Universidade Federal do Rio de Janeiro

Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL

Ramiro Ruiz Cárdenas

2007

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL

Ramiro Ruiz Cárdenas

Tese de Doutorado submetida ao Programa de Pós-graduação em Estatística do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências Estatísticas.

Orientadores: Alexandra M. Schmidt
Marco A. R. Ferreira

Rio de Janeiro, Julho de 2007.

Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL

Ramiro Ruiz Cárdenas

Orientadores: Alexandra M. Schmidt

Marco A. R. Ferreira

Tese de Doutorado submetida ao Programa de Pós-graduação em Estatística do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências Estatísticas.

Presidente, Prof^a. Alexandra M. Schmidt IM-UFRJ	Prof. Marco A. R. Ferreira University of Missouri
Prof^a. Marina S. Paez IM-UFRJ	Prof. Helio S. Migon IM-UFRJ
Prof. Ronaldo Dias IMECC-UNICAMP	Prof. Ricardo S. Ehlers DEST-UFPR

Rio de Janeiro, Julho de 2007.

Cárdenas, Ramiro Ruiz

Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL/ Ramiro Ruiz Cárdenas – Rio de Janeiro:UFRJ/IM, 2007.

vi, 133f.: il, color.; 31cm.

Orientadores: Alexandra Mello Schmidt e Marco Antonio Rosa Ferreira

Tese (Doutorado em Estatística) – UFRJ/IM/Programa de Pós-graduação em Estatística, 2007.

Referências Bibliográficas: f. 99 – 116.

1. Teoria da decisão. 2. Monte Carlo populacional. 3. Amostragem espacial. I. Schmidt, Alexandra M. e Ferreira, Marco A. R. (Orient.). II. Universidade Federal do Rio de Janeiro, Instituto de Matemática. III. Título.

RESUMO

Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL

Ramiro Ruiz Cárdenas

Orientadores: Alexandra M. Schmidt

Marco A. R. Ferreira

Resumo da Tese de Doutorado submetida ao Programa de Pós-graduação em Estatística do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências Estatísticas.

Neste trabalho são propostos alguns algoritmos para a otimização de funções objetivo em espaços de alta dimensão, usando uma abordagem baseada em algoritmos de busca estocásticos e métodos de Monte Carlo populacionais, com ênfase em problemas de redesenho de grandes redes de monitoramento ambiental e mapeamento de QTL.

Inicialmente foi abordado o problema de redução de grandes redes de monitoramento já existentes, quando o critério de otimalidade é baseado nos conceitos de entropia e informação. Um novo algoritmo genético híbrido é proposto para esse problema de otimização, o qual combina o algoritmo genético padrão com um operador de busca local. O novo algoritmo foi comparado com outros algoritmos de busca estocásticos (o *simulated annealing* e o algoritmo genético padrão), mediante simulações, assim como numa aplicação real. Um melhor desempenho do novo algoritmo, em termos de qualidade das soluções, foi verificado para todos os casos considerados.

Posteriormente, baseado num contexto de teoria da decisão, o problema de redesenho de redes foi abordado desde uma perspectiva de simulação. Um novo algoritmo foi proposto para a maximização

de utilidades esperadas que permite fazer uma melhor exploração do espaço dos desenhos. O novo algoritmo combina elementos de um conhecido algoritmo baseado em simulação (Müller et al., 2004), com um algoritmo de Monte Carlo populacional que usa operadores genéticos para atualizar uma população de N cadeias de Markov rodando em paralelo. O algoritmo foi implementando numa aplicação real envolvendo a redução de uma rede de monitoramento de ozônio em larga escala, onde a função de utilidade a ser maximizada engloba vários objetivos conflitantes. O algoritmo proposto superou o desempenho de uma abordagem prévia, baseada numa única cadeia de Markov, e que usa *simulated annealing* como ferramenta de maximização da utilidade esperada.

Finalmente, num outro contexto, o enfoque de teoria da decisão é usado para abordar o problema de mapeamento de QTL (*quantitative trait loci*), isto é, a identificação de regiões no genoma de um oganismo que estão diretamente associadas a alguma característica quantitativa de interesse. Geralmente tanto o número, como a localização, desses QTL é desconhecida, levando a um problema de dimensão variável. Uma metodologia é proposta para a estimação do número e localização desses QTL, via a minimização de uma função perda, onde o processo de minimização é feito usando um algoritmo evolucionário desenhado para tal fim.

Palavras-chave : teoria da decisão; amostragem espacial; *simulated annealing*; algoritmos genéticos; Monte Carlo populacional; *quantitative trait loci*.

ABSTRACT

On Stochastic Optimization Algorithms: Applications in Monitoring Network Redesign and QTL mapping

Ramiro Ruiz Cárdenas

Advisors: Alexandra M. Schmidt

Marco A. R. Ferreira

Abstract da Tese de Doutorado submetida ao Programa de Pós-graduação em Estatística do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências Estatísticas.

In this thesis some algorithms are proposed for the optimization of objective functions in high dimensional spaces, using an approach based on stochastic search algorithms and population Monte Carlo methods. The focus is on the redesign of large environmental monitoring networks and QTL mapping problems.

Initially, the shrinkage of a large monitoring network already existing was approached, when the optimality criterion is based on entropy and information concepts. A new hybrid genetic algorithm is proposed for this optimization problem, which combines the standard genetic algorithm with a local search operator. The new algorithm was compared with other stochastic search algorithms (simulated annealing and the standard genetic algorithm), through simulations, as well as with a real application. A better performance was verified for the new algorithm in terms of quality of the solutions obtained, for all the cases considered.

Based on a decision theory framework, the redesign problem was afterward approached from a simulation perspective. A new algorithm for expected utility maximization was proposed, which is

able to do a better exploration of the search space. The new algorithm combines elements from a well known simulation based algorithm (Müller et al., 2004), with a population Monte Carlo algorithm, where genetic operators are used to update a population of N Markov chains running in parallel. The algorithm was implemented in a real application involving the reduction of a large-scale ground-level ozone monitoring network, where the utility function to be maximized encodes several competing aims. The proposed algorithm overcome a previous approach based on the simulation of one Markov chain and using the simulated annealing algorithm as the expected utility maximization tool.

Finally, in another context, the decision theory framework was used to approach the QTL (*quantitative trait loci*) mapping problem, i.e., the identification of regions in the genome of an organism which are directly associated with some quantitative trait of interest. In general, both, the number and the locations of these QTL are unknown, leading to a variable dimension problem. One methodology is proposed for the estimation of the locations and the number of these QTL, via the minimization of a loss function, where the minimization process is governed by a genetic algorithm.

Key-words : decision theory; spatial sampling; simulated annealing; genetic algorithms; population Monte Carlo; quantitative trait loci.

AGRADECIMENTOS

O autor expressa seus mais sinceros agradecimentos às seguintes pessoas e entidades por sua valiosa contribuição para a realização deste trabalho:

- Marco A. R. Ferreira
- Alexandra M. Schmidt
- Helio S. Migon
- Dani Gamerman
- Marina Silva Paez
- Roseli Aparecida Leandro
- Edna Afonso Reis
- Alunos do PPG em Estatística da UFRJ, em especial a Romy, Ralph, Esther e Juan.
- CAPES
- FAPERJ

Sumário

Lista de Tabelas **iv**

Lista de Figuras **v**

Capítulo 1: Introdução **1**

Capítulo 2: Revisão de Literatura **4**

2.1 O Algoritmo *simulated annealing* 5

2.1.1 Descrição do algoritmo 6

2.1.2 Esquemas de temperatura 7

2.1.3 A estrutura de vizinhança 10

2.1.4 Extensões 11

2.2 Algoritmos Genéticos 12

2.2.1 Seleção 14

2.2.2 Cruzamento 15

2.2.3 Mutação 16

2.2.4 Controle de parâmetros no algoritmo genético 17

2.2.5 Extensões 18

2.3 Algoritmos de Monte Carlo Populacionais 23

2.3.1 Cadeias Múltiplas Independentes 26

2.3.2 Amostragem com direção adaptativa 26

2.3.3 Amostragem por importância dinamicamente ponderada 27

2.3.4 Parallel tempering 28

2.3.5 Algoritmos de Monte Carlo Evolucionários 30

Capítulo 3: Algoritmos de Busca Estocásticos para o Desenho Ótimo de Redes de Monitoramento 35

3.1 Introdução 35

3.2 Desenhos de Amostragem Espacial Baseados em Entropia e Informação 37

3.3 Abordagens Computacionais 39

3.3.1 O Algoritmo *simulated annealing* (SA) 39

3.3.2 O Algoritmo Genético (AG) 41

3.3.3 O Algoritmo Genético Híbrido (AGH) 43

3.4 Análise de Dados 44

3.4.1 Um Estudo Baseado em Dados Artificiais 44

3.4.2 Uma Aplicação Real 51

3.5 **Considerações Finais** 54

Capítulo 4: Algoritmos Evolucionários de Monte Carlo via Cadeias de Markov para o Desenho Ótimo de Redes de Monitoramento 56

4.1 Introdução 56

4.2 O Algoritmo de Müller 58

4.3 Um Algoritmo Evolucionário de Monte Carlo via cadeias de Markov 60

4.3.1 Os operadores Genéticos 61

4.4 Uma aplicação Ambiental 65

4.4.1 O modelo espaço-tempo 66

4.4.2 A Função de Utilidade 69

4.5 Considerações Finais 76

Capítulo 5: Mapeamento de QTL **77**

5.1 Introdução 77

5.2 Conceitos básicos de genética 79

5.3 O modelo Bayesiano para QTL 80

5.3.1 Especificação das prioris 83

5.4 A função perda 84

5.5 Um algoritmo evolucionário para minimização da perda esperada 85

5.6 Aplicação a dados simulados 90

Capítulo 6: Considerações Finais **94**

Bibliografia **98**

Apêndice A: Formulação Bayesiana de Desenhos Ótimos **117**

Apêndice B: Partição da Função de Entropia **123**

Apêndice C: Seleção de Parâmetros para o Algoritmo Genético **125**

Apêndice D: Seleção de Parâmetros para o Simulated Annealing **131**

Lista de Tabelas

3.1 Resultados obtidos no estudo de simulação para os algoritmos *simulated annealing*, Algoritmo Genético e Algoritmo Genético Híbrido. 47

3.2 Resultados obtidos na aplicação real para os algoritmos *simulated annealing*, Algoritmo Genético e Algoritmo Genético Híbrido. 53

4.1 Resultados obtidos com o algoritmo de Müller e com o algoritmo *eMCMC* proposto, para os dois casos considerados na aplicação ambiental 73

5.1 Localizações estimadas dos QTL e perda esperada associada sob a abordagem de Sisson e Hurn (2004) para vários números fixos de QTL. 93

5.2 Localizações estimadas dos QTL e perda esperada associada sob a abordagem de Sisson e Hurn (2004) para diferentes valores de J. 93

C.1 Tamanhos mínimos de população N para o algoritmo genético padrão, seguindo o critério de Harik et al. (1999). 127

Lista de Figuras

3.1 Box plots para as funções objetivo obtidas no estudo de simulação. 48

3.2 Melhores desenhos de redes obtidos no estudo de simulação. 50

3.3 Melhores desenhos de redução de uma rede de monitoramento de ozônio nos Estados Unidos 54

4.1 Representação esquemática do operador de cruzamento entre pares de desenhos. . . 63

4.2 Localização das 478 estações monitoradoras no leste dos Estados Unidos. 66

4.3 Localização dos 26 pontos de suporte do processo latente usados para construir o processo Gaussiano no modelo de convolução 67

4.4 Localização dos 145 pontos de suporte usados para interpolar a superfície de resposta média da função de utilidade 70

4.5 Melhores desenhos de redes encontrados com o algoritmo de Müller e com o algoritmo *eMCMC* para o primeiro caso da aplicação ambiental 74

4.6 Melhores desenhos de redes encontrados com o algoritmo de Müller e com o algoritmo *eMCMC* para o segundo caso da aplicação ambiental 75

5.1 Estrutura usada para definir distâncias para os C cromossomos. Linhas sólidas representam os cromossomos e, linhas ponteadas representam os "spacers" associados a cada cromossomo. 85

5.2 Estimativas da intensidade a posteriori da localização dos QTL para cada um dos cromossomos simulados. As linhas vermelhas indicam as posições verdadeiras dos QTL simulados. 91

C.1 Média (a) e desvio padrão (b) da função objetivo obtidos para as melhores soluções como uma função das probabilidades de cruzamento (*pcross*) e mutação (*pmut*) no algoritmo genético padrão. 130

D.1 Média (a), desvio padrão (b) e número de vezes que o melhor valor conhecido da função objetivo foi encontrado (c) como função dos fatores de desconto (*fdesc*) e de acréscimo (*facr*) em 100 repetições do algoritmo *simulated annealing*. 133

Capítulo 1

Introdução

A otimização de funções objetivo em espaços de alta dimensão ou com muitas modas separadas por barreiras altas é um problema que ainda desafia muitas disciplinas científicas tais como biologia, química, ciências da computação, física e estatística, entre outras. Nestas situações, algoritmos de otimização convencionais, geralmente falham em encontrar boas soluções ao ficarem presos em modas locais.

Esta tese trata sobre algoritmos de otimização para problemas de alta dimensão usando uma abordagem baseada em algoritmos de busca estocásticos e métodos de Monte Carlo populacionais. Embora os algoritmos apresentados nesta tese tenham aplicações em diversas áreas, a motivação principal para a realização deste trabalho vem de dois problemas práticos da maior importância atualmente nas áreas de estatística ambiental e genética: O redesenho de grandes redes de monitoramento ambiental e o mapeamento de QTL, aquelas regiões do genoma de um organismo responsáveis pela variação em alguma característica quantitativa (contínua) de interesse.

Diversas agências de monitoramento ambiental ao redor do mundo mantém atualmente redes de monitoramento da qualidade do ar em larga escala com diferentes objetivos, tais como avaliar a eficácia de medidas de controle reguladoras, determinar níveis atuais e tendências, assim como fornecer dados de entrada para modelos de avaliação de risco, para poluentes tais como: ozônio (O_3), dióxido de enxofre (SO_2), e material particulado fino ($PM_{2.5}$), entre outros. No entanto,

estas redes precisam ser administradas de tal forma que, diante de eventuais restrições de verbas no futuro, possam ser feitas mudanças nas prioridades e necessidades da rede, tanto em nível local quanto nacional. Em situações como esta, as agências devem estar preparadas para propor reduções nas redes existentes, de modo que as redes reduzidas mantenham ainda suficiente informação espacial para assegurar uma inferência estatística razoável sobre a poluição do ar. De outro lado, os resultados do monitoramento durante um certo período podem apontar a necessidade de instalar sítios de monitoramento adicionais para certos poluentes em áreas onde o monitoramento está sendo deficiente. Nesse caso, as agências deverão considerar o aumento da rede existente, visando selecionar os sítios mais informativos para a instalação das novas estações. Portanto, há uma necessidade em fornecer abordagens estatísticas críveis para redesenhar (reduzir ou aumentar) as redes de monitoramento existentes em uma forma ótima, no sentido de encontrar o conjunto mais informativo de sítios de monitoramento, capazes de satisfazer os múltiplos objetivos dos programas de monitoramento atuais.

Organização da tese

Esta tese consiste de cinco capítulos, um de revisão de literatura, três com os principais resultados de pesquisa, e um de considerações finais, os quais são brevemente descritos a seguir:

No capítulo 2 é feita uma revisão sobre dois dos principais e mais bem sucedidos algoritmos de busca estocásticos registrados na literatura, o algoritmo *simulated annealing* e o algoritmo genético, os quais constituem a base para abordar o problema de redesenho de redes de monitoramento no capítulo 3. Igualmente, é feita uma revisão sobre métodos de Monte Carlo que operam sobre uma população de soluções, com destaque para os algoritmos evolucionários de Monte Carlo via cadeias de Markov, os quais usam operadores de algoritmos genéticos para permitir a troca de informações entre as diferentes cadeias, a medida que o algoritmo converge. Um novo algoritmo desta classe é proposto no capítulo 4.

O capítulo 3 trata sobre algoritmos de busca estocásticos para o desenho ótimo de grandes redes de monitoramento. Este trabalho surgiu das dificuldades existentes para implementar metodologias já existentes para o desenho de redes baseadas em critérios bem conhecidos tais como entropia e

informação, no caso de redes de monitoramento de larga escala. Para tentar superar este problema, foi usada uma abordagem baseada em algoritmos de busca estocásticos, computacionalmente intensivos. Foram desenvolvidos algoritmos adequados para o desenho de redes de monitoramento baseados no algoritmo *simulated annealing* e no algoritmo genético. Igualmente foi proposto um novo algoritmo genético híbrido para o desenho de redes, combinando o algoritmo genético padrão com um operador de busca local. Os resultados obtidos mostraram que o algoritmo genético híbrido proposto superou o desempenho dos outros dois algoritmos em termos de qualidade das soluções para todos os casos considerados.

No capítulo 4, baseado num contexto de teoria da decisão, o problema de redesenho de redes é abordado desde uma perspectiva de simulação. Um novo algoritmo é proposto para a maximização de utilidades esperadas. O novo algoritmo combina elementos de um conhecido algoritmo baseado em simulação (Müller et al., 2004), com algoritmos evolucionários, onde a simulação é feita usando métodos de Monte Carlo via Cadeias de Markov. Os resultados obtidos até agora mostram a adequação do algoritmo em encontrar melhores desenhos de redes, devido à sua capacidade para maximizar a utilidade esperada fazendo uma melhor exploração do espaço dos desenhos.

O capítulo 5 considera o problema de mapeamento de QTL (*quantitative trait loci*), i.e., a identificação de regiões no genoma de um indivíduo que estão diretamente associadas a alguma característica quantitativa de interesse. Geralmente tanto o número, como a localização, desses QTL é desconhecida, levando a um problema de dimensão variável. Seguindo o enfoque de teoria da decisão iniciado no capítulo 4, é proposta uma metodologia para a estimação do número e localização desses QTL, baseada na minimização de uma função perda. O processo de minimização é feito usando um algoritmo evolucionário desenhado para tal fim.

Finalmente no capítulo 6, são brevemente descritas algumas das possíveis extensões do trabalho, assim como potenciais futuros tópicos de pesquisa.

Capítulo 2

Revisão de Literatura

A otimização de funções objetivo de interesse em espaços de alta dimensão, tem sido uma área de pesquisa crescente nos últimos anos. Se S denota o conjunto de todas as possíveis soluções de um problema, o objetivo é maximizar ou minimizar a função $f : S \to \mathbb{R}$. No caso da maximização, o interesse é encontrar uma configuração $x_{opt} \in S$ que satisfaça

$$f(x_{opt}) \geq f(x) \qquad \text{para todo} \quad x \in S.$$

É fácil ver que, a medida que a dimensão de S aumenta, esta tarefa fica mais complicada, e mais tempo será necessário para achar uma solução ótima ou perto do ótimo global. Outra dificuldade associada a problemas de alta dimensão, é que a função objetivo pode ter muitos ótimos locais, e os algoritmos de otimização utilizados deverão ter a capacidade de escapar destes ótimos locais.

Uma das abordagens mais estudadas recentemente para este tipo de problemas é a *otimização estocástica*, na qual a busca pela solução ótima envolve aleatoriedade em algum sentido. Neste capítulo é feita uma revisão de dois dos principais e mais bem sucedidos algoritmos de busca estocásticos registrados na literatura, o algoritmo *simulated annealing* (SA) e o algoritmo genético (AG). Aqui são descritas suas principais características e algumas das mais recentes extensões que têm sido propostas na literatura para aumentar o poder destes algoritmos. Igualmente, é feita uma revisão sobre métodos de Monte Carlo que operam sobre uma população de soluções, outra recente abordagem para simulação e otimização em espaços de alta dimensão. Uma referência particular

é feita sobre os algoritmos evolucionários de Monte Carlo via cadeias de Markov, os quais usam operadores de algoritmos genéticos para permitir a troca de informações entre as diferentes cadeias a medida que o algoritmo converge. O objetivo desta revisão é fazer uma introdução aos algoritmos de otimização que serão considerados nos capítulos seguintes desta tese, e não fazer uma completa enumeração de todos os trabalhos que têm sido publicados na área. Referências bibliográficas úteis são dadas ao longo das diferentes seções deste capítulo, para leitores interessados em se aprofundar mais neste assunto. Revisões recentes sobre algoritmos de busca estocásticos, incluindo SA e AG, com aplicações em estatística, podem ser encontrados em Chatterjee et al. (1996) e Fouskakis e Draper (2002). Outra revisão interessante, focada para problemas de otimização combinatória, é dada em Blum e Roli (2003).

2.1 O Algoritmo simulated annealing

O algoritmo *simulated annealing* pode ser definido como uma técnica de busca local estocástica que aproxima o máximo (ou o mínimo) de uma função objetivo $f : S \to \mathbb{R}$, sobre um conjunto finito S. O algoritmo foi introduzido na literatura por Kirkpatrick et al. (1983), baseado nas idéias de Metropolis et al. (1953) sobre simulação de um sistema de partículas ao experimentar mudanças em temperatura. Sob perturbação, o sistema tenta encontrar um ponto de equilíbrio que minimize a energia total. O termo *annealing* em termodinâmica se refere ao esfriamento de materiais sob condições controladas. Kirkpatrick et al. (1983) fizeram uma analogia entre os estados do sistema no problema de Metropolis e as possíveis configurações num problema de otimização mais geral, com os valores da função objetivo fazendo o papel dos níveis de energia, e a temperatura do sistema correspondendo a um parâmetro de controle no processo de otimização.

Este algoritmo tem sido implementado com sucesso em muitas aplicações (ver por exemplo van Laarhoven e Aarts, 1987). Particularmente em estatística, tem surgido recentemente diversas aplicações para este algoritmo em áreas como: seleção de modelos (Brooks et al., 2003), restauração de imagens (Gluhovsky, 2004), detecção de clusters espaciais (Duckzmal e Assunção, 2004), estimação de parâmetros em modelos Markovianos escondidos (Andrieu e Doucet, 2000), amostragem

espacial (Ferri e Piccioni, 1992; Zhu e Stein, 2005) e estimação de parâmetros em estruturas de covariância espaciais (Pardo-Igúzquiza, 1998b; Iovleff e Perrin, 2004), entre outras.

Ingber (1994) apresenta uma extensa revisão do algoritmo SA dirigida a potenciais usuários, descrevendo uma grande gama de áreas de aplicação do algoritmo, e apontando seus pontos fortes e fracos, assim como suas principais extensões, incluindo versões adaptativas e implementação em paralelo. Outra revisão interessante, descrevendo as origens e a evolução deste algoritmo, é dada em Fleischer (1995)

2.1.1 Descrição do algoritmo

O processo de otimização no algoritmo SA começa com uma configuração inicial x_0, escolhida aleatoriamente do conjunto S de todas as possíveis configurações. Esta configuração inicial também pode ser especificada pelo usuário, dependendo da informação disponível sobre o problema. A cada iteração o algoritmo opera escolhendo uma configuração x' na vizinhança $\partial(x)$ da configuração corrente x. A configuração candidata x' poderá ser aceita ou não como a nova configuração no processo de otimização. Seguindo a notação de Fouskakis e Draper (2002), seja $p(x, x', T)$ a probabilidade de aceitação de uma configuração candidata x', obtida a partir da configuração corrente x. Esta probabilidade é controlada por um parâmetro, T, coloquialmente chamado de *temperatura* em analogia com o processo físico de esfriamento descrito anteriormente. Uma escolha comum para $p(x, x', T)$ é a probabilidade de aceitação de Metropolis. Para o caso de maximização da função objetivo f, esta probabilidade é dada por

$$p(x, x', T) = \begin{cases} 1, & \text{se } f(x') \geq f(x), \\ \exp\left[\dfrac{f(x') - f(x)}{T}\right], & \text{se } f(x') < f(x). \end{cases}$$

Isto é, propostas melhores serão sempre aceitas, mas, propostas que levarem a uma configuração pior também poderão ser aceitas, com uma certa probabilidade que depende do parâmetro de controle T. Isto dá ao algoritmo a chance de escapar de ótimos locais. A altas temperaturas o sistema aceita propostas quase de forma aleatória. A temperatura é diminuída a cada certo número de iterações, fazendo com que a probabilidade de aceitar propostas piores também decresça. Eventualmente o

sistema ficará "congelado" num estado local ou globalmente ótimo, não aceitando mais propostas a partir daquele ponto. A taxa à qual o parâmetro T vai decrescendo ao longo das iterações, conhecido como "esquema de esfriamento", é de vital importância para o desempenho do algoritmo.

O conjunto de configurações propostas, geradas pelo algoritmo durante as iterações em que permanece com a mesma temperatura, constitui uma cadeia de Markov. Assim, o algoritmo *simulated annealing* consiste de uma série de cadeias de Markov homogêneas a diferentes temperaturas decrescentes, onde em cada estado da cadeia, as transições são geradas de acordo com o critério de Metropolis. O objetivo de cada cadeia de Markov é permitir que o sistema restaure o equilíbrio térmico, perdido toda vez que ocorre uma diminuição da temperatura.

De acordo com van Laarhoven e Aarts (1987), o algoritmo *simulated annealing* converge com probabilidade um para uma configuração globalmente ótima quando, para cada valor do parâmetro de controle, T_k, é gerado um número infinito de transições (iterações) e $lim_{k\to\infty}T_k = 0$. Uma prova formal da convergência assintótica do SA para o ótimo global sob diferentes condições, pode ser encontrada, por exemplo, em Bélisle (1992) e Locatelli (1996). No entanto, em qualquer implementação do algoritmo, a convergência assintótica unicamente poderá ser aproximada, já que, na prática, o número de transições geradas para cada valor de T_k é finito.

2.1.2 Esquemas de temperatura

A evolução da temperatura durante o processo de otimização é chamado de *esquema de esfriamento* ou *esquema de temperatura*. Desde a aparição do artigo de Kirkpatrick et al. (1983), vários esquemas de esfriamento tem sido propostos na literatura. Para uma análise detalhada e comparação de vários esquemas de esfriamento para *simulated annealing* veja, por exemplo, Nourani e Andresen (1998) e Cohn e Fielding (1999).

Um esquema de esfriamento consiste de quatro elementos:

1. uma temperatura inicial;

2. uma regra para a diminuição da temperatura;

3. uma regra para o comprimento da cadeia de Markov, isto é, quantas iterações o algoritmo realizará com uma mesma temperatura;

4. uma condição de parada.

Como o equilíbrio térmico não pode ser alcançado em um tempo finito, o conceito de quase-equilíbrio tem sido usado para guiar os esquemas de esfriamento. Inicialmente um estado de quase-equilíbrio pode ser assumido escolhendo um valor inicial de temperatura alto o suficiente para que virtualmente todas as transições sejam aceitas de acordo com o critério de Metropolis, i.e., T_0 é tal que $\exp[-((f(x')-f(x))/T_0) \simeq 1]$ para quase todo x e x'. Nesta situação todas as configurações existem com igual probabilidade de aceitação. Kirkpatrick et al. (1983) propõem calcular a temperatura inicial, T_0, de tal forma que a taxa de aceitação do algoritmo seja aproximadamente igual a um valor dado, χ_0. Para isto, primeiro é escolhida uma temperatura inicial grande e são rodadas um certo número de iterações do algoritmo usando essa temperatura. Se a proporção de propostas aceitas for menor que χ_0, então a temperatura é multiplicada por 2, e o procedimento continua até que essa proporção exceda χ_0. Vários outros métodos têm sido propostos na literatura para calcular a temperatura inicial do algoritmo SA. Para uma revisão completa sobre este assunto veja Ben-Ameur (2004).

O número de transições avaliadas em qualquer temperatura específica, T_k, corresponde ao comprimento, L_k, da k-ésima cadeia de Markov. O comprimento da cadeia é governado pela noção de *proximidade* entre a distribuição de probabilidade da configuração corrente p_{L_k,T_k} , e a distribuição estacionária da função objetivo q_{T_k} a uma temperatura específica. Vários esquemas de esfriamento exploram este conceito, e diferentes abordagens têm sido propostas para determinar quando esta proximidade é atingida. Por exemplo, Kirkpatrick et al. (1983) propõem um critério baseado no argumento intuitivo de que para cada valor T_k do parâmetro de controle, uma quantidade mínima de transições deveria ser aceita, isto é, L_k é determinado de tal forma que o número de transições aceitas seja no mínimo η_{min}, onde η_{min} é algum número previamente fixado. Usualmente é determinada uma cota superior, L, para o comprimento da cadeia, L_k, para evitar cadeias de Markov muito longas quando os valores de T_k são muito baixos.

Para simular as condições de convergência do algoritmo, a temperatura deve se aproximar do valor zero a medida que o algoritmo avança. Na prática, o valor da temperatura é reduzido para valores suficientemente pequenos, tais que virtualmente não sejam aceitas configurações piores de acordo com o critério de Metropolis. Essa redução da temperatura também está intimamente relacionada à noção de quase-equilíbrio da distribuição de probabilidade das configurações. Assim, reduções grandes da temperatura T_k para a k-ésima cadeia de Markov, precisarão de um número grande de transições antes que o estado de quase-equilíbrio seja restaurado. Portanto, deve haver um balanço entre a taxa de redução da temperatura e o comprimento das cadeias de Markov. A maioria dos esquemas de esfriamento adotam a estratégia de pequenas reduções na temperatura T_k para evitar cadeias de Markov muito longas.

Um critério usual de parada do algoritmo é determinado fixando o número de valores T_k, para os quais o algoritmo será executado, ou terminando a execução do algoritmo se as últimas configurações de cadeias de Markov consecutivas são idênticas para um determinado número de cadeias (Kirkpatrick et al., 1983). Adicionalmente pode ser requerido que a taxa de aceitação seja menor que um valor dado χ_f.

Dependendo da forma como a temperatura é diminuída, os esquemas de esfriamento para o algoritmo SA podem ser classificados como esquemas fixos ou esquemas adaptativos. No primeiro caso, os esquemas são determinados antes do início do algoritmo e não são influenciados durante o progresso do algoritmo (isto é, são independentes do valor da função objetivo das configurações). Um dos exemplos mais populares deste tipo de esquemas é o denominado esquema *geométrico*, onde a temperatura é mantida fixa durante um ciclo de R iterações, e ao final de cada ciclo, é diminuída de acordo com a regra

$$T_{k+1} = \alpha T_k,$$

onde α é uma constante escolhida no intervalo (0,1) antes de começar o algoritmo. Valores de α próximos de 1 usualmente apresentam um melhor desempenho. Segundo Fouskakis e Draper (2002), a constante α pode também ser definida em termos dos limites superior, T_0, e inferior, T_f, do parâmetro de controle T, e do número total de iterações, M de acordo com a expressão

$$\alpha = \left(\frac{T_f}{T_0}\right)^{M^{-1}}. \tag{2.1}$$

Segundo Cohn e Fielding (1999), sob este esquema, o número de iterações que o SA roda em cada temperatura, tem pouco ou nenhum efeito no desempenho do algoritmo, se o valor de α for ajustado apropriadamente.

Lundy e Mees (1986) propuseram um esquema em que a temperatura é diminuída lentamente a cada iteração do SA de acordo com a regra

$$T_{k+1} = \frac{T_k}{1 + \beta T_k},$$

onde β é um valor pequeno que pode ser escolhido de forma análoga à equação 2.1.

Muitos outros esquemas de esfriamento fixos têm sido propostos na literatura. Para uma referência mais detalhada sobre esquemas deste tipo veja por exemplo van Laarhoven e Aarts (1987) e Cohn e Fielding (1999).

Por outro lado, nos esquemas adaptativos, a temperatura é ajustada dinamicamente a medida que o algoritmo avança, com base no perfil da busca. A magnitude do ajuste pode variar ao longo do algoritmo, e pode ser em qualquer direção, incluindo a possibilidade de *reaquecimento*, isto é, aumento da temperatura se necessário. Recentes estratégias adaptativas de esfriamento para SA podem ser encontradas em Ferri e Piccioni (1992), Poupaert e Deville (2000), Azizi e Zolfaghari (2001) e Atiqullah (2004).

2.1.3 A estrutura de vizinhança

Na sua forma original, o algoritmo SA assume que a estrutura de vizinhança associada a cada configuração é bem definida e não muda ao longo do algoritmo. No entanto, em alguns casos, é possível obter um melhor desempenho do algoritmo quando a estrutura de vizinhança é ajustada a medida que a temperatura decresce. Uma forma de fazer isto é colocando restrições sobre o tamanho da vizinhança.

A idéia central é que a baixas temperaturas as propostas aceitas serão geradas a partir de perturbações cada vez menores da configuração corrente. Isto é devido a que nesta situação, o algoritmo

esteja perto de atingir a sua convergência e, provavelmente, já esteja em regiões interessantes do espaço de busca, precisando refinar cada vez mais a busca, o que é logrado com movimentos cada vez mais locais. Assim, para evitar tempo perdido propondo e rejeitando movimentos grandes, é criado um limite para o tamanho máximo da vizinhança, e este limite é forçado a decrescer a medida que o sistema vai esfriando. Uma abordagem neste sentido foi implementada por van Groenigen e Stein (1998) para otimizar esquemas de amostragem espacial, onde inicialmente foi selecionada uma vizinhança de comprimento igual à região de amostragem inteira, a qual ia decrescendo ao longo das iterações.

2.1.4 Extensões

Duas das principais extensões do algoritmo SA simples que têm surgido recentemente na literatura são a paralelização do algoritmo, e a sua combinação com outros algoritmos de busca. A seguir estas duas abordagens são brevemente descritas.

2.1.4.1 simulated annealing em paralelo

Uma desvantagem do algoritmo SA quando a dimensão do problema aumenta, é a grande quantidade de tempo computacional necessário para obter soluções perto do ótimo global. Para tentar aliviar este problema, várias tentativas de paralelização do algoritmo SA têm sido propostas na literatura. Para uma revisão sobre esquemas de paralelização em SA veja por exemplo Ingber (1994) e Onbaşoğlu e Özdamar (2001). A idéia principal é distribuir a execução das várias partes do algoritmo SA seqüencial em um número de processadores rodando em paralelo.

Por exemplo, Diekmann et al. (1995), baseados na observação de que no algoritmo SA a maioria das propostas são rejeitadas, e assumindo independência entre as propostas rejeitadas, propõem usar vários processadores trabalhando simultaneamente na avaliação de uma mesma cadeia de Markov.

Knopman e Aude (1996) propõem uma técnica adaptativa para a paralelização do algoritmo SA, a qual usa abordagens de paralelização diferentes para valores de temperatura altos e baixos durante o processo de annealing. No algoritmo usado para baixas temperaturas, o número de processadores alocados à solução do problema e o número de movidas avaliadas por processador mudam com

a temperatura. O algoritmo a altas temperaturas está baseado na avaliação paralela de cadeias independentes.

2.1.4.2 Hibridização do simulated annealing

A combinação do algoritmo SA com outros algoritmos de busca estocásticos, tem gerado algoritmos híbridos, que visam melhorar o desempenho do SA simples, incorporando características desejáveis de outros métodos. Um dos mais populares é o formado pela combinação do *simulated annealing* com o algoritmo genético, que permite introduzir o conceito de população dos algoritmos evolucionários dentro do SA, com o qual é possível injetar mais diversidade ao algoritmo. A hibridização pode ocorrer de várias formas, por exemplo Miki et al. (2003) usam um algoritmo genético para gerar as diferentes temperaturas do SA. Outros autores têm usado o SA para guiar o operador de seleção do algoritmo genético (Yip e Pao, 1995). Uma revisão interessante sobre este tipo de híbridos é dada em Mahfoud e Goldberg (1995).

Analogamente ao caso do algoritmo SA simples, estes algoritmos híbridos também suportam versões com esquemas adaptativos de controle da temperatura (Cho et al., 1998), e implementação em paralelo (Calaor et al., 2002), assim como combinações destas duas abordagens (Miki et al., 2003).

2.2 Algoritmos Genéticos

Desde sua introdução na literatura por Holland et al. (1975), os algoritmos genéticos têm se tornado um dos métodos mais populares para resolver grandes problemas de otimização. Referências clássicas sobre este algoritmo, além do próprio Holland, incluem os livros de Goldberg (1989) e Davis (1991). Aplicações em estatística mais recentes podem ser encontradas em Chatterjee et al. (1996).

Algoritmos genéticos são algoritmos de busca estocásticos baseados em princípios de evolução natural. Eles tentam encontrar uma solução ótima para um problema em particular, manipulando uma população de soluções candidatas, permitindo às soluções se reproduzirem, criando novas soluções que competirão pela sobrevivência na próxima geração. Cada geração pode ter membros que não melhoram o desempenho dos membros da geração prévia mas, em média, o desempenho das soluções,

medido por uma medida de adaptabilidade ("*fitness*"), é tipicamente melhor ao longo das gerações e a melhor solução após muitas gerações é usualmente perto do ótimo global. Uma estrutura formal para o algoritmo genético tem três componentes: (i) o ambiente e os elementos do ambiente (as soluções candidatas); (ii) um plano adaptativo (a aplicação de operadores genéticos), e (iii) seleção baseada numa medida de desempenho (o *fitness* das soluções).

O mecanismo básico nos algoritmos genéticos é a evolução Darwiniana, segundo a qual características "ruins" tendem a ser eliminadas da população porque elas aparecem em indivíduos que não sobrevivem ao processo de seleção. As características "boas" de um indivíduo sobrevivem e são recombinadas para formar melhores indivíduos. A noção de características "boas", é formalizada em algoritmos genéticos com o conceito de blocos construtores, os quais são seqüências binárias curtas, presentes numa pequena porção dos indivíduos e que conferem um alto valor de *fitness* aos indivíduos que as possuem. Segundo a teoria relacionada, os algoritmos genéticos funcionam propagando esses blocos construtores através dos operadores de seleção e cruzamento.

No algoritmo genético padrão, os indivíduos da população são compostos de cadeias binárias de comprimento fixo, l. Assim o Algoritmo explora um espaço de busca formado por 2^l pontos. Inicialmente, a população consiste de pontos escolhidos aleatoriamente, a menos que seja utilizado algum outro método mais simples para gerar boas soluções iniciais mas, mesmo neste caso, uma parte da população é ainda gerada de forma aleatória para assegurar que exista alguma diversidade nas soluções.

Cada indivíduo tem um valor de ajuste (*fitness*) associado de acordo com a qualidade da solução. Em geral o *fitness* é uma medida de remuneração que depende de quão bem cada indivíduo da população resolve o problema de interesse. Em otimização, o *fitness* de um indivíduo é o valor da sua função objetivo nos pontos representados pela cadeia binária.

Algoritmos genéticos simples usam dois operadores "pobremente" baseados em mecanismos genéticos de variação para explorar o espaço de busca: cruzamento e mutação. Esses dois operadores junto com um operador de seleção de indivíduos na população, permitem que a população evolua para soluções melhores ao longo das gerações. Esses operadores são descritos nas seguintes subseções.

Existem várias formas de parar um algoritmo genético. Uma delas é parar após um determinado número de gerações ou de avaliações da função objetivo. Outra forma é parar quando a qualidade média das soluções na população não melhore após um determinado número de gerações.

2.2.1 Seleção

O objetivo deste operador é dar aos membros com melhor ajuste da população corrente (i.e., aqueles com os melhores valores da função objetivo), uma chance maior de estarem presentes na população de próxima geração. Duas formas de seleção podem ser identificadas em algoritmos genéticos:

(i) seleção para variação, onde são escolhidos os indivíduos para as operações de mutação e recombinação, e

(ii) seleção para sobrevivência, onde são selecionados os indivíduos que farão parte de nova geração.

As estratégias de seleção para variação incluem entre outras as seguintes categorias:

Seleção proporcional ao ajuste : aqui a probabilidade de selecionar um indivíduo x_i é dada por:

$$p(x_i) = \frac{H(x_i)}{\sum\limits_{x_j \in P} H(x_j)}.$$

Uma desvantagem deste tipo de seleção é que a medida que a variância dos valores da função objetivo na população decresce, a amostragem chega a ser puramente aleatória.

Seleção por torneio : Aqui, antes de selecionar um indivíduo, são pre-selecionados aleatoriamente (independentemente do valor da sua função objetivo), δ indivíduos da população, para depois selecionar o indivíduo com a melhor medida de ajuste dentre esses δ indivíduos.

As estratégias de seleção para sobrevivência incluem:

Substituição generacional : é a estratégia mais simples, na qual todos os indivíduos parentais são substituídos pela sua descendência. Este método tem sido usado em combinação com seleção proporcional ao ajuste para variação, para forçar a pressão de seleção.

Seleção Steady state : Usada para decidir quais indivíduos parentais serão substituídos quando o número de novos indivíduos gerados por variação é menor que o número de indivíduos parentais. Existem diversas variantes, tais como a substituição dos piores indivíduos, ou dos mais antigos.

Seleção (μ, λ) : Os μ indivíduos parentais são substituidos pelos melhores indivíduos do conjunto de λ de novos indivíduos ($\lambda \geq \mu$). A pressão de seleção pode ser aumentada incrementando o número de novos indivíduos gerados.

Seleção ($\mu + \lambda$) : Os melhores μ indivíduos são escolhidos de uma população temporária contendo os μ indivíduos parentais e os λ novos indivíduos.

Outros métodos têm sido geralmente usados em combinação com as estratégias de seleção já descritas, tais como o *elitismo*, onde os melhores indivíduos sempre sobreviverão à próxima geração, e a *checagem de indivíduos duplicados*, onde novos indivíduos idênticos a seus parentais não são incluídos na próxima geração. Este último caso é especialmente importante quando o tamanho da população é pequeno.

Comparações entre estes esquemas de seleção em algoritmos genéticos foram feitas por Goldberg e Deb (1991), Hancock (1994) e Zhang e Cho (2001).

2.2.2 Cruzamento

Cruzamento é um mecanismo de exploração primária em algoritmos genéticos. O cruzamento permite que pares de indivíduos da população corrente combinem suas melhores características para criar soluções melhoradas para a próxima geração. O resultado do cruzamento é uma troca aleatorizada de material genético entre indivíduos, com a possibilidade de que soluções boas possam gerar soluções ainda melhores. O cruzamento acontece na população com uma taxa p_{cross}.

O mais conhecido operador de cruzamento usado na literatura é o chamado cruzamento de um ponto. Este operador seleciona dois indivíduos da população, representados por duas cadeias binárias, e as corta num ponto de corte p selecionado aleatoriamente. Posteriormente, ocorre uma permutação das sub-cadeias aleatórias entre as duas cadeias, produzindo assim dois novos indivíduos. Como um

exemplo, considere as cadeias binárias A_1 e A_2 de comprimento 8 bits:

$A_1 = 0\ 1\ 1\ 0 \mid 1\ 1\ 1\ 1$

$A_2 = 1\ 1\ 1\ 0 \mid 0\ 0\ 0\ 0$

Existem $l-1$ possíveis pontos de cruzamento em cadeias de comprimento l. Suponha que o ponto de corte escolhido aleatoriamente tenha sido na posição 4, (como indicado acima pelo símbolo $|$). A permutação das sub-cadeias ao redor do ponto de corte, resulta em duas novas cadeias disponíveis, A'_1 e A'_2, para fazer parte da próxima geração:

$A'_1 = 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0$

$A'_2 = 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1$

O operador de cruzamento pode tomar muitas formas. Além do cruzamento de um ponto, é possível usar um cruzamento de dois pontos, de n pontos, ou um cruzamento uniforme. Mais pontos de cruzamento resultam numa busca mais exploratória, mas ao mesmo tempo incrementam as chances de destruir blocos construtores longos.

2.2.3 Mutação

Este operador dá a oportunidade de visitar regiões ainda não exploradas do espaço de busca, alterando arbitrariamente um ou mais caracteres de uma cadeia selecionada. A mutação também serve para restaurar a diversidade perdida pela aplicação repetida dos operadores de seleção e cruzamento, e ajuda a prevenir convergência prematura e a assegurar que a probabilidade de alcançar qualquer ponto no espaço de busca nunca seja zero.

Geralmente, há duas formas de implementar o operador de mutação. A primeira é predefinir uma taxa de mutação p_{mut}, com a qual cada bit de cada solução binária será mutado. A outra forma é predefinir o número total de bits a serem mutados na população, e selecionar aleatoriamente as posições destes bits nas diferentes soluções.

2.2.4 Controle de parâmetros no algoritmo genético

A escolha das probabilidades de mutação, p_{cross}, e de cruzamento, p_{mut}, afeta criticamente o desempenho dos algoritmos genéticos. Muitos pesquisadores têm investido um considerável esforço tentando encontrar os melhores valores destes parâmetros. Dentre estes destacam-se os trabalhos de De Jong (1975); Grefenstette (1986) e Schaffer et al. (1989). Uma excelente revisão sobre controle de parâmetros em algoritmos evolucionários é dada em Eiben et al. (1999).

Valores típicos para p_{cross} variam no intervalo 0,6–0,95. Valores abaixo deste intevalo para este parâmetro são raramente usados na literatura. Já para o parâmetro p_{mut}, são adotados valores pequenos, tipicamente variando entre 0,001 e 0,05. Estes valores foram gerados a partir de estudos empíricos sobre um conjunto fixo de problemas teste, e podem não ser os mais adequados quando usados em outras situações.

Recentemente tem surgido alternativas para o controle de parâmetros, tais como o controle adaptativo, o qual é brevemente abordado na subseção seguinte.

2.2.4.1 Controle adaptativo de parâmetros

No algoritmo genético simples, as probabilidades de usar os operadores de cruzamento e mutação são definidas antes do início do algoritmo, e permanecem fixas durante a execução do mesmo. Outra abordagem que tem ganhado espaço nos últimos anos permite que as taxas de cruzamento e mutação mudem ao longo da implementação do algoritmo, em resposta à avaliação dos resultados obtidos na geração corrente e/ou na próxima geração.

De acordo com Eiben et al. (1999), os algoritmos evolucionários cujos parâmetros de controle não permanecem fixos ao longo do algoritmo podem ser divididos em três categorias:

Controle de parâmetros dinâmico : Consiste em um esquema determinístico de redução das taxas dos parâmetros de controle ao longo das gerações. Um dos esquemas deste tipo mais bem sucedido foi o proposto por Bäck e Schütz (1996), onde a taxa de mutação na k-ésima geração é dada pela expressão

$$p_{mut}(i) = \left(2 + \frac{l-2}{K-1}\, i\right)^{-1}$$

onde K é o número total de gerações do algoritmo, e l é o comprimento de cada solução. Outros esquemas determinísticos são dados, por exemplo, em Fogarty (1989) e Hesser e Männer (1991).

Controle de parâmetros adaptativo : Este esquema usa a informação obtida durante o processo de otimização para adaptar o valor dos parâmetros de controle. Esquemas deste tipo para controlar as taxas do operador mutação foram propostos por Thierens (2002). Esquemas recentes de controle adaptativo para o operador de cruzamento são apresentadas em Spears (1995), Yang (2002), Lin et al. (2003) e Ming et al. (2004).

Controle de parâmetros auto-adaptativo : Neste esquema, originalmente proposto por Bäck (1992), cada indivíduo tem seus parâmetros de controle codificados dentro de seu genótipo (i.e., existem bits adicionais em cada solução representando esses parâmetros), os quais vão sendo modificados a medida que o indivíduo evolui. A idéia é que indivíduos com valores bons desses parâmetros terão uma vantagem evolucionária para se auto-proliferar na população. Auto-adaptação tem sido aplicada com sucesso a problemas de otimização em espaços contínuos (veja, por exemplo: Deb e Beyer, 2001; Kita, 2001).

2.2.5 Extensões

Diversas extensões do algoritmo genético padrão têm surgido na literatura nos últimos anos. Destacam-se dentre esses algoritmos estendidos, os algoritmos genéticos guiados por modelos probabilísticos, os algoritmos que consideram simultaneamente múltiplos objetivos, e aqueles que exploram a capacidade de paralelização do algoritmo. A seguir estas três abordagens são brevemente descritas.

2.2.5.1 Algoritmos Genéticos Probabilísticos (AGP)

Também conhecidos na literatura com os nomes de *algoritmos genéticos baseados na construção de modelos probabilísticos* (PMBGA) e *algoritmos com estimação de distribuição* (EDA), foram

introduzidos na literatura por Mühlenbein e Paaß (1996). Esta classe de algoritmos está baseada na modelagem probabilística de soluções promissórias para guiar a exploração do espaço de busca ao invés de usar os operadores de cruzamento e mutação como no AG simples, cujos parâmetros de controle geralmente são de difícil sintonização. Duas importantes revisões sobre os principais modelos desta classe de algoritmos podem ser encontradas em Pelikan et al. (2002) e Yu (2002).

O modelo geral de um AGP considera os seguintes passos:

1. Inicializar uma população (usualmente de forma aleatória);

2. Selecionar os indivíduos mais promissores;

3. Estimar a distribuição daqueles indivíduos promissores (construção do modelo probabilístico);

4. Amostrar novas soluções de acordo com a distribuição em 3 e adicionar as novas soluções à população original substituindo algumas das soluções anteriores;

5. Retornar ao passo 2 até atingir um critério de parada do algoritmo.

O desempenho de um AGP está diretamente associado ao modelo probabilístico utilizado. Os diferentes tipos de AGP podem ser classificados de acordo com a complexidade do modelo probabilístico usado. No caso mais simples, é assumido que as diferentes soluções da população podem ser representadas por vetores binários de comprimento fixo e que não interagem entre eles. Em cada geração é guardado um vetor de probabilidades onde cada elemento é selecionado para ser 0,5. Após a seleção das soluções promissoras da população, esse vetor de probabilidades é modificado segundo algum modelo. Modelos mais sofisticados como os propostos por Wang e Wang (2004) e Shakya et al. (2005) propõem modelar as soluções do AGP usando campos aleatórios Markovianos.

Embora modelos mais precisos garantam uma melhor reprodução, eles usualmente são mais complicados e custosos de construir. Assim, deve existir um compromisso entre eficiência e precisão.

2.2.5.2 Algoritmos genéticos em paralelo

Existem duas razões principais para paralelizar um algoritmo evolucionário: uma é fazer economia no tempo de processamento, ao distribuir o esforço computacional e, a segunda, é se beneficiar das características inerentemente paralelas do algoritmo, em analogia com a evolução natural de populações espacialmente distribuídas. Tomassini (2001) faz uma revisão completa sobre os principais modelos de algoritmos evolucionários paralelos, discutindo seu funcionamento, assim como suas vantagens e desvantagens com respeito à arquitetura de computadores sobre a qual eles são executados. Outras interessantes revisões sobre o assunto são dadas em Cantú-Paz (1998), Alba e Troya (1999) e Nowostawski e Poli (1999).

Quatro tipos principais de algoritmos genéticos paralelos têm sido considerados na literatura, os quais são brevemente descritos a seguir:

1. Método de paralelização global *mestre-escravos*: Neste esquema é usada uma única população (como no caso do AG seqüencial), mas a avaliação da função objetivo dos indivíduos da população é distribuída entre os processadores disponíveis. A paralelização da avaliação da função objetivo não requer qualquer mudança no algoritmo genético seqüencial, já que o *fitness* de um indivíduo é independente do resto da população. Assim, sob este esquema, a comunicação entre processadores somente ocorre quando os processadores *escravos* recebem o seu subconjunto de indivíduos para serem avaliados, e quando retornam o valor da função objetivo ao processador *mestre*. Os operadores genéticos consideram a população inteira, isto é, cada indivíduo pode competir e se recombinar com qualquer outro como no caso seqüencial. Estas operações permanecem centralizadas no processador *mestre*.

2. Algoritmos genéticos paralelos com arquitetura de "grão-grosso": Também chamados algoritmos genéticos *distribuídos*, consistem em separar a população inicial em poucas sub-populações de tamanho relativamente grande, as quais evoluem separadamente a maior parte do tempo, mas que, ocasionalmente, podem fazer trocas de informação, permitindo a alguns indivíduos migrarem de uma sub-população para outra de acordo com um certo critério. A principal razão

para esta abordagem é re-injetar periodicamente diversidade às sub-populações para evitar convergência prematura em ótimos locais. Espera-se que as diferentes sub-populações tendam a explorar mais eficientemente diferentes regiões do espaço de busca. Entre fases de migração é executado um algoritmo genético seqüencial dentro de cada sub-população.

3. Algoritmos genéticos paralelos com arquitetura de "grão-fino": Consistem de uma única população, mas com uma estrutura espacial que limita as interações entre indivíduos. As operações de seleção e cruzamento para cada indivíduo, estão restritas a uma pequena vizinhança, mas existe alguma sobreposição entre as vizinhanças, permitindo a interação entre os indivíduos. Assim, boas soluções podem se disseminar através da população inteira. A escolha da estrutura de vizinhança influencia diretamente o desempenho do algoritmo. Uma análise sobre os efeitos do tamanho e forma da vizinhança sobre a pressão de seleção neste tipo de algoritmos foi feita por Sarma e De Jong (1996). Eles quantificaram como muda o tempo para propagar uma solução boa através da população inteira, usando vizinhanças de diferentes tamanhos.

4. Algoritmos genéticos paralelos hierárquicos: É possível combinar diferentes estratégias de paralelização formando uma hierarquia. Os algoritmos genéticos paralelos híbridos mais populares combinam um algoritmo distribuído (multi-população) no nível superior da hierarquia, e em cada uma das sub-populações aplicam um esquema *mestre-escravos*, mas outras combinações são possíveis.

Quando implementados em paralelo, o desempenho dos algoritmos genéticos melhora devido à exploração simultânea de diferentes regiões do espaço de busca, e à possibilidade de adiar a uniformização de uma população com a migração de indivíduos. Segundo Tomassini (2001), problemas multimodais estão entre os candidatos mais promissores para se beneficiar de uma implementação em paralelo de algoritmos evolucionários. No entanto, alguns autores apontam como uma desvantagem dos algoritmos evolucionários paralelos, a maior complexidade, quando comparados com suas versões seqüenciais, já que novos parâmetros devem ser considerados, tais como topologias de comunicação e critérios (taxas e intervalos) de migração de indivíduos entre sub-populações. Alguns critérios para

guiar a escolha racional dos parâmetros que controlam este tipo de algoritmos são apresentados em Cantú-Paz e Goldberg (2000).

2.2.5.3 Algoritmos genéticos multi-objetivo

Estes algoritmos consideram a otimização simultânea de múltiplos objetivos. Em situações como esta geralmente não existe uma solução que seja melhor com respeito a todos os objetivos mas, sim, um conjunto de soluções que são superiores ao resto das soluções no espaço de busca quando todos os objetivos são considerados, mas inferiores a outras soluções no espaço de busca, em um ou mais objetivos. Estas soluções são conhecidas como soluções Pareto-ótimas ou soluções não dominadas. Como nenhuma das soluções no conjunto de soluções não dominadas é absolutamente melhor que qualquer outra, qualquer uma delas é uma solução aceitável. A escolha de uma solução sobre as outras requer conhecimento específico do problema.

Algoritmos evolucionários são particularmente adequados para resolver problemas de otimização multi-objetivo e têm sido objeto de pesquisa nos últimos 25 anos, pois consideram simultaneamente um conjunto de possíveis soluções (a população). Isto permite encontrar vários membros do conjunto Pareto-ótimo a cada geração do algoritmo. Coello (2001) apresenta um tutorial interessante sobre os principais modelos de algoritmos genéticos multi-objetivo, apontando suas vantagens e desvantagens, assim como suas diversas áreas de aplicação. Uma revisão completa sobre este tipo de algoritmos pode ser encontrada em Zitzler (1999).

Muitos dos algoritmos evolucionários multi-objetivo disponíveis na literatura foram desenhados e/ou testados em problemas de baixa dimensão ($\leq$ 3 objetivos). No entanto, problemas com um alto número de objetivos apresentam dificuldades adicionais em termos da qualidade das soluções do conjunto de Pareto e do tempo de processamento do algoritmo. Como uma alternativa a este problema, Brockhoff e Zitzler (2006) apresentam uma metodologia para redução de dimensionalidade em AG multi-objetivo e discutem as condições sob as quais a redução de objetivos é adequada, e a forma de calcular o conjunto mínimo de objetivos a ser considerado.

2.3 Algoritmos de Monte Carlo Populacionais

Otimização e amostragem são processos de importância fundamental em muitas áreas científicas. Os dois estão intimamente ligados e compartilham problemas similares. A busca por áreas de interesse (alta probabilidade ou valores baixos de uma função de custo) é difícil na presença de uma função objetivo complexa ou de alta dimensão. Em otimização isto significa que, frequentemente, somente mínimos locais serão encontrados. Em amostragem isto implica que muitos passos serão requeridos para se obter uma amostra representativa.

O processo de amostragem consiste em gerar vetores de estado de uma densidade de interesse $\pi(x)$, definida para os estados $x \in \Omega$, conhecida a menos de uma constante multiplicadora. Um processo de otimização relacionado consiste em encontrar $x \in \Omega$ que minimize $H(x) = -\log \pi(x)$ (em física estatística, $-\log \pi(x)$ é conhecida como a função de energia da distribuição, e um mínimo local na superfície definida por $-\log \pi(x)$ é chamado de mínimo de energia local de $\pi(x)$), onde

$$\pi(x) = \frac{1}{Z} \exp(-H(x)),$$

com, Z, sendo uma constante que assegura que $\pi(x)$ integre 1 em Ω. Acrescentando um parâmetro de controle T (*temperatura*), tem-se a densidade de Boltzmann:

$$\pi^{(T)}(x) = \frac{1}{Z(T)} \exp\left(\frac{-H(x)}{T}\right). \tag{2.2}$$

A densidade de interesse, $\pi(x)$, é recuperada a partir de (2.2), quando $T = 1$. Fazendo T tender para zero, é obtida uma função que tem probabilidade diferente de zero somente no ótimo global de $H(.)$. Quando T assume valores grandes, os gradientes de probabilidade na densidade decrescem. Estas propriedades da densidade de Boltzmann são exploradas pelo algoritmo de otimização *simulated annealing*, descrito na seção 2.1, e pelo algoritmo *parallel tempering*, descrito na subseção 2.3.4.

Muitos métodos de amostragem usam uma densidade proposta $g(y)$ para selecionar regiões do espaço Ω que presumivelmente tenham altos valores para a densidade de interesse $\pi(x)$. A densidade de interesse é avaliada na localização proposta, e alguma regra é usada para aceitar, rejeitar ou ponderar a proposta de tal forma que as amostras geradas conformem para $\pi(x)$. Um dos métodos

mais populares é o método Monte Carlo via cadeias de Markov (MCMC), o qual gera novas propostas y usando o estado corrente x de uma "cadeia" de amostragem, de acordo com uma densidade condicional $g(y \mid x)$. A proposta y será aceita seguindo a regra de Metropolis-Hastings (Hastings, 1970), com probabilidade

$$p = \min\left\{1, \frac{\pi(y)g(x \mid y)}{\pi(x)g(y \mid x)}\right\}.$$

Se a proposta y é aceita, o novo estado da cadeia será y. Caso contrário, o estado x é retido até o próximo passo do algoritmo. Suponha que g é uma função da distância euclidiana entre x e y:

$$g(x \mid y) = f(\|y - x\|),$$

onde f é alguma densidade de probabilidade. Então, a probabilidade de aceitação é dada por:

$$p = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\} = \min\{1, \exp[H(x) - H(y)]\}.$$

Esta regra de aceitação simplificada é o método de Metropolis (Metropolis et al., 1953).

O algoritmo de Metropolis-Hastings (M-H) vem tendo uma importância fundamental em computação Monte Carlo pela sua capacidade para gerar amostras dependentes de uma cadeia de Markov com $\pi(.)$ como sua distribuição de equilíbrio. No entanto, em muitas aplicações, o algoritmo M-H pode ter um desempenho pobre, principalmente quando a função objetivo é de alta dimensão ou tem muitas modas locais separadas por barreiras altas. Em tais situações o algoritmo pode ficar facilmente preso em uma moda local.

Nas últimas duas décadas várias técnicas têm sido propostas para melhorar a capacidade de mistura do algoritmo M-H. Esses algoritmos são baseados em duas idéias principais:

A primeira idéia é o uso de variáveis auxiliares. Incluídos nesta categoria estão os algoritmos *simulated tempering* (Marinari e Parisi, 1992; Geyer e Thompson, 1995), *parallel tempering* (Geyer, 1991; Hukushima e Nemoto, 1996), *simulated sintering* (Liu et al., 2000), etc. Nestes algoritmos a temperatura é típicamente tratada como uma variável auxiliar. As simulações a altas temperaturas estendem a amostragem do espaço amostral, ajudando, assim, o sistema a escapar de mínimos de

energia local.

A segunda idéia é o uso de amostras passadas. O algoritmo Multicanônico (Berg e Neuhaus, 1991), aparentemente, é o primeiro trabalho nesta direção. Este algoritmo é essencialmente um algoritmo de amostragem por importância dinâmico, onde a distribuição proposta é obtida dinamicamente através de amostras passadas. Trabalhos relacionados incluem o algoritmo de Wang-Landau (Wang e Landau, 2001), o algoritmo de Wang-Landau generalizado (Liang, 2005) e o algoritmo de ponderação dinâmica (Wong e Liang, 1997; Liu et al., 2001).

Recentemente têm sido desenvolvidos métodos MCMC chamados de *Monte Carlo populacionais*, os quais consideram uma população de múltiplas cadeias de amostragem em paralelo e cada cadeia é atualizada pela troca de informações com outras cadeias. Em geral, esses algoritmos podem ser usados tanto para tarefas de otimização como para tarefas de simulação. Nesta seção é apresentada uma curta revisão de alguns dos métodos Monte Carlo populacionais mais relevantes propostos na literatura. Uma revisão prévia sobre o assunto pode ser encontrada em Jasra et al. (2005).

Os métodos considerados aqui têm como seu espaço de estado o espaço conjunto de todas as cadeias na população. Assim, uma população de tamanho N tem um espaço de estado conjunto de dimensão $N \times m$, onde m é a dimensão do espaço original. A idéia básica de estender o espaço de estados é que, usando a informação na população, podem ser feitos movimentos mais globais, resultando em uma mistura mais rápida dos algoritmos MCMC. Cada proposta é considerada no contexto de seu espaço de estados conjunto, ainda se esta afeta somente um membro da população. Em geral, isto leva a métodos que amostram da densidade conjunta

$$\pi(x^{(1)}, ..., x^{(N)}) \propto \exp\left(-\sum_{i=1}^{N} \frac{H(x^{(i)})}{T_i}\right). \tag{2.3}$$

Qualquer densidade marginal será então idêntica a $\pi(x)$ se $T_i = 1$. Muitos tipos de propostas satisfazem as condições necessárias para amostrar desta densidade conjunta. Estas incluem passos de Metropolis sobre estados individuais, permutações entre cadeias (possivelmente a diferentes temperaturas), e vários tipos de operações geométricas no vetor de estados. A seguir são brevemente descritos alguns dos mais relevantes esquemas de amostragem Monte Carlo que operam ao nível de

população.

2.3.1 Cadeias Múltiplas Independentes

A técnica mais direta para fazer uso de cadeias múltiplas é simplesmente rodar N algoritmos MCMC independentes em paralelo. As cadeias são inicializadas em diferentes estados iniciais e sua convergência é observada ao mesmo tempo. Espera-se que, desta forma, seja obtida uma amostragem mais confiável da distribuição de interesse $\pi(.)$. É importante notar que, sob este esquema, não ocorre troca de informações entre as cadeias. Segundo Drugan e Thierens (2005), dado que as cadeias não interagem, o algoritmo de cadeias independentes múltiplas é, no nível populacional, um MCMC com probabilidade de transição igual ao produto das probabilidades de transição das cadeias que a compõem. As recomendações na literatura são conflitantes a respeito da eficiência de se rodar cadeias independentes em paralelo, embora existam pelo menos vantagens teóricas para o uso de múltiplas cadeias MCMC independentes em relação a sua convergência para $\pi(.)$ (Gilks et al., 1996).

2.3.2 Amostragem com direção adaptativa (ADA)

Este algoritmo proposto por Gilks et al. (1994); Roberts e Gilks (1994), é um dos primeiros exemplos de métodos MCMC populacionais. Eles propõem um método de se escolher direções dinamicamente para amostrar de uma distribuição de interesse, fazendo uso de múltiplas cadeias MCMC, onde os movimentos de uma cadeia são adaptados de acordo com a informação das outras cadeias. A atualização de uma amostra da população corrente é feita usando uma linha de amostragem em direção a outra amostra da população.

Um importante caso especial do ADA é o denominado algoritmo "sinuca" ("snooker" em inglês), onde, a cada iteração, são escolhidos dois pontos da população corrente, e um desses pontos é substituído por um outro ponto, escolhido ao longo de uma linha reta através dos dois pontos selecionados inicialmente, de acordo com uma distribuição unidimensional adequada. Assim, dada uma população de N amostras na iteração corrente, $\boldsymbol{X}^{(t)} = \{X_1^{(t)}, ..., X_N^{(t)}\}$, a próxima geração $\boldsymbol{X}^{(t+1)}$ é produzida como segue:

- selecionar aleatoriamente um membro $X_c^{(t)}$ de $\boldsymbol{X}^{(t)}$;

- gerar uma direção aleatória $e^{(t)}$ como $e^{(t)} = (X_c^{(t)} - X_a^{(t)})/\|X_c^{(t)} - X_a^{(t)}\|$, onde o ponto de âncora $X_a^{(t)}$ é escolhido ao acaso de $\{\boldsymbol{X}^{(t)} \setminus \{X_c^{(t)}\}\}$;

- gerar um escalar $r^{(t)}$ de uma distribuição apropriada $f(r)$, e finalmente

- atualizar $X_c^{(t+1)} = X_a^{(t)} + r^{(t)}e^{(t)}$, e $X_j^{(t+1)} = X_j^{(t)}$ para $j \neq c$.

Gilks et al. (1994) mostram que $f(r)$ deveria ser da forma

$$f(r) \propto |r|^{k-1}\pi(X_a^{(t)} + re^{(t)}). \tag{2.4}$$

Liu et al. (2000) demonstraram que o algoritmo ADA pode ser mais efetivo se a escolha de e_t é guiada por um processo de otimização local.

O ADA tem sido usado como componente de outros algoritmos MCMC populacionais propostos recentemente, tais como os algoritmos MCMC evolucionários de Liang e Wong (2001); Ter Braak (2006) e Goswami e Liu (2007), onde o algoritmo "sinuca" é usado como um operador de recombinação em espaços de busca contínuos. Liu et al. (2000) combinam o algoritmo ADA com um algoritmo de Metropolis multi-tentativa, que permite propor múltiplos pontos candidatos e usar aqueles correspondentes a uma região de busca maior. Uma limitação do algoritmo ADA é que ele está restrito a distribuições contínuas.

2.3.3 Amostragem por importância dinamicamente ponderada (AIDP)

Este algoritmo, proposto por Liang (2002), é uma extensão do algoritmo de ponderação dinâmica de Wong e Liang (1997), que consiste em introduzir pesos de importância no processo dinâmico de Monte Carlo, para permitir ao sistema fazer transições grandes, o qual não seria possível com as regras de transição do algoritmo Metropolis padrão. Com isto, o espaço de busca pode ser explorado mais livremente.

Na ponderação dinâmica, o estado x da cadeia de Markov é aumentado por um peso de importância para (x, w). Dado o estado corrente (x_t, w_t), um movimento de ponderação dinâmica

funciona gerando um estado candidato y de uma função de transição $G(x_t, y)$ e calculando a razão dinâmica

$$r_d = w_t \frac{\pi(y)G(y, x_t)}{\pi(x_t)G(x_t, y)}.$$

Seja $\theta = \theta(x_t, w_t) > 0$, uma função positiva arbitrária, e seja $a = r_d/(r_d+\theta)$. Com probabilidade a selecione $x_{t+1} = y$ e $w_{t+1} = r_d/a$. Caso contrário, selecione $x_{t+1} = x_t$ e $w_{t+1} = w_t/(1-a)$. Assim, se uma proposta é rejeitada, então o peso será dinamicamente ajustado para um valor maior, dividindo a probabilidade de rejeição dessa proposta, rendendo uma menor probabilidade total de rejeição no próximo movimento. No entanto, os pesos de importância têm uma esperança infinita, e as estimativas resultantes são de alta variabilidade e convergem para os valores verdadeiros muito lentamente.

No AIDP, o estado x da cadeia de Markov é aumentado para uma coleção de amostras ponderadas (a população) $(\boldsymbol{X}, \boldsymbol{W}) = \{x_1, w_1; ...; x_N, w_N\}$, onde N é o tamanho da população. Cada iteração do algoritmo AIDP envolve dois passos:

1. *Ponderação dinâmica*: atualiza cada um dos estados da população corrente por uma transição de ponderação dinâmica, como explicado anteriormente;

2. *Controle da população*: divide ou replica os estados dos indivíduos com pesos grandes e descarta os estados dos indivíduos com pesos pequenos. O viés induzido por esta operação é compensado, dando diferentes pesos às novas amostras produzidas.

A inclusão destes dois passos assegura que o AIDP possa se movimentar através do espaço de busca como no algoritmo de ponderação dinâmica, mas os pesos são bem controlados e tem uma esperança finita, e as estimativas resultantes podem convergir mais rapidamente do que no algoritmo de ponderação dinâmica.

2.3.4 Parallel tempering (PT)

A idéia geral do algoritmo *parallel tempering* (Geyer, 1991) é simular N réplicas do sistema original de interesse, cada réplica associada a uma temperatura diferente. Os sistemas a alta temperatura são geralmente capazes de amostrar regiões grandes do espaço de busca, enquanto que os sistemas

a baixas temperaturas fazem uma amostragem mais precisa numa região local do espaço de busca, mas podem ficar presos num mínimo de energia local durante a execução do algoritmo. PT realiza uma amostragem eficiente, permitindo que sistemas associados a diferentes temperaturas troquem configurações completas. Assim, a inclusão de sistemas associados a uma temperatura mais alta assegura que os sistemas associados a uma temperatura mais baixa possam escapar de regiões não representativas do espaço de busca e ter acesso a outras regiões mais representativas do espaço de busca, associadas a baixas temperaturas. Uma completa revisão sobre a história, teoria e as várias generalizações deste algoritmo é apresentada em Earl e Deem (2005).

No algoritmo *Parallel tempering*, cada uma das N cadeias tem uma distribuição estacionária diferente $\pi_i(.) = \pi(.)^{1/T_i}$, $i = 1, \cdots, N$. As temperaturas têm uma magnitude crescente, $T_1 < \cdots < T_N$, com $t_1 = 1$. A distribuição estacionária da cadeia com a temperatura mais baixa é portanto igual à distribuição objetivo, isto é, $\pi_1(.) = \pi(.)$. As temperaturas T_i, $2 \leq i \leq N$, têm valores constantes, tipicamente obtidos a partir de uma série geometricamente crescente. Note a semelhança deste vetor de temperaturas com os valores de temperatura propostos pelo esquema de esfriamento no algoritmo *simulated annealing*, embora, no *simulated annealing*, as temperaturas sejam geradas sequencialmente no tempo, enquanto que, no PT, as diferentes temperaturas são geradas no espaço, isto é, na população e não mudam durante a execução do algoritmo.

As permutações são propostas entre estados associados a diferentes níveis de temperatura, e são aceitas de acordo com uma regra de Metropolis, aplicada ao espaço conjunto de todas as cadeias. A diferença nas temperaturas entre cadeias sucessivas deve ser suficientemente pequena para assegurar uma probabilidade de aceitação alta para todas as permutações. A probabilidade de aceitação para uma permutação entre o estado $x^{(i)}$ associado à temperatura T_i e o estado $x^{(j)}$ associado à temperatura T_j é dada por:

$$p = \min\left\{1,\ \exp\left[\left(H\left(x^{(i)}\right) - H\left(x^{(j)}\right)\right)\left(\frac{1}{T_i} - \frac{1}{T_j}\right)\right]\right\}. \tag{2.5}$$

Usualmente as tentativas de permutação no PT são feitas entre cadeias adjacentes. Permutações entre cadeias não adjacentes também podem acontecer, mas espera-se que a taxa de aceitação de

tais permutações seja geralmente baixa. Uma extensão do algoritmo, para permitir a troca entre sistemas com temperaturas não adjacentes, é dado em Calvo (2005).

Como escolher o número de réplicas a serem usadas nas simulações e as temperaturas associadas a essas réplicas são questões de grande importância e têm sido assunto de intensa pesquisa nos últimos anos. Recentes propostas incluem os trabalhos de Kone e Kofkea (2005); Rathore et al. (2005); Katzgraber et al. (2006) e Goswami e Liu (2007). Outras referências importantes sobre o assunto podem ser encontradas em Earl e Deem (2005).

Parallel tempering pode ser combinado com outros métodos MCMC populacionais. Em algoritmos MCMC evolucionários, por exemplo, o PT tem sido usado como um operador de permutação entre estados (ver, por exemplo, Liang e Wong, 2000, 2001; Goswami e Liu, 2007), complementando a ação dos operadores genéticos.

2.3.5 Algoritmos de Monte Carlo Evolucionários

O conceito de população nos algoritmos de Monte Carlo populacionais lembra de alguma forma a estrutura dos algoritmos evolucionários descritos na seção 2.2. Baseados nesta analogia, alguns autores em particular têm proposto métodos que integram técnicas de algoritmos evolucionários dentro de um contexto MCMC. Uma revisão destes algoritmos é feita em Drugan e Thierens (2004).

A idéia de combinar algoritmos genéticos com algoritmos MCMC para produzir uma troca de informações mais eficiente entre os membros da população tem sido explorada entre outros por Liang e Wong (2000, 2001); Zhang e Cho (2001); Laskey e Myers (2003); Strens et al. (2002); Strens (2003); Hu e Tsui (2005); Ter Braak (2006); Holloman et al. (2006), e guarda direta relação com os algoritmos *parallel tempering* (Geyer, 1991) e amostragem com direção adaptativa (Gilks et al., 1994).

A idéia básica é simular uma população de cadeias de Markov em paralelo e atualizar essa população usando operadores genéticos tais como mutação e cruzamento. Em alguns casos as cadeias podem estar associadas a diferentes temperaturas (Liang e Wong, 2000, 2001), para incorporar as características do algoritmo *parallel tempering*. Nas subseções seguintes são descritos alguns desses algoritmos.

2.3.5.1 Monte Carlo Evolucionário (MCE)

O método foi inicialmente proposto para amostrar de uma distribuição definida em um espaço de seqüências binárias finitas (Liang e Wong, 2000) e, posteriormente, estendido para amostrar de distribuições em espaços reais (Liang e Wong, 2001). A idéia central do algoritmo é incorporar os operadores mutação e cruzamento dos algoritmos evolucionários no algoritmo *parallel tempering*, combinando, assim, a capacidade de aprendizagem dos algoritmos genéticos com a rapidez de mistura do *parallel tempering*.

Seja $(\boldsymbol{x}, \boldsymbol{T}) \triangleq (\boldsymbol{x}_1, T_1, \cdots, \boldsymbol{x}_N, T_N)$ o espaço conjunto de todas as cadeias, ou a população, e $\boldsymbol{x}_i$ o i-ésimo estado de $(\boldsymbol{x}, \boldsymbol{T})$. O algoritmo MCE gera amostras de $\pi(.)$ como definida em 2.3, usando quatro tipos de operadores descritos a seguir.

Mutação

Este operador consiste em aplicar uma atualização de Metropolis-Hastings a um estado $\boldsymbol{x}_i$ da população $(\boldsymbol{x}, \boldsymbol{T})$, onde $i \in \{1, \cdots, N\}$ é escolhido de acordo com uma distribuição $p(I = i \mid \boldsymbol{x})$ (por exemplo, uniforme ou determinística). Posteriormente, é proposto um estado $\boldsymbol{y}_i$ de uma distribuição proposta $G_i(\boldsymbol{x}_i, .)$, e a nova população, $(\boldsymbol{y}, \boldsymbol{T}) = \{(\boldsymbol{x}, \boldsymbol{T}) \setminus \{(\boldsymbol{x}_i, T_i)\}, \{(\boldsymbol{y}_i, T_i)\}\}$, é aceita com probabilidade $\min(1, r_m)$, onde

$$r_m = \frac{\pi_i(y_i) G_i(y_i, x_i)}{\pi_i(x_i) G_i(x_i, y_i)} \times \frac{p(I = i \mid y)}{p(I = i \mid x)}$$

A distribuição $G_i(x_i, .)$ é ajustada de tal forma que a probabilidade de aceitação do operador de mutação esteja no intervalo (0,2 ; 0,.5).

Cruzamento real

Consiste em aplicar o operador de cruzamento usual dos algoritmos genéticos a pares de estados selecionados aleatoriamente. Assim, são possíveis cruzamentos de um ponto, de k pontos, ou uniformes (ver a seção 2.2 para detalhes sobre estes tipos de cruzamento). Por exemplo, no cruzamento real de um ponto, dois índices, i e j, $i, j \in \{1, \cdots, N\}$, $i \neq j$, são escolhidos usando as distribuições $p(I = i \mid \boldsymbol{x})$ e $p(J = j \mid \boldsymbol{x}, I = i)$. Posteriormente, é escolhido aleatoriamente um *ponto de*

cruzamento $c \in \{1, \cdots, d\}$, onde d é a dimensão dos vetores de estado $\boldsymbol{x}_i$ e $\boldsymbol{x}_j$. Um novo par de estados $\boldsymbol{y}_i$ e $\boldsymbol{y}_j$ é definido como:

$$\boldsymbol{y}_i = (x_{i1}, ..., x_{ic}, x_{j(c+1)}, ..., x_{jd})$$

$$\boldsymbol{y}_j = (x_{j1}, ..., x_{jc}, x_{i(c+1)}, ..., x_{id}).$$

Aqui, $\boldsymbol{x}_i$ e $\boldsymbol{x}_j$ são os estados parentais, e $\boldsymbol{y}_i$ e $\boldsymbol{y}_j$ os novos estados gerados. Os parentais são substituídos pela sua progênie na população com probabilidade $\min(1, r_{rc})$, onde

$$r_{rc} = \frac{\pi_i(\boldsymbol{y}_i)\pi_j(\boldsymbol{y}_j)}{\pi_i(\boldsymbol{x}_i)\pi_j(\boldsymbol{x}_j)} \times \frac{G_{i,j}(\boldsymbol{y}, \boldsymbol{x})}{G_{i,j}(\boldsymbol{x}, \boldsymbol{y})},$$

com $G_{i,j}(\boldsymbol{x}, \boldsymbol{y}) = p(I = i \mid \boldsymbol{x})p(J = j \mid \boldsymbol{x}, I = i) + p(I = j \mid \boldsymbol{x})p(J = i \mid \boldsymbol{x}, I = j)$. Usualmente, é assumido que $p(I = i \mid \boldsymbol{x})$ e $p(J = j \mid \boldsymbol{x}, I = i)$ são proporcionais a uma versão "aquecida" da distribuição de interesse. Goswami e Liu (2007) assumem que $p(I = i \mid \boldsymbol{x}) \propto \exp\{-H(\boldsymbol{x}_i)/T\}$ e $p(J = j \mid \boldsymbol{x}, I = i) \propto \exp\{-H(\boldsymbol{x}_j)/T\}$, para $j \neq i$, com temperatura $T > 0$ selecionada próxima ao valor mínimo da escada de temperaturas $\boldsymbol{T}$ do algoritmo.

Cruzamento "sinuca"

Este operador é baseado no algoritmo ADA (Gilks et al., 1994), descrito na subseção 2.3.2. A intuição deste operador é puxar uma solução aleatoriamente escolhida, na direção de uma boa solução, esperando que a nova solução resultante cairá em uma região de alta probabilidade. Primeiro, é escolhido $i \in \{1, \cdots, N\}$ usando uma distribuição $p(I = i \mid \boldsymbol{x})$. O estado $\boldsymbol{x}_i$ é chamado de estado candidato. Posteriormente, é escolhido $j \neq i$ usando uma distribuição $p(J = j \mid \boldsymbol{x}, I = i)$. O estado $\boldsymbol{x}_j$, é chamado de estado âncora. A seguir é definido $\boldsymbol{y}_i = \boldsymbol{x}_j + r.(\boldsymbol{x}_j - \boldsymbol{x}_i)/\|\boldsymbol{x}_j - \boldsymbol{x}_i\|$ e formada a nova população $(\boldsymbol{y}, \boldsymbol{T}) = \{(\boldsymbol{x}, \boldsymbol{T}) \setminus \{(\boldsymbol{x}_i, T_i)\}, \{(\boldsymbol{y}_i, T_i)\}\}$. Aqui, r é amostrada da distribuição em (2.4). Amostrar da distribuição de r é geralmente muito difícil. Algumas orientações para tratar com este problema são dadas em Goswami e Liu (2007). Segundo Liang e Wong (2001), pode ser usada uma distribuição uniforme para $p(I = i \mid \boldsymbol{x})$, e considerar $p(J = j \mid \boldsymbol{x}, I = i) \propto \exp\{-H(\boldsymbol{x}_j)/T_N\}$ para $j \neq i$.

Permutação

Este operador funciona como a permutação do algoritmo *parallel tempering*, isto é, primeiro é selecionado $i \in \{1, \cdots, N\}$; posteriormente, é selecionado $j = i \pm 1$ com igual probabilidade, e a nova configuração $(\boldsymbol{y}, \boldsymbol{T}) = (\boldsymbol{x}_1, T_1; \cdots; \boldsymbol{x}_j, T_i; \cdots; \boldsymbol{x}_i, T_j; \cdots; \boldsymbol{x}_N, T_N)$ é aceita com a probabilidade da Equação 2.5.

Implementação do algoritmo MCE

Seja $\boldsymbol{x}^{(0)} = (\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \cdots, \boldsymbol{x}_N^{(0)})$ uma configuração inicial da população, $p_{mut} \in (0, 1)$ a taxa de mutação previamente fixada, e $\{T_1 > T_2 > \cdots > T_N = \tau_{min} > 0\}$ uma escada de temperaturas predeterminada. Então, uma iteração do algoritmo MCE consiste dos seguintes passos:

1. escolher os operadores mutação, cruzamento real ou cruzamento "sinuca" com probabilidades p_{mut}, $(1 - p_{mut})/2$, and $(1 - p_{mut})/2$, respectivamente;

2. aplicar permutação na população N vezes, isto é, propor a permutação de N pares de estados vizinhos, como descrito anteriormente.

Extensões do algoritmo MCE têm sido recentemente propostas na literatura considerando, por exemplo, novas formas para o operador permutação (Goswami e Liu, 2007) ou a inclusão de novas operações, como a migração de estados entre subpopulações (Hu e Tsui, 2005).

2.3.5.2 Cadeias de Markov com evolução diferencial (CM-ED)

Este algoritmo, apresentado em Strens et al. (2002) e Ter Braak (2006), combina um algoritmo genético para otimização em espaços contínuos, chamado evolução diferencial (Storn e Price, 1997), com um algoritmo MCMC de cadeias múltiplas. As N cadeias rodam em paralelo e uma cadeia é atualizada a partir da diferença dos vetores de duas cadeias, $\boldsymbol{x}_{R_1}$ e $\boldsymbol{x}_{R_2}$, escolhidas aleatoriamente da população, multiplicada por um fator γ, segundo a expressão

$$\boldsymbol{x}_{prop} = x_i + \gamma(\boldsymbol{x}_{R_1} - \boldsymbol{x}_{R_2}) + \varepsilon,$$

onde $\varepsilon \sim N(0,b)^d$ com d sendo a dimensão da distribuição de interesse, e b é uma variância pequena em relação à variância da distribuição de interesse. Cada proposta $\boldsymbol{x}_{prop}$ é aceita com base no critério de Metropolis. Uma escolha *default* para o parâmetro γ, sugerida por Ter Braak (2006), é escolher $\gamma = 2,38\sqrt{(2d)}$. Já para o número de cadeias, Ter Braak (2006) sugere usar $N = 2d$ ou $N = 3d$ para o número de cadeias em distribuições unimodais e, para distribuições mais complexas, usar $N = 10d$ ou até $N = 20d$.

2.3.5.3 Monte Carlo evolucionário distribuído (MCED)

Baseado no algoritmo genético distribuído (Tanese, 1989), Hu e Tsui (2005) propõem introduzir um operador de migração no algoritmo MCE de Liang e Wong (2001). O algoritmo resultante, chamado de Monte Carlo evolucionário distribuído (MCED), consiste em dividir a população inteira em um conjunto de subpopulações e executar o algoritmo MCE para cada subpopulação, permitindo que alguns indivíduos sejam selecionados periodicamente de cada subpopulação e migrados para diferentes subpopulações. A idéia do o operador de migração é prevenir a convergência prematura do algoritmo, frequentemente causada, no algoritmo genético padrão, pelo uso repetido do operador de cruzamento.

Capítulo 3

Algoritmos de Busca Estocásticos para o Desenho Ótimo de Redes de Monitoramento

3.1 Introdução

O desenho eficiente de redes de monitoramento é muito importante para se obter um melhor entendimento de processos ambientais, ecológicos e epidemiológicos. O redesenho destas redes, especialmente sua redução devido a eventuais mudanças futuras nas prioridades ou objetivos da rede, assim como devido a possíveis restrições de verbas, é outro desafio que deve ser cuidadosamente abordado, visando minimizar a perda de informação, mantendo ao mesmo tempo a capacidade para fazer inferências sobre o fenômeno de interesse. Neste capítulo é desenvolvido um novo algoritmo genético híbrido (AGH) para o desenho ótimo de redes de monitoramento, o qual combina o algoritmo genético padrão com um operador de busca local. O desempenho do AGH é comparado com outros dois algoritmos de busca estocástica, *simulated annealing* (SA) e um algoritmo genético padrão (AG). O foco deste estudo é o redesenho de redes de monitoramento ambiental em larga escala.

Em geral o problema de desenho consiste em determinar um subconjunto de localizações de amostragem de forma que satisfaça algum critério de otimalidade. Muitas aplicações na literatura de desenho de redes de monitoramento ambiental consideram a minimização de alguma função de custo como, por exemplo, a variância de krigagem (ver por exemplo Boer et al., 2002; Diggle e Lophaven,

2006; Pardo-Igúzquiza, 1998; Trujillo-Ventura e Ellis, 1991, e referências). Outra abordagem baseada em teoria da informação como proposta em Caselton e Hussain (1980) e Caselton e Zidek (1984) visa achar o desenho da rede que maximize a entropia das variáveis aleatórias de interesse nos sítios monitorados. O redesenho de redes monitoradoras baseado neste critério tem sido considerado em Caselton et al. (1992); Guttorp et al. (1993) e Wu e Zidek (1992) para o caso univariado, com extensões para o caso multivariado descritas em Le e Zidek (1994); Le et al. (2003) e Zidek et al. (2000). No entanto, independentemente do critério de desenho usado, os aspectos computacionais deste tipo de metodologias são ainda uma consideração prática da maior importância em desenhos de amostragem espacial, como mencionado em Guttorp et al. (1993) e Le et al. (2003).

Quando o espaço das possíveis soluções é finito e o critério para avaliar os desenhos é bem definido, o problema de desenho pode ser visualizado como um problema de otimização combinatória. Os métodos rotineiramente usados na literatura para encontrar a melhor solução para este problema consistem na enumeração completa de todas as possíveis soluções (Le et al., 2003; Wu e Zidek, 1992); ou em regras heurísticas simples, baseadas, principalmente, em trocas de estações, onde a cada passo é adicionada (ou subtraída quando está sendo considerada a redução da rede) a estação que melhor otimiza a função objetivo (Guttorp et al., 1993; Zidek et al., 2000; Boer et al., 2002; Royle, 2002); ou em algoritmos exatos, os quais tem sido desenvolvidos com sucesso para o caso Gaussiano (Ko et al., 1995; Bueso et al., 1998; Lee, 1998). Embora estes métodos funcionem bem para o desenho de pequenas redes, eles não são adequados para a otimização de grandes redes. De fato, Ko et al. (1995) mostrou que este problema é NP-árduo, implicando que é improvável que possa ser encontrado um algoritmo eficiente para resolver de forma ótima todos os possíveis casos do problema.

Por esta razão, desenhos de redes de monitoramento baseados em algoritmos de busca estocástica computacionalmente intensivos, tais como o *simulated annealing* (Ferri e Piccioni, 1992; Pardo-Igúzquiza, 1998; Banjevic e Switzer, 2001; Ferreyra et al., 2002; Nunes et al., 2004; Fuentes et al., 2005) e o algoritmo genético (Reed et al., 2000; Wu et al., 2005), tem sido propostos recentemente na literatura. Estes algoritmos têm fornecido soluções perto do ótimo global para problemas de otimização complexos em muitas áreas de aplicação. Seguindo estas idéias, neste capítulo é proposto

um novo algoritmo genético híbrido (AGH), o qual melhora o desempenho do algoritmo genético padrão e do *simulated annealing*, na busca pelo melhor desenho.

O desempenho do AGH é comparado com o obtido com os algoritmos SA e AG, usando um conjunto de problemas simulados de diferentes tamanhos, assim como em uma aplicação real envolvendo a redução de uma rede de monitoramento ambiental de larga escala, destacando as vantagens e desvantagens de cada algoritmo. O critério de desenho escolhido foi o de entropia, devido a sua generalidade e amplo uso na literatura ambiental. Cabe ressaltar que a comparação destes algoritmos de busca estocástica num contexto de desenho de redes de monitoramento de larga escala não havia sido considerada antes na literatura ambiental ou estatística. Lee e Ellis (1996) fizeram uma comparação prévia de oito algoritmos, incluindo SA e AG, mas ela foi restrita a problemas simples (seu problema mais complexo foi escolher a localização de dez estações monitoradoras dentre 121 possíveis sítios candidatos).

3.2 Desenhos de Amostragem Espacial Baseados em Entropia e Informação

Considere um campo aleatório $\{B(s) : s \in S \subset \mathbb{R}^p\}$, gerado por um certo processo natural; neste estudo será considerado $p = 2$. Por simplicidade, a região de interesse, S, é discretizada como uma grade de m potenciais sítios de amostragem espalhados por todo o espaço. O interesse é escolher dentre esses m potenciais sítios de amostragem o melhor subconjunto de n pontos ($n < m$) para instalar uma rede de monitoramento.

O vetor de observações em todos os m sítios do campo discretrizado $\boldsymbol{B}$ pode ser subdividido num vetor $\boldsymbol{B_1}$ correspondente aos $m-n$ sítios que não serão monitorados, e um vetor $\boldsymbol{B_2}$ correspondente aos n sítios que serão monitorados. Suponha que $\boldsymbol{B}$ tem a função de densidade de probabilidade conjunta, $f(\boldsymbol{B})$. A incerteza total sobre $\boldsymbol{B}$ pode ser expressa pela entropia da sua distribuição, i.e. $H(\boldsymbol{B}) = E[-\log\{f(\boldsymbol{B})\}]$ e a informação sobre $\boldsymbol{B}$ é dada por $I(\boldsymbol{B}) = E[\log\{f(\boldsymbol{B})\}]$. Sob esta abordagem são possíveis duas formulações do problema.

Uma primeira formulação do problema foi proposta em Caselton e Zidek (1984) e consiste em escolher os n sítios que serão monitorados ($\boldsymbol{B_2}$) de tal forma que as predições do campo inteiro

baseadas naqueles sítios fornecerá a máxima informação possível sobre os $m-n$ sítios não monitorados ($\boldsymbol{B_1}$). Seguindo as idéias de Bernardo (1979), uma medida de informação ganha sobre $\boldsymbol{B_1}$ como resultado de medir $\boldsymbol{B_2}$ é dada por

$$I(\boldsymbol{B_1}, \boldsymbol{B_2}) = I(\boldsymbol{B_1} \mid \boldsymbol{B_2}) - I(\boldsymbol{B_1}) \ , \tag{3.1}$$

a qual é conhecida como índice de informação de Shannon. Assim, o conhecimento ótimo sobre $\boldsymbol{B_1}$ é encontrado ao maximizar $I(\boldsymbol{B_1}, \boldsymbol{B_2})$. Se o processo for Gaussiano, então $\boldsymbol{B} \sim N_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, e pode ser mostrado que

$$I(\boldsymbol{B_1}, \boldsymbol{B_2}) = -\frac{1}{2} \ \log|\boldsymbol{\Sigma}_{\boldsymbol{B_1}|\boldsymbol{B_2}}| + \frac{1}{2} \ \log|\boldsymbol{\Sigma}_{\boldsymbol{B_1}}| = -\frac{1}{2} \ \log\prod_{i=1}^{n}(1-\rho_i^2) \ , \tag{3.2}$$

onde os $\rho's$ são os coeficientes de correlação canônica entre $\boldsymbol{B_1}$ e $\boldsymbol{B_2}$. As provas das expressões 3.1 e 3.2 são apresentadas no apêndice A.

A segunda formulação visa minimizar a entropia condicional $H(\boldsymbol{B_1} \mid \boldsymbol{B_2})$, que corresponde à incerteza residual remanescente após $\boldsymbol{B_2}$ ser observado. A quantidade total de incerteza na distribuição conjunta de $\boldsymbol{B}$ é definida como

$$H(\boldsymbol{B_1}, \boldsymbol{B_2}) = H(\boldsymbol{B_1}|\boldsymbol{B_2}) + H(\boldsymbol{B_2}) \ . \tag{3.3}$$

Assim, como a entropia total é fixa, a minimização da entropia condicional $H(\boldsymbol{B_1}|\boldsymbol{B_2})$ é equivalente a maximizar a entropia nos sítios que serão monitorados $H(\boldsymbol{B_2})$. A prova da expressão 3.3 pode ser vista no apêndice B. Para o caso Gaussiano, isto é equivalente a maximizar o determinante da matriz de covariâncias entre as localizações de amostragem ($argmax\{\log|\boldsymbol{\Sigma}_{\boldsymbol{B_2}}|\}$).

De acordo com Caselton e Zidek (1984) e Le e Zidek (1994), critérios baseados em entropia para resolver problemas de desenho de redes são usualmente considerados quando há mais de um objetivo a ser alcançado, ou quando é difícil formular um critério que inclua todos os possíveis objetivos da rede, ou quando o número de localizações potenciais de amostragem é finita. Neste estudo é considerada a segunda formulação como critério de desenho para o desenvolvimento e comparação dos algoritmos para otimização de desenho de redes, os quais são descritos nas seções seguintes.

3.3 Abordagens Computacionais

Esta seção descreve em detalhe os passos para a implementação dos algoritmos desenvolvidos para a otimização de redes de monitoramento. Primeiramente será descrito o algoritmo *simulated annealing*, seguido do algoritmo genético e, finalmente, o algoritmo genético híbrido. Inicialmente, será assumido que o espaço dos possíveis sítios de amostragem, é um campo discretizado na forma de uma grade regular de m pontos (sítios candidatos), mas os algoritmos são aptos para funcionar sobre grades irregulares como mostrado na aplicação da sub-seção 3.4.2.

3.3.1 O Algoritmo simulated annealing (SA)

A idéia do *simulated annealing* (Kirkpatrick et al., 1983) vem da analogia com o processo físico de "annealing", o qual envolve a agregação de muitas partículas em um sistema físico enquanto este é esfriado. O algoritmo consiste em escolher uma solução inicial, e a partir daí fazer sucessivas perturbações aleatórias, pequenas, visando gerar novas soluções que serão aceitas sempre que levarem a uma melhora no valor da função objetivo a ser otimizada. O algoritmo pode ainda aceitar propostas piores com uma certa probabilidade que depende de um parâmetro de controle (T), o qual decresce com o número de iterações do algoritmo. O uso deste parâmetro permite que o algoritmo possa escapar de ótimos locais. A implementação do SA considerada neste estudo consiste dos seguintes passos:

Passo 0 : Inicializar um contador $c = 0$ e especificar o número total de iterações (*totiter*) que o algoritmo fará, ou especificar algum critério de parada para o algoritmo, por exemplo, parar o algoritmo se as últimas v iterações forneceram configurações idênticas para o melhor desenho.

Passo 1 : Escolher ao acaso uma solução inicial S, representada na forma de um vetor binário (cadeia de zeros e uns) de comprimento igual ao número de pontos candidatos, e com valor 1 nas posições correspondentes aos pontos selecionados, e zero nas posições dos pontos no selecionados.

Passo 2 : Calcular o valor da função objetivo $H(S)$ para a solução inicial.

Passo 3 : Determinar o valor inicial para o parâmetro de controle (T) de tal forma que a taxa de aceitação das soluções propostas no início do algoritmo seja alta o suficiente (por exemplo ≥ 0.95). Ver van Laarhoven e Aarts (1987, Seção 5.2) para detalhes sobre este procedimento.

Passo 4 : Gerar uma nova proposta S^* fazendo uma perturbação aleatória na configuração S: Isto é feito escolhendo aleatoriamente um dos s_i sítios selecionados, $i = 1, ..., n$, da solução S e trocando-o por um de seus vizinhos não selecionados.

Passo 5 : Calcular o valor da função objetivo $H(S^*)$ para a solução proposta S^*.

Passo 6 : Critério de aceitação: a solução proposta será aceita com probabilidade

$$p_{accept} = \min\left\{1, \exp\left[\frac{H(S^*) - H(S)}{T}\right]\right\}.$$

Passo 7 : Parar o algoritmo se o número de iterações pré-especificado for alcançado $(k = totiter)$ ou se o critério de parada for atingido. Caso contrário ir ao próximo passo.

Passo 8 : Reduzir o parâmetro de controle T com um esquema de esfriamento geométrico, isto é, a cada ℓ iterações, fazer $T = \alpha \times T$, onde $0 < \alpha < 1$ é um fator de desconto.

Passo 9 : Reiniciar o algoritmo a cada r iterações, usando como solução inicial o melhor resultado obtido até a iteração prévia, e incrementar o parâmetro de controle T, por exemplo através de $T = \beta \times T$, onde $\beta > 1$ é um fator de acréscimo.

Passo 10 : Fazer $c = c + 1$ e retornar ao passo 4.

Como esquemas de vizinhança para este estudo foram considerados um esquema de vizinhança de segunda ordem para as grades regularmente espaçadas, e um esquema dos k vizinhos mais próximos para as grades irregularmente espaçadas. No entanto, outros esquemas de vizinhança são igualmente plausíveis. Outros esquemas de esfriamento podem também ser considerados (veja por exemplo Nourani e Andresen, 1998; Cohn e Fielding, 1999).

3.3.2 O Algoritmo Genético (AG)

O algoritmo genético (Holland et al., 1975) visa melhorar uma população de possíveis soluções através de sucessivas gerações usando princípios de evolução genética tais como seleção natural, cruzamento e mutação. A implementação do algoritmo genético considerado neste estudo consiste dos seguintes passos:

Passo 0 : Inicializar um contador $c = 0$ e especificar o número total de gerações (*totger*) que o algoritmo fará, ou especificar algum critério de parada para o algoritmo.

Passo 1 : Escolher ao acaso uma população inicial de tamanho N, isto é, um conjunto de N possíveis soluções, $(S_1, ..., S_N)$, cada uma destas soluções sendo representada em forma binária, como descrito no algoritmo SA.

Passo 2 : Calcular o valor da função objetivo $H(S_i)$, $i = 1, \cdots, N$, para cada uma das soluções na população.

Passo 3 : Estabelecer uma probabilidade de seleção que servirá como critério para selecionar os indivíduos sobre os quais será aplicado o operador de recombinação.

Passo 4 : *Cruzamento*. O cruzamento entre indivíduos é feito para uma proporção, (p_{cross}), dos indivíduos na população. Os indivíduos que serão submetidos ao cruzamento são selecionados de acordo com sua probabilidade de ajuste previamente definida. O cruzamento começa formando pares de indivíduos ao acaso (dentre os indivíduos selecionados), os quais são combinados para gerar novos indivíduos de acordo com o seguinte procedimento: pontos presentes em ambos os indivíduos de um certo par serão selecionados automaticamente para fazer parte da nova solução resultante. Os pontos restantes a serem selecionados serão aleatoriamente escolhidos, dentre os pontos selecionados em pelo menos um dos indivíduos do par escolhido.

Passo 5 : *Mutação*. A mutação ocorre em uma proporção, $(propm)$, de indivíduos escolhidos ao acaso na população. O procedimento consiste em avaliar cada ponto de cada solução

escolhida para mutação e decidir se este será descartado ou não dessa solução de acordo com uma probabilidade de mutação (p_{mut}). Pontos descartados por este procedimento deverão ser substituídos, visando sempre manter o número fixo n de pontos selecionados em cada solução. Isto é feito escolhendo aleatoriamente um dos pontos não selecionados na vizinhança ∂_i do ponto que está saindo. Neste algoritmo são usados os mesmos esquemas de vizinhança descritos para o algoritmo SA.

Passo 6 : *Seleção*. Os indivíduos gerados pelos procedimentos de cruzamento e mutação são incorporados à população inicial, e sua função objetivo é calculada. A população da nova geração será selecionada a partir dessa população aumentada. Os P indivíduos com melhor ajuste ($P < N$), chamados de *elite*, entrarão diretamente nessa nova população, enquanto que os restantes $N - P$ membros da população serão escolhidos aleatoriamente de acordo com sua probabilidade de seleção.

Passo 7 : Atualizar as probalidades de seleção dos indivíduos na nova população.

Passo 8 : Parar o algoritmo se o número pré-especificado de gerações for atingido ($c = totger$), ou se o critério de parada for atingido. Caso contrário, fazer $c = c + 1$ e retornar ao passo 4.

Foram considerados inicialmente três esquemas de seleção para estabelecer as probabilidades de seleção nos passos 3 e 6:

(i) *seleção proporcional ao ajuste*, em que a probabilidade de selecionar um indivíduo i é dada por:

$$p_{\text{fitness}} = \frac{\exp\{H(S_i)\}}{\sum_{j=1}^{N} \exp\{H(S_j)\}};$$

(ii) *seleção por torneio*, onde antes de selecionar um indivíduo, são pré-selecionados, aleatoriamente (independentemente do valor da sua função objetivo), t indivíduos da população, para depois selecionar o indivíduo com a melhor medida de ajuste dentre esses t indivíduos;

(iii) *seleção* $N + O$, onde os melhores N indivíduos são escolhidos de uma população temporária aumentada, contendo os N indivíduos parentais e os O indivíduos gerados apartir deles na geração corrente.

Os melhores resultados foram obtidos com o segundo esquema de seleção, quando $t = 4$ indivíduos. Portanto, ele foi escolhido como o esquema de seleção para este algoritmo, para todos os problemas considerados neste capítulo. Detalhes sobre estes, e outros esquemas de seleção, podem ser encontrados por exemplo em Goldberg (1989).

3.3.3 O Algoritmo Genético Híbrido (AGH)

Um algoritmo de busca local (LS) foi acoplado ao algoritmo genético padrão da seção 3.3.2 visando refinar a qualidade das soluções obtidas pelo AG e, consequentemente, melhorar seu desempenho. A estrutura geral do AGH adotada neste estudo é descrita a seguir:

Passo 0 : Inicializar um contador $c = 0$ e especificar o número total de gerações (*totger*) que o algoritmo fará, ou especificar algum critério de parada para o algoritmo.

Passo 1 : Construir uma população inicial de N soluções.

Passo 2 : Aplicar o procedimento de busca local para substituir cada solução inicial com uma solução localmente ótima.

Passo 3 : Aplicar os operadores genéticos (cruzamento e mutação) para gerar novas soluções.

Passo 4 : Selecionar os N indivíduos que formarão a população na seguinte geração, como descrito no algoritmo AG.

Passo 5 : Aplicar o procedimento de busca local aos indivíduos da nova população provenientes da implementação dos operadores genéticos no passo 3 (indivíduos velhos presentes na nova população não experimentarão busca local). O operador de busca local é descrito na subseção seguinte.

Passo 6 : Repetir os passos 3 a 5 até que o número pré-especificado de gerações seja alcançado ($c = totger$) ou até que o critério de parada seja atingido.

3.3.3.1 O operador de busca local

O operador de busca local se desloca iterativamente através das estações s_{ik} de cada solução S_i, $i = 1, ..., N$, $k = 1, ..., n$. A busca local opera na vizinhança ∂_i de cada estação s_{ik}. Cada ponto desta vizinhança é selecionado com uma probabilidade p_{neigh}, e uma solução proposta é formada trocando as posições das estações corrente e selecionada. A solução proposta substituirá a solução corrente se, e só se, esta tem uma melhor medida de ajuste.

Em geral, o operador de busca local não é aplicado nas primeiras gerações do AGH (e.g. o primeiro 20%), para evitar fazer busca local em regiões do espaço de busca nas quais o ótimo global provavelmente não está localizado. A partir deste ponto, a busca local é aplicada a cada nova solução, a cada g gerações do algoritmo. Soluções com o mesmo valor da função objetivo de outras soluções já consideradas para busca local são ignoradas.

O custo deste passo de busca local é medido pelo número de avaliações da função objetivo realizadas pelo operador de busca local, e seu valor é medido por seu efeito no melhoramento da qualidade da solução obtida com o AG padrão. Nas próximas duas seções estes três algoritmos são implementados e comparados via simulações, assim como usando um conjunto de dados real.

3.4 Análise de Dados

3.4.1 Um Estudo Baseado em Dados Artificiais

Um estudo de simulação foi realizado para avaliar o desempenho dos algoritmos anteriormente descritos. Foram consideradas regiões regulares de cinco diferentes tamanhos (5×5, 10×10, 20×20, 30×30 e 50×50), os quais representam campos aleatórios discretizados com 25, 100, 400, 900 e 2500 potenciais localizações de amostragem, respectivamente. O número de pontos a serem selecionados em cada problema correspondeu a 20% e 40% das potenciais localizações de amostragem para as grades 5×5 e 10×10; 10% e 20% das potenciais localizações de amostragem para as grades

20×20 e 30×30 e 5% e 10% das potenciais localizações de amostragem para a grade 50×50. Com exceção dos primeiros dois casos (grades 5×5), a enumeração completa de todas as possíveis soluções é impossível para os problemas propostos; o terceiro problema mais simples (escolher 20 estações monitoradoras dentre 100 possíveis pontos candidatos) envolve um espaço de busca maior do que 5^{20} possíveis soluções.

O foco deste estudo foi a comparação da eficiência dos algoritmos em encontrar desenhos de amostragem próximos ao ótimo. Portanto, foi assumido que a distribuição que aproxima o processo $\boldsymbol{B}$, assim como seus parâmetros eram conhecidos a priori. Mais precisamente, foi assumido que $\boldsymbol{B} \mid \boldsymbol{\Sigma} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, com $\Sigma_{ij} = \exp\{-\phi \parallel s_i - s_j \parallel\}$, e $\phi = 0.01$, onde $\parallel . \parallel$ denota a distância Euclidiana. O critério de entropia foi considerado como critério de desenho. Assim, o objetivo foi maximizar $H(\boldsymbol{B_2}) = \frac{1}{2} \log \mid \boldsymbol{\Sigma_{B_2}} \mid$, com $\boldsymbol{B_1}$ e $\boldsymbol{B_2}$ como descritos na seção 3.2. O número total de iterações em cada rodada do algoritmo SA foi fixada em 50000 para todos os problemas considerados. O parâmetro de controle (temperatura) foi inicializado em 200 e reduzido a cada 100 iterações usando um fator de desconto $\alpha = 0.95$. O passo de reinício foi implementado a cada 1000 iterações usando um fator de acréscimo $\beta = 1,5$. O algoritmo AG foi rodado por 1000 gerações, com uma população de 100 indivíduos (soluções candidatas), uma probabilidade de cruzamento $p_{\text{cross}} = 0,75$, uma probabilidade de mutação $p_{\text{mut}} = 0,05$, e uma elite de $P = 5$ indivíduos. Estes valores foram previamente obtidos de um pequeno estudo de caso implementado sobre uma grade 10×10 (ver detalhes no apêndice C). O AGH foi rodado por 300 gerações. Foi usada uma probabilidade $p_{\text{neigh}} = 0,6$, constante através das gerações, a qual foi definida antes da aplicação do algoritmo através de algumas rodadas exploratórias. O operador de busca local não foi aplicado aos primeiros 20% das gerações do AG. A partir daí, a busca local foi aplicada a cada nova solução em cada geração do AG. Um esquema de vizinhança de segunda ordem foi adotado nos três algoritmos para representar o grupo ∂_i de vizinhos da i-ésima estação na grade.

Para comparar o desempenho dos algoritmos em encontrar o melhor desenho de amostragem, foram obtidas 100 realizações de cada um deles, inicializadas com desenhos diferentes exceto para os últimos três casos, onde por razões computacionais, foram obtidas somente 10 realizações de cada algoritmo. Para manter a comparabilidade entre eles, os três algoritmos usaram o mesmo conjunto

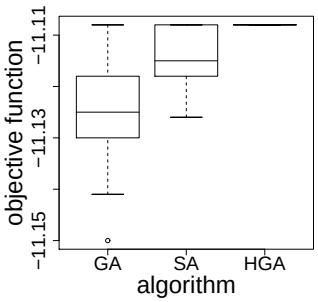
inicial de sementes para o gerador de números aleatórios. A efetividade dos algoritmos foi expressa em termos do valor da função objetivo e da freqüência com que foram encontradas boas soluções. O anterior foi feito para cada um dos problemas propostos. Também foi considerado o tempo computacional necessário para executar os algoritmos. Todos os algoritmos foram implementados usando a linguagem de programação Ox versão 3.3 (Doornik, 2002) rodando sob Windows XP. As análises foram feitas em um PC com processador AMD Athlon XP 3000+, a 2.1GHz e 512MB de memória RAM.

Um resumo dos resultados é dado na Tabela 3.1 e Figuras 3.1 e 3.2. A Tabela 3.1 registra o valor máximo, a média e o desvio padrão da função objetivo obtidos considerando todas as rodadas de cada um dos problemas propostos e para cada um dos três algoritmos. Para cada algoritmo foi obtido também o tempo médio de processamento de uma rodada e o número de vezes que a melhor solução encontrada (considerando todos os algoritmos) foi obtida com cada algoritmo. Algumas conclusões podem ser obtidas a partir da verificação destes resultados:

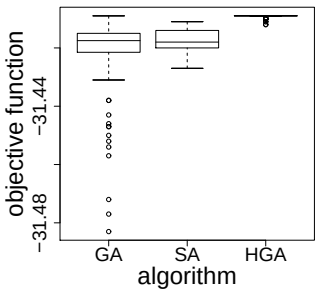
Tabela 3.1: Resultados obtidos no estudo de simulação para os algoritmos *simulated annealing* (SA), Algoritmo Genético (AG), e Algoritmo Genético Híbrido (AGH).

grade	n	SA					AG					AGH				
		máximo	média	dp	CPU	Nopt	máximo	média	dp	CPU	Nopt	máximo	média	dp	CPU	Nopt
5×5	5	-1.8056	-1.8056	0	1.0	100	-1.8056	-1.8056	0	8.2	100	-1.8056	-1.8056	0	3.3	100
5×5	10	-6.3796	-6.3796	0	1.3	100	-6.3796	-6.3796	0	9.9	100	-6.3796	-6.3796	0	8.1	100
10×10	20	-11.1081	-11.1124	0.0052	2.7	56	-11.1081	-11.1210	0.0086	19.3	16	-11.1081	-11.1081	0	59.4	100
10×10	40	-31.411	-31.4178	0.0040	11.3	0	-31.4086	-31.422	0.0140	38.9	1	-31.4086	-31.4092	0.0006	363.2	53
20×20	40	-18.034	-18.0518	0.0082	14.4	0	-18.045	-18.0952	0.0327	69.1	0	-18.0315	-18.0352	0.0038	822.1	32
20×20	80	-51.471	-51.5214	0.0241	50.7	0	-51.532	-51.6271	0.0429	171.2	0	-51.4428	-51.4686	0.0111	5874.8	1
30×30	90	-44.001	-44.0549	0.0278	98.8	0	-44.154	-44.3191	0.0725	360.3	0	-43.9355	-43.9714	0.0165	13824.6	1
30×30	180	-120.87	-121.024	0.0711	394.2	0	-122.31	-122.527	0.1858	1285.3	0	-120.7810	-120.8411	0.0378	131315.1	1
50×50	125	-44.395	-44.4936	0.0481	229.8	0	-45.161	-45.5004	0.2706	1028	0	-44.3324	-44.3718	0.0301	49062.9	1
50×50	250	-129.93	-130.106	0.1066	1049.2	0	-135.33	-135.802	0.3101	3523	0	-129.7609	-129.8404	0.0504	472140.6	1

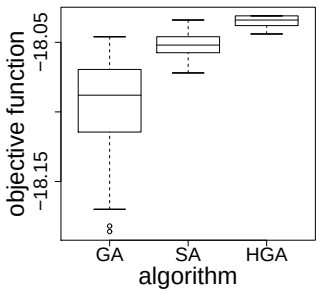
Nopt: número de vezes que a melhor solução encontrada pelos três algoritmos foi registrada; *CPU:* tempo computacional médio (em segundos) para uma rodada; n: número de estações a serem incluidas na rede; *dp:* desvío padrão.



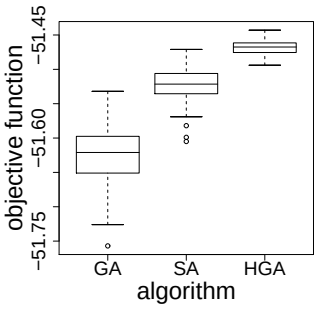
(a) 10 × 10 - 20



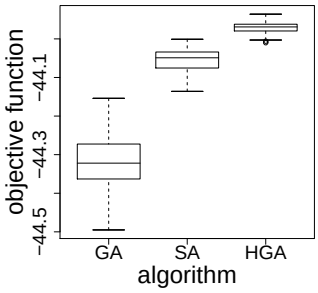
(b) 10 × 10 - 40



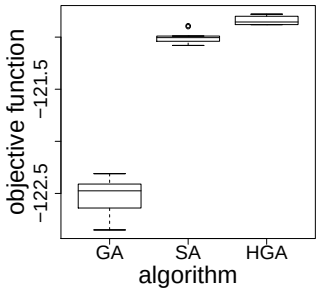
(c) 20 × 20 - 40



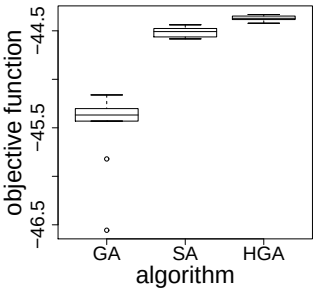
(d) 20 × 20 - 80



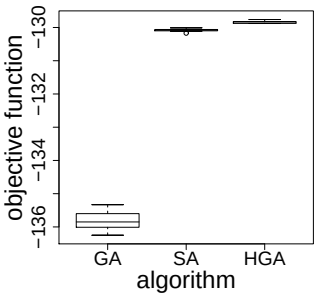
(e) 30 × 30 - 90



(f) 30 × 30 - 180



(g) 50 × 50 - 125

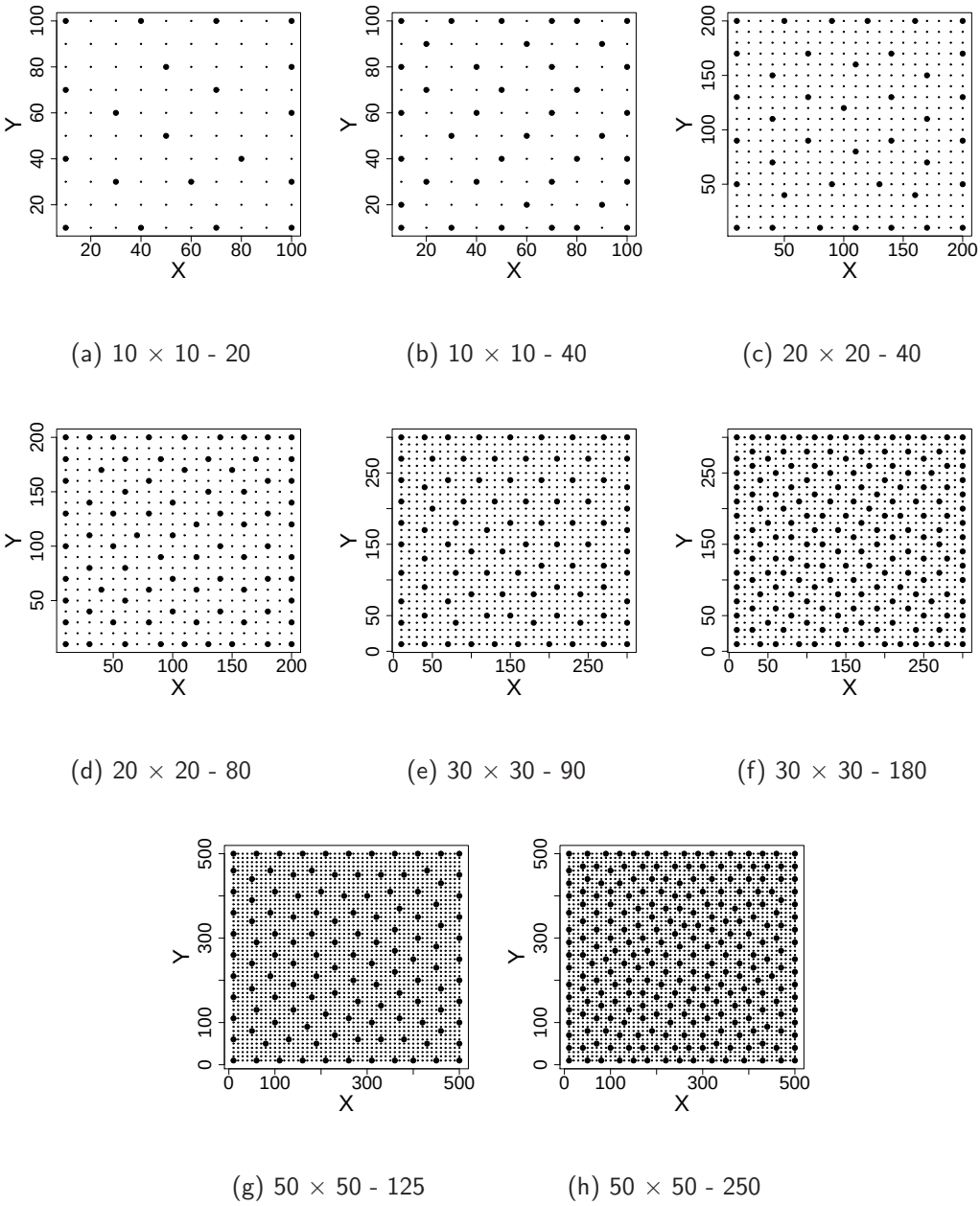


(h) 50 × 50 - 250

Figura 3.1: Box plots para as funções objetivo obtidas no estudo de simulação.

Para os problemas mais simples (grades 5×5), os três algoritmos foram capazes de encontrar a solução ótima em 100% das vezes (para estes casos simples, o ótimo verdadeiro era conhecido). Para todos os outros casos considerados, o AGH teve o melhor desempenho em termos do número de vezes em que foi registrada a melhor solução encontrada considerando os três algoritmos, assim como em termos dos valores médios da função objetivo. O desempenho do AGH foi seguido pelo desempenho do algoritmo SA, enquanto que os piores resultados foram para o AG padrão. As diferenças em desempenho entre o AG e os outros dois algoritmos, foram mais acentuadas a medida que o tamanho do problema aumentava. O SA gastou muito menos tempo que os algoritmos AG ou AGH. Esta eficiência computacional foi mais evidente para problemas de tamanho maior.

As soluções obtidas com o AGH tiveram também a menor variabilidade como mostrado nos box plots da Figura 3.1 e nos desvios padrões da Tabela 3.1, seguido de perto pelo SA. As soluções obtidas com o AG tiveram a maior variabilidade. Estes resultados sugerem que os algoritmos SA e AGH são menos sensíveis à escolha de pontos iniciais no sentido de que eles convergiram para boas soluções mais frequentemente. Esta é uma característica importante, dado que a maioria das vezes os pontos de início são escolhidos arbitrariamente. Por outro lado, aparentemente, o AG faz uma boa exploração inicial do espaço de busca, mas tem um desempenho pobre tentando refinar essas soluções intermediárias. Essa desvantagem do AG padrão é superada pelo AGH, com a inclusão do operador de busca local. As melhores subredes encontradas para cada um dos problemas simulados considerados são apresentadas na Figura 3.2.



(a) 10 × 10 - 20 (b) 10 × 10 - 40 (c) 20 × 20 - 40

(d) 20 × 20 - 80 (e) 30 × 30 - 90 (f) 30 × 30 - 180

(g) 50 × 50 - 125 (h) 50 × 50 - 250

Figura 3.2: Melhores desenhos de redes obtidos no estudo de simulação.

Os resultados anteriores coincidem com os obtidos por Lee e Ellis (1996) que registraram o algoritmo de busca TABU e o SA como sendo mais eficientes que o AG para um problema de desenho de redes de monitoramento, baseado na freqüência com que foram encontradas as melhores soluções. O desempenho pobre do algoritmo genético quando comparado ao *simulated annealing* foi referenciado também por Hurley et al. (1996) para um problema de transmissão de freqüências de rádio. Por outro lado, Drezner e Wesolowsky (2003) encontrou que o AG superou o desempenho do SA num problema de localização de instalações. De acordo com Adler (1953), o uso de taxas de mutação muito baixas, e uma alta dependência do operador de cruzamento poderiam explicar a convergência prematura do AG, levando a uma estagnação do algoritmo a medida que a população vai ficando mais homogênea. Porém, o AG considerado neste estudo foi implementado com as melhores taxas de cruzamento e mutação encontradas na análise exploratória.

3.4.2 Uma Aplicação Real

O desempenho dos algoritmos foi testado também com dados reais, em uma aplicação que considerou a escolha de um desenho ótimo para reduzir uma grande rede de estações monitoradoras de ozônio ao nível do solo nos Estados Unidos. O conjunto de dados considerado para esta aplicação foi coletado pela agência de proteção ambiental americana (EPA) e consiste em medições horárias da concentração de ozônio em partes por bilhão medidas durante o período 1994-2003. Aproximadamente metade das estações monitoradoras desta rede estiveram fora de operação durante o inverno e muitas delas operaram somente durante alguns dos dez anos considerados. Por esta razão, foram consideradas para a análise somente estações que estiveram em operação durante os dez anos, e em cada ano somente foi considerado o período correspondente à estação verão (Junho 21 a Setembro 22). Adicionalmente, somente foram consideradas as medições feitas às 15:00 horas. Esta hora do dia foi selecionada pois nesse horário são registrados os maiores níveis de ozônio e a menor proporção de dados faltantes. Portanto, o conjunto de dados que realmente foi usado na análise consistiu de um total de 671 estações monitoradoras diferentes, espalhadas irregularmente através do país, e cada estação consistiu de $94 \times 10 = 940$ observações ao longo do tempo.

Uma análise exploratória dos dados permitiu identificar um ciclo diurno de concentração de ozônio,

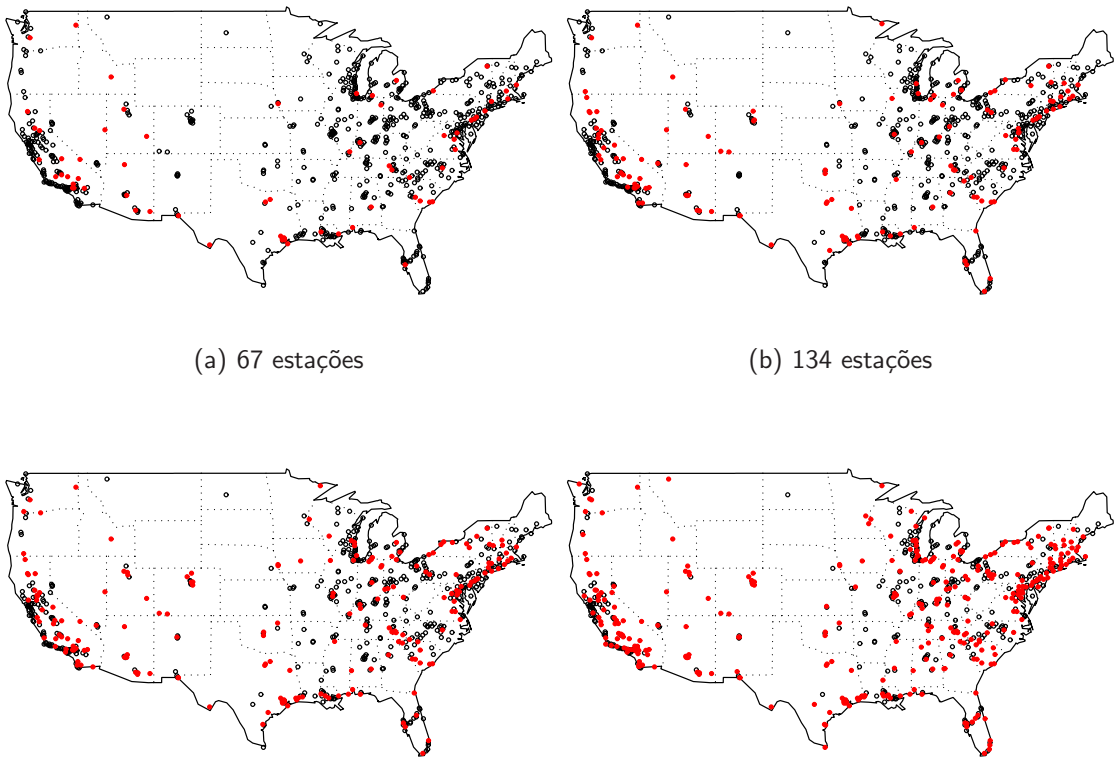
com um pico alto durante as primeiras horas da tarde, assim como com um efeito de final de semana (com concentrações de ozônio mais altas durante o fim de semana). Portanto, para remover os componentes sazonal e de tendência previamente identificados pela análise exploratória, foi ajustado, para cada estação, um modelo de regressão com variáveis indicadoras para o ano e para o dia da semana, assim como com um termo seno e coseno com periodicidade $2\pi/94$. Uma transformação raiz quadrada foi usada para aproximar normalidade e estabilizar a variância das observações. A matriz de covariâncias amostral foi então calculada a partir dos resíduos deste modelo. Acredita-se que esses resíduos contém a estrutura espacial que não foi capturada pela análise temporal. Isto foi feito para evitar qualquer confundimento das estruturas temporal e espacial o qual pode acontecer quando a matriz de covariâncias amostral é calculada a partir dos dados originais.

Os algoritmos foram implementados usando os mesmos critérios de desenho e escolha de parâmetros usados no estudo simulado da seção anterior. O esquema de vizinhança usado nesta aplicação consistiu dos 10 pontos mais próximos de cada estação. Os tamanhos das redes reduzidas considerados foram 67, 134, 201, 268 e 336 estações. Em cada caso foi considerado o melhor desenho obtido dentre 20 otimizações realizadas com cada um dos algoritmos. Os resultados mostrados na Tabela 3.2 confirman o melhor desempenho do AGH. O algoritmo SA foi novamente melhor que o AG padrão para as redes maiores (268 e 336 estações). No entanto, para redes pequenas e médias (67, 134 e 201 estações), o AG padrão superou o desempenho do SA em termos de qualidade das soluções e variabilidade. Os mapas com os melhores subconjuntos de estações encontradas em cada caso são apresentadas na Figura 3.3.

Tabela 3.2: Resultados obtidos na aplicação real para os algoritmos *simulated annealing* (SA), Algoritmo Genético (AG), e Algoritmo Genético Híbrido (AGH).

n	SA					AG					AGH				
	máximo	média	dp	CPU	Nopt	máximo	média	dp	CPU	Nopt	máximo	média	dp	CPU	Nopt
67	43.357	42.503	0.5928	39.5	0	44.033	43.792	0.1510	167.0	0	44.276	44.176	0.0647	722.7	1
134	70.370	69.312	0.6808	190.8	0	71.275	70.819	0.2202	620.3	0	71.815	71.783	0.0336	8730.1	1
201	87.284	86.227	0.5221	493.7	0	86.963	86.402	0.3211	1526.0	0	88.902	88.845	0.0381	32324.2	1
268	95.703	94.826	0.4164	1028.7	0	94.579	93.691	0.5305	3045.7	0	97.094	97.089	0.0521	72109.5	1
336	96.809	95.711	0.5356	1881.3	0	94.983	93.760	0.7160	5444.2	0	98.935	98.875	0.0855	149160.2	1

Nopt: número de vezes que a melhor solução encontrada pelos três algoritmos foi registrada; *CPU:* tempo computacional médio (em segundos) para uma rodada; *dp:* desvío padrão.



(a) 67 estações (b) 134 estações

(c) 201 estações (d) 336 estações

Figura 3.3: Melhores desenhos de redes obtidos para o caso da redução de uma rede de monitoramento de ozônio nos Estados Unidos.

3.5 Considerações Finais

Neste capítulo foi apresentado um algoritmo genético híbrido (AGH) desenvolvido para resolver o problema de redesenhar uma rede de monitoramento, incorporando um procedimento de busca local no AG padrão. Este algoritmo é capaz de realizar uma busca local no subespaço de ótimos locais de uma maneira mais eficiente. O algoritmo AGH foi comparado com dois algoritmos de busca estocásticos frequentemente usados na literatura, o *simulated annealing* e o algoritmo genético padrão. Os algoritmos foram testados e comparados em problemas artificiais de diferentes tamanhos, assim como com dados reais. Os resultados computacionais mostraram que o AGH teve um desempenho superior para todos os problemas testados, superando o desempenho dos outros dois algoritmos em termos de qualidade das soluções.

Embora o principal objetivo do estudo tenha sido o de reduzir o tamanho de uma rede pré-

existente, estes algoritmos poderiam ser adaptados para o caso oposto, isto é, o de aumentar o tamanho da rede. A incorporação do custo de instalação e manutenção da rede dentro da função objetivo como proposto em Zidek et al. (2000), ou a combinação do critério de entropia com uma função de utilidade que dê prioridade a algumas características desejáveis do problema de desenho de redes que esteja sendo considerado, (e.g. a capacidade para identificar níveis de poluentes acima de um certo limiar) como proposto em Fuentes et al. (2005), são também casos plausíveis de otimização usando os algoritmos descritos neste capítulo.

Todos os algoritmos implementados neste estudo foram rodados em forma sequencial, isto é, avaliando somente uma solução por vez. No entanto, os algoritmos AG e AGH são algoritmos inerentemente paralelos, o que significa que os indivíduos (soluções) da população podem ser avaliados simultaneamente. Assim, se os recursos computacionais disponíveis permitirem uma implementação em paralelo destes algoritmos, seus tempos computacionais podem ser significativamente reduzidos.

Dado que a troca de uma estação somente leva à mudança da linha e a coluna correspondentes a essa estação na matriz de covariâncias, o determinante $|\mathbf{\Sigma}_{\boldsymbol{B_2}}|$ poderia ser obtido de uma forma mais eficiente via a atualização de uma decomposição da matriz de covariâncias (tal como a decomposição QR ou SVD), após a troca de cada estação, sem precisar calcular explicitamente tal determinante. Este procedimento beneficiaria todos os algoritmos com passos locais tais como o SA e o AGH, assim como o operador de mutação do algoritmo genético. Um impacto maior deste procedimento seria esperado para o AGH devido à grande quantidade de avaliações da função objetivo necessárias no operador de busca local.

A utilidade de cotas superiores para $\log|\mathbf{\Sigma}_{\boldsymbol{B_2}}|$ como medidas de ajuste substitutas durante a busca local, para evitar o cálculo exato da função objetivo, deveria também ser investigada visando aumentar a eficiencia do AGH. Estas cotas superiores já existem na literatura (veja por exemplo, Anstreicher e Lee, 2004; Hoffman et al., 2001, e referências).

Capítulo 4

Algoritmos Evolucionários de Monte Carlo via Cadeias de Markov para o Desenho Ótimo de Redes de Monitoramento

4.1 Introdução

Problemas de desenho espacial em geral consistem em selecionar um subconjunto (p) de pontos em algum domínio espacial d-dimensional, visando otimizar o valor de um critério de otimalidade. Este problema pode ser resolvido em um contexto de teoria da decisão (DeGroot, 1970). Suponha que variáveis de decisão $\boldsymbol{d}$ (neste caso o desenho de uma rede de monitoramento) devem ser escolhidas sob incerteza de um conjunto de possíveis alternativas $\mathcal{D}$. Seja $\boldsymbol{y}$ o conjunto de dados a ser observado e seja $p_{\boldsymbol{d}}(\boldsymbol{y} \mid \boldsymbol{\theta})$ a distribuição de amostragem dos dados dado um vetor de parâmetros desconhecido $\boldsymbol{\theta}$, o qual depende da decisão $\boldsymbol{d}$. Seja $p(\boldsymbol{\theta})$ a distribuição a priori de $\boldsymbol{\theta}$, a qual é invariante sob a decisão $\boldsymbol{d}$. A formulação do problema de desenho é completada especificando uma função de utilidade $u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta})$ que modela preferências sobre conseqüências associadas a cada decisão. Como a decisão $\boldsymbol{d}$ tem que ser tomada antes de observar os dados, a decisão ótima $\boldsymbol{d}^*$ é obtida maximizando a esperança de $u(.,.,.)$ em relação a $(\boldsymbol{\theta}, \boldsymbol{y})$, i.e., $\boldsymbol{d}^* = arg \max_{\boldsymbol{d} \in \mathcal{D}} U(\boldsymbol{d})$, onde

$$U(\boldsymbol{d}) = \int\int_{y,\theta} u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta}) p_{\boldsymbol{d}}(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) dy d\theta. \tag{4.1}$$

Duas dificuldades principais surgem desta abordagem: (i) frequentemente a utilidade esperada (4.1) é uma integral analiticamente intratável; e (ii) é preciso fazer maximização sobre um espaço de decisão grande. Müller et al. (2004) propõem uma solução para estes problemas, baseados na simulação de uma superfície da utilidade quando vista como uma função de densidade de probabilidade sobre o espaço dos desenhos; o desenho ótimo corresponde à moda desta distribuição de probabilidade. A idéia principal é interpretar $u(\boldsymbol{d}, \boldsymbol{y}, \boldsymbol{\theta}) p_{\boldsymbol{d}}(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$ como uma distribuição conjunta de $\boldsymbol{\theta}$, $\boldsymbol{y}$ e $\boldsymbol{d}$, e então simular a distribuição a posteriori marginal de $\boldsymbol{d}$ por amostragem de Monte Carlo. Uma aproximação razoável da moda desta distribuição marginal será considerada como o melhor desenho. O método usa o algoritmo *simulated annealing* como a ferramenta de maximização para a aproximação desta moda.

No entanto, esta abordagem de maximização pode não ser eficiente para problemas de alta dimensão. Em particular, ela pode levar a uma exploração lenta do espaço dos desenhos e, consequentemente, a tempos computacionais muito grandes para obter bons resultados. Adicionalmente, nesta situação há um risco maior da cadeia de Markov ficar presa em um ótimo local.

Nos últimos anos tem sido desenvolvida uma nova família de algoritmos de otimização estocásticos conhecida como *algoritmos evolucionários de Monte Carlo via cadeias de Markov* para resolver problemas de otimização em espaços de alta dimensão de maneira mais eficiente (ver, por exemplo, Drugan e Thierens, 2004, e suas referências). Esses algoritmos combinam idéias de algoritmos genéticos ou evolucionários em um contexto de Monte Carlo via cadeias de Markov. Em geral eles consistem em rodar várias cadeias de Markov em paralelo, as quais são inicializadas em diferentes estados (pontos diferentes do espaço de busca) de forma aleatória. As cadeias podem interagir entre elas através de operadores genéticos tais como mutação e recombinação para obter melhores amostras. Essa interação permite que em qualquer tempo, o conjunto das cadeias forneça uma *população* espalhada através das diferentes regiões do espaço de busca.

Neste capítulo é proposto um novo algoritmo baseado em simulação para a maximização de

utilidades esperadas em espaços de alta dimensão, combinando o método de Müller et al. (2004) com recentes algoritmos evolucionários de Monte Carlo. A abordagem proposta visa fazer a maximização das utilidades esperadas com uma melhor exploração do espaço dos desenhos. O algoritmo desenvolvido foi comparado com o proposto por Müller et al. (2004), usando uma aplicação real envolvendo a redução de uma rede de monitoramento de ozônio de larga escala no leste dos Estados Unidos.

4.2 O Algoritmo de Müller

Em uma série de artigos, por exemplo, Bielza et al. (1999), Müller (1999), Sansó e Müller (1999), Müller et al. (2004), Peter Müller e colaboradores têm desenvolvido uma abordagem para desenhos ótimos baseada em simulações de Monte Carlo no espaço dos possíveis desenhos. Em sua abordagem, o problema de seleção de desenhos é encaixado dentro de um problema artificial de inferência Bayesiana, visando usar técnicas de amostragem modernas para inferência Bayesiana. A abordagem foi desenvolvida em um contexto de teoria da decisão, portanto, é requerida uma função de utilidade específica para medir a qualidade das soluções. A idéia central é definir uma distribuição auxiliar

$$h_J(\boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J) \propto \prod_{j=1}^{J} u(\boldsymbol{d}, \theta_j, y_j) p\left(y_j \mid \theta_j, \boldsymbol{d}\right) p\left(\theta_j\right) \tag{4.2}$$

de modo que a marginal em $\boldsymbol{d}$,

$$\begin{aligned}
h_J(\boldsymbol{d}) &= \int \cdots \int h_J(\boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J) dy_{1:J} d\theta_{1:J} \\
&\propto \int \cdots \int \prod_{j=1}^{J} u(\boldsymbol{d}, \theta_j, y_j) p_{\boldsymbol{d}}(y_j, \theta_j) dy_{1:J} d\theta_{1:J} \\
&\propto \underbrace{\int_{y_1} \int_{\theta_1} u(\boldsymbol{d}, \theta_1, y_1) p_{\boldsymbol{d}}(y_1, \theta_1) dy_1 d\theta_1}_{U(\boldsymbol{d})} \int \cdots \int \prod_{j=2}^{J} u(\boldsymbol{d}, \theta_j, y_j) p_{\boldsymbol{d}}(y_j, \theta_j) dy_{2:J} d\theta_{2:J} \\
&\vdots \\
&\propto \left[U\left(\boldsymbol{d}\right)\right]^J
\end{aligned}$$

seja proporcional a $U(\boldsymbol{d})^J$ como desejado, para um inteiro positivo J. A transformação potência $h^J(\boldsymbol{d})$ é usada ao invés da função objetivo original $h(\boldsymbol{d})$, devido ao fato de que a superfície da utilidade esperada em muitas aplicações é muito achatada ao redor da moda, requerendo assim a simulação de amostras de tamanhos excessivamente grandes para estimar a moda de $h(\boldsymbol{d})$.

A simulação da distribuição objetivo $h_J(\boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J)$ é feita usando métodos MCMC com cadeias de Markov não homogêneas, com J aumentando ao longo das iterações. Seja o vetor de estados alternativo $(\boldsymbol{d}, \nu)$, onde $\nu = \dfrac{1}{J}\sum\limits_{j=1}^{J} \log u(\boldsymbol{d}, \theta_j, y_j)$. Os passos do algoritmo são descritos a seguir:

passo 0 : Assuma que o estado corrente da cadeia é $(J, \boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J)$ ou $(\boldsymbol{d}, \nu)$.

passo 1 : Propor um novo desenho $\tilde{\boldsymbol{d}}$ de $g\left(\tilde{\boldsymbol{d}} \mid \boldsymbol{d}\right)$ (uma distribuição simétrica).

passo 2 : Simular $\left(\tilde{\theta}_j, \tilde{y}_j\right)$ de $p\left(\boldsymbol{\theta}, \boldsymbol{y} \mid \tilde{\boldsymbol{d}}\right)$, com $j = 1, \cdots, J$;

passo 3 : Calcular $\tilde{\nu}$ e considerar $\left(\tilde{\boldsymbol{d}}, \tilde{\nu}\right)$ como um novo estado proposto da cadeia de Markov;

passo 4 : Aceitar a configuração proposta com probabilidade $\alpha_J = \min\left[\, 1, \exp\left(J\tilde{\nu} - J\nu\right) \,\right]$;

passo 5 : Seja J_n o valor corrente de J. Incrementar J para $J_{n+1} \geq J_n$, seguindo um esquema de "esfriamento" escolhido, tal que $J_n \to \infty$ quando $n \to \infty$;

passo 6 : Repetir os passos 2 a 6 até convergência.

A probabilidade de aceitação α_J no passo 4 é obtida como:

$$\alpha_{\mathrm{J}} = \min\left\{1,\ \frac{h_J\left(\tilde{\boldsymbol{d}}, \tilde{y}_1, \tilde{\theta}_1, \cdots, \tilde{y}_J, \tilde{\theta}_J\right)}{h_J\left(\boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J\right)}\ \frac{\prod\limits_{j=1}^{J} p\left(\theta_j, y_j \mid \boldsymbol{d}\right) g\left(\boldsymbol{d} \mid \tilde{\boldsymbol{d}}\right)}{\prod\limits_{j=1}^{J} p\left(\tilde{\theta}_j, \tilde{y}_j \mid \tilde{\boldsymbol{d}}\right) g\left(\tilde{\boldsymbol{d}} \mid \boldsymbol{d}\right)}\right\};$$

mas, $h_J(\boldsymbol{d}, y_1, \theta_1, \cdots, y_J, \theta_J) \propto \prod\limits_{j=1}^{J} u(\boldsymbol{d}, \theta_j, y_j) p\left(\theta_j, y_j \mid \boldsymbol{d}\right)$ e $g\left(\boldsymbol{d} \mid \tilde{\boldsymbol{d}}\right) = g\left(\tilde{\boldsymbol{d}} \mid \boldsymbol{d}\right)$.

Assim:

$$\alpha_{\mathrm{J}} = \min\left\{1,\ \frac{\prod_{j=1}^{J} u(\tilde{\boldsymbol{d}}, \tilde{y}_j, \tilde{\theta}_j)}{\prod_{j=1}^{J} u(\boldsymbol{d}, y_j, \theta_j)}\right\}$$

$$= \min\left\{1,\ \exp\left[\sum_{j=1}^{J} \log u\left(\tilde{\boldsymbol{d}}, \tilde{y}_j, \tilde{\theta}_j\right) - \sum_{j=1}^{J} \log u\left(\boldsymbol{d}, y_j, \theta_j\right)\right]\right\}$$

$$= \min\left\{1,\ \exp\left(J\tilde{\nu} - J\nu\right)\right\},$$

onde $\nu = \frac{1}{J}\sum_{j=1}^{J} \log u(\boldsymbol{d}, y_j, \theta_j)$.

Müller et al. (2004) aplicam seu algoritmo na redução do tamanho de uma rede de monitoramento ambiental já existente. A distribuição usada no passo 1 para gerar as propostas para o algoritmo de Metropolis é baseada na simples troca de uma estação entre o subconjunto de estações incluídas na rede, D, e o subconjunto de estações excluídas, D^c. De acordo com Sansó e Müller (1999), esta não é uma abordagem satisfatória por ser improvável que esse algoritmo consiga explorar uma porção suficientemente grande do espaço dos desenhos. Eles também ressaltam que esta deficiência parece ser compartilhada por outros algoritmos de desenho de redes existentes na literatura, quando o número de possíveis sítios é moderadamente grande. O algoritmo *eMCMC* proposto neste capítulo, o qual é descrito em detalhe na seção seguinte, tenta superar esta deficiência.

4.3 Um Algoritmo Evolucionário de Monte Carlo via cadeias de Markov

O algoritmo proposto neste capítulo combina o algoritmo de Müller et al. (2004) com elementos do algoritmo evolucionário de Monte Carlo de Liang e Wong (2000), visando fazer a maximização da utilidade esperada com uma melhor exploração do espaço dos desenhos. O algoritmo simula uma população de N cadeias de Markov em paralelo, cada uma delas associada a um valor diferente J_i, $i \in \{1, \cdots, N\}$, do conjunto $\boldsymbol{J} = \{J_1, J_2, \cdots, J_N\}$, com $J_1 > \cdots > J_N$. Cada cadeia tem seu próprio parâmetro de controle associado (*"temperatura"*) do conjunto $\boldsymbol{T} = \{T_1, \cdots, T_N\}$, com $T_i = 1/J_i$, tal que $T_1 > \cdots > T_N$. A população inicial de desenhos $\mathcal{D} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N\}$ é escolhida aleatoriamente, onde cada $\boldsymbol{d}_i$ é representado como um vetor binário de comprimento igual ao número

de pontos candidatos, e com valor um nas posições correspondentes às estações selecionadas, e zero nas posições dos pontos não selecionados. A população é atualizada usando operadores genéticos (mutação, cruzamento e permutação), os quais permitem a troca de informação entre as cadeias ao longo das simulações. Esses operadores são descritos em detalhe na subseção seguinte.

4.3.1 Os operadores Genéticos

Os operadores genéticos mutação e cruzamento, usados para o intercambio de informações entre as cadeias, são baseados naqueles desenvolvidos no capítulo anterior especificamente para o problema de desenho de redes. Seguindo a mesma notação do algoritmo de Müller, seja $(\boldsymbol{d}_i, \nu_i)$ um vetor de estados alternativo para a i-ésima cadeia, com $\nu_i = \frac{1}{J_i} \sum_{j=1}^{J_i} \log\, u(\boldsymbol{d}_i, \theta_j, y_j)$.

4.3.1.1 Mutação

Cada desenho $\boldsymbol{d}_k$, $k = 1, \cdots, N$ da população corrente $\mathcal{D} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N\}$ experimentará mutação independentemente com uma probabilidade (q_{mut}), levando a um novo desenho $\tilde{\boldsymbol{d}}_k$, e cada estação de cada desenho escolhido para mutação, será descartada ou não desse desenho de acordo com uma probabilidade de mutação (*probm*). Estações descartadas deverão ser substituídas por outras, mantendo assim sempre fixo o número p de estações selecionadas em cada desenho. Isto é feito escolhendo aleatoriamente um dos pontos não selecionados na vizinhança ∂_i da estação que será removida. O esquema de vizinhança de segunda ordem e o esquema dos k vizinhos mais próximos são comumente usados como esquemas de vizinhança para desenhos regular e irregularmente espaçados, respectivamente. Outros esquemas são igualmente plausíveis. A nova população $\mathcal{D}' = \{\mathcal{D} \backslash \{\boldsymbol{d}_k\}, \tilde{\boldsymbol{d}}_k\}$ será aceita com probabilidade

$$\alpha_{\mathrm{J_{mut}}} = \min\left\{1,\ \frac{h_{J_k}\left(\tilde{\boldsymbol{d}}_k, \tilde{y}_1, \tilde{\theta}_1, \cdots, \tilde{y}_{J_k}, \tilde{\theta}_{J_k}\right)}{h_{J_k}\left(\boldsymbol{d}_k, y_1, \theta_1, \cdots, y_{J_k}, \theta_{J_k}\right)} \times \frac{\prod_{j=1}^{J_k} p\left(\theta_j, y_j \mid \boldsymbol{d}_k\right) g\left(\boldsymbol{d}_k \mid \tilde{\boldsymbol{d}}_k\right)}{\prod_{j=1}^{J_k} p\left(\tilde{\theta}_j, \tilde{y}_j \mid \tilde{\boldsymbol{d}}_k\right) g\left(\tilde{\boldsymbol{d}}_k \mid \boldsymbol{d}_k\right)}\right\}$$

$$= \min\left\{1,\ \frac{\prod_{j=1}^{J_k} u(\tilde{\boldsymbol{d}}_k, \tilde{y}_j, \tilde{\theta}_j)}{\prod_{j=1}^{J_k} u(\boldsymbol{d}_k, y_j, \theta_j)}\right\}$$

$$= \min\left\{1,\ \exp\left[\sum_{j=1}^{J_k} \log u\left(\tilde{\boldsymbol{d}}_k, \tilde{y}_j, \tilde{\theta}_j\right) - \sum_{j=1}^{J_k} \log u\left(\boldsymbol{d}_k, y_j, \theta_j\right)\right]\right\}$$

$$= \min\left\{1,\ \exp\left[J_k \tilde{\nu}_k - J_k \nu_k\right]\right\},$$

onde $\nu_k = \frac{1}{J_k} \sum_{j=1}^{J_k} \log u(\boldsymbol{d}_k, y_j, \theta_j)$.

4.3.1.2 Cruzamento

O cruzamento entre pares de desenhos é feito para uma proporção $q_{cross} = 1 - q_{mut}$ dos desenhos na população corrente $\mathcal{D}$. O primeiro desenho de cada par, $\boldsymbol{d}_i$, é escolhido de acordo com sua probabilidade de ajuste, definida segundo um esquema de seleção proporcional, em que a probabilidade de selecionar o desenho i é dada por:

$$p_{\mathrm{fitness}_{(i)}} = \frac{\exp\{\nu_i\}}{\sum_{j=1}^{N} \exp\{\nu_j\}}; \tag{4.3}$$

ver por exemplo, Goldberg (1989), para detalhes sobre este e outros esquemas de seleção em algoritmos evolucionários. O segundo desenho, $\boldsymbol{d}_k$ $(i \neq k)$, é aleatoriamente escolhido do resto da população (assume-se que $\nu_i \geq \nu_k$). Cada par de desenhos selecionado é recombinado para gerar novos desenhos da seguinte forma: estações presentes em ambos desenhos serão selecionadas automaticamente para fazer parte dos dois novos desenhos resultantes, $\tilde{\boldsymbol{d}}_i$ e $\tilde{\boldsymbol{d}}_k$. O restante das estações será selecionado aleatoriamente entre as estações presentes em pelo menos um dos desenhos do par parental, como representado na Figura 4.1. Uma nova população é proposta como $\mathcal{D}' = \{\mathcal{D} \setminus \{\boldsymbol{d}_i, \boldsymbol{d}_k\}, \tilde{\boldsymbol{d}}_i, \tilde{\boldsymbol{d}}_k\}$ e aceita com probabilidade

$$
\begin{aligned}
\alpha_{\mathrm{J_{cross}}} &= \min\Bigg\{1,\ \frac{h_{J_i}\left(\tilde{\boldsymbol{d}}_i, \tilde{y}_1, \tilde{\theta}_1, \cdots, \tilde{y}_{J_i}, \tilde{\theta}_{J_i}\right) h_{J_k}\left(\tilde{\boldsymbol{d}}_k, \tilde{y}_1, \tilde{\theta}_1, \cdots, \tilde{y}_{J_k}, \tilde{\theta}_{J_k}\right)}{h_{J_i}\left(\boldsymbol{d}_i, y_1, \theta_1, \cdots, y_{J_i}, \theta_{J_i}\right) h_{J_k}\left(\boldsymbol{d}_k, y_1, \theta_1, \cdots, y_{J_k}, \theta_{J_k}\right)} \\
&\qquad \times \frac{\prod_{j=1}^{J_i} p\left(\theta_j, y_j \mid \boldsymbol{d}_i\right) \prod_{j=1}^{J_k} p\left(\theta_j, y_j \mid \boldsymbol{d}_k\right) g\left(\boldsymbol{d}_i \mid \tilde{\boldsymbol{d}}_i\right) g\left(\boldsymbol{d}_k \mid \tilde{\boldsymbol{d}}_k\right)}{\prod_{j=1}^{J_i} p\left(\tilde{\theta}_j, \tilde{y}_j \mid \tilde{\boldsymbol{d}}_i\right) \prod_{j=1}^{J_k} p\left(\tilde{\theta}_j, \tilde{y}_j \mid \tilde{\boldsymbol{d}}_k\right) g\left(\tilde{\boldsymbol{d}}_i \mid \boldsymbol{d}_i\right) g\left(\tilde{\boldsymbol{d}}_k \mid \boldsymbol{d}_k\right)}\Bigg\} \\
&= \min\left\{1,\ \frac{\prod_{j=1}^{J_i} u(\tilde{\boldsymbol{d}}_i, \tilde{y}_j, \tilde{\theta}_j) \prod_{j=1}^{J_k} u(\tilde{\boldsymbol{d}}_k, \tilde{y}_j, \tilde{\theta}_j)}{\prod_{j=1}^{J_i} u(\boldsymbol{d}_i, y_j, \theta_j) \prod_{j=1}^{J_k} u(\boldsymbol{d}_k, y_j, \theta_j)}\right\} \\
&= \min\Bigg\{1,\ \exp\Bigg[\sum_{j=1}^{J_i} \log u\left(\tilde{\boldsymbol{d}}_i, \tilde{y}_j, \tilde{\theta}_j\right) + \sum_{j=1}^{J_k} \log u\left(\tilde{\boldsymbol{d}}_k, \tilde{y}_j, \tilde{\theta}_j\right) - \sum_{j=1}^{J_i} \log u\left(\boldsymbol{d}_i, y_j, \theta_j\right) \\
&\qquad - \sum_{j=1}^{J_k} \log u\left(\boldsymbol{d}_k, y_j, \theta_j\right)\Bigg]\Bigg\} \\
&= \min\left\{1,\ \exp\left[J_i(\tilde{\nu}_i - \nu_i) + J_k(\tilde{\nu}_k - \nu_k)\right]\right\}.
\end{aligned}
$$

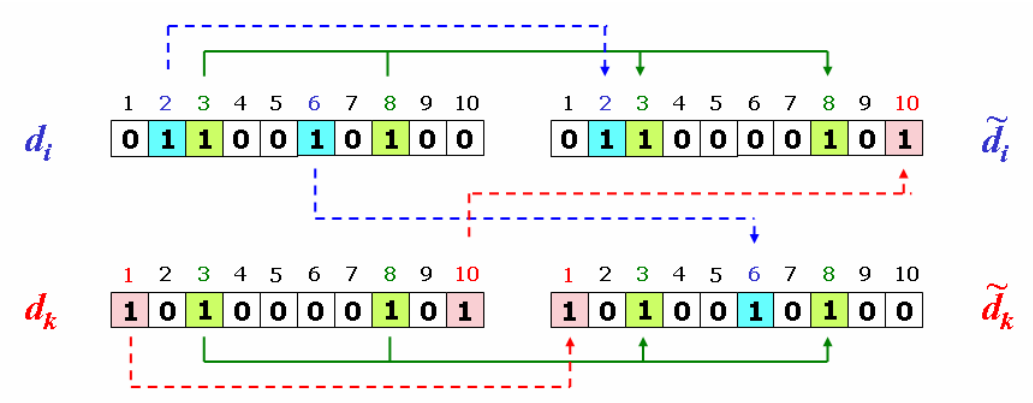


Figura 4.1: Representação esquemática do operador de cruzamento entre pares de desenhos.

4.3.1.3 Permutação

O operador de permutação propõe trocas entre desenhos associados a diferentes temperaturas. O objetivo deste operador é permitir que soluções promissórias associadas a baixas temperaturas possam escapar mais facilmente de ótimos locais. Mais especificamente, dada a população de desenhos

corrente, $\mathcal{D}$, e o vetor de temperaturas $\boldsymbol{T}$, $(\mathcal{D}, \boldsymbol{T}) = (\boldsymbol{d}_1, T_1, \cdots, \boldsymbol{d}_N, T_N)$, o operador de permutação propõe um movimento de $(\mathcal{D}, \boldsymbol{T}) = (\boldsymbol{d}_1, T_1, \cdots, \boldsymbol{d}_i, T_i, \cdots, \boldsymbol{d}_k, T_k, \cdots, \boldsymbol{d}_N, T_N)$ para $(\mathcal{D}', \boldsymbol{T}) = (\boldsymbol{d}_1, T_1, \cdots, \boldsymbol{d}_k, T_i, \cdots, \boldsymbol{d}_i, T_k, \cdots, \boldsymbol{d}_N, T_N)$. Seguindo Liang e Wong (2000), a permutação é feita unicamente entre estados com valores de temperatura adjacentes, i.e., para cada par (i, k), i é amostrada uniformemente de $\{1, \cdots, N\}$ e $k = i \pm 1$ é amostrada com probabilidade $w(\boldsymbol{d}_k \mid \boldsymbol{d}_i)$, onde $w(\boldsymbol{d}_{i+1} \mid \boldsymbol{d}_i) = w(\boldsymbol{d}_{i-1} \mid \boldsymbol{d}_i) = 0,5$ e $w(\boldsymbol{d}_2 \mid \boldsymbol{d}_1) = w(\boldsymbol{d}_{N-1} \mid \boldsymbol{d}_N) = 1$. Outros esquemas de permutação, como os recentemente propostos por Goswami e Liu (2007) também poderiam ser usados. A nova população será aceita com probabilidade

$$
\begin{aligned}
\alpha_{\mathrm{J_{exch}}} &= \min\Bigg\{1,\ \frac{h_{J_i}\left(\boldsymbol{d}_k, y_1, \theta_1, \cdots, y_{J_i}, \theta_{J_i}\right) h_{J_k}\left(\boldsymbol{d}_i, y_1, \theta_1, \cdots, y_{J_k}, \theta_{J_k}\right)}{h_{J_i}\left(\boldsymbol{d}_i, y_1, \theta_1, \cdots, y_{J_i}, \theta_{J_i}\right) h_{J_k}\left(\boldsymbol{d}_k, y_1, \theta_1, \cdots, y_{J_k}, \theta_{J_k}\right)} \\
&\qquad \times \frac{\prod_{j=1}^{J_i} p\left(\theta_j, y_j \mid \boldsymbol{d}_i\right) \prod_{j=1}^{J_k} p\left(\theta_j, y_j \mid \boldsymbol{d}_k\right) g\left(\boldsymbol{d}_i \mid \boldsymbol{d}_k\right) g\left(\boldsymbol{d}_k \mid \boldsymbol{d}_i\right)}{\prod_{j=1}^{J_i} p\left(\theta_j, y_j \mid \boldsymbol{d}_k\right) \prod_{j=1}^{J_k} p\left(\theta_j, y_j \mid \boldsymbol{d}_i\right) g\left(\boldsymbol{d}_k \mid \boldsymbol{d}_i\right) g\left(\boldsymbol{d}_i \mid \boldsymbol{d}_k\right)}\Bigg\} \\
&= \min\left\{1,\ \frac{\prod_{j=1}^{J_i} u(\boldsymbol{d}_k, y_j, \theta_j) \prod_{j=1}^{J_k} u(\boldsymbol{d}_i, y_j, \theta_j)}{\prod_{j=1}^{J_i} u(\boldsymbol{d}_i, y_j, \theta_j) \prod_{j=1}^{J_k} u(\boldsymbol{d}_k, y_j, \theta_j)}\right\} \\
&= \min\Bigg\{1,\ \exp\Bigg[\sum_{j=1}^{J_i} \log u\left(\boldsymbol{d}_k, y_j, \theta_j\right) + \sum_{j=1}^{J_k} \log u\left(\boldsymbol{d}_i, y_j, \theta_j\right) - \sum_{j=1}^{J_i} \log u\left(\boldsymbol{d}_i, y_j, \theta_j\right) \\
&\qquad - \sum_{j=1}^{J_k} \log u\left(\boldsymbol{d}_k, y_j, \theta_j\right)\Bigg]\Bigg\} \\
&= \min\left\{\ 1,\ \exp\left[J_i\nu_k + J_k\nu_i - J_i\nu_i - J_k\nu_k)\right]\ \right\} \\
&= \min\left\{\ 1,\ \exp\left[-(\nu_i - \nu_k)(J_i - J_k)\right]\ \right\}.
\end{aligned}
$$

A seguir é dado um resumo dos passos do algoritmo *eMCMC* proposto:

passo 0 : Assuma que o estado corrente da i-ésima cadeia, $i \in \{1, \cdots, N\}$, é $(J_i, \mathbf{d}_i, y_1, \theta_1, \cdots, y_{J_i}, \theta_{J_i})$ ou $(\boldsymbol{d}_i, \nu_i)$.

passo 1 : Propor novos desenhos aplicando os operadores de mutação e cruzamento à população de desenhos $\mathcal{D}$, com probabilidades q_{mut} e $1 - q_{mut}$, respectivamente.

passo 2 : Tentar permutar os desenhos $\boldsymbol{d}_i$ e $\boldsymbol{d}_k$ para N pares (i, k).

passo 3 : Para cada nova proposta $\tilde{\boldsymbol{d}}_i$, simular $(\tilde{\theta}_j, \tilde{y}_j)$ de $p(\boldsymbol{\theta}, \boldsymbol{y} \mid \tilde{\boldsymbol{d}}_i)$, com $j = 1, \cdots, J_i$.

passo 4 : Calcular $\tilde{\nu}_i$ e registrar $(\tilde{\boldsymbol{d}}_i, \tilde{\nu}_i)$ como um novo estado proposto da i-ésima cadeia de Markov.

passo 5 : Aceitar as propostas geradas pelos operadores de mutação, cruzamento e permutação, com suas correspondentes probabilidades de aceitação, $\alpha_{\mathrm{J_{mut}}}$, $\alpha_{\mathrm{J_{cross}}}$ e $\alpha_{\mathrm{J_{exch}}}$.

passo 6 : Repetir os passos 2 a 6 até convergência.

4.4 Uma aplicação Ambiental

O algoritmo proposto foi comparado com o de Müller et al. (2004), considerando o problema de redução do tamanho de uma rede de estações de monitoramento de ozônio no leste dos Estados Unidos. Os dados para este estudo foram obtidos da Agência de proteção ambiental americana (EPA). A variável resposta de interesse foi a concentração máxima diária de ozônio em partes por bilhão (ppb), obtida a partir de dados de ozônio medidos a cada hora segundo o seguinte procedimento: primeiro foi calculada a concentração média de ozônio para cada um dos 17 grupos de 8 horas consecutivas que podem ser formados com as 24 horas de um dia. A concentração máxima diária de ozônio correspondeu ao máximo desses 17 valores médios. Foram consideradas um total de $n = 478$ estações de monitoramento durante um período de 30 dias no verão de 1999 (de Julho 01 a Julho 30). Uma transformação raiz quadrada foi usada antes de realizar a análise, para aproximar normalidade e estabilizar a variância dos dados. As localizações das estações consideradas nesta aplicação são mostradas na Figura 4.2.

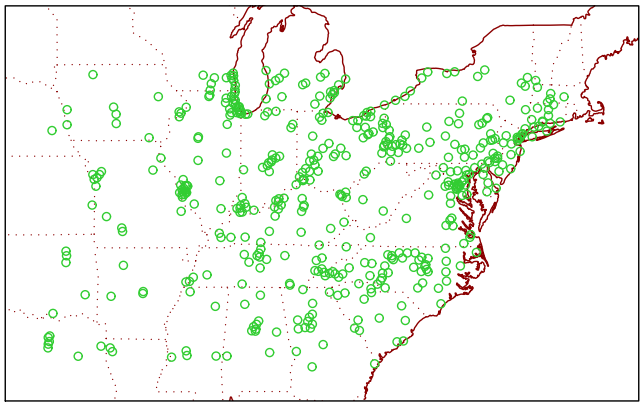


Figura 4.2: Localização das 478 estações monitoradoras no leste dos Estados Unidos.

4.4.1 O modelo espaço-tempo

Como modelo de probabilidade para os dados foi adotado o mesmo modelo espaço-tempo descrito em Müller et al. (2004), isto é, um modelo de convolução de processos (Higdon, 2002). Seja $y_t(s), s \in S$ o nível de ozônio no dia t na localização s. A idéia principal deste modelo é a construção de um processo Gaussiano $y(s)$ na região espacial S convoluindo um processo de ruído branco contínuo $\theta(s)$ com um *kernel* suavizador $k(s)$. Por simplicidade foram assumidas distribuições a priori independentes e conjugadas para os parâmetros do modelo. Assim, o modelo foi especificado como

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{K}\boldsymbol{\theta}_t + \boldsymbol{\varepsilon}_t & K_{ij} &= k(s_i - \omega_j) \\ \boldsymbol{\theta}_t &\sim N(\mathbf{0}, \lambda_\theta \boldsymbol{I}_m) & \lambda_\theta &\sim IG(\alpha_\theta, \beta_\theta) \\ \boldsymbol{\varepsilon}_t &\sim N(\mathbf{0}, \lambda_{\boldsymbol{\varepsilon}} \boldsymbol{I}_n) & \lambda_{\boldsymbol{\varepsilon}} &\sim IG(\alpha_{\boldsymbol{\varepsilon}}, \beta_{\boldsymbol{\varepsilon}}) \end{aligned} \tag{4.4}$$

onde $\boldsymbol{I}_n$ é a matriz identidade n-dimensional, $\boldsymbol{K}$ é uma matriz de *kernels*, $IG(.,.)$ denota a distribuição gama invertida, e α_θ, β_θ, $\alpha_{\boldsymbol{\varepsilon}}$ e $\beta_{\boldsymbol{\varepsilon}}$ são hiperparâmetros conhecidos. Os valores destes hiperparâmetros foram escolhidos de modo que fornecessem prioris pouco informativas. Assim, foram selecionados os valores $\alpha_{\boldsymbol{\varepsilon}} = 0.001$, $\beta_{\boldsymbol{\varepsilon}} = 0.001$, $\alpha_\theta = 0.1$ e $\beta_\theta = 0.1$. Outros valores testados para esses hiperparâmetros forneceram resultados similares (não mostrados aqui).

O domínio de $\theta_t(s)$ foi restrito a uma grade regular de $m = 26$ nos, $\{\omega_1, \cdots, \omega_{26}\}$, sobre a região de estudo, como mostrado na Figura 4.3. O *kernel* suavizador escolhido foi um *kernel* Gaussiano

com média zero, i.e.,

$$k(s) = (2\pi\sigma^2)^{-1} \exp\left\{-\frac{1}{2\sigma^2}\, s^2\right\},$$

cujas distâncias s foram transformadas para $s' = s/\sigma$ para obter isotropia.

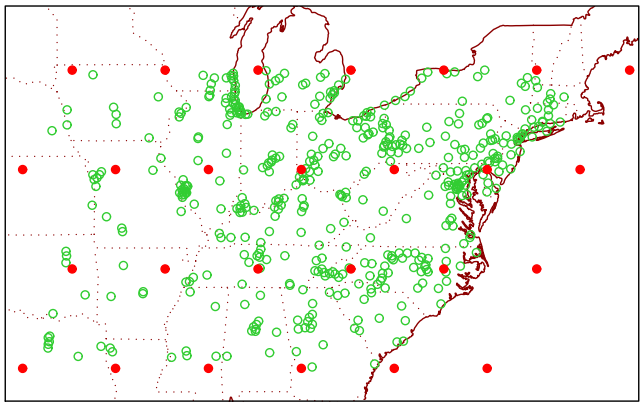


Figura 4.3: Localização dos 26 pontos de suporte do processo latente [pontos vermelhos] usados para construir o processo Gaussiano no modelo de convolução. Os pontos verdes representam as 478 estações.

Para formar o modelo espaço-temporal foi assumido que os $\boldsymbol{\theta}_t$, $t = 1, 2, \cdots, Z$, são *a priori* independentes, e que a priori condicional $p\left(\boldsymbol{\theta}_{Z+1} \mid \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_Z\right)$ é definida como $p\left(\boldsymbol{\theta}_{Z+1} = \boldsymbol{\theta}_t\right) = 1/Z$. Note-se que este é apenas um modelo para avaliar os desenhos propostos, já que o principal interesse é o problema de decisão de escolher o melhor desenho de rede. Outro tipo de modelos por exemplo, para analisar os dados de ozônio em espaço e tempo visando identificar tendências ou fazer previsões mais precisas, requereria abordagens mais sofisticadas, que estão além dos objetivos deste estudo.

Seguindo o teorema de Bayes, a distribuição a posteriori conjunta de todos os parâmetros do modelo é proporcional ao produto da verossimilhança e as distribuições a priori, ,isto é,

$$p(\boldsymbol{\Theta}, \lambda_\varepsilon, \lambda_\theta \mid \boldsymbol{Y}) \propto \prod_{t=1}^{Z} p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t, \lambda_\varepsilon) \prod_{t=1}^{Z} p(\boldsymbol{\theta}_t \mid \lambda_\theta) p(\lambda_\varepsilon) p(\lambda_\theta)$$

onde $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_Z\}$, $\boldsymbol{\theta}_t = \{\theta_{t1}, \cdots, \theta_{tm}\}$, $\boldsymbol{Y} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_Z\}$ e $\boldsymbol{y}_t = \{y_{t1}, \cdots, y_{tn}\}$, para $t \in \{1, \cdots, Z\}$.

O amostrador de Gibbs (Gelfand e Smith, 1990) foi usado para explorar esta distribuição conjunta. Este algoritmo constrói uma cadeia de Markov que amostra iterativamente das distribuições

condicionais completas dos parâmetros do modelo. Para o modelo espaço-tempo considerado, estas distribuições condicionais completas podem ser, todas, obtidas em forma fechada:

A distribuição condicional completa dos $\boldsymbol{\theta}$'s é dada por:

$$
\begin{aligned}
p(\boldsymbol{\theta}_t \mid ...) &\propto p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t, \lambda_\varepsilon) p(\boldsymbol{\theta}_t \mid \lambda_\theta) \\
&\propto \exp\left\{-\frac{1}{2\lambda_\varepsilon}\,(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)'\boldsymbol{I}_n^{-1}(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)\right\} \exp\left\{-\frac{1}{2\lambda_\theta}\,\boldsymbol{\theta}_t'\boldsymbol{I}_m^{-1}\boldsymbol{\theta}_t\right\} \\
&\propto exp\left\{-\frac{1}{2}\left[(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t) + \boldsymbol{\theta}_t'\lambda_\theta^{-1}\boldsymbol{I}_m^{-1}\boldsymbol{\theta}_t\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{y}_t'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}\boldsymbol{y}_t - 2\boldsymbol{y}_t'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}\boldsymbol{K}\boldsymbol{\theta}_t + (\boldsymbol{K}\boldsymbol{\theta}_t)'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}(\boldsymbol{K}\boldsymbol{\theta}_t) + \boldsymbol{\theta}_t'\lambda_\theta^{-1}\boldsymbol{I}_m^{-1}\boldsymbol{\theta}_t\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\theta}_t'\boldsymbol{K}'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}\boldsymbol{K}\boldsymbol{\theta}_t + \boldsymbol{\theta}_t'\lambda_\theta^{-1}\boldsymbol{I}_m^{-1}\boldsymbol{\theta}_t - 2\boldsymbol{\theta}_t'\boldsymbol{K}'\lambda_\varepsilon^{-1}\boldsymbol{I}_n^{-1}\boldsymbol{y}_t\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\theta}_t'\underbrace{(\lambda_\theta^{-1}\boldsymbol{I}_m + \lambda_\varepsilon^{-1}\boldsymbol{K}'\boldsymbol{I}_n^{-1}\boldsymbol{K})}_{\boldsymbol{C}^{-1}}\boldsymbol{\theta}_t - 2\boldsymbol{\theta}_t'\underbrace{(\lambda_\varepsilon^{-1}\boldsymbol{K}'\boldsymbol{I}_n^{-1}\boldsymbol{y}_t)}_{\boldsymbol{C}^{-1}\boldsymbol{M}}\right]\right\}.
\end{aligned}
$$

Portanto, $(\boldsymbol{\theta}_t \mid ...) \sim N_m(\boldsymbol{M}, \boldsymbol{C})$,

com $\boldsymbol{C} = \left(\lambda_\theta^{-1}\boldsymbol{I}_m + \lambda_\varepsilon^{-1}K'\boldsymbol{I}_n^{-1}\boldsymbol{K}\right)^{-1}$, e $\boldsymbol{M} = \boldsymbol{C}\left(\lambda_\varepsilon^{-1}\boldsymbol{K}'\boldsymbol{I}_n^{-1}\boldsymbol{y}_t\right)$.

A distribuição condicional completa de λ_θ é dada por:

$$
\begin{aligned}
p(\lambda_\theta \mid ...) &\propto \prod_{t=1}^{Z} p(\boldsymbol{\theta}_t \mid \lambda_\theta) p(\lambda_\theta) \\
&\propto \left(\lambda_\theta^{-\frac{m}{2}}\right)^Z \exp\left\{-\frac{1}{2\lambda_\theta}\sum_{t=1}^{Z}(\boldsymbol{\theta}_t'\boldsymbol{\theta}_t)\right\} \lambda_\theta^{-(\alpha_\theta+1)} \exp\left\{-\frac{\beta_\theta}{\alpha_\theta}\right\} \\
&\propto \lambda_\theta^{-\left(\alpha_\theta + \frac{mZ}{2} + 1\right)} \exp\left\{-\frac{1}{\lambda_\theta}\left(\beta_\theta + \frac{\sum_{t=1}^{Z}(\boldsymbol{\theta}_t'\boldsymbol{\theta}_t)}{2}\right)\right\}.
\end{aligned}
$$

Portanto, $(\lambda_\theta \mid ...) \sim IG\left(\alpha_\theta + \frac{mZ}{2}, \beta_\theta + \frac{1}{2}\sum_{t=1}^{Z}(\boldsymbol{\theta}_t'\boldsymbol{\theta}_t)\right)$.

A distribuição condicional completa de λ_ε é dada por:

$$
\begin{aligned}
p(\lambda_\varepsilon \mid ...) &\propto \prod_{t=1}^{Z} p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t, \lambda_\varepsilon) p(\lambda_\varepsilon) \\
&\propto \left(\lambda_\varepsilon^{-\frac{n}{2}}\right)^{Z} \exp\left\{-\frac{1}{2\lambda_\theta}\sum_{t=1}^{Z}(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)'(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)\right\} \lambda_\varepsilon^{-(\alpha_\varepsilon+1)} \exp\left\{-\frac{\beta_\varepsilon}{\alpha_\varepsilon}\right\} \\
&\propto \lambda_\varepsilon^{-\left(\alpha_\varepsilon+\frac{nZ}{2}+1\right)} \exp\left\{-\frac{1}{\lambda_\varepsilon}\left(\beta_\varepsilon + \frac{\sum_{t=1}^{Z}(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)'(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)}{2}\right)\right\}.
\end{aligned}
$$

Portanto, $(\lambda_\varepsilon \mid ...) \sim IG\left(\alpha_\varepsilon + \frac{nZ}{2}, \beta_\varepsilon + \frac{1}{2}\sum_{t=1}^{Z}(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)'(\boldsymbol{y}_t - \boldsymbol{K}\boldsymbol{\theta}_t)\right)$.

Após rodar o algoritmo por um número suficiente de iterações de "aquecimento", as amostras reproduzem a distribuição a posteriori conjunta.

4.4.2 A Função de Utilidade

Para formalizar o problema de decisão, foi considerada uma função de utilidade similar àquela usada em Müller et al. (2004), a qual leva em consideração os seguintes objetivos conflitantes: (i) identificar locais com níveis médios de ozônio acima do padrão de qualidade do ar para ozônio de $y^* = 85$ partes por bilhão; (ii) estimar níveis de ozônio em estações que foram removidas da rede; (iii) fazer inferência sobre a superfície de resposta do nível médio de ozônio para toda a região leste dos Estados Unidos e (iv) minimizar custos. Portanto,

$$
\begin{aligned}
u(\boldsymbol{d}, \boldsymbol{\theta}, \boldsymbol{y}) = R \sum_{i \in \boldsymbol{D}^c} \mathbf{1}\{(y_{Z+1,i} > y^*) \text{ e } (\hat{y}_i > y^*)\} \; - \; C_1 \times \sum_{i \in \boldsymbol{D}^c} (y_{Z+1,i} - \hat{y}_i)^2 \\
- \; C2 \times \int_S (\hat{f}(s) - f(s;\theta))^2 ds \; - \; \sum_{i \in \boldsymbol{D}} c_i \; + \; C_0. \qquad (4.5)
\end{aligned}
$$

Os objetivos conflitantes citados acima estão representados pelos primeiros quatro termos da equação (4.5), onde: $\mathbf{1}\{A\}$ denota a função indicadora para o evento A; $\boldsymbol{y}^o = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_Z)$ denota os dados observados; $\boldsymbol{d} = (d_1, \cdots, d_n)$ é um vetor de indicadores especificando quais estações permanecem na rede $(d_i = 1)$ e quais não $(d_i = 0)$; $\boldsymbol{D} = \{i : d_i = 1\}$ denota o conjunto de índices das estações selecionadas para permanecer na rede; $\boldsymbol{D}^c = \{i : d_i = 0\}$ denota o conjunto de

índices das estações removidas da rede; $\boldsymbol{y}_{Z+1}$ denota todos os dados futuros; $\boldsymbol{y_d} = (\boldsymbol{y}_{Z+1,i}, i \in \boldsymbol{D})$ denota os dados futuros nas estações selecionadas; $\boldsymbol{\theta}_{Z+1}$ denota os parâmetros para a superfície de resposta média no tempo $Z+1$; $\hat{\boldsymbol{y}} = E\left[\boldsymbol{y}_{Z+1} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right]$ denota o vetor de respostas futuras médias estimadas; $f(z;\boldsymbol{\theta}) = \boldsymbol{K}_{(z)}\boldsymbol{\theta}_{Z+1}$ denota a superfície de resposta média avaliada em z e $\hat{f}(z) = E\left[f(z;\boldsymbol{\theta}) \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right]$. O termo constante C_0 é adicionado para assegurar que a função de utilidade seja não negativa. A integral no terceiro termo foi aproximada por uma soma sobre uma grade regular de 145 pontos, como mostrado na figura 4.4. Os coeficientes R, C_1 and C_2 definem a importância relativa dos objetivos conflitantes. Nesta aplicação foram considerados dois casos:

caso I: os coeficientes foram fixados em $R = 100$, $C1 = 0.01$, $C2 = 1$. Esta escolha dá prioridade ao primeiro objetivo da função de utilidade, isto é, desenhos que possam predizer com mais precisão observações extremas para sítios não incluídos na rede serão favorecidos pela função de utilidade.

caso II: os coeficientes foram fixados em $R = 10$, $C1 = 1$, $C2 = 10$, diminuindo as diferenças entre os pesos das diferentes componentes da função de utilidade.

O coeficiente c_i representa o custo de incluir a estação i, o qual foi mantido constante para todas as estações nos dois casos, com $c_i = 2$. Deve ser lembrado que a especificação da função de utilidade sempre é em algum sentido arbitrária, e várias outras escolhas são possíveis.

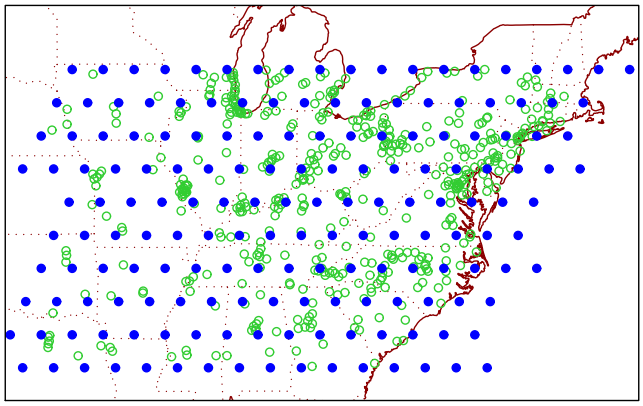


Figura 4.4: Localização dos 145 pontos de suporte (em azul) usados para interpolar a superfície de resposta média da função de utilidade (ver o texto para mais detalhes). Os pontos verdes representam as 478 estações monitoradoras.

Para o cálculo da função de utilidade foram seguidos os seguintes passos:

Assuma que na g-ésima iteração do algoritmo MCMC tem-se: $\boldsymbol{d}^{(g)}$, $\lambda_{\varepsilon}^{(g)}$, J.

1. Para $j \in \{1, \cdots, J\}$, defina

$$\boldsymbol{\varepsilon}_{(j)}^{(g)} = \left\{\varepsilon_{1,(j)}^{(g)}, \cdots, \varepsilon_{n,(j)}^{(g)}\right\}$$

$$\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \left\{\theta_{Z+1,1,(j)}^{(prop)}, \cdots, \theta_{Z+1,m,(j)}^{(prop)}\right\}$$

$$\boldsymbol{y}_{Z+1,(j)}^{(prop)} = \left\{y_{Z+1,1,(j)}^{(prop)}, \cdots, y_{Z+1,n,(j)}^{(prop)}\right\}$$

2. Para $i \in \boldsymbol{D}^c$ calcule

$$\begin{aligned}
\hat{y}_{i,(j)} &= E\left[y_{Z+1,i,(j)}^{(prop)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] \\
&= E\left[\boldsymbol{K}_{1(s_i)}\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} + \varepsilon_{i,(j)}^{(g)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] \\
&= \boldsymbol{K}_{1(s_i)} E\left[\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] + E\left[\varepsilon_{i,(j)}^{(g)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] \\
&= \boldsymbol{K}_{1(s_i)} E\left[\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right]
\end{aligned}$$

3. calcule $f(z;\boldsymbol{\theta}) = K_{2(z)}\boldsymbol{\theta}_{Z+1,(j)}^{(prop)}$;

4. calcule $\hat{f}(z) = E\left[f(z;\boldsymbol{\theta}) \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] = \boldsymbol{K}_{2(z)} E\left[\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right]$,

onde: $E\left[\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right] = \sum_{t=1}^{Z} \boldsymbol{\theta}_t^{(g)} p\left(\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right)$

com

$$\begin{aligned}
p\left(\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)} \mid \boldsymbol{y}^o, \boldsymbol{y_d}\right) &\propto p\left(\boldsymbol{y_d} \mid \boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)}\right) p\left(\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)} \mid \boldsymbol{y}^o\right) \\
&\propto p\left(\boldsymbol{y_d} \mid \boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)}\right) \\
&\propto N\left(\boldsymbol{y_d} \mid \boldsymbol{K}_{(\boldsymbol{d})}\boldsymbol{\theta}_t^{(g)}, \lambda_{\varepsilon}^{(g)}\boldsymbol{I_d}\right),
\end{aligned}$$

e $p\left(\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} = \boldsymbol{\theta}_t^{(g)} \mid \boldsymbol{y}^o\right) = \dfrac{1}{Z}$, para $t \in \{1, \cdots, Z\}$.

Assim, tem-se que

$$E\left[\boldsymbol{\theta}_{Z+1,(j)}^{(prop)} \mid \boldsymbol{y}^{o}, \boldsymbol{y_d}\right] = \frac{\sum_{t=1}^{Z} \boldsymbol{\theta}_t^{(g)} N\left(\boldsymbol{y_d} \mid \boldsymbol{K}_{(\boldsymbol{d})} \boldsymbol{\theta}_t^{(g)}, \lambda_{\varepsilon}^{(g)} \boldsymbol{I_d}\right)}{\sum_{t=1}^{Z} N\left(\boldsymbol{y_d} \mid \boldsymbol{K}_{(\boldsymbol{d})} \boldsymbol{\theta}_t^{(g)}, \lambda_{\varepsilon}^{(g)} \boldsymbol{I_d}\right)}.$$

O tamanho da rede reduzida foi fixado em 150 estações. O algoritmo de Müller foi implementado para um total de 25000 iterações, descartando as primeiras 5000 como aquecimento inicial da cadeia. A partir daí, o coeficiente J foi incrementado em uma unidade a cada 1000 iterações, levando a um poder máximo de $J = 20$. A distribuição geradora das propostas, $g(\tilde{\boldsymbol{d}} \mid \boldsymbol{d})$, foi definida selecionando aleatoriamente, a cada iteração, um dos sítios da rede corrente, e permutando-o com um dos seus 10 vizinhos mais próximos. Já o algoritmo *eMCMC* foi implementado durante 1000 gerações, com um tamanho populacional (número de cadeias de Markov) de $N = 20$. A taxa de mutação foi fixada em $q_m = 0.2$ após algumas rodadas exploratórias do algoritmo. Como no algoritmo de Müller, foi usado um esquema de vizinhança dos 10 vizinhos mais próximos para escolher as novas estações no operador de mutação. Um esquema de seleção proporcional ao ajuste (equação 4.3) foi usado para escolher o primeiro desenho de cada par no operador de cruzamento.

Os algoritmos foram implementados usando a linguagem de programação Ox versão 3.3 (Doornik, 2002) rodando sob Windows XP. As análises foram feitas em um PC com processador AMD Athlon XP 3000+, a 2.1GHz e 512MB de memória RAM.

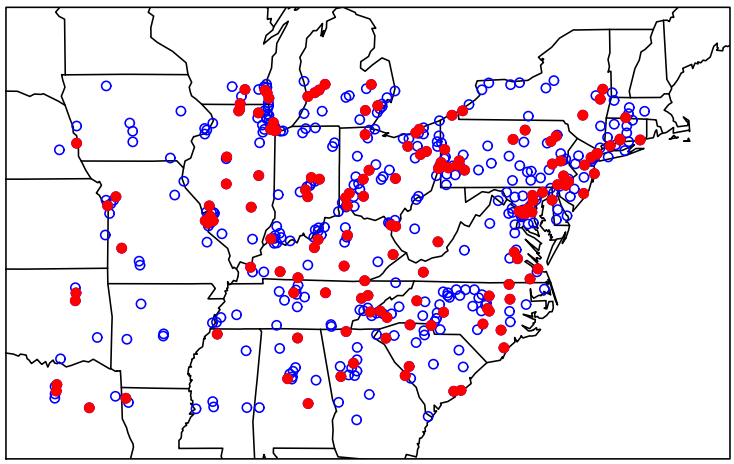
Os resultados obtidos até agora, apresentados na Tabela 4.1 e nas Figuras 4.5 e 4.6, mostraram um melhor desempenho do algoritmo $eMCMC$ quando comparado com o algoritmo de Müller, para a aplicação ambiental considerada aqui, em termos do valor de função objetivo ν a ser maximizada.

A rede desenhada pelo algoritmo *eMCMC* foi também capaz de predizer mais eficientemente observações extremas para sítios não incluídos na rede, como mostrado na coluna P_1 da Tabela 4.1.

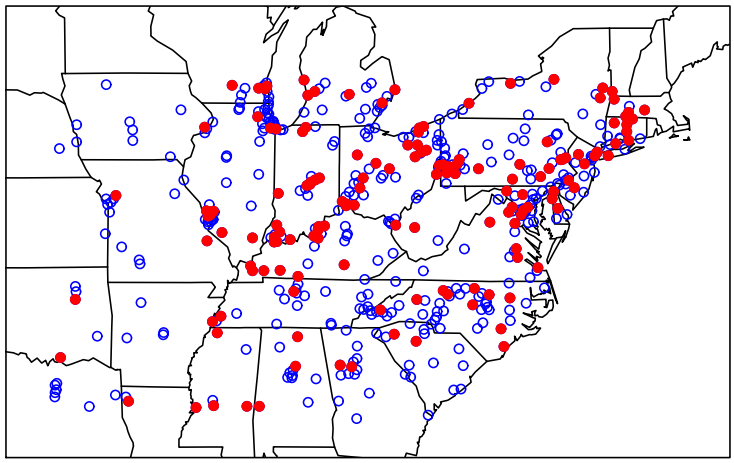
Tabela 4.1: Resultados obtidos com o algoritmo de Müller e com o algoritmo *eMCMC* proposto, para os dois casos considerados na aplicação ambiental

Caso	Algoritmo	P_1	P_2	P_3	P_4	$\exp\{\nu\}$	CPU
I	Müller	1450	-6.1055	-9.0905	-300	3617.01	51773
	eMCMC	3400	-5.6343	-0.0107	-300	$\mathbf{5594.29}$	$\mathbf{46213}$
II	Müller	90	$-518,56$	$-0,1727$	-300	4271.41	66450
	eMCMC	160	$-457,03$	-0.0052	-300	$\mathbf{4402,82}$	$\mathbf{59824}$

P_1, P_2, P_3 e P_4 correspondem aos valores médios para as primeiras quatro componentes da função de utilidade; ν é a média do logaritmo da utilidade observada; *CPU* é o tempo computacional (em segundos).

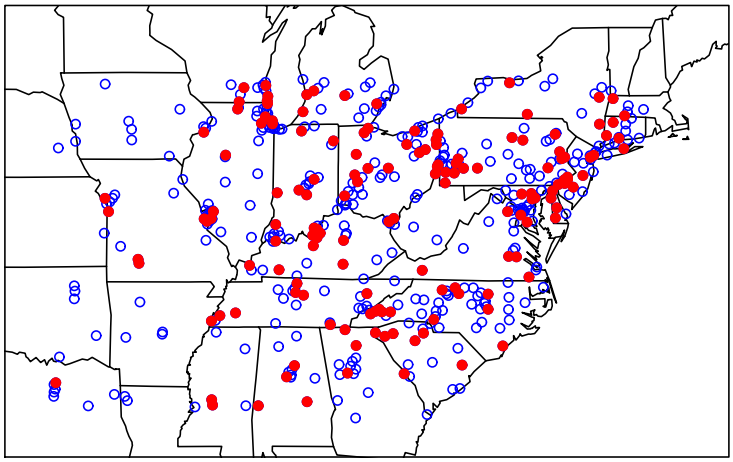


(a) algoritmo de Müller

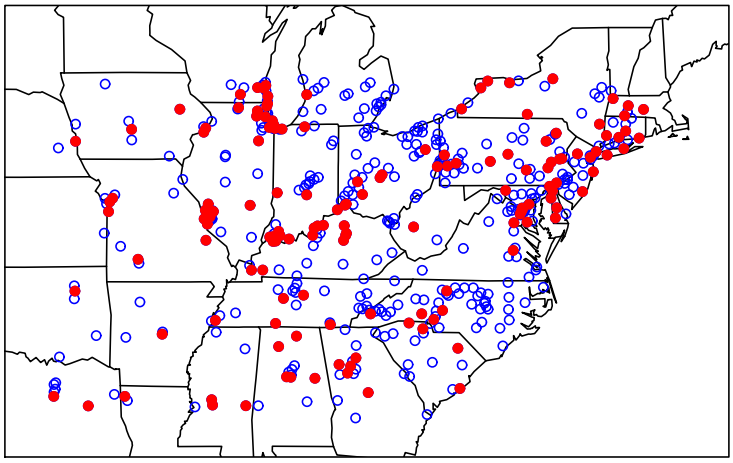


(b) algoritmo *eMCMC*

Figura 4.5: Melhores desenhos de redes encontrados com o algoritmo de Müller (a) e com o algoritmo *eMCMC* (b) para o primeiro caso da aplicação ambiental. Os pontos vermelhos correspondem às estações que permaneceram na rede; os pontos azuis correspondem às estações que foram removidas da rede.



(a) algoritmo de Müller



(b) algoritmo *eMCMC*

Figura 4.6: Melhores desenhos de redes encontrados com o algoritmo de Müller (a) e com o algoritmo *eMCMC* (b) para o segundo caso da aplicação ambiental. Os pontos vermelhos correspondem às estações que permaneceram na rede; os pontos azuis correspondem às estações que foram removidas da rede.

4.5 Considerações Finais

Neste capítulo foi apresentado um algoritmo evolucionário de Monte Carlo implementado via cadeias de Markov (*eMCMC*), desenvolvido para resolver problemas de maximização da utilidades esperadas em espaços de alta dimensão. Devido ao uso de uma população que explora simultaneamente diferentes regiões do espaço de busca, e à incorporação de operadores genéticos, os quais permitem a troca de informação entre as diferentes soluções ao longo das simulações, o algoritmo consegue fazer uma exploração mais eficiente do espaço de busca. O algoritmo *eMCMC* foi comparado com o algoritmo de Müller et al. (2004), que opera com uma única cadeia de Markov e usa *simulated annealing* no processo de otimização, usando dados reais de uma rede de monitoramento ambiental de larga escala. Os resultados computacionais mostraram que o algoritmo *eMCMC* teve um desempenho superior ao algoritmo de Müller para os casos considerados do problema, em termos de qualidade das soluções obtidas.

Embora a aplicação do algoritmo *eMCMC* apresentada aqui esteve restrita a um contexto de desenho de redes de monitoramento, o algoritmo é igualmente aplicável a outros problemas de otimização combinatória de alta dimensão, e em geral, a problemas cujo espaço de busca possa ser representado como uma coleção de seqüências binárias. A abordagem de simulação da utilidade esperada usando uma distribuição artificial, pode também ser acoplada a outros esquemas de Monte Carlo evolucionários recentemente desenvolvidos, tais como os de Strens (2003) e Drugan e Thierens (2004) para otimização em espaços discretos, e os de Liang e Wong (2001) e Hu e Tsui (2005) para otimização em espaços contínuos. Isto será objeto de futura pesquisa.

O algoritmo *eMCMC* apresentado aqui foi implementado de forma sequencial, isto é, avaliando somente uma solução por vez. No entanto, as características de algoritmos evolucionários (o fato de trabalhar com uma população de soluções, o uso de operadores genéticos, etc.), fazem dele um algoritmo inerentemente paralelizável, o que significa que, a cada iteração, os indivíduos (soluções) da população podem ser avaliados simultaneamente. Assim, uma implementação em paralelo deste algoritmo aceleraria significativamente o tempo de convergência para a moda de $U(.)$.

Capítulo 5

Mapeamento de QTL

5.1 Introdução

O uso de métodos Bayesianos em aplicações genéticas tem se tornado cada vez mais popular (Shoemaker et al., 1999; Beaumont e Rannala, 2004), particularmente na análise de regiões do genoma responsáveis pela variação em uma característica quantitativa (contínua) de interesse em um organismo. Através de técnicas moleculares, é possível identificar marcadores genéticos no genoma, isto é, regiões neutras cuja posição no genoma é conhecida, e que podem ser usados para a detecção de outras regiões no genoma associadas a uma característica específica de interesse.

Neste tipo de análise, conhecida como mapeamento de QTL (*quantitative trait loci*), a inferência geralmente consiste em identificar o número e localização destas regiões no genoma (comumente chamadas de *loci* ou *locus* em singular), assim como seus efeitos genéticos condicionados aos dados fenotípicos observados e aos dados dos marcadores. A idéia principal é que os marcadores genéticos que tendem a ser herdados conjuntamente com os valores específicos de uma característica, provavelmente estejam próximos às regiões no genoma que afetam essa característica.

Para populações provenientes de desenhos experimentais conhecidos, é possível calcular exatamente a probabilidade de que um locus, posicionado perto de um marcador de genótipo conhecido, tenha um genótipo diretamente associado com a característica de interesse. Através desta associação, as regiões do cromossomo que influenciam diretamente a característica de interesse (os QTL) podem ser identificadas para pesquisas futuras. Veja por exemplo Doerge et al. (1997) e Lynch e Walsh

(1998) para uma revisão sobre métodos de estimação de QTL e a bibliografia relacionada.

No entanto, o mapeamento de QTL é complicado pelo fato de que o número de QTL e, portanto, a dimensionalidade do espaço paramétrico são desconhecidos. Entre os métodos Bayesianos adotados para abordar este problema destaca-se o algoritmo MCMC com saltos reversíveis, RJMCMC, (Green, 1995), o qual é capaz de explorar o espaço dos modelos, permitindo que sua dimensão (o número de QTL) mude como parte da cadeia de Markov (e.g. Satagopan et al., 1996; Uimari e Hoeschele, 1997; Sillanpää e Arjas, 1998; Stephens e Fisch, 1998; George et al., 2000; Gaffney, 2001; Waagepetersen e Sorensen, 2001). No entanto, de acordo com Yi (2004), as dificuldades de implementação e sintonização de parâmetros do algoritmo e o fato de usualmente estar sujeito a uma mistura pobre e uma convergência lenta, tem atenuado a generalização de seu uso em aplicações genéticas. Alternativas recentes para a análise de QTL ainda no contexto Bayesiano incluem os trabalhos de Sen e Churchill (2001), Yi (2004); Yi et al. (2005); Wang et al. (2005); Zhang et al. (2005), entre outros.

Os resultados da análise Bayesiana, independentemente do método usado, são geralmente apresentados em termos de distribuições a posteriori marginais ou funções diretas destas, tais como o fator de Bayes ou distribuições a posteriori condicionais. Usualmente, aplicações de mapeamento de QTL usam amostras destas distribuições para construir histogramas que auxiliem na visualização da distribuição da posição dos QTL. No entanto, em algumas situações, tais como no mapeamento comparativo [1], o principal objetivo é a localização da moda a posteriori desta distribuição. Neste caso o interesse primário é encontrar o melhor marcador candidato ou a melhor posição putativa no mapa dos marcadores genéticos. A aplicação inadequada do histograma nestes casos pode levar a conclusões errôneas (Slager et al., 2001).

Algumas formas de obtenção da moda a posteriori da posição dos QTL têm sido propostas recentemente na literatura. Por exemplo, Hoti et al. (2002) sugerem o uso de um *kernel* suavizador para resumir a distribuição a posteriori da posição dos QTL de uma amostra MCMC. Sillanpää e Arjas (1998) propõem uma abordagem baseada na intensidade dos QTL, assumindo que a localização

[1] Mapeamento comparativo: abordagem baseada na observação que posições no mapa genético de genes homólogos são parcialmente conservadas através das espécies, o qual sugere que regiões candidatas para um QTL numa espécie são também regiões candidatas para um QTL em espécies relacionadas.

desses QTL segue um processo de Poisson não homogêneo sobre os cromossomos, e estimando essa intensidade a partir de amostras geradas por uma análise MCMC com saltos reversíveis.

Em outro contexto, Sisson e Hurn (2004) apresentam uma abordagem para estimação pontual no genoma via teoria de decisão Bayesiana, para estimar o número e a localização dos QTL. Eles sugerem uma função perda baseada nas idéias de Celeux et al. (2000), cujo estimador Bayes (aquele que minimiza a perda a posteriori esperada) não pode ser avaliado explicitamente. Eles tentam aproximá-lo através da decomposição da esperança da função perda em sub-funções que possam ser avaliadas explicitamente, aplicando um algoritmo em dois passos, que usa primeiro um algoritmo MCMC (para a parte de estimação) e posteriormente um algoritmo *simulated annealing* (para a parte de otimização), no espírito de Rue (1995).

Seguindo o contexto de teoria da decisão Bayesiana, neste capítulo é proposto um novo algoritmo para a minimização da perda esperada em espaços de alta dimensão. Um modelo de probabilidade aumentado é usado para derivar a perda esperada, enquanto que a minimização é feita através de um algoritmo evolucionário. Diferentemente de Sisson e Hurn (2004), o novo algoritmo realiza conjuntamente as tarefas de estimação e otimização, e permite considerar diretamente funções perda cujas esperanças não são integráveis analiticamente, não precisando decompor o problema em subproblemas explicitamente avaliáveis, o que nem sempre é possível dependendo da função perda escolhida. O algoritmo desenvolvido foi comparado com o proposto por Sisson e Hurn (2004), na estimação do número e localização de QTL usando dados simulados de uma população controlada gerada a partir de um conhecido desenho experimental.

5.2 Conceitos básicos de genética

A informação genética de um organismo vivo reside em cadeias de DNA no interior de cada célula, agrupadas em estruturas chamadas *cromossomos*, os quais podem ser divididos em uma ou mais subregiões chamados *genes*, responsáveis por alguma ação ou atributo do organismo ao qual pertencem. As diferentes configurações que um gene pode assumir são os *alelos* desse gene. A posição de cada gene no cromossomo é chamado de *locus* (*loci* no plural). A coleção de todo o material genético de um organismo constitui o seu *genoma*.

Quando os cromossomos de um organismo estão arranjados em pares, ele é dito ser *diplóide*. O *genótipo* de um indivíduo para um gene em particular é o conjunto de alelos que ele possui para esse gene. Assim, para um indivíduo diplóide, os dois alelos de cada gene constituirão o seu genótipo. Os atributos físicos de um organismo constituem seu fenótipo e são determinados pelo seu genótipo e pela interação deste com o ambiente em que ele habita.

Os cromossomos que fazem parte de um mesmo par são chamados de cromossomos *homólogos*. Genes que estiverem localizados em posições correspondentes em dois cromossomos homólogos são chamados de genes *alélicos*. Quando um indivíduo possui dois genes alélicos iguais, ele é chamado de *homozigoto*, podendo ser *dominante* ou *recessivo*. Por outro lado, se o organismo possui genes alélicos diferentes, ele é chamado de *heterozigoto*. Usualmente, emprega-se letras maiúsculas para designar genes dominantes, e minúsculas para genes recessivos como segue: AA - homozigoto dominante, aa - homozigoto recessivo, e Aa - heterozigoto.

Os desenhos experimentais usualmente considerados para gerar populações genéticas de forma controlada, assumem o cruzamento inicial de duas linhagens parentais contrastantes, P_1 e P_2, geneticamente puras. Isto é, os indivíduos da linhagem P_1 são homozigotos com genótipo AA, enquanto que os indivíduos da linhagem P_2 são homozigotos com genótipo aa. O cruzamento destas duas linhagens produzirá uma progênie denominada F1 que será heterozigota em cada locus (com genótipo Aa).

O cruzamento entre indivíduos de uma população F1, gerará uma população F2 ou segunda filial, enquanto que o cruzamento de indivíduos da população F1 com alguma das linhagens parentais gerará uma população de retrocruzamento (B1). Estas duas populações são comumente utilizadas em mapeamento de QTL.

5.3 O modelo Bayesiano para QTL

O modelo estatístico aqui adotado foi o proposto por Yi (2004), que usa uma abordagem de espaços compostos (Godsill, 2001) para modelar múltiplos QTL sob um contexto MCMC. Sob esta abordagem que inclui o algoritmo RJMCMC como um caso especial, a simulação MCMC para a estimação do número e localização dos QTL, pode ser implementada, pelo menos conceitualmente, em um espaço

de dimensão fixa, evitando assim a complexidade da abordagem de saltos reversíveis.

Dado que o modelo estatístico e o desenho experimental usados para gerar as populações genéticas não são o foco deste capítulo, foram adotadas algumas simplificações, mas a abordagem proposta é independente do modelo utilizado, permitindo o uso de modelos mais complexos. Assim, assume-se que os dados são provenientes de cruzamentos experimentais controlados, derivados de duas linhagens parentais divergentes geneticamente puras. Os dados observados consistem de um vetor com os valores fenotípicos da característica de interesse para uma população de n indivíduos, $\boldsymbol{y} = \{y_1, \cdots, y_n\}$, e os genótipos dos marcadores, $\boldsymbol{m}$, cuja localização é conhecida a priori, para todos os indivíduos da população. Assume-se que os QTL influenciam esta característica de forma independente, isto é, não existe *epistasia* (interação) entre os QTL. O objetivo é inferir conjuntamente o número de QTL, suas posições no genoma e seus efeitos genéticos.

O modelo considera uma partição do genoma inteiro em H loci, $\boldsymbol{\zeta} = \{\zeta_1, \cdots, \zeta_H\}$, e assume que os possíveis QTL ocorrem nessas posições fixas. Segundo Yi et al. (2005), quando H é grande, esta suposição introduz somente um viés menor. Em geral, os genótipos, $\boldsymbol{g}$, nos H loci não são observáveis, mas suas probabilidades de recombinação, $p(\boldsymbol{g} \mid \boldsymbol{\zeta}, \boldsymbol{m})$, podem ser inferidas a partir dos genótipos dos marcadores observados e dos genótipos das linhagens parentais sendo, portanto, uma função do desenho experimental. Existem tabelas predeterminadas para especificar estas probabilidades de recombinação. Para uma derivação detalhada dessas distribuições veja, por exemplo, Jiang e Zeng (1997) e Lynch e Walsh (1998, capítulo 15).

O problema de inferir o número e a localização dos QTL é equivalente ao problema de selecionar um subconjunto de $\boldsymbol{\zeta}$ que explique da melhor forma possível a variação fenotípica. Yi et al. (2005) assume que, no máximo L QTL contribuem para explicar esta variação, onde tipicamente L é muito menor que H. O vetor $\boldsymbol{\lambda} = \{\lambda_1, \cdots, \lambda_L\} \in \{\zeta_1, \cdots, \zeta_H\}$, denota as posições correntes dos L supostos QTL.

Assume-se que a distribuição do vetor de fenótipos, $\boldsymbol{y}$, segue um modelo linear

$$\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{X\Gamma\beta} + \boldsymbol{\varepsilon},$$

onde $\boldsymbol{\mu}$ é a média geral, $\boldsymbol{\beta}$ denota o vetor de efeitos aleatórios associados aos L potenciais QTL, $\boldsymbol{X}$ é a matriz de desenho, $\boldsymbol{\Gamma} = \text{diag}(\gamma)$, onde γ é uma variável indicadora binária associada a cada efeito aleatório, indicando se o correspondente efeito será incluído ($\gamma = 1$) ou excluído ($\gamma = 0$) do modelo, e $\boldsymbol{\varepsilon}$ é um vetor de erros normais independentes, com média 0, e variância σ^2. O número de QTL iguala o número de uns no vetor $\boldsymbol{\gamma} = \{\gamma_1, \cdots, \gamma_L\}$. A matriz de desenho $\boldsymbol{X}$ depende do desenho experimental e é determinada a partir dos genótipos $\boldsymbol{g}$ nas posições correntes $\boldsymbol{\lambda}$, onde g_{ij} (o genótipo do j-ésimo QTL para o indivíduo i) é homozigoto (AA, aa), ou heterozigoto (Aa) nos alelos das linhagens parentais. Assim, por exemplo, para um cruzamento F2

$$x_{ij} = \begin{cases} 1, & \text{se } g_{ij} = AA, \\ 0, & \text{se } g_{ij} = Aa, \\ -1, & \text{se } g_{ij} = aa. \end{cases}$$

O vetor de parâmetros desconhecidos $(\boldsymbol{\lambda}, \boldsymbol{g}, \boldsymbol{\theta})$ pode ser particionado em $(\boldsymbol{\lambda}_\gamma, \boldsymbol{g}_\gamma, \boldsymbol{\theta}_\gamma)$ e $(\boldsymbol{\lambda}_{-\gamma}, \boldsymbol{g}_{-\gamma}, \boldsymbol{\theta}_{-\gamma})$, onde os subíndices γ e $-\gamma$ representam os parâmetros desconhecidos desse vetor incluídos ou excluídos do modelo respectivamente, com $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\mu}, \sigma^2)$, $\boldsymbol{\theta}_\gamma = (\boldsymbol{\beta}_\gamma, \boldsymbol{\mu}, \sigma^2)$, e $\boldsymbol{\theta}_{-\gamma} = \boldsymbol{\beta}_{-\gamma}$. Similarmente, $\boldsymbol{X}_\gamma$ e $\boldsymbol{X}_{-\gamma}$ representam os coeficientes do modelo incluidos e excluidos respectivamente. Assim, para um vetor γ em particular, a função de verossimilhança é dada por

$$p(\boldsymbol{y} \mid \boldsymbol{\gamma}, \boldsymbol{X}, \boldsymbol{\theta}) = p(\boldsymbol{y} \mid \boldsymbol{\gamma}, \boldsymbol{X}_\gamma, \boldsymbol{\theta}_\gamma),$$

e a distribuição a priori para $(\boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{g}, \boldsymbol{\theta} \mid \boldsymbol{m})$ pode ser particionada como

$$(\boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{g}, \boldsymbol{\theta} \mid \boldsymbol{m}) = p(\boldsymbol{\gamma}) p(\boldsymbol{\lambda}_\gamma, \boldsymbol{g}_\gamma, \boldsymbol{\theta}_\gamma \mid \boldsymbol{\gamma}, \boldsymbol{m}) p(\boldsymbol{\lambda}_{-\gamma}, \boldsymbol{g}_{-\gamma}, \boldsymbol{\theta}_{-\gamma} \mid \boldsymbol{\gamma}, \boldsymbol{m}).$$

Seguindo o teorema de Bayes, o núcleo da distribuição a posteriori conjunta é proporcional a

$$p(\boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{g}, \boldsymbol{\theta} \mid \boldsymbol{y}, \boldsymbol{m}) \propto p(\boldsymbol{y} \mid \boldsymbol{\gamma}, \boldsymbol{X}_\gamma, \boldsymbol{\theta}_\gamma) p(\boldsymbol{\gamma}) p(\boldsymbol{\lambda}_\gamma, \boldsymbol{g}_\gamma, \boldsymbol{\theta}_\gamma \mid \boldsymbol{\gamma}, \boldsymbol{m}) p(\boldsymbol{\lambda}_{-\gamma}, \boldsymbol{g}_{-\gamma}, \boldsymbol{\theta}_{-\gamma} \mid \boldsymbol{\gamma}, \boldsymbol{m}), \quad (5.1)$$

onde

$$p(\boldsymbol{\lambda}_\gamma, \boldsymbol{g}_\gamma, \boldsymbol{\theta}_\gamma \mid \boldsymbol{\gamma}, \boldsymbol{m}) = p(\boldsymbol{\lambda}_\gamma \mid \boldsymbol{\gamma}, \boldsymbol{m}) p(\boldsymbol{g}_\gamma \mid \boldsymbol{\lambda}_\gamma, \boldsymbol{m}) p(\boldsymbol{\theta}_\gamma \mid \boldsymbol{g}_\gamma \boldsymbol{\gamma}, \boldsymbol{m}), \text{e}$$

$$p(\boldsymbol{\lambda}_{-\gamma}, \boldsymbol{g}_{-\gamma}, \boldsymbol{\theta}_{-\gamma} \mid \boldsymbol{\gamma}, \boldsymbol{m}) = p(\boldsymbol{\lambda}_{-\gamma} \mid \boldsymbol{\gamma}, \boldsymbol{m}) p(\boldsymbol{g}_{-\gamma} \mid \boldsymbol{\lambda}_{-\gamma}, \boldsymbol{m}) p(\boldsymbol{\theta}_{-\gamma} \mid \boldsymbol{\gamma}, \boldsymbol{m}).$$

A partir de (5.1), podem ser derivadas as distribuições condicionais completas.

5.3.1 Especificação das prioris

Foram atribuídas as seguintes especificações a priori para os parâmetros do modelo, seguindo as sugestões de Yi (2004) e Yi et al. (2005):

Cota superior para L : Assume-se que a distribuição a priori para o número de QTL, l, é Poisson com média l_0, onde l_0 obtido com base em métodos tradicionais. Um valor de L suficientemente grande, tal que $p(l > L)$ seja muito pequena, é obtido com base numa aproximação normal para a Poisson, considerando $L \approx l_0 + 3\sqrt{l_0}$.

Priori sobre γ : Foi assumida uma priori de independência da forma

$$p(\gamma) = \prod_{j=1}^{L} \omega_j^{\gamma_j} (1 - \omega_j)^{1-\omega_j},$$

onde $\omega_j = p(\gamma_j = 1)$ é a probabilidade de inclusão a priori para o j-ésimo efeito, e ω_j é aproximado como

$$\omega_j = 1 - \left[1 - \frac{l}{L}\right].$$

Priori sobre λ : Assume-se que as localizações dos QTL são a priori independentes e uniformemente distribuídas nos H possíveis loci.

Priori sobre β : Assume-se que cada efeito genético, β_j, segue uma priori de mistura hierárquica

$$\beta_j \mid (\gamma_j, \sigma^2, \boldsymbol{x}_{.j}) \sim \left(0, \gamma_j \, c \, \sigma^2 (\boldsymbol{x}_{.j}^T \boldsymbol{x}_{.j})^{-1}\right),$$

onde $\boldsymbol{x}_{.j} = (x_{ij}, \cdots, x_{nj})^T$ é o vetor dos coeficientes de β_j, e c é un fator de escala. Para os problemas considerados neste capítulo $c = n$.

Priori sobre μ : Foi assumido que a priori $\mu \sim N(\eta_0, \tau_0^2)$, com

$$\eta_0 = \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \qquad \text{e} \qquad \tau_0^2 = s_y^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

Priori sobre σ^2 : Foi assumida a priori não informativa $p(\sigma^2) \propto 1/\sigma^2$.

5.4 A função perda

Após definir o modelo de probabilidade, o próximo passo para formalizar o problema de decisão, é definir a função perda a ser minimizada. Para efeitos de comparação, foi adotada a função perda definida em Sisson e Hurn (2004), para o caso de QTL distribuídos em C cromossomos ao longo do genoma. Esta função está baseada nas idéias de Celeux et al. (2000) para estimar parâmetros em distribuições de mistura não rotuladas. A função perda é descrita a seguir.

Seja Ω a união enumerável de espaços correspondentes a nenhum QTL, um QTL, e assim por diante, e sejam $\boldsymbol{\phi}$ e $\hat{\boldsymbol{\phi}}$ duas configurações em Ω (configuração aqui significa o conjunto de QTL distribuídos nos C cromossomos, e os componentes de $\boldsymbol{\phi}$ são as posições dos QTL pertencentes a esse conjunto). Sisson e Hurn (2004) sugere definir um número grande de pontos de controle (posições), $t_1, \cdots, t_T$, no mesmo espaço dos componentes dos vetores $\boldsymbol{\phi}$, i.e., pontos cobrindo uniformemente todos os cromossomos considerados no genoma. Seja a função $d(t_i, \boldsymbol{\phi})$ uma função da distância entre o ponto de controle t_i e o mais próximo dos componentes do vetor $\boldsymbol{\phi}$, denotado por s. Sisson e Hurn (2004) definem esta função como

$$d(t_i, \boldsymbol{\phi}) = \frac{\mid t_i - s \mid}{1 + \mid t_i - s \mid}.$$

Para garantir invariância na ordenação dos cromossomos, Sisson e Hurn (2004) propõem uma "estrutura espacial" entre cromossomos, em que todos os cromossomos estão unidos em ambos extremos por "spacers" adicionais de comprimento K_{max} (o comprimento do maior cromossomo), tal como ilustrado na Figura 5.1. Esses "spacers" também têm pontos de controle t_i associados, mas nenhum "spacer" pode conter QTL, isto é, a localização dos QTL está restrita apenas aos C cromossomos.

A função perda $\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})$, a qual expressa a perda ou erro cometido ao estimar $\boldsymbol{\phi}$ por $\hat{\boldsymbol{\phi}}$, é finalmente definida como a soma sobre todos os pontos de controle (nos "spacers" e nos cromossomos) da diferença quadrática nas distâncias de cada ponto de controle ao QTL mais próximo sob as duas configurações, isto é,

$$\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}}) = \sum_{\forall i} \left[d'(t_i, \boldsymbol{\phi}) - d'(t_i, \hat{\boldsymbol{\phi}}) \right]^2, \tag{5.2}$$

onde

$$d'(t_i, \boldsymbol{\phi}) = \begin{cases} d(t_i, \boldsymbol{\phi}) & se \ \{\# \ \boldsymbol{\phi}\} \ \geq 1 \\ \\ d(t_i, t_i + 2K_{max}) & se \ \{\# \ \boldsymbol{\phi}\} \ = 0, \end{cases}$$

com $\{\# \ \boldsymbol{\phi}\}$ denotando o número de QTL na configuração $\boldsymbol{\phi}$. Essa redefinição da função de distância é necessária para garantir que a função perda seja bem definida em situações em que não há QTL.

Assim, há uma contribuição à perda se a distância de qualquer t_i ao componente ϕ_j mais próximo é diferente da distância de t_i ao componente $\hat{\phi}_j$ mais próximo. Dado que existe um número suficiente de pontos de controle em áreas relevantes do espaço Ω, esta função perda é minimizada quando $\boldsymbol{\phi} = \hat{\boldsymbol{\phi}}$. A metodologia proposta para obter o estimador de Bayes correspondente a esta função perda é descrito na seção seguinte.

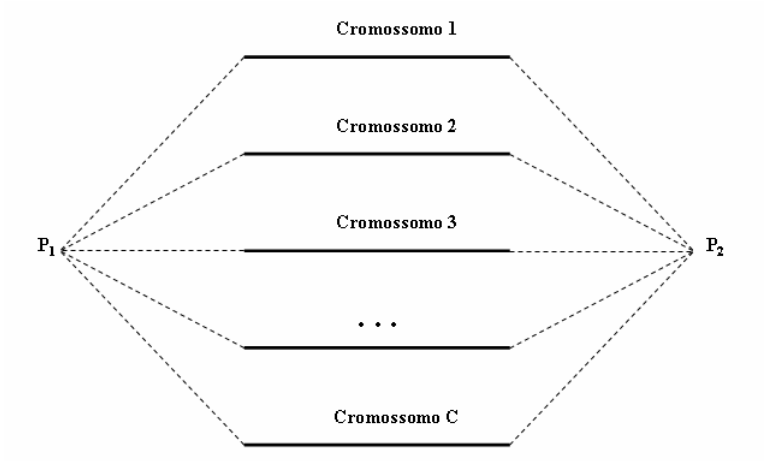


Figura 5.1: Estrutura usada para definir distâncias para os C cromossomos. Linhas sólidas representam os cromossomos e, linhas ponteadas representam os "spacers" associados a cada cromossomo.

5.5 Um algoritmo evolucionário para minimização da perda esperada

Nesta seção é proposto um novo algoritmo para minimização da perda esperada em problemas de decisão quando a dimensão do espaço das decisões é alta. O algoritmo é descrito em termos do problema de mapeamento de QTL que está sendo considerado neste capítulo. No entanto, ele é o suficientemente genérico para ser aplicado em outros contextos.

Seguindo a notação da função perda definida na seção anterior, tem-se que a perda a posteriori esperada com relação a $\hat{\boldsymbol{\phi}}$ é dada por

$$L(\hat{\boldsymbol{\phi}}) = E_{\boldsymbol{\phi}|\boldsymbol{y}}[\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})] = \int_{\boldsymbol{\phi}} \ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}}) p(\boldsymbol{\phi} \mid \boldsymbol{y}) d\boldsymbol{\phi},$$

e o objetivo é obter o estimador de Bayes, dado por

$$\boldsymbol{\phi}^* = \arg\min_{\hat{\boldsymbol{\phi}}} L(\hat{\boldsymbol{\phi}}) = \arg\min_{\hat{\boldsymbol{\phi}}} E_{\boldsymbol{\phi}|\boldsymbol{y}}[\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})].$$

Para o caso do mapeamento de QTL, considerando a função perda em (5.2), Sisson e Hurn (2004) tentam aproximar o estimador Bayes fazendo uma decomposição da perda esperada em sub-funções que possam ser avaliadas mais facilmente:

$$\begin{aligned} E_{\boldsymbol{\phi}|\boldsymbol{y}}[\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})] &= E_{\boldsymbol{\phi}|\boldsymbol{y}} \left[\sum_{\forall i} \left[d'(t_i, \boldsymbol{\phi}) - d'(t_i, \hat{\boldsymbol{\phi}}) \right]^2 \right] \\ &= \sum_{\forall i} E_{\boldsymbol{\phi}|\boldsymbol{y}} \left[d'(t_i, \boldsymbol{\phi})^2 - 2d'(t_i, \boldsymbol{\phi}) d'(t_i, \hat{\boldsymbol{\phi}}) + d'(t_i, \hat{\boldsymbol{\phi}})^2 \right] \\ &= \sum_{\forall i} \left[E_{\boldsymbol{\phi}|\boldsymbol{y}}[d'(t_i, \boldsymbol{\phi})^2] - 2d'(t_i, \hat{\boldsymbol{\phi}}) E_{\boldsymbol{\phi}|\boldsymbol{y}}[d'(t_i, \boldsymbol{\phi})] + d'(t_i, \hat{\boldsymbol{\phi}})^2 \right]. \end{aligned} \tag{5.3}$$

As funções $E_{\boldsymbol{\phi}|\boldsymbol{y}}[d'(t_i, \boldsymbol{\phi})^2]$ e $E_{\boldsymbol{\phi}|\boldsymbol{y}}[d'(t_i, \boldsymbol{\phi})]$ para $i = 1, \cdots, Z$, são estimadas tomando médias ergódicas a partir de uma amostra da distribuição a posteriori de $\boldsymbol{\phi}$, obtida previamente com um algoritmo MCMC. Esses valores esperados estimados são posteriormente inseridos em (5.3) para aproximar a perda esperada. A minimização da perda esperada é feita usando um algoritmo *simulated annealing*, onde a cada iteração é gerada uma proposta $\hat{\boldsymbol{\phi}}$, fazendo uma pequena perturbação em uma das posições do vetor de posições dos QTL. Cada proposta é avaliada usando (5.3), e aceita ou não com uma certa probabilidade que depende de um parâmetro de controle, como é usual no *simulated annealing* padrão.

Duas dificuldades principais surgem desta abordagem: a primeira é que em espaços de alta dimensão a convergência do *simulated annealing* é geralmente muito lenta, e o fato de explorar espaços desta magnitude usando uma única cadeia de Markov, aumenta o risco do algoritmo ficar preso num mínimo local. A segunda dificuldade vem do fato de que no *simulated annealing* a

função objetivo precisa ser explicitamente calculada a cada iteração, o que limita sua generalização para outros problemas de minimização complexos, cujas perdas esperadas não são analiticamente tratáveis ou não podem ser decompostas em subproblemas mais simples.

O novo algoritmo proposto aqui, diferentemente de Sisson e Hurn (2004), realiza conjuntamente as tarefas de estimação e otimização, e permite considerar diretamente funções perda cujas esperanças não são integráveis analiticamente, não precisando decompor o problema em subproblemas explicitamente avaliáveis, o que nem sempre é possível, dependendo da função perda escolhida.

A idéia principal é rodar simultaneamente um algoritmo MCMC e um algoritmo genético, de tal forma que os processos de estimação dos parâmetros do modelo e otimização da localização (e número) dos QTL ocorram conjuntamente.

O primeiro passo é rodar o modelo para o mapeamento de QTL descrito na seção 5.3, usando um algoritmo MCMC simples até atingir a convergência da cadeia de Markov. Cabe lembrar que sob essa abordagem, o vetor $\boldsymbol{\phi}$ de posições dos QTL mantém sempre uma dimensão fixa igual ao número máximo (L) de QTL permitido pelo modelo, isto é, o modelo sempre estima posições para os L possíveis QTL, mas quem determina quais dessas posições serão de fato ocupadas por QTL são os elementos com valor 1 no vetor indicador binário $\boldsymbol{\gamma}$ associado a ele.

Agora suponha um algoritmo genético com uma população de N possíveis soluções (i.e., N vetores $\hat{\boldsymbol{\phi}}$ propostos), $\hat{\Phi} = \{\hat{\boldsymbol{\phi}}_1, ..., \hat{\boldsymbol{\phi}}_N\}$, inicializadas aleatoriamente. Cada solução está associada a um valor diferente J_k, $k \in \{1, \cdots, N\}$, do conjunto $\boldsymbol{J} = \{J_1, J_2, \cdots, J_N\}$, com $J_1 > \cdots > J_N$.

A derivação da perda esperada é baseada na idéia de interpretar $\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})p(\boldsymbol{\phi} \mid \boldsymbol{y})$ como uma distribuição conjunta não normalizada de $\boldsymbol{\phi}$, $\hat{\boldsymbol{\phi}}$ e $\boldsymbol{y}$, e então simular a distribuição a posteriori marginal de $\hat{\boldsymbol{\phi}}$ por amostragem de Monte Carlo. Uma aproximação razoável do mínimo desta distribuição marginal será considerada como o estimador Bayes que minimiza a perda $\ell(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}})$.

A distribuição conjunta de $\boldsymbol{\phi}$, $\hat{\boldsymbol{\phi}}$ e $\boldsymbol{y}$ é representada por um modelo de probabilidade aumentado artificial, construído a partir de um conjunto de réplicas do vetor $\boldsymbol{\phi}$. Esse conjunto de vetores replicados é construído considerando os vetores $\boldsymbol{\phi}$ gerados a cada iteração do algoritmo MCMC após aquecimento da cadeia (*burnin*). Isto é feito a cada $R = \sum_{i=1}^{N} i$ iterações, tomando os R vetores $\boldsymbol{\phi}$ gerados pelo algoritmo MCMC, permutando-os para evitar autocorrelação e construindo

com eles uma população de conjuntos de vetores replicados $\Phi_J = \{\boldsymbol{\phi}^{(J_1)}, ..., \boldsymbol{\phi}^{(J_N)}\}$, onde cada $\boldsymbol{\phi}^{(J_k)} = \{\boldsymbol{\phi}_1^{(J_k)}, ..., \boldsymbol{\phi}_{J_k}^{(J_k)}\}$ representa um conjunto de J_k réplicas do vetor $\boldsymbol{\phi}$. Portanto, cada vetor proposto $\hat{\boldsymbol{\phi}}_k$ na população $\hat{\Phi}$, estará associado a um conjunto de réplicas $\boldsymbol{\phi}^{(J_k)}$.

Assim, com base nas idéias de Müller et al. (2004) tem-se então que

$$h_{J_k}\left(\hat{\boldsymbol{\phi}}_k, \boldsymbol{\phi}_1^{(J_k)}, \cdots, \boldsymbol{\phi}_{J_k}^{(J_k)}, \boldsymbol{y}\right) \propto \prod_{j=1}^{J_k} \ell\left(\boldsymbol{\phi}_j^{(J_k)}, \hat{\boldsymbol{\phi}}_k\right) p\left(\boldsymbol{\phi}_j^{(J_k)} \mid \boldsymbol{y}\right),$$

onde $\boldsymbol{\phi}_1^{(J_k)}, \cdots, \boldsymbol{\phi}_{J_k}^{(J_k)}$ são as J_k réplicas do vetor de posições $\boldsymbol{\phi}$ associadas ao vetor proposto $\hat{\boldsymbol{\phi}}_k$, tal que a marginal

$$\begin{aligned}
h_{J_k}(\hat{\boldsymbol{\phi}}_k) &= \int \cdots \int h_{J_k}(\hat{\boldsymbol{\phi}}_k, \boldsymbol{\phi}_1^{(J_k)}, \cdots, \boldsymbol{\phi}_{J_k}^{(J_k)}, \boldsymbol{y}) d\boldsymbol{y} d\boldsymbol{\phi}_{1:J_k} \\
&\propto \int \cdots \int \prod_{j=1}^{J_k} \ell(\boldsymbol{\phi}_j^{(J_k)}, \hat{\boldsymbol{\phi}}_k) p(\boldsymbol{\phi}_j^{(J_k)} \mid \boldsymbol{y}) d\boldsymbol{\phi}_{1:J_k} \\
&\propto \underbrace{\int_{\boldsymbol{\phi}_1^{(J_k)}} \ell(\boldsymbol{\phi}_1^{(J_k)}, \hat{\boldsymbol{\phi}}_k) p(\boldsymbol{\phi}_1^{(J_k)} \mid \boldsymbol{y}) d\boldsymbol{\phi}_1}_{L\left(\hat{\boldsymbol{\phi}}_k\right)} \int \cdots \int \prod_{j=2}^{J_k} \ell(\boldsymbol{\phi}_j^{(J_k)}, \hat{\boldsymbol{\phi}}_k) p(\boldsymbol{\phi}_j^{(J_k)} \mid \boldsymbol{y},) d\boldsymbol{\phi}_{2:J_k} \\
&\vdots \\
&\propto \left[L\left(\hat{\boldsymbol{\phi}}_k\right)\right]^{J_k} \quad \text{como desejado, para um inteiro positivo } J_k.
\end{aligned}$$

Para a avaliação da função perda são consideradas apenas as posições válidas desses vetores, i.e., aquelas associadas aos QTL identificados pelo modelo. Portanto, as réplicas do vetor $\boldsymbol{\phi}$ em cada conjunto $\boldsymbol{\phi}^{(J_k)}$ têm dimensões variáveis.

Os operadores genéticos utilizados para gerar as novas propostas $\hat{\boldsymbol{\phi}}$, seguem as idéias de Nakamichi et al. (2001). Eles consideram um operador de cruzamento e dois operadores de mutação, os quais são descritos a seguir:

- *Cruzamento* : Neste operador uma nova solução é gerada a partir de duas soluções da população corrente escolhidas usando um esquema de seleção por torneio. O número de QTL da nova solução gerada é escolhido aleatoriamente do intervalo determinado pelo número de QTL das duas soluções parentais. Do mesmo modo, as posições dos QTL da nova solução são escolhidos aleatoriamente dentre as posições ocupadas pelos QTL nas duas soluções parentais.

- *Mutação drástica* : Neste operador uma solução pode ser modificada de acordo com um dos três seguintes padrões, escolhido ao acaso:
  - *Eliminação*: seleciona aleatoriamente uma posição do vetor $\hat{\phi}$ para ser suprimida dessa solução.
  - *Inserção*: adiciona um novo QTL, cuja posição é escolhida aleatoriamente e acrescentada ao vetor de posições $\hat{\phi}$.
  - *Realocação*: seleciona aleatoriamente um QTL e o move para uma posição aleatória num dos cromossomos também escolhido aleatoriamente.

- *Mutação suave* : Este operador serve para ajudar a manter a diversidade da população, e consiste em fazer uma pequena perturbação aleatória numa das posições do vetor ϕ, escolhida aleatoriamente.

Esses operadores genéticos são aplicados na população $\hat{\Phi}$ para criar novos vetores $\tilde{\hat{\phi}}$ propostos, os quais serão aceitos com uma certa probabilidade. Para o caso dos dois operadores de mutação, a população $\hat{\Phi} = \{\hat{\phi}_1, ..., \hat{\phi}_k, ..., \hat{\phi}_N\}$ será substituida por $\hat{\Phi}' = \{\hat{\Phi} \setminus \hat{\phi}_k,\ \tilde{\hat{\phi}}\}$ com probabilidade

$$\alpha_{\mathrm{J_{mut}}} = \min\left[\, 1, \exp\left(J_k \tilde{\nu}_k - J_k \nu_k\right) \,\right],$$

onde

$$\nu_k = \frac{1}{J_k} \sum_{j=1}^{J_k} \log \ell \left(\boldsymbol{\phi}_j^{(J_k)}, \hat{\boldsymbol{\phi}}_k \right) \quad \text{e} \quad \tilde{\nu}_k = \frac{1}{J_k} \sum_{j=1}^{J_k} \log \ell \left(\boldsymbol{\phi}_j^{(J_k)}, \tilde{\hat{\boldsymbol{\phi}}}_k \right).$$

Note que o algoritmo MCMC e o algoritmo genético, embora estejam rodando simultaneamente, são processos independentes. O algoritmo genético apenas se abastece com os vetores ϕ gerados a cada iteração do MCMC, mas o algoritmo genético não fornece informações para o MCMC. Ao final desse processo de estimação-otimização vão ser obtidos, de um lado uma amostra a posteriori dos parâmetros do modelo, gerada pelo MCMC e, de outro, lado o vetor de posições dos QTL, ϕ^*, que minimiza a perda esperada, otimizado pelo algoritmo genético. A dimensão desse vetor indica o número de QTL.

5.6 Aplicação a dados simulados

O novo algoritmo para minimização da perda esperada foi comparado com a abordagem de Sisson e Hurn (2004), usando dados simulados de uma população F2 com 250 indivíduos, considerando um genoma com três cromossomos, cada um de comprimento 100 centiMorgans (cM). Foram simulados 5 QTL nas posições (1;43,6), (1;72,4), (2;10,7), (2;48,7), e (3;96,9), onde o primeiro valor de cada par entre parênteses representa o número do cromossomo que contém o QTL e, o segundo valor representa a posição do QTL (em cM) nesse cromossomo medida desde o seu extremo esquerdo. Os marcadores genéticos foram simulados a cada 5 cM ao longo de cada cromossomo, incluindo os extremos (i.e., cada cromossomo tinha 21 marcadores). Os dados foram simulados no software `R` (R Development Core Team, 2006) usando o pacote `qtlbim` (Yandell et al., 2006).

Os dados foram analisados usando o modelo descrito na seção 5.3. A função perda em (5.2) foi avaliada considerando pontos de controle uniformemente espalhados a cada 0,1 cM em cada cromossomo e em cada "spacer", isto é, um total de 1001 pontos de controle foram associados a cada cromossomo e a cada "spacer". A distribuição a priori adotada para l, o número desconhecido de QTL, foi Poisson com média 3.

A função perda a ser minimizada foi a de Sisson e Hurn (2004) descrita na seção 5.4. Note que a esperança desta função perda pode ser aproximada a partir da posteriori de ϕ. A escolha desta função perda foi apenas para efeitos de comparação do método proposto com o método de Sisson e Hurn (2004), já que este último só pode ser implementado quando a perda esperada pode ser calculada. No entanto, como mencionado anteriormente, o método proposto permite considerar perdas esperadas analiticamente intratáveis.

As estimativas da intensidade a posteriori da localização dos QTL para cada um dos cromossomos são ilustradas na Figura 5.2. Seguindo Sillanpää e Arjas (1998), estas estimativas são obtidas normalizando as contagens da localização dos QTL, de modo que a área sob o gráfico integre o número esperado a posteriori de QTL, $E(NQTL \mid \boldsymbol{y}, \boldsymbol{m}) = 5,12$ obtido pela análise MCMC. Ao integrar marginalmente sobre cada cromossomo, temos que o número a posteriori de QTL é 1,995, 1,997 e 1,128 nos cromossomos 1 a 3 respectivamente. As localizações verdadeiras dos QTL simulados

foram, em geral, corretamente identificadas, com exceção de uma partição da intensidade a posteriori a ambos lados do segundo QTL simulado no cromossomo 1 e do QTL simulado no cromossomo 3. Sisson e Hurn (2004) simularam dados similares e concluiram, apos repetidas simulações da cadeia de Markov, que estas disparides são suportadas pelos dados.

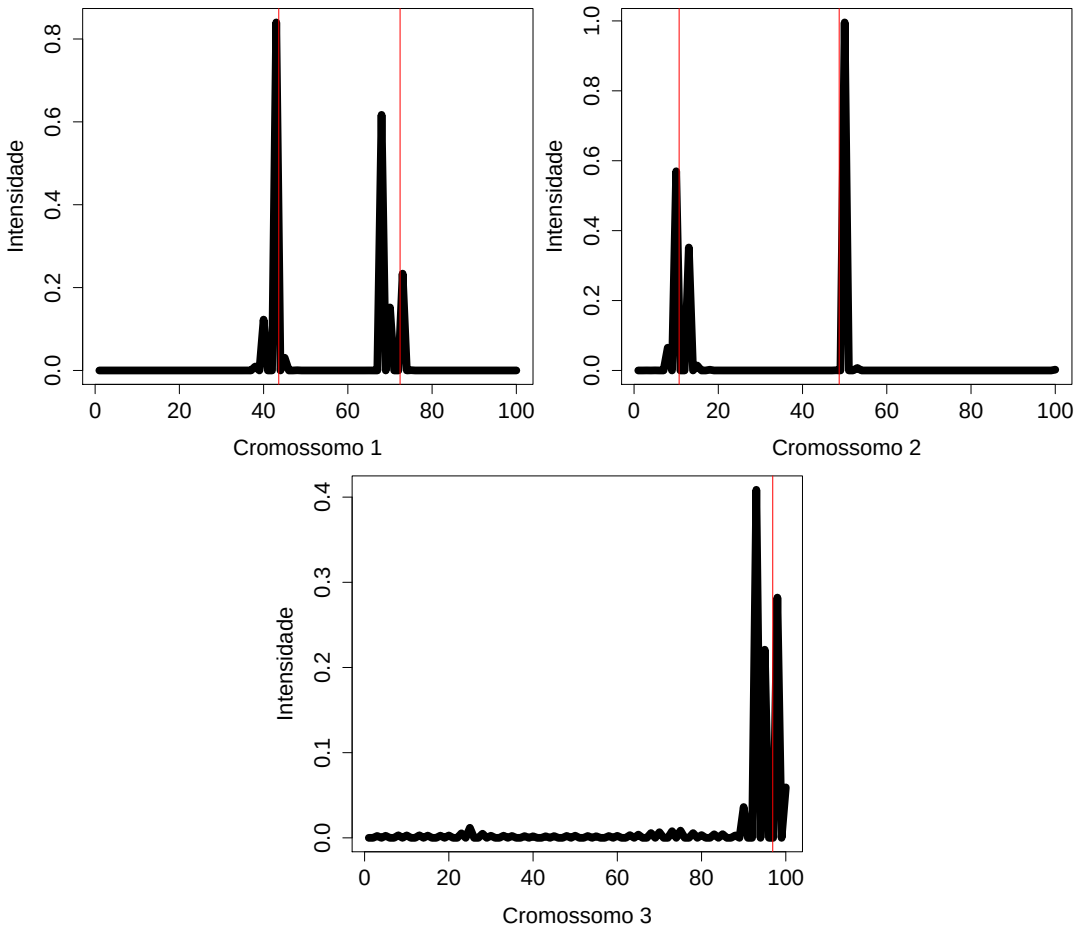


Figura 5.2: Estimativas da intensidade a posteriori da localização dos QTL para cada um dos cromossomos simulados. As linhas vermelhas indicam as posições verdadeiras dos QTL simulados.

Resultados preliminares do processo de minimização da perda esperada obtidos com os dois algoritmos são apresentados nas Tabelas 5.1 e 5.2. No algoritmo de Sisson e Hurn (2004) a busca pelas localizações dos QTL que minimizam a perda esperada é feita fixando o número de QTL. A tabela 5.1 mostra as perdas esperadas minimizadas encontradas rodando o algoritmo com diferentes números de QTL. O número total de iterações do simulated annealing em cada rodada do algoritmo foi fixada em 50000. O parâmetro de controle (temperatura) foi inicializado em 550 e reduzido a cada 100 iterações usando um fator de desconto $\alpha = 0.95$. Um passo de reinício foi implementado

a cada 1000 iterações usando um fator de acréscimo $\beta = 1,2$. As localizações propostas $\phi_i^{(prop)}$ em cada iteração do algoritmo foram sorteadas do intervalo $\phi_i \pm \delta$, onde ϕ_i representa a localização corrente do i-ésimo QTL, e δ é a largura do intervalo, a qual foi fixada em 0,5 cM. Outros valores de δ previamente testados forneceram resultados similares.

Já o algoritmo proposto procura pela melhor configuração (número e localização dos QTL) a partir de uma população inicial de configurações escolhidas aleatoriamente. O algoritmo MCMC foi implementado para um total de 200000 iterações, descartando as primeiras 20000 como aquecimento inicial da cadeia. A abordagem proposta foi testada usando diferentes tamanhos populacionais. A taxa de mutação drástica foi fixada em 0,1 após algumas rodadas exploratórias do algoritmo, enquanto que a taxa de mutação suave foi selecionada em 0,25 do número de indivíduos da população com o mesmo genótipo da nova solução criada. Na seleção por torneio o tamnaho do torneio foi de dois indivíduos. A Tabela 5.2 apresenta as melhores configurações encontradas para diferentes tamanhos de população. Para efeitos de comparação com a abordagem de Sisson e Hurn (2004), na última coluna da tabela 5.2 são apresentadas estimativas da menor perda esperada $L(\boldsymbol{\phi}*)$, calculada usando a equação 5.2, a partir das localizações dos QTL encontradas pelo algoritmo em cada caso considerado.

Tabela 5.1: Localizações estimadas dos QTL e perda esperada associada sob a abordagem de Sisson e Hurn (2004) para vários números fixos de QTL.

N° de QTL	Cromossomo 1	Cromossomo 2	Cromossomo 3	$L(\phi*)$
0	$\emptyset$	$\emptyset$	$\emptyset$	112,428
1	$\emptyset$	$\{50,002\}$	$\emptyset$	75,523
2	$\{67,650\}$	$\{50,002\}$	$\emptyset$	57,987
3	$\{67,650\}$	$\{10,155;\ 50,002\}$	$\emptyset$	45,476
4	$\{67,650\}$	$\{10,155;\ 50,002\}$	$\{92,951\}$	33,474
5	$\{42,465;\ 67,650\}$	$\{10,155;\ 50,002\}$	$\{92,951\}$	19,915

Tabela 5.2: Localizações estimadas dos QTL e perda esperada associada sob a abordagem de Sisson e Hurn (2004) para diferentes valores de J.

J	Cromossomo 1	Cromossomo 2	Cromossomo 3	$L(\phi*)$
5	$\{41,135\}$	$\{11,673\}$	$\{92,274\}$	51,829
20	$\{45,43;\ 70,336\}$	$\{9,158;\ 70,224\}$	$\{94,366\}$	34,706
30	$\{43,270;\ 67,502\}$	$\{9,3720;\ 51,825\}$	$\{94,230\}$	29,855
50	$\{42,478;\ 67,690\}$	$\{10,115;\ 50,003\}$	$\{94,702\}$	19,968

Capítulo 6

Considerações Finais

Nesta tese foi abordado o problema de otimização em espaços de alta dimensão considerando algoritmos de busca estocásticos e métodos de Monte Carlo populacionais. O foco deste estudo esteve voltado para dois problemas práticos relevantes: o redesenho de grandes redes de monitoramento ambiental e o mapeamento de QTL. Os algoritmos desenvolvidos foram testados sob diferentes cenários, usando-se dados simulados, assim como aplicações em dados reais. Os algoritmos propostos tiveram melhor desempenho quando comparados às abordagens previamente propostas na literatura, para todos os casos considerados.

O trabalho desenvolvido até agora pode ser estendido em várias direções, algumas das quais já foram mencionadas no final de cada capítulo de pesquisa. Em geral elas podem ser divididas em duas categorias:

(i) *Extensões mais gerais sobre os algoritmos de otimização*. Esta categoria inclui, por exemplo, a paralelização dos algoritmos para diminuir seu tempo computacional e o uso de abordagens adaptativas sobre os parâmetros de controle dos algoritmos SA e AG para guiar a busca de maneira mais eficiente. Algumas estratégias de paralelização e controle adaptativo de parâmetros para os algoritmos SA e AG foram descritas no Capítulo 2.

(ii) *Extensões mais específicas focadas nas duas aplicações consideradas*. Uma extensão imediata no contexto do redesenho ótimo de redes de monitoramento dos capítulos 3 e 4 é considerar o caso complementar de expandir uma rede existente, assim como o monitoramento de variáveis

resposta não Gaussianas. Previsões mais precisas também poderiam ser obtidas modificando o modelo espaço-tempo usado nos algoritmos do capítulo 4. A motivação para essas extensões é descrita a seguir.

Expansão de uma rede de monitoramento pré-existente

Seguindo a mesma notação do Capítulo 3, considere o campo aleatório $\{Z(s) : s \in S \subset \mathbb{R}^2\}$, o qual por simplicidade, é considerado como um campo aleatório espacial discreto, isto é, existe uma densa grade de, por exemplo, p potenciais sítios de amostragem espalhados por todo o espaço de busca, e $\boldsymbol{Z}$ é um vetor p-dimensional cujas coordenadas são observáveis, correspondendo aos sítios de observação. Considere também que o vetor de sítios de observação em todos os p sítios do campo discretizado $\boldsymbol{Z}$ pode ser subdividido num vetor $\boldsymbol{Z_2}$ para os g sítios que formam a rede de monitoramento corrente, e um vetor $\boldsymbol{Z_1}$ para os $u = p - g$ potenciais sítios de amostragem para expandir a rede.

Suponha que estejam disponíveis observações repetidas sobre a rede de monitoramento corrente $\boldsymbol{Z}_2 = (Z_{21}, \cdots, Z_{2g})'$ para T instantes de tempo, e que não há observações sobre o vetor $u \times 1$, $\boldsymbol{Z}_1 = (Z_{11}, \cdots, Z_{1u})'$. O objetivo é aumentar a rede corrente $\boldsymbol{Z}_2$ selecionando um número específico u_2 de sítios no subconjunto $\boldsymbol{Z}_1$. Isto é, procura-se uma partição ótima de $\boldsymbol{Z}_1$, tal que $\boldsymbol{Z}_1 = (\boldsymbol{Z}_1^{(rem)'}, \boldsymbol{Z}_1^{(add)'})$, onde $\boldsymbol{Z}_1^{(rem)'}$ é um vetor u_1-dimensional representando os futuros sítios não monitorados e $\boldsymbol{Z}_1^{(add)'}$ é um vetor u_2-dimensional representando os futuros novos sítios monitorados. A rede resultante consistirá dos sítios correspondentes às coordenadas de $(\boldsymbol{Z}_2, \boldsymbol{Z}_1^{(add)'})$, a qual é de dimensão $(g + u_2)$.

Uma forma de alcançar este objetivo, sob a abordagem do capítulo 4 é usar as T observações repetidas disponíveis da rede corrente para fazer predições para todos os $g + u$ sítios do campo aleatório discretizado. Com estas predições para o tempo $T + 1$, os algoritmos do Capítulo 4 podem ser implementados (com modificações menores), considerando o caso de encontrar o melhor subconjunto de $g + u_2$ sítios entre os $p = g + u$ sítios totais, restringindo os subconjuntos candidatos a sempre incluirem as g estações já presentes na rede corrente.

A expansão da rede sob os critérios de entropia e informação, como descritos no Capítulo 3, já

tem sido considerada na literatura (ver por exemplo Guttorp et al., 1993; Zidek et al., 2000; Le et al., 2003) e não será abordada aqui.

Melhorando o modelo espaço-temporal

O modelo espaço-tempo usado no algoritmo proposto no Capítulo 4 pode ser melhorado para fazer previsões mais precisas, assim como para identificar mais confiavelmente observações extremas. Espera-se que estas modificações melhorem o desempenho do algoritmo na busca por melhores desenhos de redes de monitoramento. Por outro lado, deve ser lembrado que os algoritmos desenvolvidos no Capítulo 4 fazem uso intensivo de recursos computacionais, e que, dependendo da complexidade do modelo de probabilidade usado, o tempo computacional empregado no processo de otimização pode ficar excessivamente grande. Portanto, as alternativas propostas para melhorar o modelo espaço-tempo devem conciliar confiabilidade e esforço computacional. Seguindo esta direção, são propostas a seguir algumas alternativas para melhorar o modelo de convolução de processos corrente, as quais modificam o modelo atuando nos termos de erro, ou no vetor de estados.

Modificando a distribuição para os termos de erro

Algumas redes de monitoramento ambiental são desenhadas para medir valores extremos (por exemplo, picos em níveis de poluição em grandes centros urbanos). Um dos principais objetivos de tais redes é a detecção de violações nos níveis permitidos de poluentes emitidos na atmosfera com fins reguladores. Este aspecto tem sido recentemente abordado na literatura de estatística ambiental (ver por exemplo Gilleland et al., 2004; Chang et al., 2005). Uma alternativa para modelar conjuntos de dados que apresentem um comportamento de cauda pesada é o uso da distribuição t-Student para o componente de erro do modelo, com o objetivo de reduzir a influência de *outliers*, fazendo a análise estatística mais robusta. Esta abordagem tem sido usada extensivamente num contexto não espacial para modelar conjuntos de dados nas áreas de Finanças e Econometria (ver por exemplo Lange et al., 1989; Jacquier et al., 2004). Uma das dificuldades principais com esta abordagem é a estimação do número de graus de liberdade, κ, da distribuição t-Student. Alguns progressos neste aspecto

têm sido conseguidos por Fonseca et al. (2006), que desenvolveram uma análise Bayesiana objetiva baseada na priori de Jeffreys para o modelo de regressão linear com erros t-Student independentes, quando κ é desconhecido.

Seguindo essas idéias, para incrementar a capacidade de manipular observações extremas, uma proposta é assumir que o termo de erro $\boldsymbol{\varepsilon}_t$ do modelo de convolução de processos do Capítulo 4 (Equação 4.4), segue agora uma distribuição t-Student, isto é, para $y_t(s), s \in S$ denotando o nível de ozônio no dia t na localização s, o modelo de probabilidade será agora definido como

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{K}\boldsymbol{\theta}_t + \boldsymbol{\varepsilon}_t & K_{ij} &= k(s_i - \omega_j) \\ \boldsymbol{\theta}_t &\sim N(\mathbf{0}, \tau^2 \boldsymbol{I}_m) & \tau^2 &\sim IG(\alpha_\tau, \beta_\tau) \\ \boldsymbol{\varepsilon}_t &\sim t(\kappa, \mathbf{0}, \delta^2 \boldsymbol{I}_n) & \delta^2 &\sim IG(\alpha_\delta, \beta_\delta), \end{aligned} \tag{6.1}$$

onde, como no modelo de convolução de processos original, $\theta(s)$ é um processo de ruído branco contínuo restrito a uma grade regular de m nos, $\{\omega_1, \cdots, \omega_m\}$, sobre a região de estudo, $k(s)$ é um *kernel* Gaussiano com média zero, $\boldsymbol{I}_n$ é a matriz identidade n-dimensional, $IG(.,.)$ denota a distribuição gama invertida, κ é o número de graus de liberdade da distribuição t-Student, e α_τ, β_τ, α_δ e β_δ são hiperparâmetros conhecidos.

A distribuição t-Student pertence à classe de distribuições normais com mistura no parâmetro de escala, com distribuição de mistura gama. Isto é, $\varepsilon_i \sim t(\kappa, 0, \delta^2)$ pode ser escrito como $\varepsilon_i \sim N(0, \delta^2/\lambda_i)$, onde $\lambda_i \sim$ Gamma$(\kappa/2, \kappa/2)$. Em outras palavras, os erros t-Student podem ser interpretados como erros Gaussianos com variâncias diferentes. Portanto, o modelo em (6.1) pode, alternativamente, ser escrito como

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{K}\boldsymbol{\theta}_t + \boldsymbol{\varepsilon}_t & K_{ij} &= k(s_i - \omega_j) \\ \boldsymbol{\theta}_t &\sim N(\mathbf{0}, \tau^2 \boldsymbol{I}_m) & \tau^2 &\sim IG(\alpha_\tau, \beta_\tau) \\ \boldsymbol{\varepsilon}_t \mid \lambda^2 &\sim N\left(\mathbf{0}, \frac{1}{\lambda^2}\boldsymbol{I}_n\right) & \lambda^2 \mid \kappa, \delta^2 &\sim \text{Gamma}\left(\frac{\kappa}{2}, \frac{\kappa\delta^2}{2}\right) \end{aligned}$$

Sob a parametrização em (6.1), a função de verossimilhança é dada por

$$L(\theta,\delta,\kappa,y)=\frac{\Gamma\left(\frac{\kappa+1}{2}\right)^n\kappa^{n\kappa/2}}{\Gamma\left(\frac{\kappa}{2}\right)^n\pi^{n/2}\delta^n}\prod_{i=1}^{n}\left[\kappa+\left(\frac{y_{ti}-K\theta_{ti}}{\delta}\right)^2\right]^{-(\kappa+1)/2}$$

A priori de Jeffreys desenvolvida por Fonseca et al. (2006), pode ser usada na estimação do parâmetro κ.

Incluindo dependência temporal nos estados θ_t

O modelo espaço-tempo corrente pode ser estendido para um modelo dinâmico de convolução de processos, permitindo que o processo de ruido branco θ_t evolua ao longo do tempo seguindo, por exemplo, um passeio aleatório, isto é,

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{K}\boldsymbol{\theta}_t+\boldsymbol{\varepsilon}_t & \boldsymbol{\varepsilon}_t &\sim N(\mathbf{0},\delta^2\boldsymbol{I}_n) \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1}+\psi_t & \psi_t &\sim N(\mathbf{0},\tau^2\boldsymbol{I}_n) \end{aligned} \tag{6.2}$$

com $\delta^2\sim IG(\alpha_\delta,\beta_\delta)$ e $\tau^2\sim IG(\alpha_\tau,\beta_\tau)$. Esta abordagem foi usada por Calder et al. (2002) para modelar dados de ozônio em espaço e tempo. No entanto, sua adequação no contexto de otimização precisa ser testada.

Referências Bibliográficas

Adler, D. (1993). Genetic Algorithms and Simulated Annealing: A Marriage Proposal. In: *IEEE International Conference on Neural Networks*, Vol. 2, San Francisco, 1104–1109.

Alba, E. and Troya, J.M. (1999) A survey of parallel distributed genetic algorithms. *Complexity*, 4, 31–52.

Andrieu, C. and Doucet, A. (2000) Simulated Annealing for Maximum A Posteriori Parameter Estimation of Hidden Markov Models. *IEEE Transactions on Information Theory*, 46, 994–1004.

Anstreicher, K. and Lee, J. (2004). A Masked Spectral Bound for Maximum-Entropy Sampling. In: A. Di Bucchiano, H. Läuter and H. Wynn (Eds.), mODa 7 - *Advances in Model-Oriented Design and Analysis. Proceedings of the 7th International Workshop on Model-Oriented Design and Analysis*, Physica-Verlag, Heidelberg, 1–12.

Atiqullah, M.M. (2004) An Efficient Simple Cooling Schedule for Simulated Annealing. *International Conference on Computational Science and Its Applications - ICCSA 2004*, Lecture Notes in Computer Science, 3045, Berlin:Springer, pp. 396–404.

Azizi, N. and Zolfaghari S. (2004). Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research*, 31, 2439–2451.

Bäck, T. (1992). Self-adaption in genetic algorithms, In *Toward a Practice of Autonomous Systems*: Proceedings of the First European Conference on Artificial Life, F. J. Varela and P. Bourgine, Eds. Cambridge, MA: MIT Press, 1992, pp. 263–271.

Bäck, T. and Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms, In *Foundations of Intelligent Systems*, Lecture Notes in Artificial Intelligence, 1079, Z. Ras and M. Michalewicz, Eds. New York:Springer-Verlag, pp. 158–167.

Banjevic M. and Switzer, P. (2001). Optimal Network Designs in Spatial Statistics. In: *Royal Statistical Society Conference on Spatial Modelling*, Glasgow, 1–15.

Beaumont, M.A. and Rannala, B. (2004). The Bayesian revolution in genetics. *Nat. Rev. Genet*, 5, 251–261.

Bélisle, C.J.P. (1992). Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. *Journal of Applied Probability*, 29, 885–895.

Ben-Ameur, W. (2004). Computing the Initial Temperature of Simulated Annealing. *Computational Optimization and Applications*, 29, 369–385.

Berg, B.A. and Neuhaus, T. (1991). Multicanonical Algorithms for First-Order Phase Transitions, *Physics Letters B*, 267, 249–253.

Bernardo, J.M. (1979). Expected information as expected utility. *The Annals of Statistics*, 7, 686–690.

Bielza, C., Müller, P. and Rios Insua, D. (1999). Decision Analysis by Augmented Probability Simulation. *Management Science*, **45**, 995–1007.

Boer, E.P.J., Dekkers, A.L.M. and Stein A. (2002). Optimization of a Monitoring Network for Sulfur Dioxide. *Journal of Environmental Quality*, 31, 121–128.

Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35, 268–308.

Brockhoff D. and Zitzler E. (2006). Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Parallel Problem Solving from Nature - PPSN IX, Proceedings*, Lecture Notes in Computer Science vol. 4193, Berlin: Springer, pp. 533–542.

Brooks, S.P., Friel, N. and King, R. (2003). Classical model selection via simulated annealing. *Journal of the Royal Statistical Society series B*, 65, 503–520.

Bueso, M.C., Angulo, J.M. and Alonso, F.J. (1998). A state-space-model approach to optimal spatial sampling design based on entropy. *Environmental and Ecological Statistics*, 5, 29–44.

Calaor, A.E., Hermosilla, A.Y. and Corpus Jr., B.O. (2002). Parallel Hybrid Adventures with Simulated Annealing and Genetic Algorithms. In: *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02)*, IEEE Computer Society, pp 1–6.

Calder, C. A., Holloman, C. and Higdon, D. (2002). Exploring Space-Time Structure in Ozone Concentration Using a Dynamic Process Convolution Model, in *Case Studies in Bayesian Statistics 6*, C. Gatsonis, R. E. Kass, B. Carlin, A. Carriquiry, A. Gelman, D. Higdon, D. K. Pauler, I. Verdinelli and M. West, (eds.), New York:Springer-Verlag, pp. 165–176.

Calvo, F. (2005). All-exchanges parallel tempering. *Journal of Chemical Physics*, 123, 124106-1 – 124106-7.

Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10, 141–171.

Cantú-Paz, E. and Goldberg, D.E. (2000). Efficient parallel genetic algorithms: theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 186, 221–238.

Caselton, W. F. and Hussain, T. (1980). Hydrologic networks: Information transmition. *Journal of the Water Resources Planning and Management* Division, A.S.C.E., 106 (WR2), 503–520.

Caselton, W. F. and Zidek, J. V. (1984). Optimal monitoring network designs. *Statistics & Probability Letters*, 2, 223–227.

Caselton, W.F., Kan, L. and Zidek, J.V. (1992). Quality Data Networks that Minimizes Entropy. In: P. Guttorp and A. Walden (Eds.), *Statistics in the Environmental and Earth Sciences*, Halsted Press, New York, pp 10–38.

Celeux, G., Hurn, M. and Robert, C.P. (2000). Computational and Inferenctial Difficulties with Mixture Posterior Distributions. *Journal of the American Statistical Association*, 95, 957–970.

Chang, H., Fu, A. Q., Le, N. D. and Zidek, J. V. (2005). Designing Environmental Monitoring Networks for Measuring Extremes, *Technical Report #2005-4*, Statistical and Applied Mathematical Sciences Institute, 30pp.

Chatterjee, S, Laudato, M. and Lynch, L.A. (1996). Genetic algorithms and their statistical applications: an introduction. *Computational Statistics & Data Analysis*, 22, 633–651.

Cho, H.J., Oh,S.Y. and Choi, D.H. (1998). A New Population-Oriented Simulated Annealing Technique Based On Local Temperature Concept. *Electronics Letters*, 34, 312–316.

Coello, C.A. (2001). A short tutorial on evolutionary multiobjective optimization, In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization: EMO 2001, Zurich, Switzerland, March 2001*, Lecture Notes in Computer Science vol. 1993, Berlin: Springer, pp. 21–40.

Cohn, H. and Fielding, M. (1999). Simulated Annealing: Searching for an Optimal Temperature Schedule. *SIAM Journal of Optimization*, 9, 779–802.

Davis, L. (1991). Handbook of genetic algorithms, New York : Van Nostrand Reinhold.

Deb, K. and Beyer, H.G. (2001). Self-Adaptive Genetic Algorithms with Simulated Binary Crossover. *Evolutionary Computation*, 9, 197–221.

DeGroot, M. (1970). *Optimal Statistical Decisions*. New York: McGraw Hill, Chapter 7.

De Jong, K. (1975). The analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, Deptartment of Computer Science, University of Michigan, Ann Arbor.

Diekmann, R., Lüling, R. and Simon, J.(1995). A General Purpose Distributed Implementation of Simulated Annealing. In *Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing, 1992*, Arlington, pp. 94–101.

Diggle, P. and Lophaven, S. (2006). Bayesian Geostatistical Design. *Scandinavian Journal of Statistics*, 33, 53–64.

Doerge, R.W., Zeng, Z.B. and Weir, B.S. (1997). Statistical Issues in the Search for Genes Affecting Quantitative Traits in Experimental Populations. *Statistical Science*, 12, 195–219.

Doornik, J.A. (2002). *Object-Oriented Matrix Programming Using Ox*, $3th$ edition, Timberlake Consultants, London. and www.nuff.ox.ac.uk/Users/Doornik.Ox programming.

Drezner, Z. and Wesolowsky, G.O. (2003). Network design: selection and design of links and facility location. *Transportation Research* Part A, 37, 241–256.

Drugan, M. and Thierens, D. (2004). Evolutionary Markov Chain Monte Carlo, In *Artificial Evolution: 6th International Conference, Evolution Artificielle, EA 2003*, Lecture Notes in Computer Science 2936, eds. P. Liardet, P. Collet, C. Fonlupt, E. Luton and M. Shoenauer, Berlin: Springer, pp. 63–76.

Drugan, M. and Thierens, D. (2005). Recombinative EMCMC algorithms, In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 2005*, vol. 3, pp. 2024–2031.

Duckzmal, L. and Assunção, R. (2004). A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Computational Statistics & Data Analysis*, 45, 269–286.

Earl, D.J. and Deem, M.W. (2005). Parallel tempering: Theory, applications and new perspectives. *Physical Chemistry Chemical Physics*, 7, 3910–3916.

Eiben, A.E., Hinterding, R. and Michalewics, Z. (1999). Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3, 124–141.

Ferreyra, R.A., Apezteguía, H.P., Sereno, R. and Jones, J.W. (2002). Reduction of soil water spatial sampling density using scaled semivariograms and simulated annealing. *Geoderma*, 110, 265–289.

Ferri, M. and Piccioni, M. (1992). Optimal selection of statistical units An approach via simulated annealing. *Computational Statistics & Data Analysis*, 13, 47–61.

Fleischer, M. (1995). Simulated annealing: past, present and future. , In *Proceedings of the 1995 Winter Simulation Conference*, C. Alexopoulos, K. Kang, W. R Lilegdon, and D. Goldsman (eds.), pp. 155–161.

Fogarty, T. (1989). Varying the probability of mutation in the genetic algorithm, In *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), San Mateo: Morgan Kaufmann.

Fonseca, T. C. O., Ferreira, M. A. R. and Migon, H. S. (2006). Objective Bayesian analysis for the Student-t regression model. *Technical Report* N^o189, Departamento de Métodos Estatísticos, Universidade Federal do Rio de Janeiro, 20p.

Fouskakis, F. and Draper, D. (2002). Stochastic optimization: A review. *International Statistical Review*, 70, 315–349.

Fuentes, M., Chaudhuri, A. and Holland D.M. (2005). Bayesian entropy for spatial sampling design of environmental data. Mimeo Series N○ 2571, (Institute of Statistics, North Carolina State University, Raleigh, NC, 2005).

Gaffney, P.J. (2001). *An efficient reversible jump Markov chain Monte Carlo Approach to detect multiple loci and their effect in inbred crosses*. Doctoral dissertation, University of Wisconsin, Madison, USA.

Gelfand, A.E. and Smith A.F.M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85, 398–409.

George, A.W., Mengersen, K.L. and Davis G.P. (2000). Localization of a Quantitative Trait Locus via a Bayesian Approach. *Biometrics*, 56, 40–51.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, eds. E. M. Keramigas, Interface Foundations, Fairfax, pp. 156-163.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to pedigree analysis. *Journal of the American Statistical Association*, 90, 909–920.

Gilks, W.R., Roberts, R.O. and George E.I. (1994). Adaptive Direction Sampling. *The Statistician*, 43, 179–189.

Gilks, W.R., Richardson, S. and Spiegelhalter, D.J. (eds.) (1996). Makov Chain Monte Carlo in Practice. London:Chapman & Hall.

Gilleland, E., Nychka, D. and Schneider, U. (2004). Spatial Models for the Distribution of Extremes, in *Uncertainty and Variability in Ecological Inference, Forecasting and Decision Making. An Introduction to Modern Statistical Computation*, Summer Institute at the Center of Global Change, Duke University, Durham.

Gluhovsky, I. (2004). Evolutionary simulated annealing with application to image restoration, *Journal of Computational and Graphical Statistics*, 13, 871–885.

Godsill, S.J. (2001). On the Relationship Between Markov Chain Monte Carlo Methods for Model Uncertainty. *Journal of Computational and Graphical statistics*, 10, 230–248.

Goffe, W.L., Ferrier, G.D. and Rogers, J. (1994). Global optimization of statistical functions with simulated annealing, *Jounal od Econometrics*, 60, 65–99.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Leaning, Reading:Addison-Wesley.

Goldberg, D.E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, In *Foundations of Genetic Algorithms*, G.J.E. Rawlins (ed), San Mateo:Morgan Kaufmann Publishers, pp. 69–93.

Goswami, G.R. and Liu, J.S. (2007) On Learning Strategies for Evolutionary Monte Carlo. *Statistics and Computing*, 17, 23–38.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82, 711–732.

Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms, *IEEE Transanctions on Systems, Man and Cybernetics*, 16, 122–128.

Guttorp, P., Le, N.D., Sa mpson, P.D. and Zidek, J.V. (1993). Using entropy in the redesign of an environmental monitoring network. In: G.P. Patil and C.R. Rao (Eds.), *Multivariate Environmental Statistics*, North-Holland Elsevier Science Publishers, New York, pp 175–202.

Hancock, P.J.B. (1994). An Empirical Comparison of Selection Methods in Evolutionary Algorithms, In *Selected Papers from AISB Workshop on Evolutionary Computing 1994*, Lecture Notes In Computer Science, Vol. 865, pp. 80–94.

Harik, G., Cantú-Paz, E., Goldberg, D.E. and Miller, B.L. (1999). The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. *Evolutionary Computation*, 7, 231–253.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109.

Hesser, J. and Männer, R. (1991). Toward an optimal mutation probability for genetic algorithms, In *Proceedings of the First Conference on Parallel Problem Solving from Nature*, H.P. Schwefel and R. Männer (eds.), Lecture Notes in Computer Science, 496, Berlin:Springer-Verlag, pp. 23–32.

Higdon, D.M. (2002). Space and Space-Time Modeling Using Process Convolutions, In *Quantitative Methods for Current Environmental Issues*, eds. C.W. Anderson, V. Barnett, P.C. Chatwin and A.H. El-Shaarawi, New York: Springer-Verlag, pp. 37–54.

Hoffman, L., Lee, J. and Williams, J. (2001). New Upper Bounds for Maximum-Entropy Sampling. In: A.C. Atkinson, P. Hackl and W. Müller (Eds.), mODa 6 - *Advances in Model-Oriented Design and Analysis. Proceedings of the 6th International Workshop on Model-Oriented Design and Analysis*, Physica-Verlag, Heidelberg, 143–153.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holloman, C.H., Lee, H.K.H. and Higdon, D.M. (2006). Multiresolution genetic algorithms and Markov chain Monte Carlo. *Journal of Computational and Graphical Statistics*, 15, 861–879.

Hoti, F.J. , Sillanpää M.J. and Holmström, L. (2002). A Note on Estimating the Posterior Density of a Quantitative Trait Locus From a Markov Chain Monte Carlo Sample. *Genetic Epidemiology*, 22, 369–376.

Hu, B. and Tsui, K.W. (2005). Distributed Evolutionary Monte Carlo with Applications to Bayesian Analysis. *Technical Report* $N^o 1112$, Department of Statistics, University of Wisconsin, Madison, 22p.

Hukushima, K. and Nemoto, K. (1996). Exchange Monte Carlo Method and Application to Spin Glass Simulations *Journal of the Physical Society of Japan*, 65, 1604–16??.

Hurley, S., Thiel, S.U. and Smith, D.H. (1996). A Comparison of Local Search Algorithms for Radio Link Frequency Assignment Problems. *Proceedings of the 1996 ACM Symposium on Applied Computing*, Philadelphia, 251–257.

Ingber, L. (1994). Simulated annealing: Practice versus theory. *Journal of Mathematical and Computational Modelling*, 18, 29–57.

Iovleff, S. and Perrin, O. (2004). Estimating a Nonstationary Spatial Structure Using Simulated Annealing *Journal of Computational & Graphical Statistics*, 13, 90–105.

Jacquier, E., Polson, N. G. and Rossi, P. E. (2004). Bayesian Analysis of Stochastic Volatility Models with Fat-Tails and Correlated Errors, *Journal of Econometrics*, 122, 185–212.

Jasra, A., Stephens, D.A. and Holmes, C.C. (2005). On Population-Based Simulation for Static Inference. *Technical Report*, Department of Statistics, Imperial College, London, 29p.

Jiang, C., and Zeng, Z.B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica*, 101, 47–58.

Katzgraber, H.G., Trebst, S., Huse, D.A. and Troyer, M. (2006). Feedback-optimized parallel tempering Monte Carlo. *Journal of Statistical Mechanics*, P03018.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, 220, 671–680.

Kita, H. (2001). A Comparison Study of Self-Adaptation in Evolution Strategies and Real-Coded Genetic Algorithms. *Evolutionary Computation*, 9, 223–241.

Knopman, J. and Aude, J.S. (1996). Parallel Simulated Annealing: an Adaptive Approach. In *Proceedings of the 11th IEEE International Symposium on Parallel Processing, 1997*, Genova, pp. 522–526.

Ko, C.W., Lee, J. and Queyranne, M. (1995). An exact algorithm for maximum entropy sampling. *Operations Research*, 43, 684–691.

Kone, A. and Kofkea, D.A. (2005). Selection of temperature intervals for parallel-tempering simulations. *Journal of Chemical Physics*, 122, 206101-1 – 206101-2.

Lange, K. L., Little, R. J. A. and Taylor, J. G. M. (1989). Robust Statistical Modelling Using the t Distribution, *Journal of the American Statistical Association*, 84, 881–896.

Laskey, K.B. and Myers, J.W. (2003). Population Markov Chain Monte Carlo. *Machine Learning*, 50, 175–196.

Le, N.D. and Zidek, J.V. (1994). Network designs for monitoring multivariate random spatial fields. In: J.P. Vilaplana and M.L. Puri (Eds.), *Advances in Probability and Statistics*, p. 191–206.

Le, N.D., Sun, L. and Zidek, J.V. (2003). Designing Networks for Monitoring Multivariate Environmental Fields Using Data with Monotone Pattern. Technical Report # 2003-5, (Statistical and Applied Mathematical Sciences Institute, NC, 2003).

Lee, Y.M. and Ellis, J. H. (1996). Comparison of algorithms for nonlinear integer optimization. Application to monitoring network design. *Journal of Environmental Engineering*, 122, 524–531.

Lee, J. (1998). Constrained Maximum-Entropy Sampling. *Operations Research*, 46, 655–664.

Li, Y., Protopopescu, V.A. and Gorin, A. (2004). Accelerated simulated tempering. *Physics Letters A*, 328, 274–283.

Liang, F. (2002). Dynamically Weighted Importance Sampling in Monte Carlo Computation. *Journal of the American Statistical Association*, 97, 807–821.

Liang, F. (2005). Generalized Wang-Landau algorithm for Monte Carlo Computation. *Journal of the American Statistical Association*, 100, 1311–1327.

Liang, F. and Wong, W.H. (2000). Evolutionary Monte Carlo: Applications to C_p Model Sampling and Change Point Problem. *Statistica Sinica*, 10, 317–342.

Liang, F. and Wong, W.H. (2001). Real-parameter Evolutionary Monte Carlo with Applications to Bayesian Mixture Models. *Journal of the American Statistical Association*, 96, 653–666.

Lin, W.Y., Lee, W.Y. and Hong, T.P. (2003). Adapting Crossover and Mutation Rates in Genetic Algorithms. *Journal of Information Science and Engineering*, 19, 889–903.

Lynch, M. and Walsh, B. (1998). *Genetics and Analysis of Quantitative Traits*. Sunderland:Sinauer Associates Inc. Publishers.

Lindley, D.V. (1956). On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*. 27, 986–1005.

Liu, J.S., Liang, F. and Wong, W.H. (2000). The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, 95, 121–134.

Liu, J.S., Liang, F., and Wong. W.H. (2001). A theory for dynamic weighting in Monte Carlo. *Journal of the American Statistical Association*, 96, 561–573.

Liu, J. S. and Sabatti, C. (1999) Simulated sintering: Markov chain Monte Carlo with spaces of varying dimension, In: *Bayesian Statistics 6*, eds. J.M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, Clarendon Press, New York: Oxford University Press, pp. 389–413.

Locatelli, M. (1996). Convergence properties of simulated annealing for continuous global optimization. *Journal of Applied Probability*, 33, 1127–1140.

Lundy, M. and Mees, A. (1986). Convergence of an annealingalgorithm. *Mathematical Programming*, 34, 111–124.

Mahfoud, S.W. and Goldberg, D.E. (1995). Parallel recombinative simulated annealing: A genetic algorithm, *Parallel Computing*, 21, 1–28.

Marinari, E. and Parisi, G. (1992). Simulated tempering: a new Monte Carlo scheme. *Europhysics Letters*, 19, 451–458.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953). Equations of State Calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1091.

Miki, M., Hiroyasu, T. and Wako, J. (2003). Adaptive Temperature Schedule Determined by Genetic Algorithm for Parallel Simulated Annealing. *The 2003 Congress on Evolutionary Computation*, vol. 1, 459–466.

Ming, L., Cheung, Y.M. and Wang, Y.P. (2004). A Dynamically switched crossover for genetic algorithms, In *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, Shanghai, IEEE, pp. 3254–3257.

Mühlenbein, H. and Paaß, G. (1996). From recombination of genes to the estimation of distributions: I. binary parameters. In, W. Ebeling et al., (eds.), Lecture Notes in Computer Science, 1141, Proceedings of the Parallel Problem Solving from Nature IV Conference, Berlin:Springer-Verlag, pp. 178–187.

Müller, P. (1999). Simulation Based Optimal Design, In *Bayesian Statistics 6*, eds. J.M. Bernardo, J.O. Berger, A.P. Dawid and A.F.M. Smith, Oxford, U.K.: Oxford University Press, pp. 459–474.

Müller, P., Sansó, B. and De Iorio, M. (2004). Optimal Bayesian Design by Inhomogeneous Markov Chain Simulation. *Journal of the American Statistical Association*, 99, 788–798.

Nakamichi, R., Ukai, Y. and Kishino, H. (2001). Detection of Closely Linked Multiple Quantitative Trait Loci Using a Genetic Algorithm. *Genetics*, 158, 463–475.

Nourani, Y. and Andresen, B. (1998). A comparison of Simulated Annealing Cooling Strategies. *Journal of Physics A: Math. Gen.*, 31, 8373–8385.

Nowostawski, M. and Poli, R. (1999). Review and taxonomy of parallel genetic algorithms. *Technical Report CSRP-99-11*, University of Birmingham, School of Computer Science, Birminghan, available at ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1999/CSRP-99-11.ps.gz.

Nunes, L. M., Cunha, M. C. C., and Ribeiro, L. (2004). Groundwater monitoring networks optimization with redundancy reduction. *Journal of Water Resources Planning and Management*. 130, 33–43.

Onbaşoğlu, E. and Özdamar, L. (2001). Parallel Simulated Annealing Algorithms in Global Optimization. *Journal of Global Optimization*. 19, 27–50.

Pardo-Igúzquiza, E. (1998). Optimal selection of number and location of rainfall gauges for areal rainfall estimation using geostatistics and simulated annealing. *Journal of Hydrology*, 210, 206–220.

Pardo-Igúzquiza, E. (1998). Maximum Likelihood Estimation of Spatial Covariance Parameters. *Mathematical Geology*, 30, 95–108.

Pelikan, M., Goldberg, D.E. and Lobo, F.G. (2002). A Survey of Optimization by Building and Using Probabilistic Models. *Computational Optimization and Applications*, 21, 5–20.

Poupaert, E. and Deville, Y. (2000). Simulated Annealing with estimated temperature. *AI Communications*, 13, 19–26.

R Development Core Team (2006). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Rathore, N., Chopra, M. and Pablo J.J. (2005). Optimal allocation of replicas in parallel tempering simulations. *Journal of Chemical Physics*, 122, 024111-1 – 024111-8.

Reed, P.M., Minsker, B.S., Valocchi, A.J. (2000). Cost effective long-term groundwater monitoring design using a genetic algorithm and global mass interpolation. *Water Resources Research*, 36, 3731–3741.

Reed, P., Minsker, B. S., and Goldberg, D. E. (2000). Designing a competent simple genetic algorithm for search and optimization. *Water Resources Research*, 36, 3757–3761.

Roberts, R.O. and Gilks, W.R. (1994). Convergence of Adaptive Direction Sampling. *Journal of Multivariate Analysis*, 49, 287–298.

Royle, J.A. (2002). Exchange algorithms for constructing large spatial designs. *Journal of Statistical Planning and Inference*, 100, 121–134.

Rue, H. (1995). New Loss Functions in Bayesian Imaging. *Journal of the American Statistical Association*, 90, 900–908.

Sansó, B. and Müller, P. (1999). Redesigning a network of rainfall stations, In *Case Studies in Bayesian Statistics IV*, eds. C. Gatsonis, R.E. Kass, B. Carlin, A. Carriquiry, A. Gelman, I. Verdinelli and M. West, New York: Springer-Verlag, pp. 383–394.

Sarma, J. and Jong, K.D. (1996) An analysis of the effects of neighborhood size and shape on local selection algorithms, In *Parallel Problem Solving from Nature IV*, Berlin:Springer-Verlag, pp. 236–244.

Satagopan, J.M., Yandell, B.S., Newtont, M.A. and Osborn, T.C. (1996). A Bayesian Approach to Detect Quantitative Trait Loci Using Markov Chain Monte Carlo. *Genetics*, 144, 805–816.

Schafer, J.D., Caruana, R.A., Eshelman, L.J. and Das, R. (1989). A study of control parameters affecting online performance of genetics algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), San Mateo:Morgan Kaufmann.

Sen, S. and Churchill, G.A. (2001). A Statistical Framework for Quantitative Trait Mapping. *Genetics*, 151, 371–387.

Shakya, S.K., McCall, J. and Brown, D.F. (2005). Estimating the distribution in an EDA, In *Adaptive and Natural Computing Algorithms Proceedings of the International Conference in Coimbra,*

Portugal, 2005, eds. B. Ribeiro, R.F. Albrecht, A. Dobnikar, D.W. Perarson and N.C. Steele, New York: Springer-Verlag, pp. 202–205.

Shoemaker, J.S., Painter, I.S. and Weir, B. (1999). Bayesian Statistics in Genetics. A guide for the uninitiated. *Trends in Genetics*, 15, 354–358.

Sillanpää M.J. and Arjas, E. (1998). Bayesian Mapping of Multiple Quantitative Trait Loci From Incomplete Inbred Line Cross Data. *Genetics*, 148, 1373–1388.

Sisson, S.A. and Hurn, M.A. (2004). Bayesian Point Estimation of Quantitative Trait Loci. *Biometrics*, 60, 60–68.

Slager, S.L., Juo, S.H., Durner, M., and Hodge, S.E. (2001). Markov chain Monte Carlo linkage analysis: effect of bin width on the probability of linkage. *Genetic epidemiology*, 21, 700-705.

Spears W. M. (1995). Adapting Crossover in Evolutionary Algorithms, In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pp. 367–384.

Stephens, D.A. and Fisch, R.D. (1998). Bayesian analysis of quantitative trait locus data using reversible jump Markov chain Monte Carlo. *Biometrics*, 54, 1334–1347.

Storn, R. and Price, K. (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11, 341–359.

Strens, M.J.A., Bernhardt, M. and Everett, N. (2002) Markov chain Monte Carlo Sampling using Direct Search Optimization, in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002)*, Sidney, Australia.

Strens, M.A.J. (2003). Evolutionary MCMC Sampling and Optimization in Discrete Spaces, In *Proceedings of the Twentieh International Conference on Machine Learning (ICML-2003)*, eds. T. Fawcett and N. Mishra, Menlo Park, California: AAAI Press, pp. 736–743.

Tanese, R. (1989). Distributed Genetic Algorithm, In: *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 434–440.

Ter Braak, C,J,F. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing*, 16, 239–249.

Thierens, D. (1995). *Analysis and design of genetic algorithms*. Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium.

Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms, In *IEEE Proceedings of the 2002 congress on Evolutionary Computation, CEC'02*, Vol. 1, Honolulu, pp. 980–985.

Tomassini, M. (2001). Parallel and Distributed Evolutionary Algorithms. In, *Evolutionary Algorithms in Engineering and Computer Science:* Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications, K. Miettinen, M. M. Mäkelä, P. Neittaanmäki and J. Périaux (Eds.), New York:Wiley, pp. 113–134.

Trujillo-Ventura, A. and Ellis, J.H. (1991). Multiobjective air pollution monitoring network design. *Atmospheric Environment*, 25A, 469–479.

Uimari, P. and Hoeschele, I. (1997). Mapping-linked quantitative trait loci using Bayesian analysis and Markov chain Monte Carlo algorithms. *Genetics*, 146, 735–743.

Van Groenigen, J.W. and Stein, A. (1998). Constrained Optimization of Spatial Sampling using Continuous Simulated Annealing. *Journal of Environmental Quality*, 27, 1078–1086.

van Laarhoven, P.J.M. and Aarts, E.H.L. (1987). *Simulated Annealing: Theory and Applications*. Kluwer, Amsterdam, 187p.

Waagepetersen, R. and Sorensen, D. (2001). A tutorial on Reversible Jump MCMC with a view toward applications in QTL-mapping. *International Statistical Review*, 69, 49–62.

Wang, F. and Landau, D.P. (2001). Effcient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, 86, 2050–2053.

Wang, X.. and Wang, H. (2004). Evolutionary Optimization With Markov Random Field Prior. *IEEE Transactions on Evolutionary Computation*, 8, 567–579.

Wang, H., Zhang, Y.M., Li, X., Masinde, Y.L., Mohan, S., Baylink, D.J. and Xu, S. (2005). Bayesian Shrinkage Estimation of Quantitative Trait Loci Parameters. *Genetics*, 170, 465–480.

Wong, W.H. , and Liang, F. (1997). Dynamic Weighting in Monte Carlo and Optimization. *Proceedings of the National Academy of Science*, 94, 14220–14224.

Wu, J., Zheng, C. and Chien, C.C. (2005). Cost-effective sampling network design for contaminant plume monitoring under general hydrogeological conditions. *Journal of Contaminant Hydrology*, 77, 41–65.

Wu, S. and Zidek, J.V. (1992). An entropy-based analysis of data from selected NADP/NTN network sites for 1983-1986. *Atmospheric Environment*, 26A(11), 2089–2103.

Yandell, B.S. and Yi, N., with contributions from Tapan Mehta, Samprit Banerjee, Daniel Shriner, Ramprasad Venkataraman (UA-Birmingham), Jee Young Moon, W. Whipple Neely (UW-Madison), Hao Wu and Randy von Smith (Jackson Laboratory). (2006). qtlbim: QTL Bayesian Interval Mapping. R package version 1.6.5. http://www.qtlbim.org.

Yang, S. (2002). Adaptive crossover in genetic algorithms using statistics mechanism, In *Proceedings of the eighth international conference on Artificial life, 2002*, Standish, Abbass, Bedau (eds), New York:MIT Press, pp. 182–185.

Yi, N. (2004). A Unified Markov Chain Monte Carlo Framework for Mapping Multiple Quantitative Trait Loci. *Genetics*, 167, 967–975.

Yi, N., Yandell, B.S., Churchill, G.A., Allison, D.B., Eisen, E.J. and Pomp, D. (2005). Bayesian Model Selection for Genome-Wide Epistatic Quantitative Trait Loci Analysis. *Genetics*, 170, 1333–1344.

Yip, P.C. and Pao, Y.H. (1995). Combinatorial Optimization with Use of Guided Evolutionary Simulated Annealing, *IEEE Transactions On Neural Networks*, 6, 290–295.

Yu, T.L. (2002). A survey of estimation of distribution algorithms. *Technical report 2002-07, Illi-GAL*, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois at Urbana-Champaign, USA, 12pp.

Zhang, B.T. and Kim, J.J. (2000). Comparison of Selection Methods for Evolutionary Optimization. *Evolutionary Optimization*, 2, 55–70.

Zhang, B.T. and Cho, D.Y. (2001). System Identification Using Evolutionary Markov Chain Monte Carlo. *Journal of Systems Architecture*, 47, 587–599.

Zhang, M., Montooth, K.L., Wells, M.T., Clark, A.G. and Zhang, D. (2005). Mapping Multiple Quantitative Trait Loci by Bayesian Classification. *Genetics*, 169, 2305–2318.

Zhu, Z. and Stein, M.L. (2005). Spatial sampling design for parameter estimation of the covariance function. *Journal of statistical planning and inference*, 134, 583–603.

Zidek, J.V., Sun, W. and Le, N.D. (2000). Designing and integrating composite networks for monitoring multivariate Gaussian pollution fields. *Applied Statistics*, 49, 63–79.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Doctoral dissertation, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland.

Apêndice A

Formulação Bayesiana de Desenhos Ótimos

Uma das primeiras discussões sobre o desenho de experimentos desde o ponto de vista Bayesiano foi dada por Lindley (1956). Ele propôs um critério para a maximização da informação de Shannon esperada a ser ganha em um experimento quando o objetivo "não é tomar decisões, mas ganhar conhecimento sobre o mundo". Mais tarde, Bernardo (1979) mostrou que esse critério pode também ser derivado desde o ponto de vista de teoria da decisão Bayesiana, dadas certas suposições "razoáveis" sobre a forma da função de utilidade, como descrito a seguir.

Suponha que um certo experimento E renderá dados X distribuídos de acordo com um modelo $X \sim p\,(. \mid \theta)$, onde $p\,(. \mid \theta)$ é uma distribuição de probabilidade conhecida que depende do verdadeiro valor θ de um parâmetro aleatório desconhecido Θ. Suponha também que o objeto de interesse é uma função de Θ, denotada por Ψ ou $\Psi(\Theta)$, com seu verdadeiro valor denotado por ψ, que a densidade a priori de Θ é $\pi_\Theta(\theta)$ e que a densidade a priori de Ψ é $\pi_\Psi(\psi)$. A densidade marginal de X será denotada por $P_X(x) = \int p(x \mid \theta)\pi_\Theta(\theta)d\theta$. Como Θ só entrará na seguinte discussão, através de Ψ, será usada a expressão $\pi(.)$ ao invés de $\pi_\Psi(.)$. A densidade a posteriori de Ψ dados os dados, $X = x$, avaliada em $\Psi = \psi$, será denotada por $\pi(\psi \mid x)$.

Seguindo o trabalho prévio de Lindley (1956), Bernardo (1979) propôs a seguinte medida de informação contida em E quando a densidade a priori é $\pi(.)$:

$$I\{E, \pi\} = \int p_X(x) \int \pi(\psi \mid x) \log \left\{ \frac{\pi(\psi \mid x)}{\pi(\psi)} \right\} d\psi dx. \tag{A.1}$$

Dado que a integral interna é essencialmente uma divergência de informação entre as densidades a priori e a posteriori, a expressão (A.1) pode ser interpretada como um "ganho de informação esperado" resultante da implementação do experimento E.

Em um contexto de monitoramento de redes, se E denota a decisão de selecionar um conjunto específico de estações de monitoramento, X são os dados gerados pela rede, e ψ é um parâmetro específico da variável aleatória de interesse, então (A.1) sugere um critério que pode ser usado para discriminar entre diferentes propostas para E, com a "melhor" rede sendo aquela que maximiza $I\{E, \pi\}$ para alguma densidade a priori π.

A seguir é descrita brevemente, a derivação da equação (A.1) feita por Bernardo (1979), a partir de alguns princípios básicos de teoria da decisão Bayesiana.

Uma formulação baseada em teoria da decisão

Seja $\pi^\dagger(\psi)$ a distribuição de probabilidade representando nossa crença reportada sobre ψ após conduzir o experimento. Em uma formulação do problema baseada em teoria da decisão, haverá uma função de utilidade, $u(\pi^\dagger(.), \psi)$, a qual representa o ganho em reportar a densidade $\pi^\dagger$ quando o valor verdadeiro da variável aleatória Ψ é ψ.

Suponha que o experimento produziu dados x, e seja $\pi(. \mid x)$ a distribuição a posteriori de Ψ dado x. A utilidade esperada é então

$$\int u(\pi^\dagger(.), \psi)\pi(\psi \mid x)d\psi. \tag{A.2}$$

Supõe-se que a função de utilidade possua as seguintes propriedades:

(a) $u(.)$ é *própria* se (A.2) é maximizada sobre todas as distribuições de probabilidade $\pi^\dagger$ ao selecionar $\pi^\dagger(\psi) = \pi(\psi \mid x)$. Esta propriedade está baseada na propriedade natural de que a solução ótima para o problema de decisão é a densidade a posteriori.

(b) u é *local* se $u(\pi^\dagger(.), \psi)$ depende da função $\pi^\dagger(.)$ unicamente através de seu valor em ψ, i.e., $\pi^\dagger(\psi)$. Isto quer dizer que a função de utilidade para um valor particular de ψ não deveria depender de outros valores de Ψ que ainda não ocorreram.

Bernardo (1979, p.689) enunciou o seguinte teorema: *Se u é própria e local, então ela deve ser da forma*

$$u(\pi^\dagger(.), \psi) = A \log \pi^\dagger(\psi) + V(\psi), \tag{A.3}$$

onde A é constante e V é uma função só de ψ.

Esboço da prova de equação (A.3):

Como u é local, $u(\pi^\dagger(.), \psi) = u(\pi^\dagger(\psi), \psi)$. Assim, u ser própria implica que:

Entre todas as funções $\pi^\dagger(.)$, e para todas as densidades de probabilidade $\pi(.)$, o valor de $\int u(\pi^\dagger(.), \psi)\pi(\psi)d\psi$, sujeito a $\int \pi^\dagger(\psi)d\psi = 1$, é maximizado quando $\pi^\dagger(.) = \pi(.)$.

Pelo princípio de multiplicadores de Lagrange, o $\pi^\dagger(.)$ ótimo é um ponto extremo do funcional

$$F\{\pi^\dagger(.)\} = \int u(\pi^\dagger(\psi), \psi)d\psi - A\left[\int \pi^\dagger(\psi)d\psi - 1\right] \tag{A.4}$$

para alguma constante A.

No entanto, para isto acontecer, é preciso ter

$$F_1\{\pi^\dagger(.)\} \equiv \frac{\partial}{\partial \epsilon} F\{\pi^\dagger(.) + \epsilon\tau(.)\}\Big|_{\epsilon=0} = 0, \tag{A.5}$$

para toda função τ suficientemente suave e pequena. Para verificar que (A.5) é verdadeira, basta expandir (A.4) em uma série de Taylor para ϵ pequeno e τ fixo: $F\{\pi^\dagger(.) + \epsilon\tau(.)\} = F\{\pi^\dagger(.)\} + \epsilon F\{\pi^\dagger(.)\}\tau(.) + o(\epsilon)$. Se $F_1 \neq 0$, o segundo termo pode ser positivo ou negativo para um $|\epsilon|$ suficientemente pequeno. Logo, $\pi^\dagger$ não pode ser um valor extremo do funcional F. Portanto, $F_1 = 0$, o qual corresponde à equação (A.5).

Ao avaliar a derivada em (A.5), esta condição se reduz a

$$\int u_1(\pi^\dagger(\psi), \psi)\pi(\psi)\tau(\psi)d\psi - A\int \tau(\psi)d\psi = 0 \tag{A.6}$$

onde u_1 denota a derivada parcial de primeira ordem de u com respeito ao seu primeiro argumento.

No entanto, para (A.6) ser verdadeira para todo τ, deve-se ter

$$u_1(\pi^\dagger(\psi), \psi)\pi(\psi) = A \tag{A.7}$$

para cada ψ.

Para u ser uma função de utilidade própria, (A.7) deve ser verdadeira quando $\pi^{\dagger}(\psi) = \pi(\psi)$, e assim

$$u_1(\pi(\psi), \psi)\pi(\psi) = A \tag{A.8}$$

para cada ψ.

Mas (A.8) nada mais é do que uma equação diferencial na quantidade (escalar) $\pi(\psi)$, e pode ser verificado que esta tem como solução

$$u(\pi^{\dagger}(\psi), \psi) = A \log \pi(\psi) + V(\psi)$$

onde $B(\psi)$ é uma constante dependendo unicamente de τ. Isto corresponde à equação (A.3).

Portanto, a utilidade esperada antes de realizar o experimento é

$$\int \{A \log \pi(\psi) + V(\psi)\}\pi(\psi)d\psi, \tag{A.9}$$

onde $\pi(\psi)$ é a distribuição a priori. O valor esperado da utilidade após realizar o experimento é

$$\int \left\{ \int \{A \log \pi(\psi \mid x) + V(\psi)\}\pi(\psi \mid x)d\psi \right\} p_X(x)dx, \tag{A.10}$$

onde $\pi(\psi \mid x)$ é a distribuição a posteriori dado $X = x$ e $p_X(x)$ é a distribuição marginal de X. Portanto, a diferença entre (A.10) e (A.9) é o ganho esperado na utilidade como resultado de realizar o experimento.

No entanto, o termo envolvendo $V(\psi)$ é o mesmo nas equações (A.9) e (A.10). Isto é devido à identidade $\pi(\psi) = \int \pi(\psi \mid x)p_X(x)dx$.

Assim, ignorando $V(\psi)$, a diferença entre (A.10) e (A.9) é

$$A \iint \{\log[\pi(\psi \mid x)]\pi(\psi \mid x) - \log[\pi(\psi)]\pi(\psi)\}d\psi p_X(x)dx. \tag{A.11}$$

No entanto, note-se também a identidade

$$A \iint \log \pi(\psi)\{\pi(\psi) - \pi(\psi \mid x)\}d\psi p_X(x)dx = 0, \tag{A.12}$$

a qual surge após permutar a ordem das integrais, devido ao fato de que $\int \pi(\psi \mid x)p_X(x)dx = \pi(\psi)$ para cada ψ.

Substituindo de (A.12), (A.11) passa a ser

$$A \iint \{\log[\pi(\psi \mid x)] - \log[\pi(\psi)]\} d\psi \pi(\psi \mid x) p_X(x) dx$$

que é a equação (A.1).

Seguindo a mesma notação do capítulo 3, suponha agora que o vetor aleatório p-dimensional $\boldsymbol{B}$ representa os valores do campo aleatório em todos os p sítios, e que $\boldsymbol{B} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, onde $\boldsymbol{\mu}$ e $\boldsymbol{\Sigma}$ são conhecidos. Suponha que $\boldsymbol{B}$ é subdividida como $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$, com $\boldsymbol{B}_1$ e $\boldsymbol{B}_2$ correspondendo aos conjuntos de estações não selecionadas e selecionadas da rede, respectivamente, e similarmente,

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_{B_1}, \boldsymbol{\mu}_{B_2})^T, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{B_1} & \boldsymbol{\Sigma}_{B_1|B_2} \\ \boldsymbol{\Sigma}_{B_2|B_1} & \boldsymbol{\Sigma}_{B_2} \end{bmatrix}.$$

No contexto de (A.1) pode-se identificar ψ com $\boldsymbol{B}_1$, a variável a ser predita, e X com $\boldsymbol{B}_2$, os dados medidos. Assim, de acordo com a equação (3.1), a informação ganha sobre $\boldsymbol{B}_1$ como resultado de medir $\boldsymbol{B}_2$ pode ser escrita na forma

$$I\{\boldsymbol{B}_1, \boldsymbol{B}_2\} = \int p_{B_2}(b_2) \int p_{B_1|B_2}(b_1 \mid b_2) \log \left\{ \frac{p_{B_1|B_2}(b_1 \mid b_2)}{p_{B_1}(b_1)} \right\} db_1 db_2.$$

ou, equivalentemente,

$$I(\boldsymbol{B_1}, \boldsymbol{B_2}) = I(\boldsymbol{B_1} \mid \boldsymbol{B_2}) - I(\boldsymbol{B_1}),$$

com $I(Y)$ denotando a informação em uma variável aleatória Y, i.e., $\int f(y) \log f(y) dy$, onde $f(y)$ é a densidade de Y avaliada em y. Assim, para o caso Gaussiano, tem-se que

$$f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-p/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{ -\frac{1}{2} (\boldsymbol{B} - \boldsymbol{\mu})^{Tr} \boldsymbol{\Sigma}^{-1} (\boldsymbol{B} - \boldsymbol{\mu}) \right\}, \quad \text{e}$$

$$I(\boldsymbol{B}) = E[\log f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})] = \int f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \log f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) d_{\boldsymbol{B}}, \quad \text{onde}$$

$$\log f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{p}{2} \log(2\pi) - \frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \left\{ (\boldsymbol{B} - \boldsymbol{\mu})^{Tr} \boldsymbol{\Sigma}^{-1} (\boldsymbol{B} - \boldsymbol{\mu}) \right\}, \quad \text{logo,}$$

$$E[\log f(\boldsymbol{B} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})] = -\frac{p}{2} \log(2\pi) - \frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} E\left[(\boldsymbol{B} - \boldsymbol{\mu})^{Tr} \boldsymbol{\Sigma}^{-1} (\boldsymbol{B} - \boldsymbol{\mu}) \right].$$

$$\text{mas} \quad E\left\{(\boldsymbol{B}-\boldsymbol{\mu})^{Tr}\boldsymbol{\Sigma}^{-1}(\boldsymbol{B}-\boldsymbol{\mu})\right\} = p, \quad \text{portanto,}$$

$$I(\boldsymbol{B}) = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{p}{2} = -\frac{p}{2}[1+\log(2\pi)] - \frac{1}{2}\log|\boldsymbol{\Sigma}|.$$

A distribuição condicional de $\boldsymbol{B}_1$ dado $\boldsymbol{B}_2$ é normal com variância $\boldsymbol{\Sigma}_{B_1|B_2} = \boldsymbol{\Sigma}_{B_1} - \boldsymbol{\Sigma}_{B_1|B_2}\boldsymbol{\Sigma}_{B_2}^{-1}\boldsymbol{\Sigma}_{B_2|B_1}$, enquanto que a variância não condicional de $\boldsymbol{B}_1$ é $\boldsymbol{\Sigma}_{B_1}$. Logo,

$$\begin{aligned}
I(\boldsymbol{B_1},\boldsymbol{B_2}) &= I(\boldsymbol{B_1} \mid \boldsymbol{B_2}) - I(\boldsymbol{B_1}) \\
&= -\frac{p}{2}\{1+\log(2\pi)\} - \frac{1}{2}\log\left|\boldsymbol{\Sigma}_{B_1|B_2}\right| + \frac{p}{2}\{1+\log(2\pi)\} + \frac{1}{2}\log|\boldsymbol{\Sigma}_{B_1}| \\
&= -\frac{1}{2}\log\left|\boldsymbol{\Sigma}_{B_1|B_2}\right| + \frac{1}{2}\log|\boldsymbol{\Sigma}_{B_1}| \\
&= -\frac{1}{2}\log\left|\boldsymbol{\Sigma}_{B_1}^{-1}\boldsymbol{\Sigma}_{B_1|B_2}\right| \\
&= -\frac{1}{2}\log\left|\boldsymbol{\Sigma}_{B_1}^{-1}[\boldsymbol{\Sigma}_{B_1} - \boldsymbol{\Sigma}_{B_1|B_2}\boldsymbol{\Sigma}_{B_2}^{-1}\boldsymbol{\Sigma}_{B_2|B_1}]\right| \\
&= -\frac{1}{2}\log\left|I - \boldsymbol{\Sigma}_{B_1}^{-1/2}\boldsymbol{\Sigma}_{B_1|B_2}\boldsymbol{\Sigma}_{B_2}^{-1}\boldsymbol{\Sigma}_{B_1}^{-1/2}\right| \qquad \text{(A.13)} \\
&= -\frac{1}{2}\log|I-P|\,.
\end{aligned}$$

Mas $P = \boldsymbol{\Sigma}_{B_1}^{-1/2}\boldsymbol{\Sigma}_{B_1|B_2}\boldsymbol{\Sigma}_{B_2}^{-1}\boldsymbol{\Sigma}_{B_1}^{-1/2}$ é uma matriz simétrica e, portanto, pode ser escrita na forma QDQ^T, com Q ortogonal e D diagonal. Assim, outra forma de escrever (A.13) é

$$\begin{aligned}
I(\boldsymbol{B_1},\boldsymbol{B_2}) &= -\frac{1}{2}\log|I-D| \\
&= -\frac{1}{2}\log\prod_{i=1}^{n}(1-\rho_i^2),
\end{aligned}$$

onde $\rho_1^2, \cdots, \rho_n^2$ são os autovalores de P ou, equivalentemente, as correlações canônicas ao quadrado entre $\boldsymbol{B}_1$ e $\boldsymbol{B}_2$. Portanto, outra forma de expressar este critério é escolher a subdivisão que faz as correlações canônicas entre $\boldsymbol{B}_1$ e $\boldsymbol{B}_2$ grandes, no sentido de minimizar $\prod_{i=1}^{n}(1-\rho_i^2)$.

Apêndice B

Partição da Função de Entropia

Seja $\boldsymbol{B} = (\boldsymbol{B_1}, \boldsymbol{B_2})$, uma partição dos m potenciais sítios de amostragem do campo aleatório discretizado $\boldsymbol{B}$, onde $\boldsymbol{B}_1$ representa o subconjunto dos $m - n$ sítios que não serão monitorados, e $\boldsymbol{B}_2$ representa o subconjunto dos n sítios que serão monitorados

Definição: A quantidade total de incerteza na distribuição conjunta de $\boldsymbol{B}$ é dada por

$$H(\boldsymbol{B_1}, \boldsymbol{B_2}) = H(\boldsymbol{B_1} \mid \boldsymbol{B_2}) + H(\boldsymbol{B_2})$$

prova:

$$\begin{aligned}
H(\boldsymbol{B_1}, \boldsymbol{B_2}) &= E_{(\boldsymbol{B_1}, \boldsymbol{B_2})}[-\log p(\boldsymbol{B_1}, \boldsymbol{B_2})] \\
&= E_{(\boldsymbol{B_1}, \boldsymbol{B_2})}[-\log p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) p(\boldsymbol{B_2})] \\
&= \underbrace{E_{(\boldsymbol{B_1}, \boldsymbol{B_2})}[-\log p(\boldsymbol{B_1} \mid \boldsymbol{B_2})]}_{(1)} + \underbrace{E_{(\boldsymbol{B_1}, \boldsymbol{B_2})}[-\log p(\boldsymbol{B_2})]}_{(2)}
\end{aligned}$$

$$\begin{aligned}
E_{(\boldsymbol{B_1}, \boldsymbol{B_2})}[-\log p(\boldsymbol{B_1} \mid \boldsymbol{B_2})] &= \iint p(\boldsymbol{B_1}, \boldsymbol{B_2}) \log p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) d_{\boldsymbol{B_1}} d_{\boldsymbol{B_2}} \\
&= \iint p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) p(\boldsymbol{B_2}) \log p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) d_{\boldsymbol{B_1}} d_{\boldsymbol{B_2}} \\
&= \int p(\boldsymbol{B_2}) \left[\int p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) \log p(\boldsymbol{B_1} \mid \boldsymbol{B_2}) d_{\boldsymbol{B_1}} \right] d_{\boldsymbol{B_2}} \\
&= H(\boldsymbol{B_1} \mid \boldsymbol{B_2}) \int p(\boldsymbol{B_2}) d_{\boldsymbol{B_2}} \\
&= H(\boldsymbol{B_1} \mid \boldsymbol{B_2})
\end{aligned}$$

$$
\begin{aligned}
E_{(\boldsymbol{B_1},\boldsymbol{B_2})}[-\log p(\boldsymbol{B_2})] &= \iint p(\boldsymbol{B_1},\boldsymbol{B_2}) \log p(\boldsymbol{B_2}) d_{\boldsymbol{B_1}} d_{\boldsymbol{B_2}} \\
&= \int \log p(\boldsymbol{B_2}) \left[\int p(\boldsymbol{B_1},\boldsymbol{B_2}) d_{\boldsymbol{B_1}} \right] d_{\boldsymbol{B_2}} \\
&= \int \log p(\boldsymbol{B_2}) p(\boldsymbol{B_2}) d_{\boldsymbol{B_2}} \\
&= H(\boldsymbol{B_2})
\end{aligned}
$$

Assim, de (1) e (2) tem-se que $H(\boldsymbol{B_1},\boldsymbol{B_2}) = H(\boldsymbol{B_1} \mid \boldsymbol{B_2}) + H(\boldsymbol{B_2})$ $\square$

Apêndice C

Seleção de Parâmetros para o Algoritmo Genético

Tamanho da população

O tamanho da população usado na implementação do algoritmo genético foi estimada seguindo o critério desenvolvido por Harik et al. (1999), o qual está baseado em dois fatores que influenciam diretamente a qualidade das soluções que o algoritmo genético pode alcançar: o fornecimento inicial de blocos construtores, e a seleção dos melhores blocos construtores sobre seus competidores. Aqui a noção de blocos construtores está restrita aos "schematas" (seqüências binárias curtas que quando presentes em uma solução conferem um alto valor de ajuste) de mínima ordem que contribuem para o ótimo global.

A equação proposta por Harik et al. (1999) é dada por:

$$N \geq -2^{K-1} \ln(\alpha) \left(\frac{\sigma_{bb}\sqrt{\pi(m-1)}}{d} \right), \tag{C.1}$$

onde N representa o tamanho da população, K representa a ordem (número de bits) do bloco construtor, α é a probabilidade de falha do algoritmo, σ_{bb} é o desvio padrão do ajuste dos blocos construtores, m é o número máximo de blocos construtores em cada solução, e d é a diferença entre o ajuste da melhor e da segunda melhor solução. A aplicação da Equação C.1 para avaliar o tamanho

populacional não é plausível devido ao fato de que os blocos construtores são desconhecidos. Mas, segundo Harik et al. (1999), a equação C.1 pode ser aproximada por:

$$N \geq -2^{K-1} \ln(\alpha) \left(\frac{\sigma_f}{d} \right)$$

onde σ_f é o desvio padrão do ajuste dos indivíduos de uma população grande inicializada aleatoriamente. Para este estudo foi considerada uma população inicial de 1000 indivíduos. O parâmetro d foi aproximado pela diferença no ajuste dos melhores membros dessa população, selecionados com uma abordagem baseada em teoria de análise de freqüências, na qual o histograma da função de ajuste é usado para representar as diferentes classes de ajuste que podem ser estatisticamente identificadas. d é avaliado como o tamanho da primeira classe desse histograma.

A ordem verdadeira dos blocos construtores, K, na equação C.1 também é desconhecida, mas pode-se assumir que varia entre 1 e 4 (Reed et al., 2000). Neste estudo, o tamanho populacional foi calculado para $K = 1$, $K = 2$, $K = 3$, e $K = 4$. De acordo com Reed et al. (2000), blocos construtores de ordem maior, usualmente são desintegrados pelo operador de cruzamento e, portanto, não foram considerados.

Cabe ressaltar que as estimativas de tamanho populacional obtidas com a equação C.1 são conservadoras, já que o valor de σ_f diminui a medida que o algoritmo converge.

A tabela C.1 mostra os tamanhos populacionais estimados para as diferentes ordens dos blocos construtores

Tabela C.1: Tamanhos mínimos de população N para o algoritmo genético padrão, seguindo o critério de Harik et al. (1999).

grade	n	K				
		1	2	3	4	5
5×5	5	8	16	32	64	128
5×5	10	7	14	28	56	111
10×10	20	8	16	31	61	122
10×10	40	6	11	21	42	84
20×20	40	11	21	42	83	166
20×20	80	6	12	23	45	90
30×30	90	5	10	20	40	80
30×30	180	6	11	21	41	81
50×50	125	7	13	26	51	101
50×50	250	9	17	33	65	129

n: número de estações a serem incluídas na rede; *K:* ordem dos blocos construtores.

Com base nesta análise, foi fixado para todos os problemas considerados, um tamanho da população de $N = 100$ indivíduos.

Probabilidades de Cruzamento e mutação

Para estudar como o valor da função objetivo dos melhores indivíduos da população varia como uma função das probabilidades de cruzamento e mutação, foi realizado um pequeno estudo de caso, implementando o algoritmo genético padrão sobre uma grade 10×10, correspondente a um campo aleatório discretizado com 100 potenciais sítios de amostragem. O número de pontos a serem selecionados foi fixado em 20. O espaço de busca do algoritmo genético para este caso foi, portanto, de 5^{20} possíveis soluções. O critério de entropia foi considerado como critério de desenho. Assim, o objetivo foi maximizar $H(\boldsymbol{Z_2}) = \frac{1}{2}\log \mid \boldsymbol{\Sigma_{Z_2}} \mid$, com $\boldsymbol{Z_1}$ e $\boldsymbol{Z_2}$ como descritos na seção 3.2.

O número de gerações do algoritmo foi fixado em 100, e o tamanho da população foi de 100 indivíduos como definido previamente. As probabilidades de cruzamento consideradas variaram entre 0 e 1, com intervalos de 0,1, enquanto que as probabilidades de mutação variaram entre 0 e 0,1, com

intervalos de 0,01. A figura C.1 mostra os valores médios e o desvio padrão da função objetivo para os melhores indivíduos encontrados pelo algoritmo genético como uma função destas probabilidades.

Os resultados apresentados nesta figura ilustram um desempenho pobre do algoritmo quando as probabilidades de mutação são muito baixas (abaixo de 0,03), para qualquer valor da probabilidade de cruzamento.

Considerando um esquema de seleção por torneio, para torneios de tamanho s, Thierens (1995) mostrou que a cota superior de probabilidade de cruzamento P_{cross} é dada por:

$$P_{cross} < \frac{s-1}{s}.$$

Para assegurar máxima inovação, Reed et al. (2000) recomendam selecionar P_{cross} igual à cota superior. Assim, levando em consideração os resultados da figura C.1, e considerando que neste estudo foi usada uma seleção por torneio com parâmetro $s = 4$, foi adotada uma probabilidade de cruzamento $P_{cross} = 0.75$.

De acordo com De Jong (1975), um bom desempenho do algoritmo genético padrão pode ser obtido quando a probabilidade de mutação P_{mut} é selecionada usando a relação:

$$P_{mut} \approx \frac{1}{N},$$

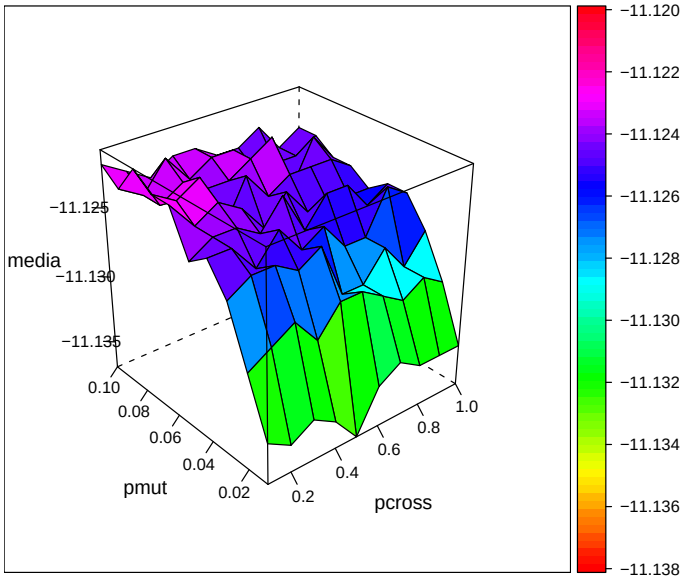
onde N é o tamanho populacional.

Uma expressão alternativa para avaliar a probabilidade de mutação é dada por:

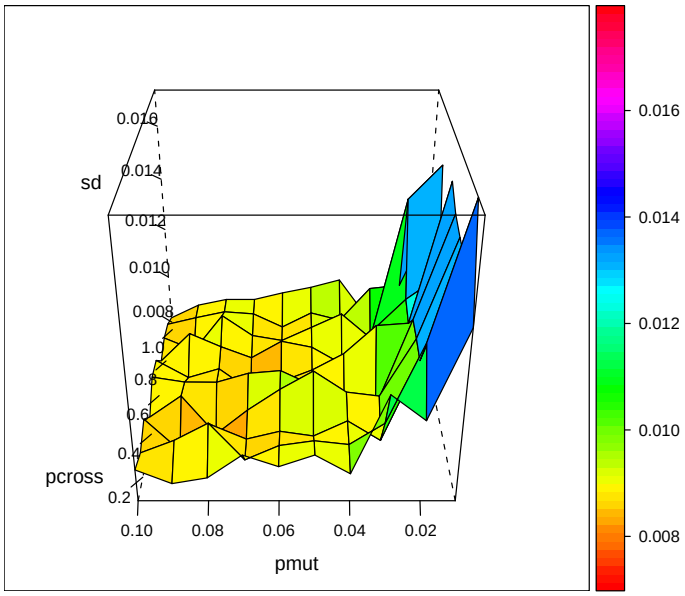
$$P_{mut} \approx \frac{1}{L},$$

onde L é o comprimento das soluções.

No entanto, a aplicação destes critérios com os valores de N e L usados neste estudo levaria à seleção de probabilidades de mutação muito pequenas, que de acordo com os resultados da figura C.1, levaria por sua vez a algoritmos genéticos com um desempenho pobre. Portanto, para este estudo foi selecionada uma probabilidade de mutação $P_{mut} = 0.05$.



(a) média



(b) desvio padrão

Figura C.1: Média (a) e desvio padrão (b) da função objetivo obtidos para as melhores soluções como uma função das probabilidades de cruzamento (*pcross*) e mutação (*pmut*) no algoritmo genético padrão.

Apêndice D

Seleção de Parâmetros para o Algoritmo Simulated Annealing

Um estudo de caso similar ao descrito no apêndice C foi usado para avaliar como o valor da função objetivo das melhores soluções varia como uma função dos fatores de desconto e de acréscimo no algoritmo *simulated annealing*. Assim, o algoritmo foi implementado sobre uma grade 10×10, correspondente a um campo aleatório discretizado com 100 potenciais sítios de amostragem. O número de pontos a serem selecionados foi fixado em 20. O espaço de busca do algoritmo para este caso foi, portanto, de 5^{20} possíveis soluções. O critério de entropia foi considerado como critério de desenho. Assim, o objetivo foi maximizar $H(\boldsymbol{B_2}) = \frac{1}{2}\log \mid \boldsymbol{\Sigma_{B_2}} \mid$, com $\boldsymbol{B_1}$ e $\boldsymbol{B_2}$ como descritos na seção 3.2.

Os fatores de desconto considerados variaram entre 0,85 e 1, com intervalos de 0,01, enquanto que os fatores de acréscimo para reiniciar o algoritmo variaram entre 1 e 1,5, com intervalos de 0,1. O exercício anterior foi repetido considerando diferentes cenários, tais como: fator de desconto aplicado a cada 50, 100 ou 150 iterações, fator de acréscimo para reiniciar o algoritmo aplicado a cada 500, 1000 ou 1500 iterações, e número total de iterações do algoritmo variando entre 10000 e 50000 iterações com intervalos de 10000 iterações.

O melhor desempenho do algoritmo foi encontrado com um número total de iterações de 50000, e fatores de desconto e acréscimo aplicados a cada 100 e 1000 iterações, respectivamente. Para estas

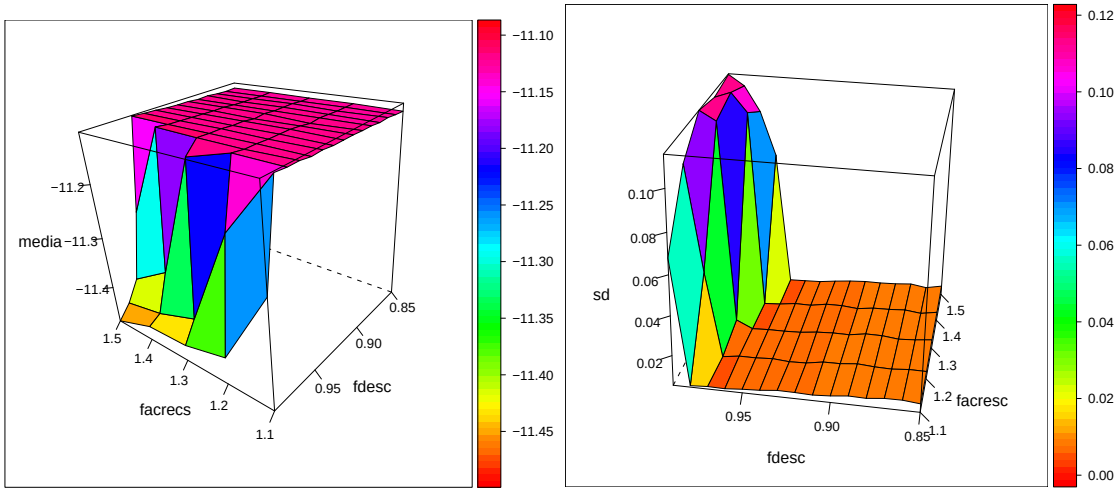
escolhas, a figura D.1 mostra os valores médios e o desvio padrão dos melhores valores da função objetivo encontrados pelo algoritmo *simulated annealing* como uma função dos fatores de desconto e de acréscimo em 100 repetições, assim como o número de vezes que o melhor valor conhecido da função objetivo foi encontrado.

Os resultados apresentados nesta figura ilustram um pobre desempenho do algoritmo (menores médias e desvios padrões maiores) quando são usados fatores de desconto maiores do que 0,96, para qualquer valor de fator de acréscimo no intervalo considerado. No entanto, considerando somente fatores de desconto até 0,95, pode ser notada uma tendência crescente nos valores médios da função objetivo e decrescente nos desvios padrões.

Por outro lado, o número de vezes que o melhor valor conhecido da função objetivo foi encontrado, foi maior quando usados fatores de acréscimo maiores do que 1,3, e fatores de desconto maiores do que 0,93.

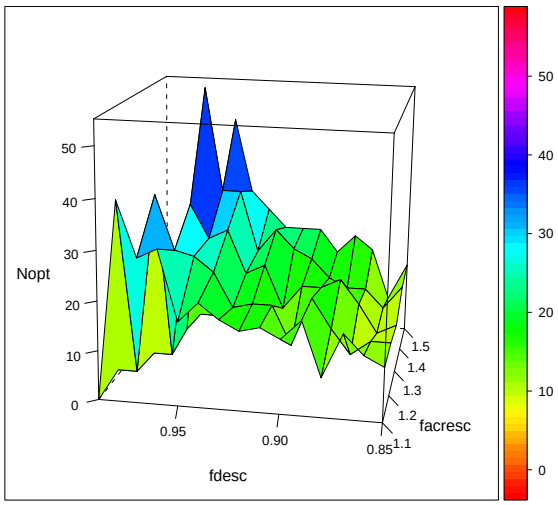
Baseado nesses resultados, foram escolhidos os seguintes parâmetros, para a implementação do algoritmo *simulated annealing*:

- fator de desconto de 0,95 aplicado a cada 100 iterações
- fator de acréscimo de 1,5 aplicado a cada 1000 iterações
- Número total de iterações = 50000



(a) média

(b) desvio padrão



(c) Nopt

Figura D.1: Média (a), desvio padrão (b) e número de vezes que o melhor valor conhecido da função objetivo foi encontrado (c) como função dos fatores de desconto (*fdesc*) e de acréscimo (*facr*) em 100 repetições do algoritmo *simulated annealing*.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo