UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

Projeto de Inovação com Algoritmos Genéticos

Relatório de Disciplina

Orientador: ALEXANDRE C. B. DELBEM

DANIEL BONETTI, DANILO SANCHES,
DANILO VARGAS, HELSON MINEIRO,
HERMES CASTELO, JEAN MARTINS,
MÁRCIO BELO FILHO, MARCIO CROCOMO
MARCILYANNE GOIS, RAPHAEL PHILIPE,
TIAGO VIEIRA DA SILVA

São Carlos
2010

Resumo

C. B. Delbem, Alexandre. **Projeto de Inovação com Algoritmos Genéticos**. São Carlos, 2010. 37p. Relatório de Disciplina. Departamento de Sistemas de Computação, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

Texto gerado pelos alunos da disciplina durante os anos 2009 e 2010 no qual busca-se esclarecer de forma didática os conceitos apresentados em [7]. A partir disso, a linguagem do texto adicionado aos exemplos que serão apresentados, torna-se possível a leitura e aprendizado por um iniciante da área. A leitura busca elucidar os principais pontos que vêm permitindo a criação de Algoritmos Genéticos (AGs) que sejam eficientes, precisos e confiáveis. Neste contexto, visa-se o entendimento dos mecanismos básicos de funcionamento dos AGs, para que, desta forma, seja possível uma exploração mais consciente dos parâmetros envolvidos e a formação de uma teoria que descreva o comportamento dos AGs em critérios como: Otimalidade e Tempo de convergência.

Palavras–chave:

Algoritmos Genéticos Competentes, Modelos Teóricos, Projeto de Inovação

Sumário

Capítulo 1

Introdução

Algoritmos Genéticos (AG) podem ser definidos como meta-heurísticas populacionais inspiradas em elementos de Biologia evolutiva [6]. Este texto tem como foco elucidar os principais pontos que vêm permitindo a criação de Algoritmos Genéticos Competentes, ou seja, AGs que solucionem problemas difíceis de forma rápida, precisa e confiável [8].

Há muitos anos os AGs são utilizados como ferramentas experimentais, de modo que diversos tipos de operadores foram desenvolvidos com o objetivo de torná-los mais robustos. No entanto, como será visto ao decorrer do texto, grande parte desses mecanismos exploram apenas superficialmente a capacidade dos AGs, não sendo possível, desta forma, uma compreensão generalizada e a identificação dos pontos realmente importantes.

Neste contexto, o estudo do projeto de AGs Competentes é um ponto importante à evolução dos AGs, visto que possibilita o entendimento dos mecanismos básicos de funcionamento dos algoritmos, permitindo assim uma exploração mais consciente dos parâmetros envolvidos.

Utiliza-se como ponto de partida, neste texto, o trabalho de Holland [9], no qual conceitos básicos, como o de *Building Block*, são elaborados e o trabalho de Goldbert [8] no qual tais conceitos são adicionados à noções como: *Fornecimento de Building Blocks* e *Identificação de Building Blocks*. Concluindo, e descrevendo, a partir deles uma teoria sobre o projeto de AGs Competentes que garanta Otimalidade em um tempo de convergência esperado.

Inicia-se, no capítulo 1.1, a descrição dos elementos fundamentais utilizados como referência para a investigação, seguido, nos Capítulos 2 e 3, pela especificação de conceitos importantes à garantia de otimalidade e tempo de convergência. Já no Capítulo 4 a complexidade de tempo do AGs selecto-recombinativo descrito no Capítulo 1.1 é especificada, seguido, no Capítulo 5, pelas conclusões e considerações finais.

1.1 Introdução a Modelos Teóricos

Uma vez que o tempo de convergência é um fator importante, que pode determinar se um problema é ou não tratável. E sabendo que os parâmetros dos algoritmos em geral e em especial dos algoritmos genéticos podem ser difíceis de ajustar. O objetivo da construção de modelos teóricos é a estimação do tempo de convergência e a estimação dos parâmetros dos algoritmos genéticos (AGs). Além disso, com o uso destes modelos será possível garantir a qualidade das soluções e determinar em que classe de problemas pode-se assegurar isso.

Em virtude disso, uma consequência de ter os modelos teóricos é poder utilizá-los como um guia em projetos de algoritmos genéticos (por exemplo na escolha de parâmetros ou mesmo na determinação automática destes), podendo também auxiliar o desenvolvimento de meta heurísticas em geral.

A técnica de modelagem a ser discutida envolve a escolha de um AG minimal, representativo de sua classe e capaz de resolver problemas relativamente complexos. Para a modelagem é necessário também escolher problemas, os quais necessitam ser relativamente complexos, com controle de complexidade por meio de parâmetros e que enfatizam características necessárias nos algoritmos para que estes problemas sejam resolvidos.

Por problemas complexos pode-se considerar funções multimodais, as quais possuem vários pontos de ótimo, podendo ou não conter mais de um ótimo global. Em problemas desta natureza, por exemplo, o AG enfrenta dificuldades e mudar os parâmetros pode não ajudar. Por exemplo, aumentar a taxa de mutação pode aumentar a exploração do algoritmo (aproximando-o cada vez mais de uma busca aleatória), o que não necessariamente garante uma busca adequada, pois pode gerar muitas soluções de baixa qualidade.

Existem, porém, modelagens teóricas que descrevem as relações de parâmetros para certas classes de problemas e além de se modificar os parâmetros e os operadores dos AGs é possível lidar com problemas complexos a partir de decomposições. Almejando encontrar seus subproblemas.

Considerando mutação baixa ou ausente (tornando o AG menos parecido com uma busca aleatória) e utilizando crossover ideal (apenas trabalhando dentro de subproblemas, ou seja, aonde há independência entre variáveis). Consequentemente, este AG se resume a um algoritmo genético selecto recombinativo (AGsr) com população fixa, seleção e crossover apenas.

1.2 Algoritmo Genético Selecto Recombinativo (AGsr)

O AGsr utiliza três parâmetros: tamanho da população, pressão de seleção (*ps*) e probabilidade de crossover (*pc*).

O *ps* é definido pelo número de vezes que os selecionados serão copiados para compor a nova população, enquanto o *pc* é a probabilidade do crossover ser aplicado em cima de dois indivíduos selecionados.

1.3 Problema a ser Modelado

Para modelar o efeito de funções multimodais foi escolhido combinações lineares de funções simples. Esta função é descrita pela equação 1-1, considerando um vetor x com m blocos de tamanho k aonde $d = \frac{1}{k}$.

$$max\ f_{deceptiva}(x),\ x \in \{0,1\}^{mk}$$

$$f_{deceptiva}(x) = \sum_{i=0}^{m-1} f_{trap}(x_{ki} + x_{ki+1} + ... + x_{ki+k-1})$$

$$f_{trap}(u) = \begin{cases} 1 & u = k \\ 1 - d - u * \frac{1-d}{k-1} & otherwise \end{cases} \quad (1\text{-}1)$$

A Figura 1.1 ilustra a função f_{trap} descrita anteriormente. Observe que a relação entre as funções f_{trap} na função deceptiva são combinações lineares, porém combinações mais complexas como lineares ou com sobreposição de variáveis poderia ter sido usada.

A próxima seção definirá o critério de convergência.

1.4 Critério de Convergência

O critério de convergência considerado não pode assumir 100% de acerto, uma vez que isso necessitaria de parâmetros infinitos. Para tanto assume-se que quando pelo menos $\frac{m-1}{m}$ blocos convergiram corretamente, que o algoritmo convergiu corretamente.

1.5 Otimização Black-Box

Os algoritmos evolutivos [7], *differential evolution* [14], *simulated annealing* [11] e busca aleatória são exemplos de algoritmos de otimização black-box. Eles são chamados desta maneira, porque encaram o problem a ser otimizado como um black-box,

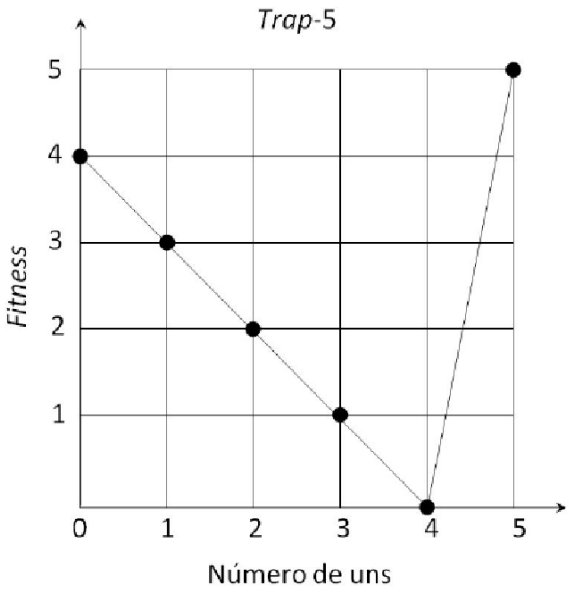

Figura 1.1: *Função Deceptiva*

ou seja, o problema é visto apenas em função de susas entradas e saídas, sendo inexistente qualquer conhecimento a priori. Neste contexto, esperar-se-ia comparar algoritmos em relação à sua eficiência média em todas as funções de otimização de um dado grupo. Esta comparação, porém, é contestada pelo *No Free Lunch Theorem* (detalhes na Seção 1.5.1). No entanto, como será visto na Seção 1.5.2, o *No Free Lunch Theorem* é mais uma excessão do que a regra.

1.5.1 *No Free Lunch Theorem* (NFL)

O *No Free Lunch Theorem* (NFL) define que, quando avaliado sobre todas os problemas de otimização possíveis eles produzem na média a mesma eficiência.

Segundo o NFL, dado uma função objetivo f, a qual pode ser representada por um mapa entre o espaço de busca e o espaço finito de avaliações, isto é, $f : X \to Y$. Tendo como verdade que um conjunto de funções F seja fechado em permutações (FEP), isto é, qualquer permutação do mapa $f : X \to Y$ está dentro de F [13]. Então, dado um algoritmo a iterado m vezes em uma função f, a probabilidade de obter um conjunto de d_m^y de amostras de avaliações em um conjunto ordenado temporalmente d_m com distintos pontos visitados é dado por $P(d_m^y|f,m,a)$ e, de acordo com [17],[16] o Teorema 1 é obtido.

Teorema 1 *Para qualquer par de algoritmos a_1 e a_2.*

$$\sum_f P(d_m^y|f,m,a_1) = \sum_f P(d_m^y|f,m,a_2).$$

Os mesmos resultados se aplicam para problemas de otimização dinâmicas [17].

Limitações do NFL

A condição para que o NFL seja válida acontece raramente na prática, pois subconjuntos dos dados são raramente FEP. Para ilustrar o quão raro, suponha-se o seguinte exemplo, dado uma função Booleana $f : 0,1^3 \rightarrow 0,1$ a fração de subconjuntos não vazios FEP é $<< 10^{-170}$[10]. Além disso, [10] mostra que com o aumento do tamanho do espaço de busca $|X|$ ou com o aumento do tamanho do espaço de avaliações $|Y|$, a fração decresce rapidamente. Exemplos de limitações do NFL são observados em [2], neste artigo os autores mostram que em cenários realísticos as técnicas de otimização diferem em sua eficiência.

1.5.2 *Free Lunches*

Free Lunches (FL) foram encontrados no uso de *encondings* [15] e em cenários coevolutivos [18]. Adicionalmente, foi mostrado em [12] que encontra-se alguns FL na prática, quando se usa algoritmos como *hyper-heuristics, genetic programming, computer scientists* e outros algoritmos aonde se busca num espaço de métodos para encontrar uma solução. É importante notar, que o campo das *hyper heuristics* é fortemente correlacionado com o de meta-otimização, no qual inclui-se os *Estimation of Distribution Algorithms*. Consequentemente, este resultado abre possibilidades para se justificar a pesquisa algoritmos genéticos mais abrangentes sem necessitar de informações específicas do problema em questão.

Capítulo 2

Reduzindo os Riscos de Ótimos Locais

A competição entre instâncias de um bloco é um dos principais fatores que podem direcionar a busca a ótimos locais [8]. Caso os subproblemas não tenham sido bem amostrados, pode ocorrer de *instâncias ruins* proliferarem pela população, impedindo que instâncias ótimas dos blocos aumentem em quantidade e, consequentemente, impedindo que ótimos globais sejam encontrados.

O risco que esse tipo de situação ocorra varia a cada geração, no entanto, é sabido que tal risco diminui à medida que aumenta o número de amostras de blocos ótimos na população, ou seja, quanto mais instâncias ótimas de blocos na população menor a probabilidade de que a busca termine em um ótimo local.

Neste capítulo, essa situação é abordada de modo a explicitar por quais meios a quantidade de instâncias ótimas de blocos na população pode ser mantida em um nível alto suficiente para que se garanta a obtenção de ótimos globais. Inicia-se com a Seção 2.1 em que a competição entre instâncias de blocos é discutida com mais detalhes, seguida pela Seção 2.2, na qual é feita uma analogia entre a competição de instâncias de blocos e o problema da Ruína do Jogador, em sequência, na Seção 2.3, um tamanho aproximado para o tamanho da população é estimado, usando como base o mesmo problema.

2.1 Competição entre Instâncias de Blocos

Um bloco (ou subproblema) é constituído por um conjunto de variáveis que possuem forte dependência entre si. Neste texto, uma "instância de bloco" se refere a um conjunto de valores assumido pelas variáveis do bloco. A Tabela 2.1 descreve exemplos de diferentes instâncias para o subproblema formado pelas variáveis $\{5,6,7,8\}$.

É através da mistura de instâncias ótimas para os subproblemas que soluções ótimas podem ser encontradas, no entanto, soluções ótimas para os subproblemas podem estar em minoria na população, e nesses casos existe a possibilidade de que elas sejam descartadas ao decorrer das gerações.

Variáveis	1	2	3	4	5	6	7	8
Instância 1					1	0	0	1
Instância 2					0	0	0	0
Instância 3					1	1	1	1

Tabela 2.1: *Diferentes instâncias de um mesmo bloco.*

Neste contexto é que ocorre a competição entre instâncias de blocos, em que soluções subótimas para os subproblemas têm a oportunidade de se perpetuarem fortemente ao longo das gerações, dificultando o crescimento de instâncias ótimas, e consequentemente impedindo soluções ótimas de serem encontradas.

É importante ressaltar que essa competição ocorre porque a seleção de boas soluções não pode ser feita de acordo com o *fitness* dos subproblemas, pois tais valores não são conhecidos. A seleção é feita sobre soluções completas, e nada impede que uma solução de baixa qualidade possua algumas instâncias ótimas para determinados subproblemas.

Para entender melhor este problema, considere que a solução ótima a ser encontrada é dada pela *string* `11111`. Considere também um esquema que contenha essa *string*: `1####`. O *fitness* deste esquema é dado pela média dos *fitness* de todas as soluções que possuam este esquema. É possível, que o *fitness* do esquema `0####` seja maior do que do esquema `1####`. Isto faz com que o esquema selecionado para a próxima geração, não esteja conduzindo a solução ótima de nosso problema.

Nosso problema então, é o de seleção de esquemas. Diferente do problema de selecionar uma melhor solução, ao selecionar um melhor esquema, estamos olhando não para a seleção de valores pontuais, mas para a seleção de uma distribuição de valores.

2.2 A Ruína do Jogador e a Competição entre Blocos

Um jogador de azar, com uma quantidade inicial a de recursos, decide apostar para aumentar seus recursos, sendo n a quantidade máxima a ser obtida. Cada aposta possui uma probabilidade p de vencer, e uma probabilidade $q = 1 - p$ de perder. Ao perder a aposta, a quantidade de recursos do jogador diminui em uma unidade, e ao vencer, a quantidade de recursos do jogador aumenta em uma unidade [3]. O jogador para de apostar quando:

1. Sua quantidade de recursos chega a 0, ou
2. Atinge a quantidade de recursos n desejada.

Queremos saber qual é a probabilidade P_n de que o jogador vença (segunda opção listada acima). A solução para este problema é conhecida, e dada pela Equação (2-1).

$$P_n = \frac{1-(q/p)^a}{1-(q/p)^n} \tag{2-1}$$

Ou seja, a partir de uma quantidade a de dinheiro, e podendo atingir uma quantidade n, o jogador vencerá, conseguindo para si todo o dinheiro com uma probabilidade dada por P_n.

Esse problema é análogo à competição entre instâncias de blocos, ou seja, supondo duas instâncias (dois jogadores) pode-se considerar que cada instância de bloco tem como objetivo aumentar a sua quantidade a de ocorrências na população (dinheiro), de forma que tal instância possa vencer o "jogo" dominando as instâncias (n) contidas na população [8].

Suponha os dois esquemas, `H1=1####` e `H2=0####`, citados anteriormente, competindo para dominar uma população. `H1` representando o melhor e `H2` o segundo melhor esquema. A Figura 2.1 ilustra a distribuição destes esquemas em um população.

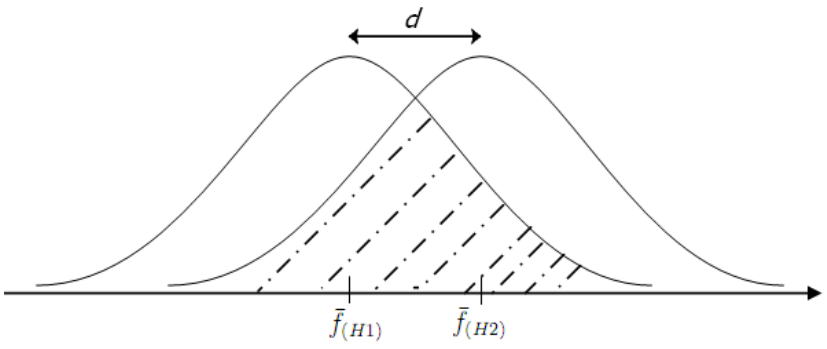

Figura 2.1: *Distribuições de esquemas concorrentes. A área de intersecção entre as duas distribuições possibilita que o pior destes esquemas seja o selecionado.*

A área de intersecção entre as duas distribuições possibilita que o pior destes esquemas seja o selecionado. Esta probabilidade de erro aumenta quando nossa amostra é pequena: ao amostrar apenas um indivíduo para cada um dos esquemas, podemos ter uma alta chance selecionar o esquema não desejado.

Agora imagine a seleção de uma amostra maior, como por exemplo, 25 amostras de cada esquema. Admita que o esquema vencedor será aquele com melhor *fitness* médio, dentre $\overline{f}(H1)$ e $\overline{f}(H2)$. A Figura 2.2 ilustra a distribuição destas médias.

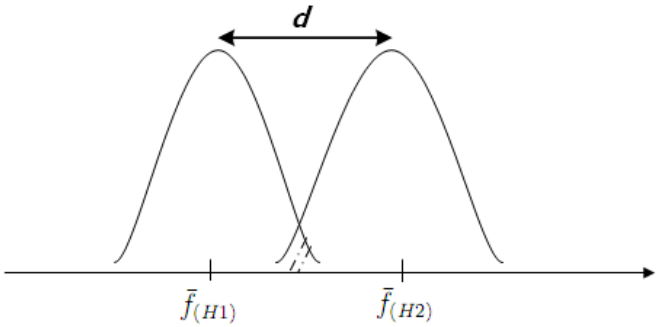

Figura 2.2: *Utilizando a média de 25 amostras a área de intersecção entre as duas distribuições diminui. Reduzindo assim, a chance de selecionar o segundo melhor esquema.*

Pode-se observar, que à medida que aumentamos nossas amostras, diminuímos a área de intersecção entre as distribuições, reduzindo assim a probabilidade de selecionarmos o esquema pior.

2.3 A Ruína do Jogador e o Tamanho da População

Pode-se então utilizar um teste de hipótese de diferenças entre médias para calcular a probabilidade de seleção do indivíduo, como mostrado na Equação 2-2. Deve-se primeiramente calcular o desvio padrão (σ^2) da média dos blocos da população para a hipótese da instância ótima ($H_1 = 11111$), ou seja, $\sigma^2_{H_1}$ (Figura 2.4). É necessário também fazer o cálculo para a hipótese alternativa subótima ($H_2 = 00000$) para encontrar $\sigma^2_{H_2}$. É considerado somente duas hipóteses porque são elas que irão dominar a população após um certo número de gerações, isto é, as chances de uma outra solução como, por exemplo, 10010 dominar a população é baixa e por isso é também insignificante para os cálculos [8].

$$p = \mathbb{N}\left(\frac{d}{\sqrt{\sigma^2_{H_1} + \sigma^2_{H_2}}}\right). \tag{2-2}$$

Para encontrar H_1 e H_2 (Figura 2.3) deve basear-se somente no ótimo global e o ótimo local que antecede a solução ótima, representados pela linha tracejada amarela e laranja, respectivamente. Pode ocorrer em determinadas funções onde existam dois subótimos iguais, ilustrado na Figura 2.5. Nesse caso, a escolha de H_2 é indiferente, pois ambas as hipóteses irão produzir o mesmo valor de d, isto é, a distância entre o ótimo e qualquer um dos dois subótimos é a mesma. No entanto, a Figura 2.6 ilustra uma hipótese para H_2 mais próxima de H_1. Com a diminuição do valor de d o problema torna-se cada vez mais difícil, pois o algoritmo pode apresentar dificuldades em distinguir

boas soluções das quase-boa soluções. Nesse caso, o valor para o ponto representado por H_3 não irá fazer parte dos cálculos. Pode ainda haver a possibilidade de a função possuir dois ótimos locais (Figura 2.7). Pode então ser considerado um dos dois H_1s, pois ambos representam o mesmo *fitness*.

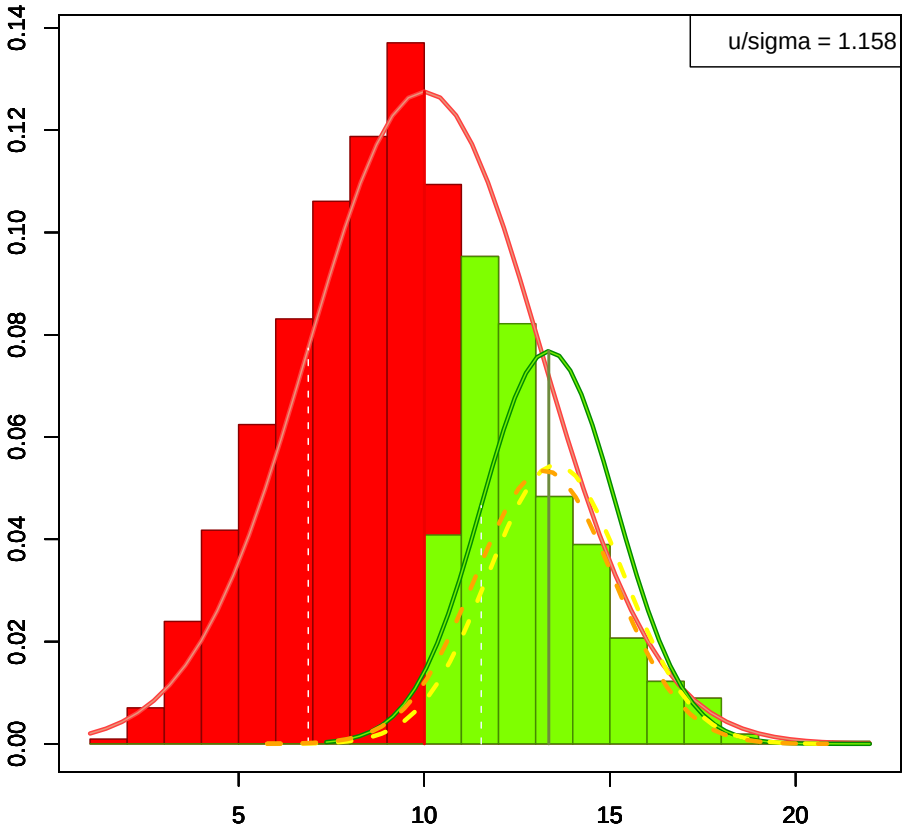

Figura 2.3: *Dado um conjunto de indivíduos aleatórios (barras em vermelho) é selecionado uma porção desses indivíduos pelo método de* ranking *e encontrado a distribuição de H1 (linha tracejada amarela) e H2 (linha tracejada laranja).*

Considerando que a *string* do indivíduo tem m subproblemas, deve-se considerar a influência dos $m-1$ blocos sobre o primeiro bloco. Assim, é necessário adicionar os $m-1$ blocos à formula, representado pela Equação 2-3, onde H^m representa a hipótese do bloco m.

$$p = \mathbb{N}\left(\frac{d}{\sqrt{\sigma^2_{H_1^1} + \sigma^2_{H_2^1} + \ldots + \sigma^2_{H_1^m} + \sigma^2_{H_2^m}}}\right). \tag{2-3}$$

É possível supor que o desvio padrão de cada bloco aproxima-se do desvio padrão médio σ^2_{BB} em todos os blocos. Sendo assim, a Equação 2-3 pode então ser

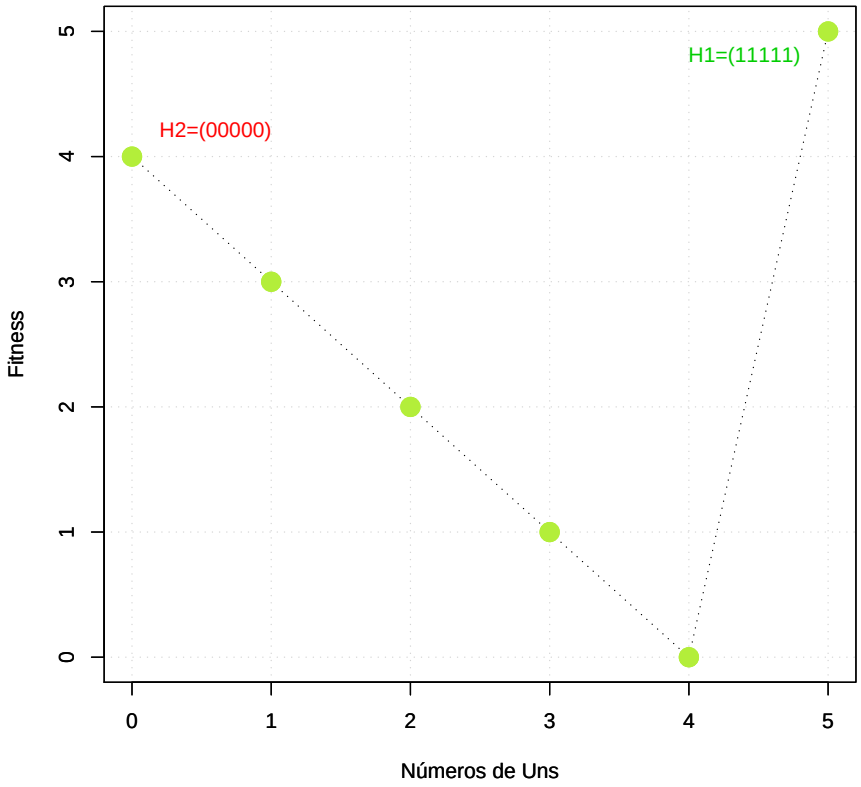

Figura 2.4: *Função* trap-5 *com H1 e H2.*

generalizada para a Equação 2-4, onde 2 é o número de hipóteses, m o número de blocos e σ_{BB}^2 é o desvio padrão médio em todos os blocos.

$$p = \mathbb{N}\left(\frac{d}{\sqrt{2m\sigma_{BB}^2}}\right). \tag{2-4}$$

Por fim, utilizando uma expansão em série de potências em $\mathbb{N}$ resulta na Equação 2-5. O primeiro termo da equação $(1/2)$ representa os 50% de chances de escolha padrão. O desvio padrão pode sair da raíz quadrada tornando um membro multiplicador da mesma. Nota-se também que a raíz quadrada da variância é igual ao desvio padrão, ou seja, o termo σ_{BB} pode também ser entendido como o desvio padrão médio dos blocos da população. Nessa equação, nota-se que valores altos para d resultam em maior probabilidade de escolha dos blocos certos no processo de seleção. Isso está relacionado diretamente com a variância média dos blocos, pois, quanto maior é a variância dos blocos mais baixo será o valor de p. Além disso, o aumento do número de blocos do problema

também resulta valores de p menores.

$$p = \frac{1}{2} + \frac{1}{2}\frac{d}{\sigma_{BB}\sqrt{2m}}. \tag{2-5}$$

A Figura 2.8 ilustra um exemplo no qual foi utilizado uma *string* de tamanho $k \times m$ para a função Trap-5, para dois subproblemas, ou seja, $m = 2$. Foi realizada uma análise para o estado da população onde 50% dos 70 indivíduos utilizados eram totalmente aleatórios e os outros 50% tinham chances uniformes de serem $(0000000000); (0000011111); (1111100000)$ ou (1111111111). Com isso, foi encontrado os valores:

$$\bar{f}_{H_1} = 6,66,$$

$$\bar{f}_{H_2} = 5,72,$$

$$d = \bar{f}_{H_1} - \bar{f}_{H_2} = 6,66 - 5,72 = 0,94$$

e

$$\sigma_{BB} = \frac{\sigma_{H_1} + \sigma_{H_2}}{2} = \frac{1,16 + 1,19}{2} = 1,17.$$

Substituindo na Equação 2-5, tem se:

$$p = \frac{1}{2} + \frac{1}{2}\frac{0,94}{1,17 \times \sqrt{2 \times 2}}$$

$$p = \frac{1}{2} + \frac{0,94}{2,35}$$

$$p = 0,50 + 0,40$$

$$p = 0,90$$

É interessante dizer que devido a Equação 2-5 ser uma aproximação da expansão da série de potência, onde somente são utilizados os dois primeiros termos da expansão, o valor de p é aproximado. Isso pode implicar em $p > 1$, pois os outros termos da expansão poderiam aparecer no formato de subtração e adição, mantendo $p < 1$.

Voltando à equação da probabilidade de sucesso de H1, escrita pela Equação 2-6, é possível substituir q por $1 - p$, isto é, a probabilidade de q ocorrer é a probabilidade de p não ocorrer. Considerando o valor de $a = \frac{n}{2^k}$, ou seja, menor que n, é possível verificar que o denominador aproxima-se de 1 antes que o numerador. Por isso, o termo $1 - (q/p)^n$ é aproximado para 1. Sendo assim, a equação pode ser reescrita como mostra a Equação 2-7.

$$P_n = \frac{1 - (q/p)^a}{1 - (q/p)^n}. \tag{2-6}$$

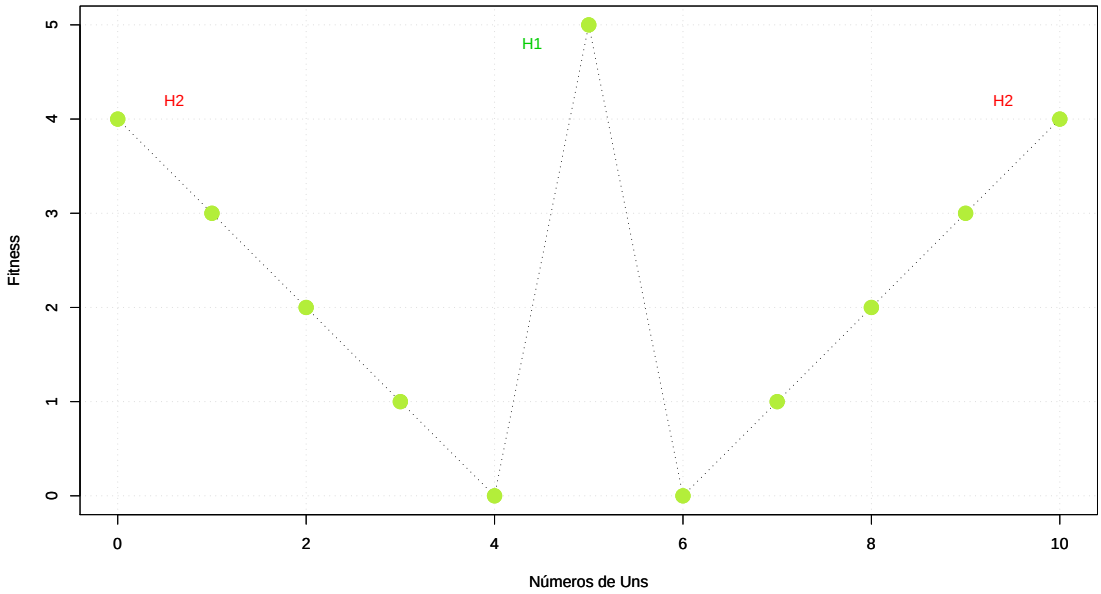

Figura 2.5: *Função* trap-5 *modificada para representar dois subótimos iguais.*

$$P_n \approx 1 - \left(\frac{1-p}{p}\right)^{\frac{n}{2^k}} \tag{2-7}$$

Considerando que $p = \frac{1}{2} + \frac{1}{2}x$, onde $x = \frac{d}{\sigma_{BB}\sqrt{2m}}$, a Equação 2-5 poderá ser reescrita pela Equação 2-8,

$$P_n = 1 - \left(\frac{1-\left(\frac{1}{2}+\frac{1}{2}x\right)}{\frac{1}{2}+\frac{1}{2}x}\right)^{\frac{n}{2^k}}, \tag{2-8}$$

resolvendo $1-p$, tem-se a Equação 2-9

$$P_n \approx 1 - \left(\frac{\frac{1}{2}-\frac{1}{2}x}{\frac{1}{2}+\frac{1}{2}x}\right)^{\frac{n}{2^k}}. \tag{2-9}$$

A divisão por 2 nas frações podem ser desconsideradas, pois não tem impacto nos resultados, e multiplicando os dois lados da equação por -1, tem-se a Equação 2-10. Está equação agora mostra a probabilidade de fracasso, devido a $1-P_n$.

$$1 - P_n \approx \left(\frac{1-x}{1+x}\right)^{\frac{n}{2^k}}. \tag{2-10}$$

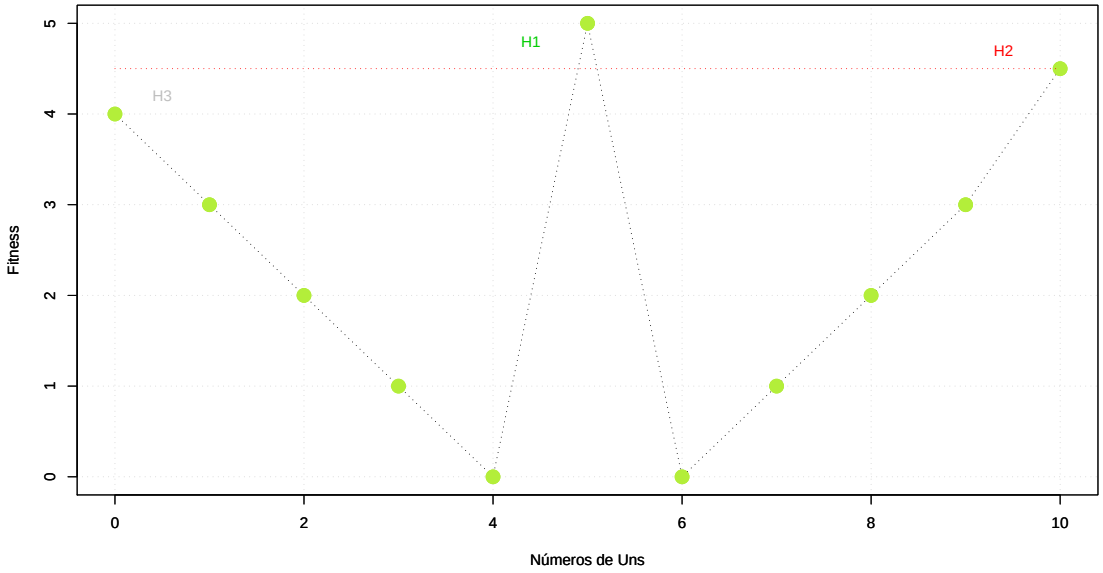

Figura 2.6: *Função* trap-5 *modificada para representar um subótimo mais próximo do ótimo.*

Sendo α a probabilidade de fracasso $1-P_n$, tem-se a Equação 2-11

$$\alpha \approx \left(\frac{1-x}{1+x}\right)^{\frac{n}{2^k}}. \tag{2-11}$$

Considerando que n é o valor a ser encontrado, isto é, o tamanho da população, deve-se aplicar ln em ambos os lados da Equação 2-11 para tirar o n do expoente. Sendo assim, essa a fórmula é reescrita pela Equação 2-12

$$\ln\alpha \approx \frac{n}{2^k}\ln\left(\frac{1-x}{1+x}\right). \tag{2-12}$$

Baseando na regra de divisão de ln, a equação é escrita no forma da Equação 2-14

$$\ln\alpha \approx \frac{n}{2^k}\left(\ln(1-x)-\ln(1+x)\right). \tag{2-13}$$

Usando que $\ln(1-x) \approx -x$ e $\ln(1+x) \approx x$, tem-se a Equação 2-14

$$\ln\alpha \approx \frac{n}{2^k}(-2x). \tag{2-14}$$

Assumindo também que $\frac{-2x}{2^k}$ é igual a $\frac{-x}{2^{k-1}}$, tem-se a Equação 2-15

$$\ln\alpha \approx \frac{-nx}{2^{k-1}} \Rightarrow \ln\alpha 2^{k-1} \approx -nx \Rightarrow -nx \approx \ln\alpha 2^{k-1} \tag{2-15}$$

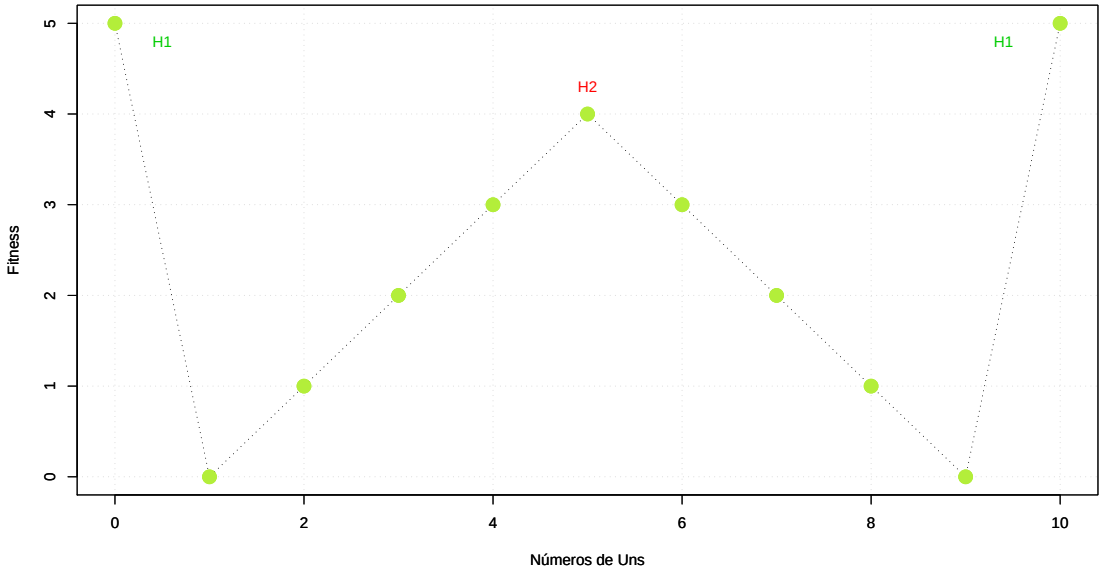

Figura 2.7: *Função* trap-5 *modificada para representar dois ótimos e um subótimo.*

Assim, a Equação 2-15 pode ser reescrita na forma da Equação 2-16

$$n = \frac{-\ln(\alpha)2^{k-1}}{x} \tag{2-16}$$

Por fim, considerando que $x = \frac{d}{\sigma_{BB}\sqrt{2m}}$, a Equação 2-17 mostra o tamanho da população para ser utilizado no AG. Um exemplo para essa equação será dado a seguir.

$$n = \frac{-\ln(\alpha)2^{k-1}\sigma_{BB}\sqrt{2m}}{d} \tag{2-17}$$

Utilizando os parâmetros:

- taxa de erro: $\alpha = 10^{-5}$;
- tamanho do bloco: $k = 5$;
- média das hipóteses de todos os blocos: $d/\sigma_{BB} = 1,158$;
- número de subproblemas: $m = 2$;

tem-se

$$n = \frac{-\log(10^{-5}) \cdot 2^{5-1}1 \cdot \sqrt{4}}{1}$$

$$n = \frac{6,64 \cdot 16 \cdot 1 \cdot 2}{1}$$

$$n = 6,64 \cdot 16 \cdot 2$$

$$n = 212$$

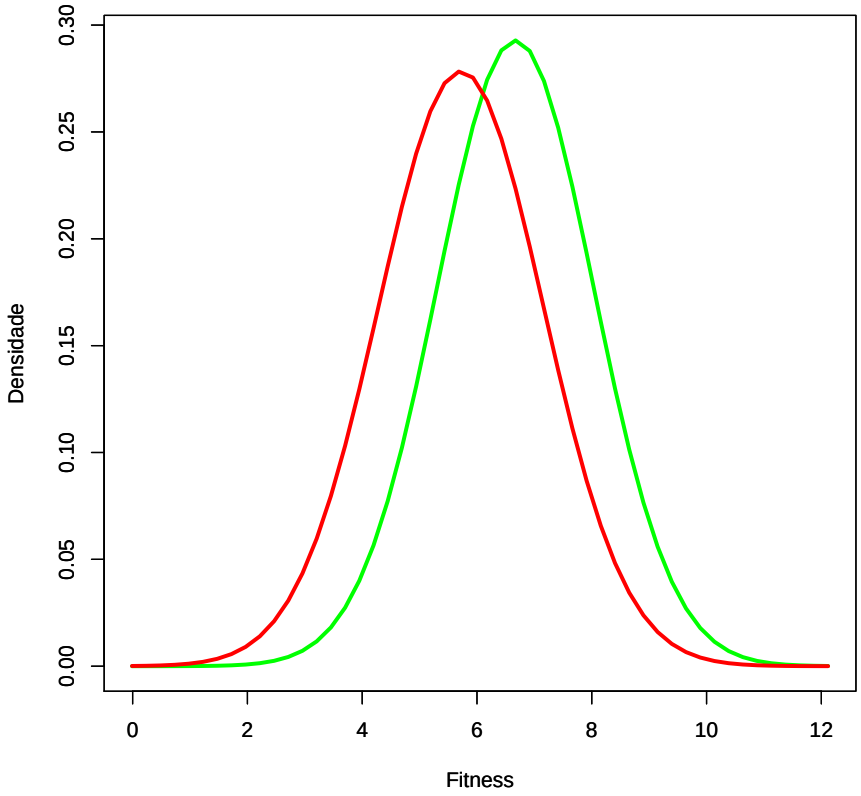

Figura 2.8: *Representação da distribuição do* fitness *médio para a função* trap-5.

A Figura 2.9[1] ilustra uma comparação entre os resultados experimentais, representados pelos pontos nos gráficos, com o modelo, ilustrado pelas linhas no gráfico. É possível notar que o tamanho da população cresce quadraticamente com base no tamanho dos blocos. A função do cálculo de n é sensível ao tamanho do bloco (k), pois na medida que k cresce, o tamanho da população cresce quadraticamente. Pode-se dizer que os outros parâmetros da equação causam pouco impacto no tamanho da população se levar em consideração o expoente k.

Pode-se ainda modelar o ruído exercido sobre o *fitness*, como interferência de sinais externo sobre o sinal de cada instância (ótima e subótima) de cada bloco para uma melhor precisão no valor de n. Isso pode ser feito pela soma das variâncias dos sinais devido a interferências externas, chamada de σ_x^2. Assim, o tamanho da população mínimo necessário para evitar a convergência a ótimos locais é incrementado pelo novo parâmetro.

[1] Código para geração deste gráfico está no Apêndice A

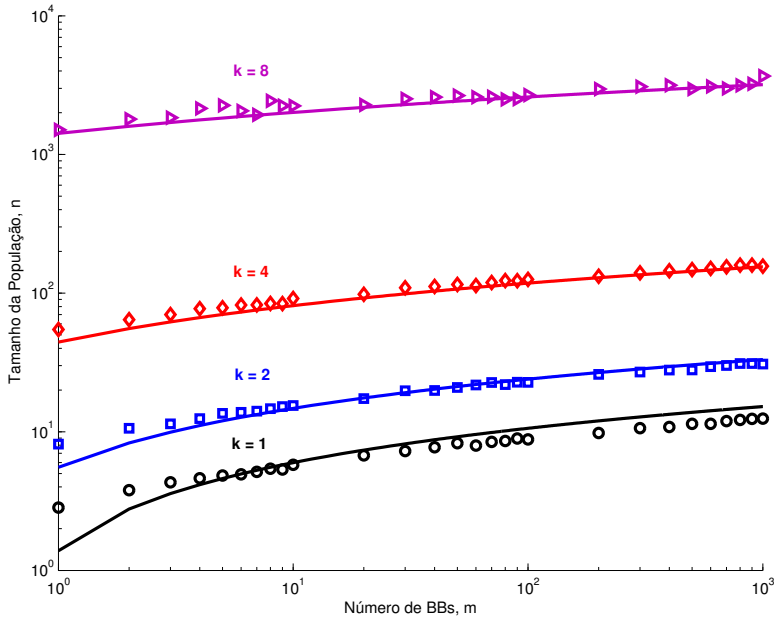

Figura 2.9: *Resultados experimentais com curva do modelo para vários valores de k.*

Dessa forma, a Equação 2-5 pode ser reescrita na forma da Equação 2-18.

$$p = \frac{1}{2} + \frac{1}{2}\frac{d}{\sqrt{2m(\sigma_{BB}^2 + \sigma_x^2)}}. \tag{2-18}$$

Outro fator que pode ainda ser adicionado ao modelo da ruína do jogador para o cálculo de n está relacionado com a pressão de seleção. Dependendo da pressão de seleção, o tamanho mínimo necessário para garantir que o AG não convirja para o ótimo local é alterado. Por exemplo, utilizando seleção por torneio, para torneios maiores que dois, os indivíduos passam a não competir mais com a média de *fitness* dos indivíduos. Eles passam a competir com os indivíduos que estão mais próximos da cauda dos melhores indivíduos. Assim, a Equação 2-19 define um novo valor de d em substituição ao valor anterior formulado pela Equação 2-2. Assim, conforme s diminui, o valor de d' aumenta.

$$d' = d + \mathbb{N}^{-1}\left(\frac{1}{s}\right)\sigma_{BB}. \tag{2-19}$$

O ruído do *fitness* (Equação 2-18) e o fator da pressão de seleção (Equação 2-19) alteram, de fato, o valor de n. Para uma modelagem convencional, onde não é adicionado o ruído do *fitness* e para torneio de dois, a Equação 2-17 é satisfatória. Adicionando o fator do ruído o valor de n sofre pouca alteração, pois está dentro da raíz. No entanto, o fator da pressão de seleção tem uma relação direta com o tamanho da população que

deverá ser utilizado, pois aparece no denominador da Equação 2-17.

Esta seção mostrou os passos necessários para construir o modelo do tamanho da população baseado no problema da ruína do jogador. Foi determinado vários parâmetros que influenciam no tamanho de n. No entanto, o valor de k da Equação 2-17 é o principal determinante do tamanho de n, pois este cresce exponencialmente de acordo com k. Isso quer dizer que valores de k superiores a 20 (blocos de tamanho 20) deverá requerer população com mais de 1 milhão de indivíduos.

Capítulo 3

Tempo de convergência

Já foi mostrado anteriormente que pode-se evitar o risco da população convergir para um ótimo local utilizando uma população suficientemente grande (Capítulo 2). Agora é preciso ter conhecimento sobre o número de gerações necessárias para que ocorra a convergência dos blocos.

Neste capítulo, será apresentado o teorema de Fisher porque consegue mostrar a relação entre uma geração e a seguinte. Logo após, esse teorema será utilizado para quantificar o tempo de convergência de dois problemas: OneMax e BinInt. A escolha desses problemas se deu pelo fato da relação entre os blocos ser diferente em cada um deles. No problema OneMax não existe relação entre blocos, enquanto que o BinInt há relação entre blocos. Assim, esses problemas podem servir de base para outros similares.

3.1 Teorema de Fisher

O teorema de Fisher afirma que existe uma relação entre o crescimento do *fitness* médio de uma população, de uma geração para outra, com a variância do *fitness* dos indivíduos. A seguir, é possível observar que essa relação pode ser deduzida de forma relativamente simples. Considera-se a seleção proporcional [5]:

$$P_{i,t+1} = \frac{f_i}{f_t} P_{i,t}, \tag{3-1}$$

em que f_i é o *fitness* do indivíduo i, $P_{i,t}$ é a proporção que o indivíduo i aparece na geração t da população e f_t é o *fitness* médio da população na geração t. Assim, f_{t+1} pode ser calculado como a seguir:

$$f_{t+1} = \sum_{i=0}^{n} P_{i,t+1} f_i. \tag{3-2}$$

Substituindo a Equação 3-1 em 3-2 obtêm-se:

$$f_{t+1} = \sum_{i=0}^{n} \frac{f_i}{f_t} P_{i,t} f_i, \tag{3-3}$$

ou

$$f_{t+1} = \sum_{i=0}^{n} \frac{f_i^2}{f_t} P_{i,t}. \tag{3-4}$$

Como o *fitness* médio da população atual é dado por:

$$f_t = \sum_{i=0}^{n} P_{i,t} f_i, \tag{3-5}$$

subtraindo-se 3-13 de 3-4 resulta em:

$$\begin{aligned} f_{t+1} - f_t &= \sum_{i=0}^{n} \frac{f_i^2}{f_t} P_{i,t} - \sum_{i=0}^{n} P_{i,t} f_i \\ f_{t+1} - f_t &= \frac{1}{f_t} \sum_{i=0}^{n} f_i^2 P_{i,t} - f_t \sum_{i=0}^{n} P_{i,t} f_i \\ f_{t+1} - f_t &= \frac{1}{f_t} \left[\sum_{i=0}^{n} f_i^2 P_{i,t} - f_t^2 \right]. \end{aligned} \tag{3-6}$$

Sabendo que a variância de uma população com distribuição uniforme (fatia da população igual para todos os indivíduos: $p_i = \frac{1}{n}$) é dada por:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (f_i - \overline{x})^2}{n} \tag{3-7}$$

e que pode ser reescrita como:

$$\sigma^2 = \sum_{i=1}^{n} p_i (x_i^2 - \overline{x})^2. \tag{3-8}$$

Como essa relação mantém-se para um p_i não uniforme, a equação 3-6 pode ser reescrita como:

$$f_{t+1} - f_t = \frac{\sigma_t^2}{f_t} \tag{3-9}$$

Para o caso de uma seleção por truncamento [5] em uma população com distribuição normal, o seu efeito pode ser observado na Figura 3.1.

Pode-se observar que s gera praticamente uma $\mathbb{N}$ truncada pela esquerda, em que $\overline{f}_{t+1}$ (μ') fica mais próxima do limite esquerdo. Além disso, ao longo das gerações,

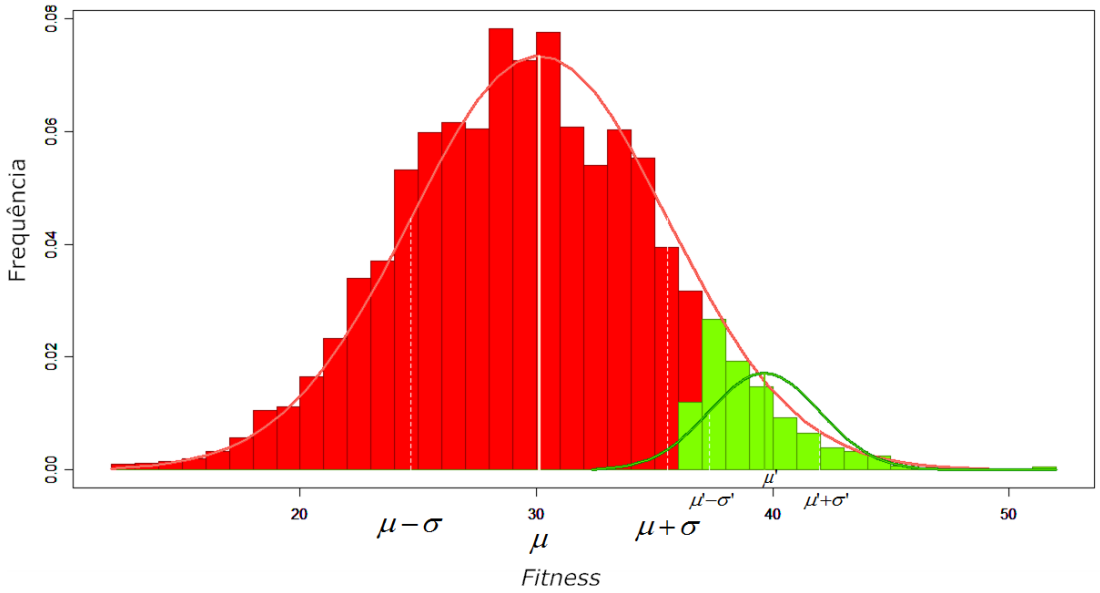

Figura 3.1: *Efeitos de s por truncamento sobre* $\mathbb{N}$

pode-se observar um valor constante [8] da seguinte relação:

$$I = \frac{\sigma_t}{\overline{f}_{t+1}}. \tag{3-10}$$

Dessa forma, substituindo 3-10 em 3-11 obtêm-se:

$$f_{t+1} - f_t = I\sigma_t, \tag{3-11}$$

em que a constante I é chamada intensidade de seleção.

3.2 Tempo de Convergência para o problema UmMax

O problema UmMax (*OneMax problem*) é um problema de otimização que consiste em atingir uma solução pré-especificada do domínio do problema (ótimo) a partir de um conjunto de soluções iniciais aleatórias. Para isso deve-se maximizar a função objetivo $fo(x) = \sum_{i=1}^{l} |x_i^k - x_i^o|$, em que x_i^k (x_i^o) é o valor do *bit* i da solução k (da solução ótima x^o) e l é o comprimento da *string* de *bits* x.

Deve-se notar que o problema UmMax pode ser resolvido em tempo linear por um algoritmo de busca local, pois basta pesquisar o valor que cada *bit* atribui a função objetivo de forma independente (BBs de tamanho 1), verificando qual dos valores atribuídos a cada *bit* aumenta a função objetivo.

Por simplificação, considera-se $x_o = (1, ..., 1)$, fazendo assim com que a função objetivo tenha valor proporcional a quantidade de *bits* com o valor 1, podendo ser reescrita

na forma da Equação 3-12.

$$f(x) = \sum_{i=1}^{l} x_i. \tag{3-12}$$

Levando-se em conta que a população tem uma distribuição de Bernoulli [4], a média de $f(x)$ (f_t) e a variância (σ^2) na geração t podem ser obtidos pela distribuição. Em outras palavras, a *string* x pode ser vista como um conjunto de l experimentos com resultado aleatório 0 ou 1 (chamado conjunto de experimentos de Bernoulli). Dados que p é a probabilidade de x_i ser 1, então $(1-p)$ é a probabilidade deste ser 0. A distribuição de um conjunto de *strings* com valores 0 ou 1 é uma distribuição de Bernoulli.

Verificando que f_t é a média de $f(x)$, então:

$$f_t = \frac{\sum_{j=1}^{n} f_j}{n}, \tag{3-13}$$

em que f_j é o *fitness* de cada indivíduo na população.

Substituindo a Equação 3-12 na Equação 3-13

$$f_t = \frac{\sum_{j=1}^{n} \sum_{i=1}^{l} x_{j,i}}{n}. \tag{3-14}$$

Sabendo que o valor esperado de x_i na geração t é dado por:

$$E(x_i) = 0(1-p_t) + 1p_t = p_t, \tag{3-15}$$

e sua variância por:

$$\sigma^2_{x_i} = E(x_i^2) - E(x_i)^2 = p_t - p_t^2 = p_t(1-p_t), \tag{3-16}$$

pois:

$$E(x_i^2) = 0^2(1-p_t) + 1^2 p_t = p_t. \tag{3-17}$$

A função objetivo (Equação 3-12) pode ser encarada como um conjunto de l experimentos de Bernoulli, isto é, que o resultado de cada experimento independe do resultado dos outros. Assim:

$$f_t = \frac{\sum_{j=1}^{n} lE(x)}{n} = \frac{\sum_{j=1}^{n} lp_t}{n} = lp_t \frac{\sum_{j=1}^{n} 1}{n} = lp_t \frac{n}{n} = lp_t \tag{3-18}$$

Sabendo-se que a variância pode ser calculada utilizando a esperança da seguinte

forma:

$$\begin{aligned}\sigma_t^2 = E(f_t^2) - E(f_t)^2 = E((lE(x))^2) - E(lE(x))^2 = \\ E(l^2E(x)^2) - E(lE(x))^2 = E(l^2p_t^2) - E(lp_t)^2 = \\ l^2E(p_t^2) - (lE(p_t))^2 = l^2p_t - l^2p_t^2 = l^2p_t(1-p_t)\end{aligned} \tag{3-19}$$

Em, resumo têm-se que $f_t = lp_t$ e $\sigma_t^2 = lp_t(1-p_t)$. Assim, como $f_t = lp_t$, então $f_{t+1} = lp_{t+1}$. Aplicando estas conclusões à equação de Fisher (3-11) para o truncamento, obtêm-se:

$$\begin{aligned}f_{t+1} - f_t &= I\sigma_t \\ lp_{t+1} - lp_t &= I\sigma_t \\ p_{t+1} - p_t &= \frac{I}{l}\sigma_t\end{aligned} \tag{3-20}$$

Substituindo $\sigma = \sqrt{lp_t(1-p_t)}$ em 3-20:

$$\begin{aligned}p_{t+1} - p_t &= \frac{I}{l}\sqrt{lp_t(1-p_t)} \\ p_{t+1} - p_t &= \frac{I}{\sqrt{l}}\sqrt{p_t(1-p_t)}\end{aligned} \tag{3-21}$$

Aproximando-se a equação de diferenças por uma equação diferencial:

$$\frac{\partial p_t}{\partial t} = \frac{I}{\sqrt{l}}\sqrt{p_t(1-p_t)} \tag{3-22}$$

Uma solução aproximada para a equação diferencial descrita acima é:

$$p_t \approx A\cos^2\left(\frac{I}{\sqrt{l}}t\right) \tag{3-23}$$

Para $t = 0$ e $p_0 = \frac{1}{2}$, observa-se que $A = \frac{1}{2}$ e:

$$p_t \approx \frac{1}{2}\cos^2\left(\frac{i}{\sqrt{l}}t\right) \tag{3-24}$$

Então:

$$\begin{aligned}\cos\left(\frac{i}{\sqrt{l}}t\right) &\approx \pm\sqrt{2p_t} \\ \frac{I}{\sqrt{l}}t &\approx \arccos\left(\pm\sqrt{2p_t}\right)\end{aligned} \tag{3-25}$$

Realizando a aproximação $\arccos(z) \approx \frac{\pi}{2} - z$, obtêm-se:

$$\frac{I}{\sqrt{l}}t \approx \frac{\pi}{2} \mp \sqrt{2p_t}$$
$$t \approx \frac{\sqrt{l}}{I}\frac{\pi}{2} \mp \sqrt{2p_t} \tag{3-26}$$

Utilizando o critério de convergência para Algoritmos Genéticos selecto-recombinativos em que $p_t = 1 - \frac{1}{n}$, isto é, para que ocorra a convergência de pelo menos uma faixa (gene) de todos indivíduos de uma população:

$$t \approx \frac{\sqrt{l}}{I}\frac{\pi}{2} \mp \sqrt{2(1-\frac{1}{n})} \tag{3-27}$$

Com a escolha de uma população inicial n suficientemente grande, para análise de convergência, temos que:

$$t_g \approx \frac{\sqrt{l}}{I}\frac{\pi}{2} \tag{3-28}$$

3.3 Tempo de Convergência para o problema BinInt

O problema BinInt é um problema de otimização cujo objetivo é maximizar a função

$$f(x) = \sum_{j=1}^{l} 2^{l-j}x_j, \tag{3-29}$$

em que x_j é o valor do j-ésimo *bit* e l é o comprimento da *string* de *bits* $x = (x_1, x_2, \ldots, x_l)$.

A Figura 3.2 demonstra o efeito de cada *bit* de duas soluções na função *fitness*. A primeira é a solução ótima, dada por `111111`, resultando num valor de função *fitness* de 63 unidades. A segunda solução é dada pela *string* `010011` e tem por valor de função *fitness* 19 unidades.

Vale ressaltar que os primeiros *bits* possuem uma importância maior pra função *fitness*. Por exemplo, a solução `100000` é uma unidade maior que a solução `011111`. É suficiente para esse caso estudar blocos de 1 *bit*, já que na competição entre blocos o fator crítico é a escala de cada coeficiente e não o tamanho do bloco. Porém, devido a saliência dos blocos este problema é mais difícil que o problema UmMax, pois o algoritmo genético tende a convergir os primeiros bits para 1, podendo acarretar na perda de bits 1 em alguma posição menos saliente.

O algoritmo genético selecto-recombinativo foi utilizado com a população inicial aleatória e definida por `RRRRRR`. Pode-se definir $\lambda = 0$ para esse conjunto de soluções

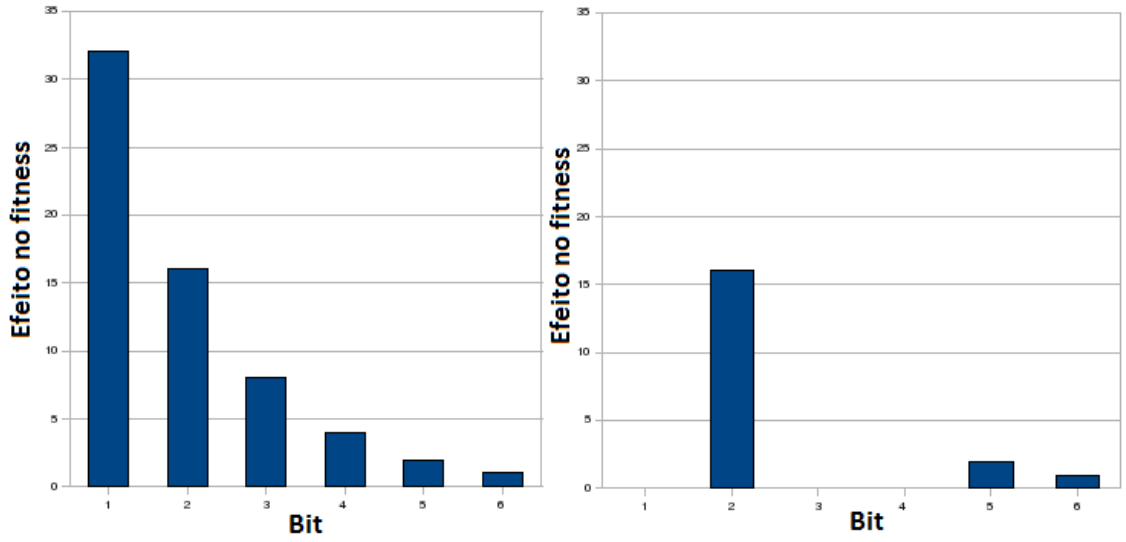

Figura 3.2: *Efeito no fitness das soluções* 111111 *e* 010011.

inicial. Seja `1RRRRR` o conjunto de soluções com 1 no primeiro *bit* e $\lambda = 1$. Continuando, para a solução `111111`, $\lambda = l = 6$. Assim, dado λ, têm-se que as soluções deste conjunto possuem o valor da função *fitness* dado por:

$$f(x) = \sum_{j=1}^{\lambda} 2^{l-j} + \sum_{j=\lambda+1}^{l} 2^{l-j} x_j \tag{3-30}$$

$$= \mathbb{C}_\lambda + \sum_{j=\lambda+1}^{l} 2^{l-j} x_j. \tag{3-31}$$

Prosseguindo, dado λ, e sabendo que para todos os *bits* de $x_1, \ldots, x_\lambda$ convergiram para 1, e dado que os que não convergiram tem valor esperado igual a $\frac{1}{2}$, temos que o valor esperado da função *fitness* é:

$$\overline{f(\lambda)} = 1 \sum_{j=1}^{\lambda} 2^{l-j} + \frac{1}{2} \sum_{j=\lambda+1}^{l} 2^{l-j} \tag{3-32}$$

$$= 1 \sum_{j=l-1}^{l-\lambda} 2^{j} + \frac{1}{2} \sum_{j=l-\lambda-1}^{0} 2^{j} \tag{3-33}$$

$$= 1 \sum_{j=l-\lambda}^{l-j} 2^{j} + \frac{1}{2} \sum_{j=0}^{l-\lambda-1} 2^{j} \tag{3-34}$$

$$= 1 (\sum_{j=0}^{l-1} 2^{j} - \sum_{j=0}^{l-\lambda-1} 2^{j}) + \frac{1}{2} \sum_{j=0}^{l-\lambda-1} 2^{j} \tag{3-35}$$

$$= 1 (2^{l-1+1} - 2^{l-\lambda-1+1}) + 2^{-1} 2^{l-\lambda-1+1} \tag{3-36}$$

$$= 2^{l-1} (2 - 2^{-\lambda}) \tag{3-37}$$

Dado que a variância é dada pela equação $\sigma^2(\lambda) = \overline{f^2(\lambda)} - \overline{f(\lambda)}^2$, e sabendo que a variância dos *bits* de $j = 0, \ldots, \lambda$ é zero, deve-se calcular apenas a variância total dos *bits* de $j = \lambda+1, \ldots, l$.

Dado que $f(\lambda) = \mathbb{C}_\lambda + \sum_{j=\lambda+1}^{l} 2^{l-j} x_j$, temos que a diferença entre soluções é dada por $f'(\lambda) = f(\lambda) - \mathbb{C}_\lambda = \sum_{j=\lambda+1}^{l} 2^{l-j} x_j$. Assim podemos utilizar $\sigma^2(\lambda) = \overline{f'^2(\lambda)} - \overline{f'(\lambda)}^2$ para calcular a variância $\sigma(\lambda)$.

Sabemos que $\overline{f'(\lambda)}$ é o valor médio da soma de todos os números inteiros representáveis por $l - \lambda$ *bits*. Assim, a média é dada pela equação 3-40.

$$\overline{f'(\lambda)} = \frac{\sum_{j=0}^{2^{l-\lambda}-1} j}{2^{l-\lambda}} \tag{3-38}$$

$$= \frac{(2^{l-\lambda}-1)(2^{l-\lambda}-1+1)}{2(2^{l-\lambda})} \tag{3-39}$$

$$= \frac{2^{l-\lambda}-1}{2} \tag{3-40}$$

De maneira análoga, podemos calcular $\overline{f'^2(\lambda)}$:

$$\overline{f'^2(\lambda)} = \frac{\sum_{j=0}^{2^{l-\lambda}-1} j^2}{2^{l-\lambda}} \tag{3-41}$$

$$= \frac{(2^{l-\lambda}-1)(2^{l-\lambda}-1+1)[2(2^{l-\lambda}-1)+1]}{6(2^{l-\lambda})} \tag{3-42}$$

$$= \frac{(2^{l-\lambda}-1)(2^{l-\lambda+1}-1)}{6} \tag{3-43}$$

Desta forma, podemos calcular a variância $\sigma(\lambda)$:

[1] Dado que $\sum_{j=0}^{n} j = \frac{n(n+1)}{2}$.

[2] Dado que $\sum_{j=0}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$.

$$\sigma^2(\lambda) = \overline{f'^2(\lambda)} - \overline{f'(\lambda)}^2 \tag{3-44}$$

$$\sigma^2(\lambda) = \frac{(2^{l-\lambda}-1)(2^{l-\lambda+1}-1)}{6} - \left(\frac{2^{l-\lambda}-1}{2}\right)^2 \tag{3-45}$$

$$\sigma^2(\lambda) = \frac{2^{2(l-\lambda)}-1}{12} \tag{3-46}$$

$$\sigma(\lambda) = \sqrt{\frac{2^{2(l-\lambda)}-1}{12}} \tag{3-47}$$

$$\sigma(\lambda) \approx \frac{2^{l-\lambda-1}}{\sqrt{3}}. \tag{3-48}$$

Podemos utilizar agora o Teorema de Fisher para o truncamento:

$$\overline{f_{t+1}} - \overline{f_t} = I\sigma_t \tag{3-49}$$

$$2^{l-1}(2-2^{-\lambda_{t+1}}) - 2^{l-1}(2-2^{-\lambda_t}) = I\frac{2^{l-\lambda_t-1}}{\sqrt{3}} \tag{3-50}$$

$$2^{l-1}(2^{-\lambda_t} - 2^{-\lambda_{t+1}}) = 2^{l-1}2^{-\lambda_t}\frac{I}{\sqrt{3}} \tag{3-51}$$

$$2^{-\lambda_t} - 2^{-\lambda_{t+1}} = 2^{-\lambda_t}\frac{I}{\sqrt{3}} \tag{3-52}$$

$$2^{-\lambda_{t+1}} = 2^{-\lambda_t}\left(1 - \frac{I}{\sqrt{3}}\right) \tag{3-53}$$

$$2^{-\lambda_{t+1}} = 2^{-\lambda_{t-1}}\left(1 - \frac{I}{\sqrt{3}}\right)^2 \tag{3-54}$$

$$2^{-\lambda_t} = 2^{-\lambda_0}\left(1 - \frac{I}{\sqrt{3}}\right)^t. \tag{3-55}$$

Para $t = 0$, $\lambda = 0$, assim, $2^{-\lambda_0} = 1$, e podemos substituir na Equação (3-55). Podemos então calcular o tempo de convergência t_g:

$$2^{-\lambda_t} = \left(1 - \frac{I}{\sqrt{3}}\right)^t \tag{3-56}$$

$$\ln(2^{-\lambda_t}) = \ln\left(1 - \frac{I}{\sqrt{3}}\right)^t \tag{3-57}$$

$$-\lambda_t \ln(2) = t \ln\left(1 - \frac{I}{\sqrt{3}}\right) \tag{3-58}$$

$$t = \frac{-\lambda_t \cdot \ln(2)}{\ln\left(1 - \frac{I}{\sqrt{3}}\right)} \tag{3-59}$$

$$t_g \overset{3}{=} \frac{-l \cdot \ln(2)}{\ln\left(1 - \frac{I}{\sqrt{3}}\right)} \tag{3-60}$$

$$t_g \overset{4}{=} \frac{-l \cdot \ln(2)}{-\frac{I}{\sqrt{3}}} \tag{3-61}$$

$$t_g = l\left(\frac{\sqrt{3} \cdot \ln(2)}{I}\right). \tag{3-62}$$

Assim, temos que o problema BinInt converge em $t_g = O(l)$ gerações. Dado que o problema UmMax converge em $t_g = O(\sqrt{l}) = O(l^{\frac{1}{2}})$ gerações, o tempo de convergência é sublinear ou linear com relação ao tamanho da *string* de *bits* da solução.

[3] Supondo que a população convirja para a solução ótima, assim $\lambda_{t_g} = l$.

[4] Dado que $\ln 1 - x \approx -x$, para valores de x pequenos.

Capítulo 4

Complexidade computacional

A partir dos resultados obtidos nas seções anteriores, onde foi analisado a ordem do tamanho da população para evitar que ela convirja para uma instância ótima local:

$$n \approx -\ln \alpha 2^{k-1} \frac{\sigma_{BB}\sqrt{2m}}{d}, \tag{4-1}$$

o tempo de convergência para o problema UmMax:

$$t_g = \sqrt{l}\left(\frac{\pi}{2I}\right), \tag{4-2}$$

e o tempo de convergência para o problema BinInt:

$$t_g = l\left(\frac{\sqrt{3}\ln 2}{I}\right), \tag{4-3}$$

podemos obter a complexidade de tempo do algoritmo genético selecto-recombinativo (*AGsr*) utilizado, por meio da relação:

$$n \cdot t_g. \tag{4-4}$$

Primeiramente precisamos calcular a complexidade do tamanho da população n. Para isso, temos que o número de blocos é dado por $m = \frac{l}{k}$, ou seja o tamanho da *string* de *bits* dividido pelo tamanho do bloco. Para problemas com o tamanho do bloco k e a variância média dos blocos limitados por uma constante $c_n > 0$, obtêm-se:

$$n \leq -c_n \cdot \ln \alpha \cdot \sqrt{l}. \tag{4-5}$$

O valor de α se refere à precisão estipulada e pode ser constante, tornando a complexidade do tamanho da população igual a:

$$n = O\sqrt{l}, \tag{4-6}$$

e pode ser dada pelo critério de convergência do *AGsr*, $\alpha = \frac{c}{l}$, deixando a complexidade do tamanho da população igual a:

$$n = O(\sqrt{l} \cdot \ln l). \tag{4-7}$$

De qualquer forma, ambas complexidades são sublineares.

Em segundo lugar, devemos calcular a complexidade do tempo de convergência. De acordo com os dois problemas analisados (problema UmMax e BinInt, respectivamente), temos que os seus tempos de convergência são dados por:

$$t_g = \sqrt{l}\left(\frac{\pi}{2I}\right), \tag{4-8}$$

$$t_g = l\left(\frac{\sqrt{3}\ln 2}{I}\right). \tag{4-9}$$

Assumindo que s independe do tamanho l do problema, I é uma constante, o que torna a complexidade dos tempos de convergência de ambos problemas:

$$t_g = O(\sqrt{l}), \tag{4-10}$$

$$t_g = O(l), \tag{4-11}$$

respectivamente.

Combinando os resultados das ordens do tamanho da população (4-8 e 4-9) e tempos de convergência (4-10 e 4-11), temos que a complexidade do *AGsr* é dado pela Tabela 4.1. De qualquer forma a complexidade computacional do *AGsr* é subquadrática.

	UmMax	BinInt
α constante	$O(\sqrt{l} \cdot \sqrt{l}) = O(l)$	$O(\sqrt{l} \cdot l) = O(l^{1,5})$
$\alpha = \frac{c}{l}$	$O(\sqrt{l} \cdot \ln l \cdot \sqrt{l}) = O(l \cdot \ln l)$	$O(\sqrt{l} \cdot \ln l \cdot l) = O(l^{1,5} \cdot \ln l)$

Tabela 4.1: *Complexidade de tempo do* AGsr

Capítulo 5

Considerações Finais

Ao longo do texto, diversos pontos importantes relacionados à teoria dos AGs foram descritos. No Capítulo 2 a competição entre instâncias de blocos foi investigada e relacionada ao Problema da Ruína do Jogador, através dessa analogia foi possível descrever: o processo que leva um AG selecto-recombinativo (AGsr) a convergir a algum ótimo global, e as condições necessárias para que isso seja possível, através da estimação do tamanho da população.

Já no Capítulo 3, o tempo de convergência do AGsr foi estimado através da análise de dois problemas. A ideia principal dessa abordagem é que conhecidos os tempos de convergência para um problema tão simples como UmMax, $\sqrt{l}\left(\frac{\pi}{2I}\right)$, e para um problema tão complexo quanto o BinInt, $l\left(\frac{\sqrt{3}\ln 2}{I}\right)$, seja possível concluir (Capítulo 4) o tempo de convergência médio de um AGsr como sendo subquadrático.

Através dessa conclusão fica clara a robustez dos AGs na resolução de problemas difíceis de forma precisa e confiável, visto que, a determinação dos parâmetros necessários para o seu bom funcionamento pode ser efetuada utilizando como base os modelos teóricos anteriormente descritos.

Outro ponto importante a ser considerado, e que fica evidente a partir da análise mais criteriosa dos AGs, é a grande dificuldade encontrada em efetuar a mistura entre *Building Blocks* (um dos pontos cruciais dos modelos teóricos descritos) [8]. Essa dificuldade é um fator que corrobora para uma nova abordagem na solução de problemas, que relaciona AGs e decomposição em subproblems, e torna secundário o papel do *crossover*. O Algoritmo Filogenético é um bom exemplo de algoritmo desse tipo [1].

Apêndice A

Código MatLab

```
function [experimentos mediaExperimentos] = verificaBbK1248(tamBB, nMaxVezes)
pontos=[1:9 10:10:99 100:100:1000];

colFlags=2^tamBB;
%num de repetiçoes do experimento para retirarmos a media
for nvezes=1:nMaxVezes
    %variando a quantidade de BB em cada repeticao do experimento
    for qtBB=1:length(pontos)
        %qt de flags==qt de BB
        linFlags=(pontos(qtBB));
        tamPop=0;

        % cria matriz de flags onde a linha indica a ocorrencia
        % daquela instancia do bb relativo aquela linha
        flag=zeros(linFlags,colFlags);

        %repete ate todas as intancias de todos os bbs existirem
        while (sum(sum(flag))<(linFlags*colFlags))
            %cria um novo individuo para verificar se ja existem
            %todas as intancias do BB
            tamPop=tamPop+1;
            %cria todos os bbs do novo individuo
            for i=1:pontos(qtBB)
                %coloca aleatoriamente valores para os alelos do bb
                for k=1:tamBB
                    rnd(k)=randi([0 1]);
                end
```

```
                IntanciaBB(i,:)=rnd;
            end
            %verifica se todas as intancias de todos os bbs existem
            for i=1:pontos(qtBB)
                %converte a instancia em decimal
                nFlagI=ConvBin2Dec(IntanciaBB(i,:));
                %para o bb i coloca 1 na posicao relativa ao valor
                %decimal da instancia
                flag(i,nFlagI+1)=1;
            end
        end
        %armazena o tamanho da pop q foi criada para garantir todas as
        %instancias do bb, para cada qt de bb
        n(pontos(qtBB))=tamPop
    end
    % armazena o valor de n para cada um dos experimentos;
    experimentos(nvezes,:)=n
    n=0;
end

mediaExperimentos=sum(experimentos)/nvezes;

end
```

Referências Bibliográficas

[1] DELBEM, A. C. B.; DE MELO, V. V.; VARGAS, D. V. **Algoritmo filo-genético**. In: *2ª Escola Luso-Brasileira de Computação Evolutiva (ELBCE)*. APDIO, 2010.

[2] DROSTE, S.; JANSEN, T.; WEGENER, I. **Perhaps not a free lunch but at least a free appetizer**. *HT014601767*, 1998.

[3] EDWARDS, A. **Pascal's problem: The'Gambler's Ruin'**. *International Statistical Review/Revue Internationale de Statistique*, 51(1):73–79, 1983.

[4] EVANS, M.; HASTINGS, N.; PEACOCK, B. **Statistical distributions**. *Measurement Science and Technology*, 12:117, 2001.

[5] GOLDBERG, D. E.; DEB, K. **A comparative analysis of selection schemes used in genetic algorithms**. In: *Foundations of Genetic Algorithms*, p. 69–93, 1991.

[6] GOLDBERG, D. **Genetic algorithms in search, optimization, and machine learning**. Addison-wesley, 1989.

[7] GOLDBERG, D. **The design of innovation: Lessons from and for competent genetic algorithms**. Springer, 2002.

[8] GOLDBERG, D. **The design of innovation: Lessons from and for competent genetic algorithms**. Springer, 2002.

[9] HOLLAND, J. **Adaptation in natural and artificial systems**. MIT Press Cambridge, MA, USA, 1992.

[10] IGEL, C.; TOUSSAINT, M. **On classes of functions for which no free lunch results hold**. *Arxiv preprint cs/0108011*, 2001.

[11] KIRKPATRICK, S. **Optimization by simulated annealing: Quantitative studies**. *Journal of Statistical Physics*, 34(5):975–986, 1984.

[12] POLI, R.; GRAFF, M. **There is a free lunch for hyper-heuristics, genetic programming and computer scientists**. *Genetic Programming*, p. 195–207, 2009.

[13] SCHUMACHER, C.; VOSE, M.; WHITLEY, L. **The no free lunch and problem description length**. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, p. 565–570. Citeseer, 2001.

[14] STORN, R.; PRICE, K. **Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces**. *Journal of global optimization*, 11(4):341–359, 1997.

[15] WHITLEY, D. **A free lunch proof for gray versus binary encodings**. In: *In*, p. 726–733. Morgan Kaufmann, 1999.

[16] WOLPERT, D. H.; G., W. **No free lunch theorems for search**. Technical report, The Santa Fe Institute.

[17] WOLPERT, D.; MACREADY, W. **No Free Lunch Theorems for Optimization**. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[18] WOLPERT, D.; MACREADY, W. **Coevolutionary free lunches**. *Evolutionary Computation, IEEE Transactions on*, 9(6):721–735, 2005.