtido por meio de estratégias relativamente simples e eficientes. Essa característica das abordagens propostas nesta Tese também as diferencia várias das pesquisas recentes da área, que têm focado em estratégias e modelos probabilísticos cada vez mais complexos.

De forma semelhante ao DOA e SS, a PSM não altera a metaheurística aplicada para otimização global em si, possibilitando um maior desempenho de uma metaheurística populacional por meio de um procedimento simples e eficiente.

Previamente ao desenvolvimento das abordagens propostas, foram realizados estudos sobre o comportamento de alguns algoritmos de otimização (GA, PSO e DE) usando diversos problemas de *benchmark*. Os estudos e as conclusões mais relevantes podem ser sintetizados como segue:

1. Estudo caracterizando os efeitos de modelos probabilísticos com maior (ou menor) Qualidade e menor (ou maior) Custo Computacional. Nesse estudo, utilizaram-se técnicas de estatística e de ML para aproximação de funções, dentre as quais se destacou a SVM (ver Seção 4.4);

2. Análise da redução do espaço de busca em problemas contínuos (ver Seção 4.5): foi possível concluir que o desempenho dos algoritmos de otimização global, para os problemas de *benchmark* testados, pode não ser influenciado por uma inicialização heurística caso a complexidade do problema seja alta e/ou a configuração do algoritmo seja inadequada. Por outro lado, se o algoritmo for capaz de encontrar o ótimo global em espaços de busca maiores, mesmo com baixa taxa de sucesso, uma redução prévia possibilita um aumento significativo dessa taxa, o que motivou o desenvolvimento das abordagens DOA (ver Seção 4.6) e SS (ver Capítulo 5);

3. Análise da qualidade do modelo no algoritmo ECGA para problemas discretos (binários) (ver Seção 7.2): foi verificado que a qualidade do modelo inicial é crucial para que o algoritmo obtenha sucesso no processo de busca pelo ótimo global. Esse resultado motivou o desenvolvimento de abordagens que possibilitem a criação de um modelo inicial de alta qualidade;

4. Estudo de métricas para comparação de metaheurísticas: buscou-se identificar as características favoráveis e desfavoráveis de métricas presentes na literatura da área, para então escolher as métricas que poderiam ser mais adequadas para a comparação das metaheurísticas. Como conclusão desse estudo, destacam-se:

   (a) Para problemas cujo ótimo global é conhecido, foram escolhidas as seguintes métricas: 1) quantidade de avaliações da função objetivo até se encontrar o ótimo ($Aval_{O*}$); 2) taxa de sucesso (TS); e 3) desempenho de sucesso (DS).

   (b) Para um melhor entendimento sobre o comportamento do algoritmo e para possibilitar uma melhor comparação com outros trabalhos, informações estatísticas (ver Tabelas do Capítulo 8) tais como: 1) mínimo; 2) 1^o quartil; 3) mediana; 4) média; 5) 3^o quartil; 6) máximo; e 7) desvio-padrão;

   (c) Com base na análise de redução de espaço de busca (ver item 2), explorou-se esse tipo de redução em conjunto com a TS como uma métrica adicional para comparação de metaheurísticas.

A partir desses estudos, a pesquisa buscou a criação de estratégias capazes de aumentar o desempenho de metaheurísticas populacionais. Para problemas contínuos, duas técnicas foram desenvolvidas. A primeira proposta utiliza conceitos estatísticos de DoE, denominada DOA, para eliminar porções do espaço de busca com baixa probabilidade de conter o ótimo global (ver Seção 4.6). A segunda técnica proposta, denominada arquitetura de Amostragem Inteligente (SS, ver Capítulo 5), foi investigada mais profundamente por ter apresentado resultados melhores. A SS utiliza técnicas de ML para identificar características das soluções de alta qualidade com o objetivo de gerar soluções ainda melhores, orientando o algoritmo a uma ou mais regiões promissoras. Após determinadas, essas regiões devem ser exploradas de maneira mais refinada para que o ótimo local seja encontrado. Para essa tarefa, foi utilizado nos experimentos apresentados (ver Capítulo 8) o algoritmo DE, metaheurística populacional que apresentou melhor desempenho de acordo com os testes descritos nas Seções 4.5 e 4.6. A combinação do SS com a DE, denominada SSDE, foi avaliada em 62 casos de teste de problemas de otimização global contínua e comparada com outros algoritmos avançados de mesma finalidade, obtendo melhorias significativas em 51 casos, cerca de 82%.

A SS é uma técnica para determinação de uma ou mais regiões promissoras sem o uso de heurísticas como cálculo de derivadas, matriz hessiana ou gradiente. Essa característica viabiliza sua aplicação a uma ampla gama de problemas. Em certo grau, pode-se dizer que a SS transforma um problema de otimização global em um ou mais problemas de otimização local, que ocorrem em uma ou mais regiões promissoras. Nesse sentido, a SS, associada a um algoritmo de busca local, pode resultar em uma combinação interessante como método de otimização.

A SS também pode ser caracterizada como uma técnica de pré-processamento, uma vez que sua aplicação produz uma ou mais populações iniciais, que podem ser então utilizadas em qualquer algoritmo de otimização populacional, sem a necessidade de qualquer modificação no mesmo.

Embora os resultados apresentados sejam relevantes, mostrando que a SS pode ser adequada para a determinação de regiões promissoras, alguns aspectos podem ser ainda investigados, buscando o aumento do desempenho, como os destacados a seguir:

1. Uma dificuldade do emprego de técnicas de ML em geral é determinar uma adequada amostragem de exemplos de todas as classes. No presente estudo, foi adotada uma divisão conservadora de 50% para cada classe, mas essa pode não ser a melhor opção;

2. O KNN é um dos classificadores mais simples. A opção de usar $K = 1$ implica o armazenamento de todos os exemplos de treinamento. Desse modo, o consumo de memória e o tempo computacional da busca pelo vizinho mais próximo podem

tornar-se altos para problemas de larga-escala. Além disso, o KNN não é capaz de revelar correlação entre atributos dos dados;

3. A quantidade de regiões determinadas pelo RBL pode ser muito grande se o problema possuir um número muito alto de ótimos locais. O algoritmo de otimização global é executado para cada uma dessas regiões de acordo com uma ordem definida pelo melhor valor obtido em cada região. Contudo, se o ótimo global estiver entre as últimas regiões, a eficiência do método pode diminuir consideravelmente;

4. A etapa de reamostragem da SS, conforme proposta, restringe-se à adição aleatória de ruído às soluções promissoras. Tal ruído pode ser mais adequado para certos tipos de problemas e ruim para outros.

Buscando melhorar a SS segundo os aspectos de (1) a (4) apresentados acima, propõe-se o seguinte conjunto de trabalhos futuros:

1. Desenvolvimento de outros operadores de reamostragem, incluindo a possibilidade de uso inteligente de mais de um operador em uma mesma execução do SS. Podem ser utilizados, por exemplo, operadores de recombinação de metaheurísticas conhecidas;

2. Realizar um estudo sobre a qualidade dos classificadores induzidos na SS e, possivelmente, uma análise do uso de outras técnicas no lugar do KNN e do RBL;

3. Estudo sobre melhor divisão da amostra (soluções promissoras *versus* não-promissoras). Isso pode ser realizado, por exemplo, por meio de informação estatística extraída da população atual ou do *fitness* da população;

4. Desenvolvimento de uma SS com parâmetros auto-ajustáveis;

5. Combinação da SS com outros algoritmos populacionais de otimização global, como o QODE, o CGRASP, e o ECS;

6. Avaliação da SS em problemas práticos do mundo real, como problemas das áreas de Física e Engenharia.

No contexto de problemas discretos (binários) com correlação entre variáveis, o algoritmo de otimização necessita encontrar tais correlações. Para esse fim, a literatura da área de EDAs propõe a construção de modelos probabilísticos. A qualidade dos modelos pode orientar o algoritmo de otimização em direção ao ótimo, mas também na direção

contrária. Para evitar modelos equivocados, deve-se utilizar uma amostra relativamente grande, reduzindo com isso a eficiência computacional do EDA.

Buscando superar essa dificuldade, duas técnicas de melhoria de eficiência (EETs) para guiar o algoritmo (no caso o ECGA) na direção do ótimo global com menor custo computacional foram desenvolvidas. A primeira proporciona a criação de modelos de alta qualidade por meio do Gerenciamento do Tamanho da População (PSM, ver Seção 7.3). A segunda técnica explora a identificação de uma tendência na população para gerar descendentes de qualidade maior (BIP, ver Seção 7.4).

Outro aspecto importante do PSM é que este possibilitou encontrar tamanhos de população que se aproximaram consideravelmente do modelo teórico proposto em (Pelikan et al., 2000), com $DG = 0,97$ para $k = 4$ e $DG = 1,11$ para $k = 5$. Isso reforça a validade do modelo teórico e mostra a consistência da descoberta experimental, que revelou a importância do primeiro modelo probabilístico para um EDA.

Tanto a PSM quanto a BIP foram desenvolvidas buscando o compromisso entre complexidade computacional e precisão do modelo, com o objetivo de que o ECGA tenha a mesma taxa de sucesso ($TS = 100\%$), mantendo o comportamento de crescimento populacional (ver Seção 7.2) em relação ao tamanho do problema, porém com uma redução na quantidade total de avaliações. Com o PSM, obteve-se aproximadamente 50% de redução. Por outro lado, com o BIP, não foi possível manter o valor de TS. Apesar disso, houve contribuições no entendimento do efeito do BIP nos problemas testados, mostrando que existe uma redução aproximadamente linear na quantidade de avaliações à medida que se aumenta a tendência. Foi também determinada qual a tendência mínima necessária para que haja uma mudança significativa no valor da TS.

Apesar da tendência gerada pelo BIP poder tanto acelerar a convergência para o ótimo global quanto para a armadilha, essa EET possibilita a obtenção de MPMs de alta qualidade (poucos erros nos BBs) na grande maioria dos casos e de forma relativamente rápida. Essa informação, obtida a menor custo computacional devido ao BIP poderia, por exemplo, viabilizar um algoritmo de busca que trabalhasse em um espaço de variáveis que correspondesse apenas aos BBs, ao invés do problema completo. Com base nessa informação, por exemplo, um algoritmo de busca local ou exaustiva poderia ser empregado para cada BB.

As duas EETs, PSM e BIP, possibilitam o tratamento de problemas de maior escala, pois necessitam de populações menores e, conseqüentemente, menos recursos computacionais. Apesar disso, as seguintes limitações permanecem:

1. Apesar do PSM apresentar resultados experimentais de redução do tamanho da população que se aproximam do comportamento do modelo teórico, não há um modelo teórico que preveja tamanhos ideais para a população inicial e para as populações

seguintes. É importante lembrar que o tamanho pode ser reduzido a cada geração, porém a taxas menores;

2. Com o BIP não foi possível desenvolver uma estratégia capaz de utilizar adequadamente a tendência de modo a orientar o algoritmo de otimização sempre ao ótimo global, apesar de ter evidenciado que seu efeito pode ser positivo.

Com base nessas considerações sobre as duas EETs desenvolvidas, os três principais objetivos futuros seriam:

1. Realizar um estudo do funcionamento do PSM gerando modelo teórico capaz de prever o tamanho ideal da população inicial e o grau de redução aceitável para as próximas gerações, uma vez que o estudo atual baseou-se em resultados experimentais. Esse modelo poderia ser desenvolvido a partir do modelo teórico de tamanho populacional existente para o ECGA (ver Seção 7.2);

2. Pesquisar técnicas que possibilitem o funcionamento adequado do BIP, visto que foi demonstrado haver ganho de desempenho a partir dessa abordagem;

3. Avaliação do comportamento dessas EETs em outros problemas discretos, não apenas binários, em que haja presença de correlação entre variáveis;

Em síntese, foi proposta uma métrica para avaliação de algoritmos de otimização global, bem como dois algoritmos para determinação de regiões promissoras, DOA e SS, que podem ser utilizados como pré-processamento em problemas contínuos de otimização global e trabalhar em conjunto com diversas metaheurísticas, possibilitando um aumento significativo no desempenho do processo de otimização.

Para problemas discretos, as duas EETs desenvolvidas são capazes de reduzir consideravelmente a quantidade de avaliações da função objetivo, por trabalharem com modelos de melhor qualidade e população reduzida. Essas EETs, que foram aplicadas ao ECGA, podem, também, ser exploradas diretamente em outros EDAs.

Como resultado direto deste trabalho, foram publicados 6 artigos em anais de congressos nacionais e internacionais. Além disso, 2 artigos estão sendo finalizados para serem submetidos a periódicos internacionais. Uma lista completa e detalhada do que foi publicado pode ser encontrada na Lista de Artigos Publicados, no preâmbulo desta Tese.

Referências Bibliográficas

Alba, E., C. Blum, P. Asasi, C. Leon, & J. A. Gomez (2009). *Optimization techniques for solving complex problems.* New Jersey: Wiley.

Alevizos, P. & D. Tasoulis (2002). Improving the orthogonal range search k-windows algorithm. In *In Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, pp. 239–245.

Alexandrov, N., J. E. Dennis, Jr., R. M. Lewis, & V. Torczon (1998). A trust-region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization 15*(1), 16–23.

Arabshari, P., A. Gray, I. Kassabalidis, & D. A. Adaptive routing in wireless communication networks using swarm intelligence. In *Proceedings of 9th AIAA Int. Communications Satellite Systems Conference*, France, pp. 17–20.

Ardia, D. (2008). Deoptim package: Differential evolution optimization in r language.

Bäck, T., D. Fogel, & Z. Michalewicz (2000a). *Evolutionary Computation 1 Basic Algorithms and Operators.* Bristol and Philadelphia: Institute of Phisics Publishing.

Bäck, T., D. Fogel, & Z. Michalewicz (2000b). *Evolutionary Computation 2 Basic Algorithms and Operators.* Bristol and Philadelphia: Institute of Phisics Publishing.

Bäck, T., D. B. Fogel, & Z. Michalewicz (1997). *Handbook of Evolutionary Computation.* Institute of Physics; Ringbound.

Bäck, T., F. Hoffmeister, & H.-P. Schwefel (1991). A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon University, Piscataway.

Baluja, S. & S. Davies (1997). Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 30–38. Morgan Kaufmann.

Bandyopadhyay, S., S. K. Pal, & B. Aruna (2005). Multiobjective gas, quantitative indices, and pattern classification. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 34*(5), 2088–2099.

Bartz-Beielstein, T. (2006). *Experimental Research in Evolutionary Computation: The New Experimentalism (Natural Computing Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Beraldi, P. & A. Ruszczyński (2005). Beam search heuristic to solve stochastic integer problems under probabilistic constraints. *European Journal of Operational Research 167*(1), 35–47.

Bhaskar, V., S. K. Gupta, & A. K. Ray (2000). Applications of multiobjective optimization in chemical engineering. *Reviews in Chemical Engineering 16*(1), 1–54.

Birru, H. K., K. Chellapilla, & S. Rao (1999). Local search operators in fast evolutionary programming. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'99).*, Washington, pp. 1506–1513.

Bonabeau, E., M. Dorigo, & G. Theraulaz (2000). *Swarm Intelligence From Natural to Artificial Systems*. Bristol and Philadelphia: Oxford University Press.

Bonet, De, J. S., C. L. Isbell, & P. Viola (1996). Mimic: finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, pp. 424–430. The MIT Press.

Box, G. E. P., W. G. Hunter, & J. S. Hunter (1978). *Statistics for experimenters*. New York: John Willey.

Boyd, S. & L. Vandenberghe (2004). *Convex Optimization*. New York, NY, USA: Cambridge University Press.

Bożejko, W. & M. Wodecki (2006). Population-based heuristics for hard permutational optimization problems. *International Journal of Computational Intelligence Research 2*(2), 151–158.

Cantú-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer Academic Pub.

Carlisle, A. & G. Dozier (2001). An off-the-shelf pso. In *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, IN, pp. 1–6.

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

Chelouah, R. & P. Siarry (2003). Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multi-minima functions. *European Journal of Operational Research. 148*(2), 335–348.

Chen, S. & S. Smith (1999). Improving genetic algorithms by search space reduction (with applications to flow shop scheduling). In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Choi, S. W., J. H. Park, & I.-B. Lee (2004). Process monitoring using a gaussian mixture model via principal component analysis and discriminant analysis. *Computers and Chemical Engineering 28*(18), 1377–1387.

Chou, H. H., G. Premkumar, & C. H. Chu (2001). Genetic algorithms for communications network design: An empirical study of factor that influence performance. In *IEEE Transactions on Evolutionary Computation.*, Volume 5, pp. 236–249.

Chow, C. & C. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory 14*(3), 462–467.

Chuang, L.-Y., H.-W. Chang, C.-J. Tu, & C.-H. Yang (2008). Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry 32*(1), 29–38.

Clerc, M. (2006). *Particle swarm optimization*. London: ISTE.

Cohen, W. W. (1995). Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123. Morgan Kaufmann.

Coletti, M., T. D. Lash, R. S. Michalski, C. Mandsager, & R. E. Moustafa (1999). Comparing performance of the learnable evolution model and genetic algorithms. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, & R. E. Smith (Eds.), *GECCO*, pp. 779. Morgan Kaufmann.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

Deb, K. & D. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In *In Schaffer, J., editor, Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, USA, pp. 42–50. Morgan Kaufmann.

Deb, K. & D. E. Goldberg (1993). Analyzing deception in trap functions. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2*, pp. 93–108. San Mateo, CA: Morgan Kaufmann.

Dennis, Jr., J. E. & R. B. Schnabel (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia: Society for Industrial and Applied Mathematics.

Ding, N., S.-D. Zhou, & Z.-Q. Sun (2008). Histogram-based estimation of distribution algorithm: a competent method for continuous optimization. *Journal of Computer Science and Technology 23*(1), 35–43.

Dorigo, M. (Ed.) (2004). *Ant colony optimization*. Cambridge: MIT Press.

Duque, T., D. Goldberg, & K. Sastry (2008a). Enhancing the Efficiency of The ECGA. *Proceedings of the X Parallel Problem Solving From Nature (PPSN2008): Dortmund, Germany, September 13-17, 2008*, 165.

Duque, T. S., D. E. Goldberg, & K. Sastry (2008b). Enhancing the efficiency of the ecga. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, Berlin, pp. 165–174. Springer.

Eberhart, R. C. & J. Kennedy (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science.*, Nagoya, Japan, pp. 39–43.

Eshelman, L. & J. Schaffer (1993). Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms 2*, 187–202.

Glover, F. & M. Laguna (1997). *Tabu search*. Boston: Kluwer Academic Publishers.

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

Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers.

Goldberg, D. E., K. Deb, H. Kargupta, & G. Harik (1993). Rapid accurate optimization of difficult problems using fast messy genetic algorithms. In S. Forrest (Ed.), *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA, pp. 56–64. Morgan Kaufmann.

Granelli, G. & M. Montagna (2007). Genetic algorithm applications to the solution of electric power system problems. In C. M. Lefebvre (Ed.), *Electric power: generation, transmission and efficiency*, pp. 75–120. New York: Nova Science Publishers.

Guo, L. & I. Matta (2003). Search space reduction in qos routing. *Comput. Networks 41*(1), 73–88.

Harik, G. (1999). Linkage Learning via probabilistic modeling in the ECGA. Technical report, University of Illinois at Urbana Chapaign, Urbana, IL.

Harik, G., E. Cantú-Paz, D. E. Goldberg, & B. L. Miller (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation 7*(3), 231–253.

Harik, G., F. Lobo, & D. Goldberg (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation 3*(4), 287–297.

Harik, G. R. & D. E. Goldberg (2000). Learning Linkage through probabilistic expression. *Computer Methods in Applied Mechanics and Engineering 186*, 295–310.

Haupt, R. & S. Haupt (2004). *Practical Genetic Algorithms*. Hoboken: John Wiley.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

Heckerman, D., D. Geiger, & M. Chickering. (1994). Learning bayesian networks: The combination of knowledge and statistical data. Technical report, Microsoft Research, Redmond, WA.

Hirsch, M. J., C. N. de Meneses, P. M. Pardalos, & M. G. C. Resende (2007). Global optimization by continuous grasp. *Optimization Letters 1*(2), 201–212.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM. 9*(3), 297–314.

Holland, J. H. (1975). *Adaptation in natural artificial systems*. Ann Arbor: University of Michigan Press.

Hooke, R. & T. A. Jeeves (1961). "direct search" solution of numerical and statistical problems. *J. ACM 8*(2), 212–229.

Hyvarinen, A., J. Karhunen, & E. Oja (2001). Gradients and optimization methods. In S. Haykin (Ed.), *Independent component analysis*, pp. 57–76. New York: John Wiley.

Jelasity, M., P. Ortigosa, & I. Garcia (2001). Uego, an abstract clustering technique for multimodal global optimization. *Journal of Heuristics. 7*(3), 215–233.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput. 9*(1), 3–12.

Jong, K. A. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Tese de Doutorado, University of Michigan, Ann Arbor.

Julstrom, B. A. (1999). Comparing darwinian, baldwinian, and lamarckian search in a genetic algorithm for the 4-cycle problem. In *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pp. 134–138.

Kennedy, J., R. Eberhard, & Y. Shi (2003). *Swarm Intelligence.* Morgan Kaufmann.

Kvasnicka, V., M. Pelikan, & J. Pospichal (1996). Hill climbing with learning (an abstraction of genetic algorithm). *Neural Network World 6*, 773–796.

Laarhoven, P. J. M. v. & E. H. L. Aarts (1987). *Simulated annealing : theory and applications.* Canada: Kluwer Academic Publishers.

Lane, P. & F. Gobet (2005). Discovering predictive variables when evolving cognitive models. In *Pattern Recognition and Data Mining*, Volume 3686, pp. 108–117. Lecture Notes in Computer Science, Springer-Verlag.

Larrañaga, P. & J. A. Lozano (Eds.) (2002). *Estimation of distribution algorithms: a new tool for evolutionary computation.* Boston: Kluwer Academic Publishers.

Liang, K.-H., X. Yao, & C. Newton (1999). Combining landscape approximation and local search in global optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, Volume 2, Piscataway, NJ, pp. 1514–1520. IEEE Press.

Liang, K.-H., X. Yao, & C. Newton (2000, July). Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems 4*(3), 172–183.

Lima, C., K. Sastry, D. Goldberg, & F. Lobo (2005). Combining competent crossover and mutation operators: a probabilistic model building approach. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, 735–742.

Linden, D. S. (2002). Innovative antenna design using genetic algorithms. In P. J. Bentley & D. W. Corne (Eds.), *Creative evolutionary systems*, pp. 487–510. San Francisco: Morgan Kaufmann Publishers.

Mauttone, A. & M. E. Urquhart (2009). A route set construction algorithm for the transit network design problem. *Computers and Operations Research 36*(8), 2440–2449.

Melo, V. V. & A. C. B. Delbem (2008a). Easy efficiency-enhancement technique for the ecga. In M. M. B. R. Vellasco, M. C. P. de Souto, & J. J. F. Cerqueira (Eds.), *Brazilian Symposium on Neural Networks (SBRN'08)*, pp. 69–74. IEEE Computer Society.

Melo, V. V. & A. C. B. Delbem (2008b). On promising regions and optimization effectiveness of continuous and deceptive functions. In *IEEE Congress on Evolutionary Computation*, pp. 4184–4191. IEEE.

Melo, V. V., A. C. B. Delbem, D. L. Pinto Junior, & F. M. Federson (2007a). Discovering promising regions to help global numerical optimization algorithms. In *6th Mexican International Conference on Artificial Intelligence (MICAI'07)*, pp. 72–82.

Melo, V. V., A. C. B. Delbem, D. L. Pinto Junior, & F. M. Federson (2007b). Improving global numerical optimization using a search-space reduction algorithm. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, New York, NY, USA, pp. 1195–1202. ACM.

Merz, P. (2003). The compact memetic algorithm. In *Workshop on Memetic Algorithms (WOMA).*, Chicago, pp. 234–241. Morgan Kaufmann.

Michalewicz, Z., D. Dasgupta, R. G. Le Riche, & M. Schoenauer (1996). Evolutionary algorithms for constrained engineering problems. *Comput. Ind. Eng. 30*(4), 851–870.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Montgomery, D. C. (1997). *Design and Analysis of Experiments*, Volume 1. John Wiley and Sons, Inc.

Montgomery, D. C. & G. C. Runger (1998). *Applied Statistics and Probability for Engineers* (2 ed.). John Wiley and Sons.

Mühlenbein, H. & T. Mahnig (1999). Fda - a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation 7*(4), 353–376.

Mühlenbein, H. & G. Paass (1996). From recombination of genes to the estimation of distributions i. binary parameters. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, London, UK, pp. 178–187. Springer-Verlag.

Myers, R. H. & D. C. Montgomery (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments* (second ed.). New York: Wiley.

Nair, P. & A. Keane (2001). Passive vibration suppression of flexible space structures via optimal geometric redesign. *American institute of aeronautics and astronautics journal 39*(7), 1338–1346.

Nelder, J. A. & R. Mead (1965, January). A simplex method for function minimization. *The Computer Journal 7*(4), 308–313.

Nelles, O. (2000). *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. New York: Springer.

Nievergelt, J. (2000). Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *SOFSEM '00: Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*, London, pp. 18–35. Springer-Verlag.

Nocedal, J. & S. J. Wright (2006). *Numerical Optimization*. New York: Springer.

Noman, N. & H. Iba (2008, Feb.). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation 12*(1), 107–125.

Oliveira, A. C. M. d. (2004, Julho). *Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos*. Tese de Doutorado, Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

Ong, Y., P. Nair, A. Keane, & K. Wong (2004). Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin (Ed.), *Knowledge Incorporation in Evolutionary Computation*, Volume 167, pp. 307–332. Berlin: Springer.

Ortigosa, P. M., I. Garcia, I. García, & M. Jelasity (2001). Reliability and performance of uego, a clustering-based global optimizer. *Journal of Global Optimization 19*, 2001.

Osman, I. & J. Kelly (Eds.) (1996). *Meta-heuristics: Theory and Applications*. Boston: Kluwar Academic Publishers.

Papadimitriou, C. H. & K. Steiglitz (1998). *Combinatorial Optimization; Algorithms and Complexity*. Dover Publications.

Patel, J. K. & C. B. Read (1996). *Handbook of the normal distribution*. New York: Marcel Dekker.

Paul, T. K. & H. Iba (2003). Linear and combinatorial optimizations by estimation of distribution algorithms. In *Proceedings of the 9th MPS Symposium on Evolutionary Computation*, Japan.

Pelikan, M., D. E. Goldberg, & E. Cantú-Paz (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic And Evolutionary Computation Conference*, pp. 524–532.

Pelikan, M., D. E. Goldberg, & E. Cantu-Paz (2000). Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation 8*(3), 311–340.

Pelikan, M., D. E. Goldberg, & F. G. Lobo (2002). A Survey of Optimization by Building and Using Probabilistic Models. In *Computational Optimization and Applications*, pp. 5–20. Berlin, Heidelberg, and New York: Kluwer Academic Publishers.

Pelikan, M. & H. Mühlenbein (1999). Marginal distributions in evolutionary algorithms. In *In Proceedings of the International Conference on Genetic Algorithms Mendel'98*, pp. 90–95.

Pelikan, M., M. Pelikan, D. E. Goldberg, & D. E. Goldberg (2000). Bayesian optimization algorithm, population sizing, and time to convergence. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 275–282.

Pelikan, M., K. Sastry, & D. E. Goldberg (2008). Sporadic model building for efficiency enhancement of the hierarchical BOA. *Genetic Programming and Evolvable Machines 9*(1), 53–84.

Pfeiffer, P. E. (1965). *Concepts of probability theory*. New York: McGraw-Hill.

Price, K. V., R. M. Storn, & J. A. Lampinen (Eds.) (2005). *Differential evolution: a practical approach to global optimization*. Germany: Springer.

Pukelsheim, F. (1994). The three sigma rule. *The American statistician 48*(2), 88–91.

Rahnamayan, S., H. R. Tizhoosh, & M. M. A. Salama (2007). Quasi-oppositional differential evolution. In *IEEE Congress on Evolutionary Computation*, pp. 2229–2236. IEEE.

Rahnamayan, S., H. R. Tizhoosh, & M. M. A. Salama (2008). Opposition versus randomness in soft computing techniques. *Appl. Soft Comput. 8*(2), 906–918.

Resende, M. G. C. & C. Ribeiro (2003). Greedy randomized adaptive search procedures. In F. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics*, pp. 219–249. Boston: Kluwer Academic Publishers.

Resende, M. G. C. & J. P. d. Sousa (2004). *Metaheuristics: computer decision-making*. Boston: Kluwer Academic Publishers.

Reynolds, R. G. & W. Sverdlik (1994). Problem solving using cultural algorithms. In *International Conference on Evolutionary Computation*, pp. 645–650.

Rinne, H. (Ed.) (2008). *The Weibull distribution: a handbook*. Boca Raton: CRC Press.

Rudolph, G. (1999). Self-adaptation and global convergence: a counter-example. In *Proceedings of the 1999 Congress on Evolutionary Computation*, Volume 1, pp. 646–651. IEEE Press.

Sastry, K. (2001). Evaluation-relaxation Schemes for Genetic and Evolutionary Algorithms. Dissertação de Mestrado, University of Illinois at Urbana-Champaign.

Sastry, K., D. Goldberg, & M. Pelikan (2004). Efficiency enhancement of probabilistic model building genetic algorithm. Technical report, Illinois Genetic Algorithms Laboratory , Univeristy of Illinois at Urbana Champaign, Urbana, IL.

Sastry, K. & D. E. Goldberg (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. Technical report, In Deb, K., et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004), Part II, LNCS 3103.

Schervish, M. J. (1996). P values: what they are and what they are not. *The American Statistician 50*(3), 203–206.

Schlkopf, B. & A. J. Smola (2001, December). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press.

Sheskin, D. J. (2000). *Handbook of parametric and nonparametric statistical procedures*. Boca Raton: Chapman & Hall/CRC.

Siarry, P. & Z. Michalewicz (Eds.) (2008). *Advances in Metaheuristics for Hard Optimization*. Natural Computing Series. Springer.

Srinivas, M. & L. M. Patnaik (1991, 18-21 Nov). Learning neural network weights using genetic algorithms- improving performance by search-space reduction. *1991 IEEE International Joint Conference on Neural Networks 3*(IEEE Cat. No. 91CH3065-0), 2331–2336.

Srivastava, R. (2002). Time continuation in genetic algorithms. Dissertação de Mestrado, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL.

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

Suganthan, P. N., N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, & S. Tiwari (2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report 2005005, Nanyang Technological University, Singapore.

Tan, P.-N., M. Steinbach, & V. Kumar (2005, May). *Introduction to Data Mining* (1 ed.). Addison Wesley.

Tasoulis, D. K., V. P. Plagianakos, & M. N. Vrahatis (2005). Clustering in evolutionary algorithms to efficiently compute simultaneously local and global minima. In *Congress on Evolutionary Computation*, pp. 1847–1854. IEEE.

Thissen, U., H. Swierenga, A. P. Weijer, De, R. Wehrens, W. J. Melssen, & L. Buydens (2005). Multivariate statistical process control using mixture modelling. *Journal of Chemometrics 19*(1), 23–31.

Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. In *CIMCA '05 Vol-1 (CIMCA-IAWTIC'06)*, Washington, DC, USA, pp. 695–701. IEEE Computer Society.

Tsutsui, S., M. Pelikan, & A. Ghosh (2006). Edge histogram based sampling with local search for solving permutation problems. *International Journal of Hybrid Intelligent Systems 3*(1), 11–22.

Von Zuben, F. J. (2004). Computação evolutiva, uma abordagem pragmática. Technical report, Unicamp, Online.

Vrahatis, M. N., B. Boutsinas, P. Alevizos, & G. Pavlides (2002). The new k-windows algorithm for improving the k-means clustering algorithm. *J. Complex. 18*(1), 375–391.

Whitley, L., A. Howe, S. Rana, J.-P. Watson, & L. Barbulescu (1998). Comparing heuristic search methods and genetic algorithms for warehouse scheduling. In *In Systems, Man and Cybernetics*, pp. 2430–2435.

Wolpert, D. H. & W. G. Macready (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM.

Wolpert, D. H. & W. G. Macready (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*(1), 67–82.

Xu, P. (2003). A hybrid global optimization method: the multi-dimensional case. *Journal of Computational and Applied Mathematics 155*(2), 423–446.

Yan, L. & Y. Changrui (2007). A new hybrid algorithm for feature selection and its application to customer recognition. In A. Dress, Y. Xu, & B. Zhu (Eds.), *Combinatorial optimization and applications: first international conference*, Volume 4616, pp. 102–111. Berlin: Springer.

Yen, J. & B. Lee (1997). A simplex genetic algorithm hybrid. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97).*, Indianapolis, pp. 175–180.

Yu, T., K. Sastry, D. Goldberg, & M. Pelikan (2007). Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 601–608. ACM Press New York, NY, USA.

Zhu, C., R. H. Byrd, P. Lu, & J. Nocedal (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw. 23*(4), 550–560.

Zhu, Q.-Y., A. K. Qin, P. N. Suganthan, & G.-B. Huang (2005). Evolutionary extreme learning machine. *Pattern Recognition 38*(10), 1759–1763.

Zlochin, M., M. Birattari, & M. Dorigo (2004). Towards a theory of practice in metaheuristics design. A machine learning perspective. Technical Report MCS04-01, Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel.