

ALGORITMO EVOLUCIONÁRIO NÃO PARAMETRIZADO APLICADO AO
PROBLEMA DA OTIMIZAÇÃO DE RECARGAS DE REATORES
NUCLEARES

Gustavo Henrique Flores Caldas

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS
PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE
FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS
EM ENGENHARIA NUCLEAR.

Aprovada por:

Prof. Roberto Schirru, D.Sc.

Prof. Eduardo Gomes Dutra do Carmo, D.Sc.

Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

Dr. Celso Marcelo Franklin Lapa, D.Sc.

Dr. Sergio de Queiroz Bogado Leite, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 2006

CALDAS, GUSTAVO HENRIQUE FLORES

Algoritmo Evolucionário Não Parame-
trizado Aplicado ao Problema da Otimiza-
ção de Recargas de Reatores Nucleares
[Rio de Janeiro] 2006

XV, 103 p. 29,7 cm (COPPE/UFRJ,
D.Sc., Engenharia Nuclear, 2006)

Tese - Universidade Federal do Rio de
Janeiro, COPPE

1. FPBIL
 2. Algoritmos Evolucionários
 3. Otimização
 4. Recarga de Reatores Nucleares
- I. COPPE/UFRJ II. Título (série)

*Dona Dulce,
minha avozinha, que poderia muito bem ser chamada Dona Doce.
Incansável intercessora, a quem dedico este trabalho.*

Agradecimentos

*A Deus,
que é fiel, apesar da minha infidelidade.*

*A Anna Stella Flores Caldas,
minha mãezinha querida, que já partiu; mas cujo exemplo ainda ecoa forte.*

*A Lúcia,
minha segunda mãe, que nunca vai admitir, mas adora me mimar.*

*A meu querido pai,
meu herói, quem botou o primeiro livro de cálculo na minha mão.*

*A Renata Alves de Souza,
a moça mais fascinante que existe, pelo seu amor e compreensão.*

*A Gisele,
minha irmã preferida!!! Exímia conhecedora da Língua Portuguesa,
com quem adoro passar o tempo.*

*A Elso,
meu cunhadão, que nunca me negou um único favor sequer.*

*A Ricardinho e Anninha,
as crianças mais graciosas que existem, porque me lembram de sempre batalhar por
um mundo melhor.*

*A D. Rose e Sr. Antônio,
meus segundos pais, pela forma carinhosa com que sempre me tratam.*

*A Roberto Schirru,
orientador e amigo, pelo direcionamento, confiança e apoio.*

*Aos colegas do PEN,
pelo companheirismo e espírito de grupo.*

*A Simone, Jô e Tania,
que nunca mediram esforços para me ajudar nas mais variadas situações.*

*Aos colegas da CNEN,
que sempre demonstraram um apoio muito grande, motivando-me a prosseguir,
especialmente nos momentos mais críticos.*

*Aos meus amigos do orkut e MSN,
q contribuíram MUITO para q eu permanecesse vivo durante meu período de
enclausuramento. Vcs são D+ !!!!! =]*

*Aos amigos
que, de uma forma ou de outra, contribuíram para a existência desta tese:*

*Júnior, Rosimar, Tuca, Wander, Renatinha, Rodney, Izabelli, Beatrice, Tarsila,
Ticiano, Tarso, Thais, Áureo, Orli, Bianca, Henrique, Iana, Elina, Nathalia,
Negra, Marcele, Carlinhos, Carol, Elton, Juliane, Ewerton, Rachael, Bebel, André,
Maurinho, Gidel, Carine, VHS, Foca, Priscila, Erika, Luciana, Bruna, Pancho,
Clhystynnine, Daniel, Júlia, Elaine, Samira, Patrícia, Bloblô, Amaral, Fire, Lire,
Amanda, Vicente, Fernandinha, Flávia, Geison, Kelly, Izabela, Taís, Miojo,
Laurentina, César, Luhem, Luiz Henrique, Luiza Helena, Luiz Felipe, Michelle,
Jéssiii, Neurotiko, Oz, Paulinha, Ruivo, Renatinho, Rian, Eric, Lucas, Toy Story,
Márcio, Bruna, Gabi, Danizinha, Victor, Thiago, Vinícius, Paolo, Wanderson,*

*João, William, Djanira, Marcelo, Alessandro, Sergio, Cláudia, Pedro, Mára,
Lééelio, Cláudia, João Márcio, Paulo, Pedro Saldanha, Eustério, Rossana, Evaldo,
Joana, Rogério, Sandra, Cristiane, Marcelo, Max, Josilto, Ricardo, Maria Helena,
Cláudio Márcio, Alan, Marcelo, Carlão, o pessoal da EBA, do Coral Jovem, do
Classe A, do MC...
... e muitos outros!!!*

Valeu!!!!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMO EVOLUCIONÁRIO NÃO PARAMETRIZADO APLICADO AO
PROBLEMA DA OTIMIZAÇÃO DE RECARGAS DE REATORES
NUCLEARES

Gustavo Henrique Flores Caldas

Julho/2006

Orientador: Roberto Schirru

Programa: Engenharia Nuclear

Neste trabalho desenvolve-se um algoritmo evolucionário sem parâmetros, baseado no PBIL, denominado FPBIL. Ademais, mostra-se, de forma rigorosa, como os parâmetros do PBIL podem ser substituídos por mecanismos auto-adaptáveis que surgem da forma radicalmente diferente pela qual a evolução é processada. Apesar dessas vantagens, o FPBIL mostra-se compacto e relativamente modesto no uso de recursos computacionais. O FPBIL é testado e, por fim, aplicado ao problema da otimização de recargas de reatores nucleares. Os resultados experimentais observados são comparados aos de outros trabalhos e corroboram para afirmar a superioridade do novo algoritmo.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PARAMETERLESS EVOLUTIONARY ALGORITHM APPLIED TO THE
NUCLEAR REACTOR RELOAD OPTIMIZATION PROBLEM

Gustavo Henrique Flores Caldas

July/2006

Advisor: Roberto Schirru

Department: Nuclear Engineering

In this work an evolutionary algorithm with no parameters called FPBIL is developed based on PBIL. Moreover, the analysis reveals rigorously how the parameters from PBIL can be replaced by self-adaptable mechanisms which appear from the radically different form by which the evolution is processed. Despite the advantages, the FPBIL reveals itself compact and relatively modest in the use of computational resources. The FPBIL is then tested and, at the end, applied to the nuclear reactor reload optimization problem. The experimental results observed are compared to those of other works and corroborate to affirm the superiority of the new algorithm.

Sumário

Lista de Figuras	p. xii
Lista de Tabelas	p. xv
1 Introdução	p. 1
2 O Algoritmo PBIL	p. 5
2.1 Estrutura do PBIL	p. 5
2.2 O Algoritmo PBIL Original	p. 9
3 FPBIL: <i>parameter Free</i> PBIL	p. 14
3.1 O Algoritmo FPBIL	p. 14
3.2 Eliminando os Parâmetros α e β	p. 16
3.3 Eliminando os Parâmetros P_{μ} e γ	p. 22
3.4 Eliminando o Parâmetro \mathcal{P}	p. 27
4 Comparação Entre os Algoritmos	p. 36
4.1 Considerações Iniciais	p. 37
4.1.1 Código de Gray	p. 37
4.1.2 Minimizando o Impacto sobre a Escolha da Aptidão	p. 41

4.1.3	Procedimento Geral	p. 44
4.2	Problemas Testes	p. 45
4.2.1	Quatro Picos	p. 45
4.2.1.1	Representação	p. 47
4.2.1.2	Aptidões Bruta e Padrão	p. 47
4.2.1.3	Resultados	p. 47
4.2.2	Banana	p. 52
4.2.2.1	Representação	p. 53
4.2.2.2	Aptidões Bruta e Padrão	p. 54
4.2.2.3	Resultados	p. 55
4.2.3	PCV Rykel48	p. 58
4.2.3.1	Representação <i>Random Keys</i>	p. 59
4.2.3.2	Aptidões Bruta e Padrão	p. 61
4.2.3.3	Resultados	p. 62
5	FPBIL Aplicado ao Problema da Recarga Nuclear	p. 72
5.1	Descrição do Problema	p. 72
5.2	O Núcleo de Angra 1 e suas Simetrias	p. 73
5.3	Metodologia	p. 75
5.3.1	RECNOD	p. 76
5.3.2	Representação das Configurações Nucleares	p. 78
5.4	Resultados	p. 82
5.5	Comparação com Outros Trabalhos	p. 91

6 Conclusão	p. 94
Referências	p. 98
Índice Remissivo	p. 101

Lista de Figuras

2.1	Algoritmo PBIL original.	p. 10
3.1	Algoritmo FPBIL.	p. 15
3.2	Mutação 1.	p. 25
3.3	Mutação 2.	p. 25
3.4	Mutação 3.	p. 26
3.5	Mutação 4.	p. 26
3.6	Comportamento de \mathcal{P}_c	p. 29
3.7	Distribuição de N	p. 29
3.8	\mathcal{P} como função de c e \mathcal{P}_0	p. 32
4.1	Comportamento da aptidão ajustada.	p. 43
4.2	Gráfico de $Q_T(\mathcal{I})$, função objetivo do problema quatro picos.	p. 46
4.3	Comparação entre o FPBIL e o PBIL.	p. 48
4.4	Comparação entre o FPBIL* e o PBIL*.	p. 49
4.5	Comparação entre o FPBIL e o FPBIL*.	p. 49
4.6	Comparação entre o PBIL e o PBIL*.	p. 50
4.7	Detalhe do PBIL contra o PBIL*.	p. 50
4.8	Evolução típica do vetor de probabilidades do FPBIL.	p. 51
4.9	Evolução típica do vetor de probabilidades do PBIL.	p. 52


4.10	Gráfico de $B(x, y)$, função objetivo do problema banana.	p. 53
4.11	Curvas de nível de $\log_{10} B(x, y)$	p. 54
4.12	Comparação entre o FPBIL e o PBIL.	p. 55
4.13	Comparação entre o FPBIL* e o PBIL*.	p. 56
4.14	Detalhe do FPBIL* contra o PBIL*.	p. 57
4.15	Comparação entre o FPBIL e o FPBIL*.	p. 57
4.16	Comparação entre o PBIL e o PBIL*.	p. 58
4.17	Comparação entre os algoritmos FPBIL.	p. 62
4.18	Detalhe da figura 4.17 na página 62.	p. 63
4.19	Comparação entre os algoritmos FPBIL.	p. 64
4.20	Detalhe da figura 4.19 na página 64.	p. 64
4.21	Comparação entre o FPBIL e o PBIL.	p. 65
4.22	Detalhe do FPBIL contra o PBIL.	p. 66
4.23	Valores mínimos e máximos ao final de um número de execuções.	p. 66
4.24	Comparação entre o FPBIL* e o PBIL*.	p. 67
4.25	Detalhe do FPBIL* contra o PBIL*.	p. 68
4.26	Comparação entre o FPBIL e o FPBIL*.	p. 68
4.27	Detalhe do FPBIL contra o FPBIL*.	p. 69
4.28	Detalhe da figura 4.27 na página 69.	p. 69
4.29	Comparação entre o PBIL e o PBIL*.	p. 70
4.30	Detalhe do PBIL contra o PBIL*.	p. 70
5.1	Esquema do núcleo de Angra 1.	p. 74

5.2	Elementos central, de quarteto e de octeto.	p. 75
5.3	Evolução de $\mathcal{A}_b(\mathcal{S})$ para o teste 1.	p. 83
5.4	Melhores soluções ao final de cada geração do teste 1.	p. 84
5.5	Evolução de $\mathcal{A}_b(\mathcal{S})$ para o teste 2.	p. 85
5.6	Melhores soluções ao final de cada geração do teste 2.	p. 86
5.7	Evolução de $\mathcal{A}_b(\mathcal{S})$ para o teste 3.	p. 87
5.8	Melhores soluções ao final de cada geração do teste 3.	p. 88
5.9	A melhor configuração de núcleo encontrada pelo FPBIL.	p. 89
5.10	Núcleo construído a partir da figura 5.9.	p. 90
5.11	Melhor resultado das diferentes técnicas.	p. 92

Lista de Tabelas

3.1	Valores de $P(k)$ para os 10 primeiros múltiplos de σ	p. 31
4.1	Definição da função aptidão \mathcal{A}_1	p. 38
4.2	Definição da função aptidão \mathcal{A}_2	p. 39
4.3	Representação binária de 3 cidades.	p. 59
4.4	Representação binária das possíveis rotas.	p. 59
4.5	Possíveis rotas. A maioria delas inválidas.	p. 60
4.6	Rotas válidas utilizando a representação <i>random keys</i>	p. 61
5.1	Formato de entrada usado pelo RECNOD.	p. 77
5.2	<i>Random keys</i> aplicado a $\mathcal{S} Q$	p. 79
5.3	<i>Random keys</i> aplicado a $\mathcal{S} O$	p. 80
5.4	Uma linha (de quarteto) da entrada de dados do RECNOD.	p. 80
5.5	Uma linha (de octeto) da entrada de dados do RECNOD.	p. 80
5.6	Nova entrada das configurações de núcleo para o RECNOD.	p. 81
5.7	Soluções não penalizadas do teste 1 com $C_B(\mathcal{S}) \geq 1.300$	p. 84
5.8	Soluções não penalizadas do teste 2 com $C_B(\mathcal{S}) \geq 1.300$	p. 87
5.9	Soluções não penalizadas do teste 3 com $C_B(\mathcal{S}) \geq 1.300$	p. 88
5.10	Comparação do desempenho das diferentes técnicas.	p. 91
5.11	DEEP e Receita, a mais, em relação a otimização manual.	p. 93

1 *Introdução*

 XISTEM problemas de interesse prático cuja complexidade é de tal ordem que muitos dos algoritmos clássicos de busca sequer podem deles tratar. Problemas com espaços de busca descontínuos ou com múltiplos máximos e mínimos locais, ou ambos simultaneamente, tornam quaisquer algoritmos de busca baseados em gradiente, por exemplo, inadequados.

Com o surgimento dos algoritmos evolucionários, muitas barreiras caíram. Os algoritmos genéticos [1, 2] são exemplos dos que se mostraram robustos o suficiente para abordar, virtualmente, qualquer problema. A forma de representar potenciais soluções utilizando-se vetores binários é o método de busca inspirado na adaptação das espécies [3] — que impõe um paralelismo implícito — e que torna possível a exploração dos mais intrincados espaços de busca.

O PBIL [4] (do inglês: “*Population-Based Incremental Learning*”) é um algoritmo evolucionário que surgiu como tentativa de imitar, de forma simplificada, o comportamento dos Algoritmos Genéticos em uma fase adiantada de sua execução, “em equilíbrio”. O resultado mostra, de forma inesperada, que o PBIL superou [5] os algoritmos genéticos em praticamente todos os aspectos. O PBIL é mais rápido e encontra resultados melhores [6].

Existem diferentes variações [4–8] do PBIL, mas todas dependem da escolha de parâmetros. Há, ainda, outras técnicas de busca sofisticadas, como o ANT-Q [9] — inspirado no comportamento de formigas, cujo desempenho se destaca em problemas combinatórios —; outra técnica é o PSO [10] — inspirado no comportamento

de uma nuvem de aves, destacando-se em problemas numéricos —, porém ambas, indistintamente, requerem parâmetros.

É importante ressaltar que um algoritmo que depende de algum parâmetro pode ser muito eficiente para um conjunto particular de parâmetros, porém muito ineficaz para algum outro conjunto. No caso do PBIL, por exemplo, variações na taxa de aprendizado produzem comportamentos completamente diferentes [4]. Desse modo, quanto menos parâmetros um algoritmo necessitar ajustar, menor o risco deste não atingir todo o seu potencial em alguma situação específica.

O objetivo desta tese é propor um novo algoritmo evolucionário, baseado no PBIL, denominado FPBIL, *parameter Free* PBIL, que é, ao mesmo tempo, livre de parâmetros, mais poderoso que as variações atuais do PBIL e comparável às melhores técnicas de busca disponíveis; e, por fim, aplicá-lo ao problema da otimização de recargas nucleares.

A otimização de recargas nucleares é um problema combinatório de extrema complexidade e de grande interesse da engenharia nuclear. Devido ao grande número de combinações possíveis, mais o fato de ser o espaço de busca altamente fragmentado e multimodal e, ainda, levando-se em consideração as várias restrições técnicas e geométricas as quais precisam ser respeitadas, o problema da otimização de recargas nucleares é um desafio para qualquer algoritmo de busca.

A motivação inicial do presente trabalho para o desenvolvimento do FPBIL era tornar o PBIL o mais compacto e simples possível, de modo que pudesse ser incorporado facilmente ao código do sistema de programação genética [11, 12] CGP lil-gp 2.1;1.02 [13]. Era, também, a oportunidade de pôr em prática o novo método de atualização do vetor de probabilidades, concebido no decorrer do curso de doutorado. O objetivo da simplificação era fazer com que o sistema de programação genética co-evoluísse com o PBIL, permitindo um mecanismo mais adequado de manipulação de constantes numéricas. Os primeiros resultados foram muito animadores, como pode ser visto em [14]. Contudo uma análise mais pormenorizada da variação resul-

tante do PBIL demonstrou possuir tal variação uma estrutura muito rica, digna de atenção exclusiva.

Diante da constatação de tal riqueza, concebe-se esse trabalho de modo a contemplar devidamente os vários aspectos da natureza do FPBIL, visando a tirar de tal análise o melhor proveito possível.

Para isso, no capítulo 2, o algoritmo PBIL, tal como em [5], é apresentado juntamente com as estruturas que fundamentam e sustentam seu funcionamento. Conceitos como espaço de busca, hipercubo, vetor de probabilidades, indivíduo, população e geração são igualmente definidos.

A seguir, o capítulo 3 descreve o algoritmo FPBIL, desenvolvido durante o curso de doutorado. No decorrer do capítulo, mostra-se como cada um dos parâmetros do PBIL é eliminado de forma profunda e simples. Em especial, o vetor de probabilidades é atualizado de maneira simples e direta; todavia, diferentemente do PBIL, utilizando-se de toda a informação disponível em cada geração e, implicitamente, incorporando um poderoso mecanismo que é semelhante a atualização usual do PBIL, porém com parâmetros auto-ajustáveis. A forma radicalmente diferente da mutação observada no FPBIL é fundamentada na distribuição de probabilidades inerente ao próprio vetor de probabilidades, garantindo que a referida mutação seja aplicada sob medida. Tais modificações tornam o FPBIL mais robusto e, ainda, possibilitam que este adquira uma população de tamanho variável, também auto-ajustável, que o torna mais eficiente.

Em seguida, no capítulo 4, o FPBIL é comparado com o PBIL. Para tanto são selecionados três problemas de diferentes classes, todos bem conhecidos. São eles: “quatro picos” — um problema altamente deceptivo, onde não é necessária a decodificação dos vetores binários; “banana” — um problema numérico cuja dificuldade está na forma da superfície em que se deseja localizar o ponto de mínimo; e o “PCV Rykel48” — um problema combinatório que necessita de um tipo de representação especial, *random keys*. Os resultados são apresentados, analisados e comentados.

No Capítulo 5, o FPBIL é utilizado para maximizar o comprimento do ciclo 7 de Angra 1, um caso particular do problema da otimização de recargas nucleares, que já foi abordado sob a ótica de diferentes técnicas, sendo considerado um *benchmark*, a nível nacional, da área. Vale ressaltar que em nenhum dos resultados o FPBIL faz uso de heurísticas e, ainda assim, concorre com as melhores técnicas. Os resultados são apresentados e comparados aos das mais bem-sucedidas abordagens.

Finalmente, o capítulo 6 encerra o trabalho ressaltando as vantagens da nova técnica e sugerindo possíveis aperfeiçoamentos.

2 O Algoritmo PBIL



ESTE capítulo será apresentado o algoritmo PBIL, *Population-Based Incremental Learning*. O PBIL é um algoritmo de busca extremamente versátil e tem se mostrado superior aos algoritmos genéticos, tanto em simplicidade quanto em eficiência e desempenho.

Inicialmente será examinada a estrutura geral relacionada a qualquer algoritmo do tipo PBIL, seguida da apresentação do algoritmo PBIL original.

2.1 Estrutura do PBIL

Com o PBIL, um subconjunto \mathcal{S}_B do espaço de busca \mathcal{B} de um determinado problema de otimização é explorado a partir de um hipercubo $\mathcal{H}_n = [0, 1]^n$, de modo que cada vértice de \mathcal{H}_n , ou seja, cada ponto de $\check{\mathcal{H}}_n = \{0, 1\}^n$ corresponda a um ponto de \mathcal{S}_B . Essa correspondência é feita a partir de um mapeamento sobrejetivo¹

$$\begin{aligned} \mathcal{M}_{\mathcal{S}_B}^n : \check{\mathcal{H}}_n &\longrightarrow \mathcal{S}_B \\ \mathcal{I} \equiv (I_1, I_2, \dots, I_n) &\longmapsto \mathcal{M}_{\mathcal{S}_B}^n(\mathcal{I}), \end{aligned} \tag{2.1}$$

com $I_k \equiv \mathcal{I}[k] \in \{0, 1\}$. As escolhas de \mathcal{S}_B e n dependem do problema em questão. Entretanto, mesmo com \mathcal{S}_B e n fixos, a escolha de $\mathcal{M}_{\mathcal{S}_B}^n$ geralmente não é única e pode influenciar no desempenho do PBIL².

¹O mapeamento precisa ser sobrejetivo a fim de representar todo o conjunto \mathcal{S}_B .

²A razão para isto é que a busca se dá em \mathcal{H}_n e, geralmente, diferentes $\mathcal{M}_{\mathcal{S}_B}^n$ produzem diferentes topologias em \mathcal{H}_n .

A fim de ilustrar os diversos conceitos, considere o seguinte exercício de encontrar o valor de x , no intervalo $[0, 10] \subset \mathbb{R}$, que maximiza $f(x) = x^2$. De imediato podemos fazer a seguinte associação:

$$\mathcal{B} = \mathbb{R}.$$

$\mathcal{S}_{\mathcal{B}}$ e o valor de n irão depender da precisão com que se deseja a solução. Suponhamos que duas casas decimais sejam suficientes. Neste caso, os candidatos a solução podem ser

$$\mathcal{S}_{\mathcal{B}} = \{0,00; 0,01; 0,02; \dots; 10,00\},$$

1001 candidatos. Devemos, portanto, escolher n de modo que o número de vértices de \mathcal{H}_n seja no mínimo 1001. Precisamos, então, satisfazer $2^n \geq 1001$, o que equivale a $n \gtrsim 9,97$. Podemos, portanto, escolher

$$n = 10.$$

$\mathcal{M}_{[0,10]}^{10}$ pode ser implementada explorando-se simplesmente a representação binária usual:

$$\begin{aligned} \mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0) &= 0,00 \\ \mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 1) &= 0,01 \\ \mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 1, 0) &= 0,02 \\ \mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 1, 1) &= 0,03 \\ \mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 1, 0, 0) &= 0,04 \\ &\vdots \\ \mathcal{M}_{[0,10]}^{10}(1, 1, 1, 1, 1, 0, 1, 0, 0, 0) &= 10,00. \end{aligned} \tag{2.2}$$

Neste caso, os 23 elementos restantes de $\tilde{\mathcal{H}}_{10}$ podem ser associados, por exemplo³, ao número 10,00.

³Embora válida, esta prática deve ser desencorajada pois, como será visto, pode confundir os algoritmos PBIL e FPBIL.

Uma outra possibilidade é tornar $\mathcal{M}_{[0,10]}^{10}$ bijetiva por meio de

$$\begin{aligned}
\mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0) &= \frac{10 \cdot 0}{1023} = 0,0000 \\
\mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 1) &= \frac{10 \cdot 1}{1023} \sim 0,0098 \\
\mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 1, 0) &= \frac{10 \cdot 2}{1023} \sim 0,0196 \\
\mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 0, 1, 1) &= \frac{10 \cdot 3}{1023} \sim 0,0293 \\
\mathcal{M}_{[0,10]}^{10}(0, 0, 0, 0, 0, 0, 0, 1, 0, 0) &= \frac{10 \cdot 4}{1023} \sim 0,0391 \\
&\vdots \\
\mathcal{M}_{[0,10]}^{10}(1, 1, 1, 1, 1, 1, 1, 1, 1, 1) &= \frac{10 \cdot 1023}{1023} = 10,0000.
\end{aligned} \tag{2.3}$$

Neste caso, $\mathcal{S}_{\mathcal{B}} = \{0,0000; 0,0098; 0,0196; \dots; 10,0000\}$.

A diferença entre cada par de pontos \mathcal{R} e \mathcal{S} de \mathcal{H}_n gera o conjunto $\mathcal{V}_{\mathcal{H}_n}$ de vetores sobre \mathcal{H}_n :

$$\mathcal{V}_{\mathcal{H}_n} = \{\mathcal{R} - \mathcal{S} \mid \mathcal{R}, \mathcal{S} \in \mathcal{H}_n\}, \tag{2.4}$$

cujas coordenadas estão em $[-1, 1]$. Pode-se então perceber que cada ponto $\mathcal{P} \in \mathcal{H}_n$ é equivalente ao vetor $\mathcal{P} - \mathcal{O} \equiv \mathcal{P} \in \mathcal{V}_{\mathcal{H}_n}$, onde $\mathcal{O} \equiv (0, 0, \dots, 0)$ pode ser considerado como a origem de \mathcal{H}_n .

Considere agora um ponto $\mathcal{P} \in \mathcal{H}_n$. Suas n componentes $p_k \equiv \mathcal{P}[k] \in [0, 1]$ são apropriadas para representar probabilidades. Se, por exemplo, p_k representar a probabilidade de se sortear o número 1 em um espaço amostral $\Omega = \{0, 1\}$, então \mathcal{P} representa uma distribuição de probabilidades sobre $\check{\mathcal{H}}_n$:

$$\begin{aligned}
\mathcal{P}: \check{\mathcal{H}}_n &\longrightarrow [0, 1] \\
\mathcal{I} &\longmapsto p = \mathcal{P}(\mathcal{I}).
\end{aligned} \tag{2.5}$$

Mais especificamente, considere o vetor distância $\mathbf{d}^{\mathcal{I}} = \mathcal{I} - \mathcal{P}$ que vai de \mathcal{P} até \mathcal{I} , cujas componentes $d_k^{\mathcal{I}}$ são, em módulo, as distâncias entre \mathcal{P} e os n hiperplanos

$(I_1, \#, \dots, \#), (\#, I_2, \dots, \#), \dots, (\#, \#, \dots, I_n)$ que se interceptam em \mathcal{I} — com $\#$ podendo substituir tanto um “0” quanto um “1”. $\mathcal{P}(\mathcal{I})$ pode, então, ser expresso simplesmente por:

$$\mathcal{P}(\mathcal{I}) = \prod_{k=1}^n (1 - |d_k^{\mathcal{I}}|). \quad (2.6)$$

Por exemplo, $\mathcal{P}' = (0,5; 0,1; 0,7; 1; 0) \in \mathcal{H}_5$ determina que a probabilidade de $\mathcal{I}_1 = (1, 1, 1, 1, 1) \in \check{\mathcal{H}}_5$ ser sorteado é $\mathcal{P}'(\mathcal{I}_1) = 0,5 \cdot 0,1 \cdot 0,7 \cdot 1 \cdot 0 = 0$. Já para $\mathcal{I}_2 = (0, 0, 0, 0, 0)$ ser sorteado a probabilidade é $\mathcal{P}'(\mathcal{I}_2) = 0,5 \cdot 0,9 \cdot 0,3 \cdot 0 \cdot 1 = 0$, e para $\mathcal{I}_3 = (1, 0, 0, 1, 0)$, a probabilidade é $\mathcal{P}'(\mathcal{I}_3) = 0,5 \cdot 0,9 \cdot 0,3 \cdot 1 \cdot 1 = 0,135$. Observe que qualquer ponto de $\check{\mathcal{H}}_5$ com a forma $(\#, \#, \#, 0, \#)$ ou $(\#, \#, \#, \#, 1)$ nunca será sorteado a partir de \mathcal{P}' , significando que a presença das duas últimas componentes de \mathcal{P}' conferem probabilidades não nulas apenas a elementos de $\check{\mathcal{H}}_3 \times \{(1, 0)\} \subset \check{\mathcal{H}}_5$.

Verifica-se, a partir da equação (2.6), que o ponto $\mathcal{I} \in \check{\mathcal{H}}_n$ mais provável de ser sorteado a partir de \mathcal{P} é o vértice de \mathcal{H}_n mais próximo de \mathcal{P} . O segundo ponto mais provável de ser sorteado a partir de \mathcal{P} é o segundo vértice de \mathcal{H}_n mais próximo de \mathcal{P} , e assim por diante. Também é notável o fato de que basta uma componente $d_k^{\mathcal{I}}$ de $\mathcal{I} - \mathcal{P}$ satisfazer $|d_k| = 1$ para que $\mathcal{P}(\mathcal{I}) = 0$, condição que é equivalente a

$$p_k = \begin{cases} 1, & \text{se } I_k = 0; \\ 0, & \text{se } I_k = 1. \end{cases} \quad (2.7)$$

Em outras palavras, se $p_k = 0$, por exemplo, então qualquer ponto de $\check{\mathcal{H}}_n$ com $I_k = 1$ terá probabilidade zero de ser sorteado, ou ainda, as probabilidades não nulas ficam restritas ao conjunto $\check{\mathcal{H}}_{k-1} \times \{0\} \times \check{\mathcal{H}}_{n-k}$, que possui a mesma estrutura de $\check{\mathcal{H}}_{n-1}$: a “ordem” de $\check{\mathcal{H}}$ é reduzida por 1. À medida que outras componentes de $\mathbf{d}^{\mathcal{I}}$ satisfazem a equação (2.7), as probabilidades não nulas ficam restritas a um número cada vez menor de vértices — $\check{\mathcal{H}}$ de ordem cada vez mais reduzida — até o caso extremo em que $\mathcal{P} \equiv \mathcal{I}_i$, um vértice qualquer de \mathcal{H}_n , quando $\mathcal{P}(\mathcal{I}_j) = \delta_{ij}$, ou seja, \mathcal{I}_i sempre será o único sorteado.

A próxima seção mostra como o PBIL utiliza os pontos de \mathcal{H}_n a fim de tentar

localizar soluções ótimas em um espaço de busca \mathcal{B} .

2.2 O Algoritmo PBIL Original

O PBIL [4] foi criado em 1994 por *Shumeet Baluja*. Foi inspirado em seu trabalho anterior com *Ari Juels* [15] em uma tentativa de simular o comportamento dos algoritmos genéticos [1] em “estado de equilíbrio”, após repetidas aplicações do operador *crossover*.

O algoritmo aqui referenciado por “PBIL original” teve sua publicação posteriormente, em 1995 [5], por seu criador, na qual 27 problemas, comumente explorados na literatura de algoritmos genéticos, são examinados por 7 diferentes técnicas de otimização, com o PBIL tendo o melhor desempenho em mais de 80% dos casos. O algoritmo PBIL original é mostrado na figura 2.1 na próxima página.

Após os vértices de \mathcal{H}_n serem devidamente mapeados em um subconjunto \mathcal{S}_B do espaço de busca, o PBIL funciona exatamente da mesma forma, independentemente do problema em questão. Essa característica é que torna o PBIL versátil, de modo que a condição necessária e suficiente para que um problema de otimização seja abordado pelo PBIL é a existência de $\mathcal{M}_{\mathcal{S}_B}^n$.

Embora o PBIL possa ser aplicado em praticamente qualquer caso, a sua utilização é recomendada somente em problemas mais complexos, quando as técnicas tradicionais falham, como em problemas da classe NP, busca em espaços multimodais, descontínuos, etc.

No início de cada PBIL todos os pontos de \mathcal{S}_B devem ser tratados como potenciais soluções e \mathcal{P} vértices de \mathcal{H}_n são, portanto, escolhidos aleatoriamente a partir de uma distribuição de probabilidades uniforme. Essa distribuição uniforme de probabilidades nada mais é que $\mathcal{P}_0 = (0,5, 0,5, \dots, 0,5)$, o centro de \mathcal{H}_n . Esta é a linha 01 da figura 2.1 na página seguinte.

Na terminologia do PBIL, os \mathcal{P} vértices \mathcal{I}_k de \mathcal{H}_n selecionados a partir de \mathcal{P}_G


```

**   *** Inicialização do Vetor de Probabilidades
01   for (j = 1...n)  P[j] = 0,5
**   ***
02   while (not Critério_de_Término)
03       for (i = 1...P)
04           for (j = 1...n)
**               *** Criação de  $\mathcal{I}_i$  a partir de  $\mathcal{P}$ 
05               if (random (0, 1) < P[j])   $\mathcal{I}_i[j] = 1$ 
06               else                         $\mathcal{I}_i[j] = 0$ 
**               ***
07           Calcular  $\mathcal{A}(\mathcal{I}_i)$ 
08           Designar  $\mathcal{I}^+$ , o indivíduo mais apto
09           Designar  $\mathcal{I}^-$ , o indivíduo menos apto
10           for (j = 1...n)
**               *** Atualização de  $\mathcal{P}$  na direção de  $\mathcal{I}^+$ 
11                $\mathcal{P}[j] = (1 - \alpha) \cdot \mathcal{P}[j] + \alpha \cdot \mathcal{I}^+[j]$ 
**               ***
12               if ( $\mathcal{I}^-[j] \neq \mathcal{I}^+[j]$ )
**                   *** Atualização de  $\mathcal{P}$  na direção oposta a de  $\mathcal{I}^-$ 
13                    $\mathcal{P}[j] = (1 - \beta) \cdot \mathcal{P}[j] + \beta \cdot \mathcal{I}^+[j]$ 
**                   ***
**               *** Aplicação da Mutação em  $\mathcal{P}$ 
14           for (j = 1...n)
15               if (random (0, 1) <  $P_{\mathcal{M}}$ )
16                   if (random (0, 1) < 0,5)   $D_{\mathcal{M}} = 1$ 
17                   else                         $D_{\mathcal{M}} = 0$ 
18                    $\mathcal{P}[j] = (1 - \gamma) \cdot \mathcal{P}[j] + \gamma \cdot D_{\mathcal{M}}$ 
**               ***

```

\mathcal{P} : Tamanho da População (100).
 α : Taxa de Aprendizado (0,1).
 β : Taxa de Aprendizado Negativo (0,075).
 $P_{\mathcal{M}}$: Probabilidade de Mutação (0,02).
 γ : Taxa de Mutação (0,05).

Figura 2.1: Algoritmo PBIL original.

(linhas 03 – 06 da figura 2.1) formam uma “população” — a “geração” \mathcal{G} — e são chamados de “indivíduos”. O algoritmo PBIL consiste em, baseado nos indivíduos da geração 0, construir \mathcal{P}_1 , que dará origem a uma outra população — a geração 1. O processo se repete, até que um indivíduo de uma geração qualquer seja considerado bom o suficiente (linha 06 da figura 2.1 na página precedente).

A medida do quanto um indivíduo é bom ou ruim é dada pela função “aptidão”⁴:

$$\begin{aligned} \mathcal{A}: \check{\mathcal{H}}_n &\longrightarrow \mathbb{R}^+ \\ \mathcal{I} &\longmapsto \mathcal{A} = \mathcal{A}(\mathcal{I}). \end{aligned} \tag{2.8}$$

Na maioria das vezes, entretanto, \mathcal{I} não é avaliado diretamente, sendo necessária uma composição $\mathcal{A} = \mathcal{M}_{\mathcal{S}_B}^n \circ \mathcal{A}'$, com $\mathcal{A}': \mathcal{S}_B \longrightarrow \mathbb{R}$. Esta é a linha 07 da figura 2.1.

Pode-se escolher \mathcal{A} de forma que $\mathcal{A}(\mathcal{I}_i) \geq \mathcal{A}(\mathcal{I}_j)$ implique em \mathcal{I}_i melhor que ou tão bom quanto \mathcal{I}_j , tornando trivial a tarefa de designar o “indivíduo mais apto”, \mathcal{I}^+ e o “indivíduo menos apto”, \mathcal{I}^- (Linhas 08 e 09 da figura 2.1 na página precedente).

A construção de $\mathcal{P}_{\mathcal{G}+1}$ a partir dos indivíduos da geração \mathcal{G} é o processo mais importante do PBIL. Qualquer ponto $\mathcal{P}_{\mathcal{G}+1}$ de $\check{\mathcal{H}}_n$ diferente de \mathcal{P}_0 gera uma distribuição de probabilidades não-uniforme sobre $\check{\mathcal{H}}_n$. A estratégia do PBIL é a de, geração após geração, modificar essa distribuição de probabilidades tornando cada vez mais provável o surgimento de \mathcal{I}^\dagger , a solução ótima. No PBIL original, $\mathcal{P}_{\mathcal{G}+1}$ é construído em duas fases.

Na primeira fase, são realizadas as seguintes operações (equivalentes as das linhas

⁴A medida natural da qualidade de um indivíduo $Q(\mathcal{I})$ em um determinado problema de otimização pode ser negativa. Neste caso, a aptidão deve ser qualquer transformação de Q que leve a \mathbb{R}^+ , preservando a relação de qual indivíduo é melhor que outro. Esta questão, apesar de não causar nenhum efeito no PBIL, será indispensável para o FPBIL, apresentado no capítulo 3 na página 14.

10–13 da figura 2.1):

$$\mathcal{P}_{\mathcal{G}+1/2} = \mathcal{P}_{\mathcal{G}} + \alpha \cdot (\mathcal{I}^+ - \mathcal{P}_{\mathcal{G}}) \quad (2.9)$$

$$\mathcal{P}'_{\mathcal{G}+1}[j] = \begin{cases} \mathcal{P}_{\mathcal{G}+1/2}[j] + \beta \cdot (\mathcal{I}^+[j] - \mathcal{P}_{\mathcal{G}+1/2}[j]) & \text{se } \mathcal{I}^+[j] \neq \mathcal{I}^-[j] \\ \mathcal{P}_{\mathcal{G}+1/2}[j] & \text{se } \mathcal{I}^+[j] = \mathcal{I}^-[j] \end{cases} \quad (2.10)$$

ou seja, $\mathcal{P}_{\mathcal{G}}$ é inicialmente deslocado na direção de \mathcal{I}^+ por uma fração α da distância entre $\mathcal{P}_{\mathcal{G}}$ e \mathcal{I}^+ . Em seguida, $\mathcal{P}_{\mathcal{G}+1/2}$ é alterado de modo que a projeção de $\mathcal{P}_{\mathcal{G}+1/2}$ no subespaço $\mathcal{H}_l \subset \mathcal{H}_n$ — formado pelas l coordenadas para as quais $\mathcal{I}^+[j] \neq \mathcal{I}^-[j]$ — seja afastada da projeção de \mathcal{I}^- , também em \mathcal{H}_l . Em \mathcal{H}_l , \mathcal{I}^+ e \mathcal{I}^- são diametralmente opostos e temos $\mathcal{I}^+[j] = 1 - \mathcal{I}^-[j]$, ou seja, aproximar-se de \mathcal{I}^+ é o mesmo que afastar-se de \mathcal{I}^- . Portanto, se olharmos de dentro desse subespaço, observaremos $\mathcal{P}_{\mathcal{G}+1/2}$ sendo deslocado na direção oposta a \mathcal{I}^- por uma fração β da distância entre $\mathcal{P}_{\mathcal{G}+1/2}$ e \mathcal{I}^+ (e não \mathcal{I}^-). Uma vez que o vértice de \mathcal{H}_n mais próximo de $\mathcal{P}_{\mathcal{G}+1}$ é o mais provável de ser escolhido, o PBIL original tenta alcançar seu objetivo direcionando a distribuição de probabilidades $\mathcal{P}_{\mathcal{G}+1}$ para o melhor indivíduo disponível em \mathcal{G} .

Na segunda fase, $\mathcal{P}'_{\mathcal{G}+1}$ sofre uma “mutação”. Pela forma como os indivíduos são criados a partir de $\mathcal{P}_{\mathcal{G}}$, e como $\mathcal{P}_{\mathcal{G}+1}$ é construído em seguida; uma vez que uma componente de $\mathcal{P}_{\mathcal{G}}$ atinge o valor 0 (ou 1), essa componente continuará sendo 0 (ou 1) até o final do algoritmo. Se, eventualmente, a solução ótima possuir a correspondente componente com valor oposto ao 0 (ou 1) de $\mathcal{P}_{\mathcal{G}}$, ela nunca será encontrada em nenhuma geração posterior, a menos que algum mecanismo adjacente seja capaz de alterar ocasionalmente as componentes de $\mathcal{P}'_{\mathcal{G}+1}$.


Tal mutação consiste em deslocar as componentes de $\mathcal{P}'_{\mathcal{G}+1}$ aleatoriamente na direção de 0 ou 1. Isto significa que cada componente $\mathcal{P}'_{\mathcal{G}+1}[j]$ poderá, ou não, sofrer o deslocamento (de acordo com a “probabilidade de mutação”), conforme a expressão:

$$\mathcal{P}_{\mathcal{G}+1}[j] = \mathcal{P}'_{\mathcal{G}+1}[j] + \gamma \cdot (D_{\mathcal{M}} - \mathcal{P}'_{\mathcal{G}+1}[j]), \quad (2.11)$$

ou seja, o deslocamento é na direção de $D_{\mathcal{M}}$ (0 ou 1, aleatoriamente) com uma amplitude igual à fração γ da distância entre $\mathcal{P}'_{g+1}[j]$ e $D_{\mathcal{M}}$ (Linhas 14–18 da figura 2.1 na página 10).

Como pode ser verificado na figura 2.1 na página 10, o PBIL original necessita de 5 parâmetros para funcionar e parece não existir nenhuma evidência de que estes sejam independentes de uma aplicação em particular — de fato, os valores da figura 2.1 foram determinados experimentalmente de modo a maximizar o desempenho médio do algoritmo em uma gama de diferentes aplicações. No próximo capítulo, propõe-se uma variação do PBIL livre de parâmetros.

3 *FPBIL: parameter Free PBIL*

 presente capítulo apresenta o algoritmo FPBIL, *parameter Free Population-Based Incremental Learning*, desenvolvido durante o trabalho de doutorado. Nota-se que o “p” de *parameter* não entra na sigla do algoritmo. O motivo é não sobrecarregar a nomenclatura nem dificultar demasiadamente sua pronúncia. O nome do algoritmo se encontra em língua inglesa pelo fato de qualquer tentativa de adaptação desse para a língua portuguesa resultar na descaracterização de sua relação com o PBIL.

O FPBIL é uma variação do PBIL que procura eliminar a necessidade dos parâmetros do PBIL modificando fundamentalmente alguns de seus princípios. O resultado é um algoritmo mais eficiente, com um poder de busca superior e sem parâmetros.

3.1 O Algoritmo FPBIL

Como no algoritmo PBIL original, o FPBIL apresenta um vetor de probabilidades \mathcal{P} , com n componentes $p_k \in [0, 1]$, a partir do qual os \mathcal{P} indivíduos \mathcal{I}_k de uma determinada geração são criados. A característica que os diferencia é que o FPBIL procura utilizar mecanismos suficientemente genéricos para torná-lo livre de parâmetros, especialmente pela forma como \mathcal{P} é atualizado e como a mutação é implementada. O algoritmo FPBIL é apresentado na figura 3.1 na página seguinte.

```

**   *** Definição das Funções  $C_x$  e  $D_x$ 
01    $D_x \equiv 1/(1+x)$ 
02    $C_x \equiv D_x^{-1} \equiv 1/x - 1$ 
**   ***
**   *** Inicialização do Vetor de Probabilidades, de  $\mathcal{P}_0$  e de  $d$ 
03   for ( $j = 1 \dots n$ )  $\mathcal{P}[j] = 0,5$ 
04    $\mathcal{P}_0 = 7(1 + 1/n)^n$ 
05    $d = 1/3$ 
**   ***
06   while (not Critério_de_Término)
**       *** Determinação de  $\mathcal{P}$ 
07       if (Flutuação em  $c$ )
08            $\mathcal{P}_0 = \mathcal{P}_0 + 1$ 
09           if ( $\Delta\langle c \rangle_{\mathcal{G}} < 1\%$ )
10                $d = 1/3$ 
11                $\mathcal{P} = \mathcal{P}_0$ 
12        $c = C_d$ 
13        $\mathcal{P} = (\text{int})\mathcal{P}_0(1 + 1/c)^c(\mathcal{P}_0/7)^{-c/n}$ 
**       ***
14       for ( $i = 1 \dots \mathcal{P}$ )
**           Criação de  $\mathcal{I}_i$  a partir de  $\mathcal{P}$ , como no PBIL original
15           Calcular  $\mathcal{A}(\mathcal{I}_i)$ 
16       for ( $j = 1 \dots n$ )
**           *** Atualização de  $\mathcal{P}$ 
17            $\mathcal{P}[j] = (\sum_{i=1}^{\mathcal{P}} \mathcal{A}(\mathcal{I}_i) \cdot \mathcal{I}_i[j]) / (\sum_{i=1}^{\mathcal{P}} \mathcal{A}(\mathcal{I}_i))$ 
**           ***
**       *** Aplicação da Mutaç o em  $\mathcal{P}$ 
18        $c =$  contagem dos casos ( $\mathcal{P}[j] \leq d$  ou  $\mathcal{P}[j] \geq 1 - d$ )
19        $c' =$  contagem dos casos ( $\mathcal{P}[j] \leq D_{C_d-1}$  ou  $\mathcal{P}[j] \geq 1 - D_{C_d-1}$ )
20       if ( $c > C_d$ )  $d = D_{c+1}$ 
21       else if ( $c' < C_d - 1$ )  $d = D_{c-1}$ 
22       if ( $d > 1/3$ )  $d = 1/3$ 
23       for ( $j = 1 \dots n$ )
24           if ( $\mathcal{P}[j] < d$ )  $\mathcal{P}[j] = d$ 
25           if ( $\mathcal{P}[j] > 1 - d$ )  $\mathcal{P}[j] = 1 - d$ 
**       ***

```

Figura 3.1: Algoritmo FPBIL: *parameter Free* PBIL.

3.2 Eliminando os Parâmetros α e β

No algoritmo PBIL original, o vetor de probabilidades é atualizado sofrendo um pequeno deslocamento que o aproxima do melhor indivíduo e outro deslocamento que o afasta do pior indivíduo. Em algumas variantes do PBIL, utiliza-se apenas o melhor indivíduo, ou apenas o pior indivíduo, ou também, em substituição ao melhor indivíduo, a média dos l melhores indivíduos:

$$\mathcal{Q}_l = \frac{\sum_{i=1}^l \mathcal{I}_i}{l} = \frac{\mathcal{I}_1 + \mathcal{I}_2 + \cdots + \mathcal{I}_l}{l}, \quad (3.1)$$

procedendo a atualização de \mathcal{P}_G conforme

$$\mathcal{P}'_{G+1} = \mathcal{P}_G + \chi \cdot (\mathcal{Q}_l - \mathcal{P}_G). \quad (3.2)$$

O fato é que para avaliar quem são os melhores e piores indivíduos, todos os indivíduos devem ser avaliados, o que significa que todos os algoritmos PBIL desperdiçam quase toda informação disponível sobre o espaço de busca.

Uma primeira tentativa de aproveitar toda a informação seria generalizar a expressão (3.1) para $l = \mathcal{P}$. Entretanto, $\lim_{l \rightarrow 2^n} \mathcal{Q}_l = \mathcal{P}_0$, o que resultaria em deslocamentos para o centro de \mathcal{H}_n , independentemente de \mathcal{P}_G . A soma na equação (3.1) pressupõe que não há repetição de nenhum indivíduo.

Uma segunda tentativa seria, portanto, tomar $\mathcal{Q}_{\mathcal{P}} = \sum_{i=1}^{\mathcal{P}} \mathcal{I}_i / \mathcal{P}$, onde i , agora, é a ordem em que os indivíduos são sorteados, podendo haver repetições de indivíduos de acordo com a probabilidade desses serem sorteados. Porém, neste caso, $\lim_{\mathcal{P} \rightarrow \infty} \mathcal{Q}_{\mathcal{P}} = \mathcal{P}_G$, o que resulta em $\mathcal{P}'_{G+1} = \mathcal{P}_G$: nenhuma modificação de \mathcal{P}_G . Talvez, por esta razão, não se costume empregar todos os indivíduos no processo de atualização do vetor de probabilidades.

A regra segundo a qual o FPBIL atualiza seu vetor de probabilidades é

$$\mathcal{P}'_{G+1} = \frac{\sum_{i=1}^{\mathcal{P}} A_i \cdot \mathcal{I}_i}{\sum_{i=1}^{\mathcal{P}} A_i}, \quad (3.3)$$

que é justamente uma média (permitindo a repetição de indivíduos) na qual utilizam-se todos os \mathcal{P} indivíduos. A diferença é que esta média é ponderada pela aptidão $\mathcal{A}_i \equiv \mathcal{A}(\mathcal{I}_i)$ de cada indivíduo. A fim de poder apreciar a mudança causada por esse detalhe, pode-se deduzir que

$$\mathcal{P}'_{\mathcal{G}+1} = \frac{\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \mathcal{A}_i \cdot \mathcal{I}_i + \sum_{\mathcal{A}_i = \langle \mathcal{A} \rangle} \mathcal{A}_i \cdot \mathcal{I}_i + \sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} \mathcal{A}_i \cdot \mathcal{I}_i}{\sum_{i=1}^{\mathcal{P}} \mathcal{A}_i}, \quad (3.4)$$

onde $\langle \mathcal{A} \rangle \equiv \sum_{i=1}^{\mathcal{P}} \mathcal{A}_i / \mathcal{P}$ é a aptidão média da geração \mathcal{G} . Somando e subtraindo os termos $\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \langle \mathcal{A} \rangle \cdot \mathcal{I}_i$ e $\sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} \langle \mathcal{A} \rangle \cdot \mathcal{I}_i$ no numerador, é possível observar que

$$\mathcal{P}'_{\mathcal{G}+1} = \frac{\langle \mathcal{A} \rangle \cdot \sum_{i=1}^{\mathcal{P}} \mathcal{I}_i + \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) \cdot \mathcal{I}_i + \sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) \cdot \mathcal{I}_i}{\mathcal{P} \cdot \langle \mathcal{A} \rangle}. \quad (3.5)$$

Multiplicando e dividindo esta expressão por $\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle)$, temos:

$$\mathcal{P}'_{\mathcal{G}+1} = \frac{\sum_{i=1}^{\mathcal{P}} \mathcal{I}_i}{\mathcal{P}} + \frac{1}{\mathcal{P}} \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \frac{\mathcal{A}_i - \langle \mathcal{A} \rangle}{\langle \mathcal{A} \rangle} \cdot \mathcal{D}, \quad (3.6)$$

$$\mathcal{D} \equiv \frac{\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) \cdot \mathcal{I}_i + \sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) \cdot \mathcal{I}_i}{\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle)}. \quad (3.7)$$

E já que $\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) = \sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} (\langle \mathcal{A} \rangle - \mathcal{A}_i)$, chega-se ao resultado:

$$\mathcal{P}'_{\mathcal{G}+1} = \frac{\sum_{i=1}^{\mathcal{P}} \mathcal{I}_i}{\mathcal{P}} + \frac{1}{\mathcal{P}} \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \frac{\mathcal{A}_i - \langle \mathcal{A} \rangle}{\langle \mathcal{A} \rangle} \cdot \mathcal{D}, \quad (3.8)$$

$$\mathcal{D} \equiv \frac{\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle) \cdot \mathcal{I}_i}{\sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} (\mathcal{A}_i - \langle \mathcal{A} \rangle)} - \frac{\sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} (\langle \mathcal{A} \rangle - \mathcal{A}_i) \cdot \mathcal{I}_i}{\sum_{\mathcal{A}_i < \langle \mathcal{A} \rangle} (\langle \mathcal{A} \rangle - \mathcal{A}_i)}. \quad (3.9)$$

Para ajudar na interpretação do resultado, considere a seguinte notação:

$${}^r \langle O_i \rangle_{w_i}^{c_i} = \frac{\sum_{c_i} w_i \cdot O_i}{\sum_{c_i} w_i}, \quad (3.10)$$

que significa a média dos objetos O_i , ponderada pelo peso w_i , restrita à condição c_i . Caso $w_i = \text{const}, \forall i$, ${}^r \langle O_i \rangle^{c_i}$ torna-se uma média simples e w_i pode ser omitido. c_i também pode ser omitido caso a restrição esteja implícita. Até mesmo o índice

i pode ser omitido, como em $\langle \mathcal{A} \rangle$. A presença do “ r ” significa que a repetição de objetos O_i é permitida, caso contrário o mesmo é omitido. Por exemplo, a equação (3.3) na página 16 pode ser escrita simplesmente como $\mathcal{P}'_{\mathcal{G}+1} = {}^r \langle \mathcal{I}_i \rangle_{\mathcal{A}_i}^{i=1\dots\mathcal{P}}$, ou até mesmo como $\mathcal{P}'_{\mathcal{G}+1} = {}^r \langle \mathcal{I}_i \rangle_{\mathcal{A}_i}^i$, enquanto a maneira correta de escrever $\langle \mathcal{A} \rangle$ seria, pelo menos, ${}^r \langle \mathcal{A} \rangle$. Assim, as equações (3.8) e (3.9) tornam-se

$$\mathcal{P}'_{\mathcal{G}+1} = {}^r \langle \mathcal{I}_i \rangle^i + \xi \cdot \mathcal{D} \quad (3.11)$$

$$\mathcal{D} \equiv \left({}^r \langle \mathcal{I}_i \rangle_{\mathcal{A}_i > \langle \mathcal{A} \rangle} - {}^r \langle \mathcal{I}_i \rangle_{\langle \mathcal{A} \rangle - \mathcal{A}_i} \right), \quad (3.12)$$

com

$$\xi = \frac{1}{\mathcal{P}} \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \frac{\mathcal{A}_i - \langle \mathcal{A} \rangle}{\langle \mathcal{A} \rangle}. \quad (3.13)$$

Neste ponto alguns comentários são cabíveis e necessários. Primeiramente, a equação (3.11) possui uma estrutura semelhante a da combinação das equações (2.9) e (2.10) na página 12. Como ${}^r \langle \mathcal{I}_i \rangle^i \approx \mathcal{P}_{\mathcal{G}}$, o vetor de probabilidades é atualizado deslocando-se $\mathcal{P}_{\mathcal{G}}$ na direção de \mathcal{D} por uma fração ξ da distância entre ${}^r \langle \mathcal{I}_i \rangle_{\mathcal{A}_i > \langle \mathcal{A} \rangle}^{\mathcal{A}_i > \langle \mathcal{A} \rangle}$ e ${}^r \langle \mathcal{I}_i \rangle_{\langle \mathcal{A} \rangle - \mathcal{A}_i}^{\mathcal{A}_i < \langle \mathcal{A} \rangle}$.

Durante a execução do FPBIL, \mathcal{P} traça um caminho em \mathcal{H}_n enquanto duram as gerações. A direção e intensidade de cada passo pode ser interpretada como a velocidade com que \mathcal{P} percorre este caminho. Como o caminho é percorrido discretamente, uma possível representação da velocidade é

$$\frac{\partial \mathcal{P}_{\mathcal{G}}}{\partial \mathcal{G}} \equiv \frac{\mathcal{P}_{\mathcal{G}+1} - \mathcal{P}_{\mathcal{G}}}{1}. \quad (3.14)$$

No PBIL original, não é simples identificar a forma exata da velocidade (mesmo sem considerar a mutação), no entanto, a combinação das equações (2.9) e (2.10) resulta em

$$\frac{\partial \mathcal{P}_{\mathcal{G}}}{\partial \mathcal{G}} \approx f(\alpha, \beta) \cdot (\mathcal{I}^+ - \mathcal{I}^-). \quad (3.15)$$

No FPBIL tem-se

$$\frac{\partial \mathcal{P}_{\mathcal{G}}}{\partial \mathcal{G}} \approx \xi \cdot \mathcal{D}. \quad (3.16)$$

A vantagem em se utilizar \mathcal{D} é que a direção do deslocamento não é baseada apenas no melhor e no pior indivíduo, mas em toda a informação disponível sobre o espaço de busca em uma geração (\mathcal{P} indivíduos avaliados). Um outro detalhe sobre \mathcal{D} é que as médias não são simples, mas ponderadas pelas diferenças entre as aptidões de cada indivíduo e a aptidão média, de modo que um indivíduo muito ruim ou muito bom exerce mais influência do que outros com aptidões próximas a da média.

É interessante notar que cada ponto \mathcal{P} de \mathcal{H}_n pode ser associado a uma aptidão média ${}_{\mathcal{P}}\langle \mathcal{A} \rangle$ por meio de

$$\begin{aligned} {}_{\mathcal{P}}\langle \mathcal{A} \rangle : \mathcal{H}_n &\longrightarrow \mathbb{R} \\ \mathcal{P} &\longmapsto {}_{\mathcal{P}}\langle \mathcal{A} \rangle = \sum_{i=1}^{2^n} \mathcal{A}_i \cdot \mathcal{P}(\mathcal{I}_i). \end{aligned} \quad (3.17)$$

O motivo é que, a partir de \mathcal{P} , cada indivíduo \mathcal{I}_i possui uma probabilidade $\mathcal{P}(\mathcal{I}_i)$ de ser sorteado. Após \mathcal{P} sorteios, o indivíduo \mathcal{I}_i é sorteado \mathcal{P}_i vezes. No limite em que \mathcal{P} torna-se muito grande, tem-se

$${}_{\mathcal{P}}\langle \mathcal{A} \rangle = \lim_{\mathcal{P} \rightarrow \infty} \langle \mathcal{A} \rangle \quad (3.18)$$

$$= \lim_{\mathcal{P} \rightarrow \infty} \frac{\sum_{j=1}^{\mathcal{P}} \mathcal{A}_j}{\mathcal{P}} = \lim_{\mathcal{P} \rightarrow \infty} \sum_{i=1}^{2^n} \mathcal{A}_i \cdot \frac{\mathcal{P}_i}{\mathcal{P}} \quad (3.19)$$

$$= \sum_{i=1}^{2^n} \mathcal{A}_i \cdot \mathcal{P}(\mathcal{I}_i). \quad (3.20)$$

Em cada geração do FPBIL, tem-se $\langle \mathcal{A} \rangle \approx {}_{\mathcal{P}}\langle \mathcal{A} \rangle$ e $\tilde{\mathcal{H}}_n$ é aproximadamente dividido em duas regiões: a de indivíduos com aptidão superior a $\langle \mathcal{A} \rangle$ e a de indivíduos com aptidão inferior a $\langle \mathcal{A} \rangle$. Cada região é representada por um ponto: ${}^r\langle \mathcal{I}_i \rangle_{\mathcal{A}_i > \langle \mathcal{A} \rangle}$, ${}^r\langle \mathcal{I}_i \rangle_{\mathcal{A}_i < \langle \mathcal{A} \rangle}$,

para $\mathcal{A}_i > \langle \mathcal{A} \rangle$ e ${}^r \langle \mathcal{I}_i \rangle_{\langle \mathcal{A} \rangle - \mathcal{A}_i}^{\mathcal{A}_i < \langle \mathcal{A} \rangle}$, para $\mathcal{A}_i < \langle \mathcal{A} \rangle$, tais pontos são uma espécie de centros de gravidade de cada região. Então $\mathcal{P}_{\mathcal{G}}$ é deslocado na direção de ${}^r \langle \mathcal{I}_i \rangle_{\langle \mathcal{A} \rangle - \mathcal{A}_i}^{\mathcal{A}_i < \langle \mathcal{A} \rangle}$ para ${}^r \langle \mathcal{I}_i \rangle_{\mathcal{A}_i - \langle \mathcal{A} \rangle}^{\mathcal{A}_i > \langle \mathcal{A} \rangle}$, a direção em que $\langle \mathcal{A} \rangle$ aumenta. Tem-se, aproximadamente¹, que

$$\mathcal{D} \propto \text{grad}_{\mathcal{P}} \langle \mathcal{A} \rangle, \quad (3.21)$$

que resulta, aproximadamente, em

$$\frac{\partial \mathcal{P}_{\mathcal{G}}}{\partial \mathcal{G}} \propto \xi \cdot \text{grad}_{\mathcal{P}} \langle \mathcal{A} \rangle. \quad (3.22)$$

Evidentemente o PBIL segue uma estratégia mais direta; e mais arriscada. A velocidade seguida pelo PBIL pode variar muito de direção até encontrar um indivíduo bom o suficiente e, então, segue direto até ele, correndo um grande risco de ficar “aprisionado” em um ótimo local (convergência prematura). Comparada à equação (3.15) na página 18, a velocidade do FPBIL está sujeita a menores variações em sua direção, já que é muito menos provável que $\langle \mathcal{A} \rangle$ varie desordenadamente de geração a geração, se comparada ao par \mathcal{I}^+ e \mathcal{I}^- . Da mesma forma, é menos provável que o FPBIL convirja prematuramente, uma vez que teria menos chance de chegar a uma região de ótimo local.

Com relação a ξ , pode-se verificar que este exerce função semelhante a de α ou β , relacionada com a intensidade do deslocamento sofrido por \mathcal{P} . Enquanto α ou β são constantes, ξ varia de acordo com a distribuição de aptidões de cada geração. Para ser mais preciso, ξ é a metade do desvio absoluto médio, relativo à média, das

¹É importante enfatizar que esta aproximação é um pouco exagerada, uma vez que não há prova formal, mas é útil para uma melhor compreensão do método.

aptidões:

$$\xi = \frac{1}{\mathcal{P}} \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \frac{\mathcal{A}_i - \langle \mathcal{A} \rangle}{\langle \mathcal{A} \rangle} \quad (3.23)$$

$$= \frac{1}{\langle \mathcal{A} \rangle} \frac{1}{\mathcal{P}} \sum_{\mathcal{A}_i > \langle \mathcal{A} \rangle} \mathcal{A}_i - \langle \mathcal{A} \rangle \quad (3.24)$$

$$= \frac{1}{2} \frac{1}{\langle \mathcal{A} \rangle} \left(\frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} |\mathcal{A}_i - \langle \mathcal{A} \rangle| \right) \quad (3.25)$$

$$= \frac{1}{2} \frac{\delta}{\langle \mathcal{A} \rangle} = \frac{1}{2} \delta_r \quad (3.26)$$

O desvio absoluto médio (δ) é uma medida de dispersão de uma distribuição, da mesma forma que o desvio padrão. δ_r é somente uma maneira de expressar a mesma dispersão em relação à média.

Deste modo, ξ apresenta um comportamento muito interessante. Considere, a seguir, 4 diferentes cenários²:

1. Todos os indivíduos da população são ruins ($\delta_r = \delta_r^1$);
2. Todos os indivíduos da população são bons ($\delta_r = \delta_r^2$);
3. Os indivíduos da população são, em média, ruins, mas existem uns poucos bons ($\delta_r = \delta_r^3$);
4. Os indivíduos da população são, em média, bons, mas existem uns poucos muito bons ($\delta_r = \delta_r^4$).

O que normalmente ocorre é que $\delta_r^3 > \delta_r^4 > \delta_r^1 > \delta_r^2$, o que é muito apropriado. No início de uma execução do FPBIL, geralmente os indivíduos possuem uma aptidão muito ruim. Enquanto não surge nenhum indivíduo que se destaque (cenário 1), ξ é pequeno — o algoritmo não se arrisca a tomar uma decisão sobre qual direção seguir.

²Deve estar claro que a intenção aqui não é a de listar exaustivamente os possíveis cenários, mas de destacar uns poucos cenários que marcam fases importantes da evolução da aptidão no algoritmo.

Quando surgem os primeiros bons indivíduos (cenário 3), ξ aumenta consideravelmente. Conforme a média das aptidões sobe (cenário 4), ξ diminui gradativamente — é a hora de se preocupar com a convergência prematura. Finalmente quando a solução ótima está próxima (cenário 2), ξ se torna bem pequeno, certificando-se que não haverá oscilação em torno da solução ótima, mas que esta será alcançada.

É óbvio que este não somente é o transcorrer ideal do FPBIL, mas é, também, o mais provável. Neste ponto, vale a pena ressaltar que a intensidade de cada passo $\|\partial_{\mathcal{G}}\mathcal{P}_{\mathcal{G}}\|$ não depende apenas de ξ , mas também do módulo de \mathcal{D} .

Utilizando-se as equações (3.21) na página 20 e (3.26) na página anterior, nos permite reescrever a equação (3.11) na página 18 como

$$\mathcal{P}'_{\mathcal{G}+1} \approx \mathcal{P}_{\mathcal{G}} + \frac{1}{2} \vartheta(\mathcal{P}_{\mathcal{G}}) \cdot \delta_r \cdot \text{grad}_{\mathcal{P}_{\mathcal{G}}}\langle \mathcal{A} \rangle, \quad (3.27)$$

que exibe toda a profundidade dos mecanismos implícitos na equação (3.3) na página 16, onde $\vartheta(\mathcal{P}_{\mathcal{G}})$ é uma função (positiva) que torna $\|\mathcal{D}\| \approx \|\vartheta(\mathcal{P}_{\mathcal{G}}) \cdot \text{grad}_{\mathcal{P}_{\mathcal{G}}}\langle \mathcal{A} \rangle\|$, possivelmente dependente da distribuição de probabilidades $\mathcal{P}_{\mathcal{G}}$.

Combinando as características de ξ e \mathcal{D} , o FPBIL segue uma estratégia que merece, no mínimo, alguma atenção. Esta é a linha 16 da figura 3.1 na página 15. Mas existem outros detalhes do FPBIL que devem ser explanados.

3.3 Eliminando os Parâmetros $P_{\mathcal{M}}$ e γ

Um dos cuidados que devem ser tomados no ajuste dos parâmetros α e β do PBIL original é não escolher valores muito próximos de 0, tampouco valores próximos de 1. No primeiro caso o algoritmo é desnecessariamente demorado, enquanto que no segundo a convergência é prematura. Por mais eficiente que seja o ajuste, entretanto, ainda é necessária a ação da mutação para minimizar ainda mais a chance de convergência prematura, além de aumentar o poder de busca.

A função da mutação é dar uma “segunda oportunidade” às componentes de \mathcal{P}

que atingem os valores 0 ou 1 equivocadamente. O PBIL original realiza esta tarefa probabilisticamente (de acordo com $P_{\mathcal{M}}$) através de deslocamentos (proporcionais a γ) aleatórios ($D_{\mathcal{M}}$, que é 0 ou 1) nas componentes de \mathcal{P} .

O FPBIL segue uma estratégia mais direta, explorando o significado do vetor de probabilidades. Em primeiro lugar, o algoritmo impede que qualquer componente de \mathcal{P} atinja os valores 0 ou 1. Deste modo sempre é possível o sorteio de qualquer indivíduo de $\tilde{\mathcal{H}}_n$. Isto é feito restringindo-se cada componente p_k de \mathcal{P} ao intervalo $[d, 1 - d]$. Como consequência, a probabilidade de se sortear qualquer indivíduo, a partir de \mathcal{P} estará sempre entre d^n e $(1 - d)^n$.

Surge, então, a questão de que valor d é mais apropriado. $d = 0$ é o mesmo que não ter mutação e $d = 0,5$ transforma o FPBIL em uma busca aleatória. Para solucionar essa questão será feita a seguinte hipótese: Das n componentes de $\mathcal{P}'_{\mathcal{G}+1}$, as c que satisfazem $p_k \leq d$ (ou $p_k \geq 1 - d$) estão no caminho correto para atingir a melhor solução—significando que $p_k \leq d \Rightarrow \mathcal{I}^\dagger[k] = 0$ (e que $p_k \geq 1 - d \Rightarrow \mathcal{I}^\dagger[k] = 1$). d será escolhido de forma a maximizar a probabilidade p^1 de sorteio de um indivíduo com as correspondentes c componentes corretas

$$p^1 \approx (1 - d)^c \quad (3.28)$$

ao mesmo tempo em que maximiza a probabilidade p^2 desse mesmo indivíduo possuir uma outra componente correta a partir da probabilidade (componente correspondente em $\mathcal{P}'_{\mathcal{G}+1}$) mais desfavorável. Na pior das hipóteses, existirá uma componente l tal que $\mathcal{I}^\dagger[l] = 1$ com $p_l \gtrsim d$ (ou $\mathcal{I}^\dagger[l] = 0$ com $p_l \lesssim 1 - d$) de modo que

$$p^2 \approx d. \quad (3.29)$$

A probabilidade final que se deseja maximizar é, portanto,

$$p(d) = p^1 p^2 \approx d(1 - d)^c, \quad (3.30)$$

bastando, para isso, encontrar a solução para $p'(d_c) = 0$:

$$p'(d_c) \approx (1 - d_c)^c - cd_c(1 - d_c)^{c-1} = 0 \quad (3.31)$$

$$\Rightarrow 1 - d_c \approx cd_c \quad (3.32)$$

$$\therefore d_c \approx \frac{1}{1 + c}. \quad (3.33)$$

É simples; porém paradoxal quando se tenta estimar c . Saber c exatamente significa saber qual é o indivíduo ótimo de antemão. A solução adotada é tal que os valores de d são construídos iterativamente.

O algoritmo de mutação do FPBIL inicialmente adota $d = d_2 = 1/3$ — o maior valor de d_c diferente de 0,5. Após a atualização do vetor de probabilidades, verifica-se quantas (c) componentes de $\mathcal{P}'_{\mathcal{G}+1}$ satisfazem $p_k \leq d_2$ (ou $p_k \geq 1 - d_2$). Se $c \geq 3$, d torna-se $d_3 = 1/4$. Se $d = d_3$ e $c \geq 4$ (o número de componentes de $\mathcal{P}'_{\mathcal{G}+1}$ que satisfazem $p_k \leq d_3$ (ou $p_k \geq 1 - d_3$)), d torna-se $d_4 = 1/5$, e assim sucessivamente. Deste modo, é possível diminuir o valor de d gradualmente, à medida que \mathcal{P} aproxima-se de algum indivíduo de $\check{\mathcal{H}}_n$ — \mathcal{J}^\dagger , de preferência.

Mas existe um mecanismo que permite d aumentar também. Se, por exemplo, $d = d_5 = 1/6$ mas $c \not\geq 6$, verifica-se quantas (c') componentes de $\mathcal{P}'_{\mathcal{G}+1}$ satisfazem $p_k \leq d_4$ (ou $p_k \geq 1 - d_4$). Se $c' < 5$, d torna-se $d_4 = 1/5$. Se $d = d_4$, $c \not\geq 5$ e $c' < 4$ (o número de componentes de $\mathcal{P}'_{\mathcal{G}+1}$ que satisfazem $p_k \leq d_3$ (ou $p_k \geq 1 - d_3$)), d torna-se $d_3 = 1/4$, e assim por diante, até que d atinja o valor $d_2 = 1/3$, o maior permitido.

Após as contagens de c e c' e a atualização do valor de d , este encontra a sua real função: trazer de volta a d (ou a $1 - d$) quaisquer componentes de $\mathcal{P}'_{\mathcal{G}+1}$ menores que d (ou maiores que $1 - d$), transformando $\mathcal{P}'_{\mathcal{G}+1}$ em $\mathcal{P}_{\mathcal{G}+1}$. O procedimento da mutação é ilustrado em diferentes situações nas figuras 3.2 a 3.5.

Como pode ser observado, os vários valores de d funcionam como “portas” que se abrem ou fecham dependendo das quantidades de pontos em “área proibida” (cinza

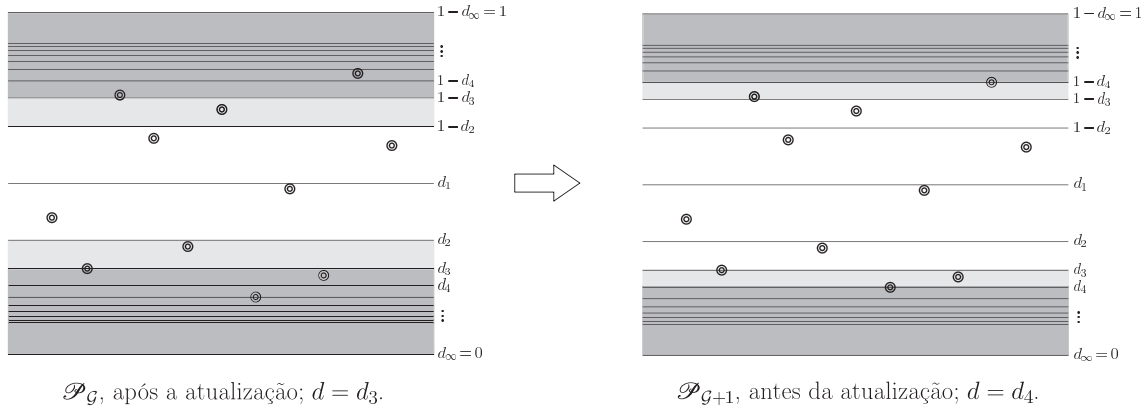


Figura 3.2: Mutaç o 1: Cada ponto \bullet da figura representa uma componente de \mathcal{P} . Nesta situaç o, $d = d_3$ e, ap s a atualizaç o de \mathcal{P} , existem mais de 3 pontos na “ rea proibida” ( rea cinza escuro), o que permite a abertura da “porta 3” (d se torna d_4). Em seguida, todos os pontos que ainda permanecem na “ rea proibida” (dois pontos, neste exemplo) s o recuados para a nova fronteira (correspondente a $d = d_4$).

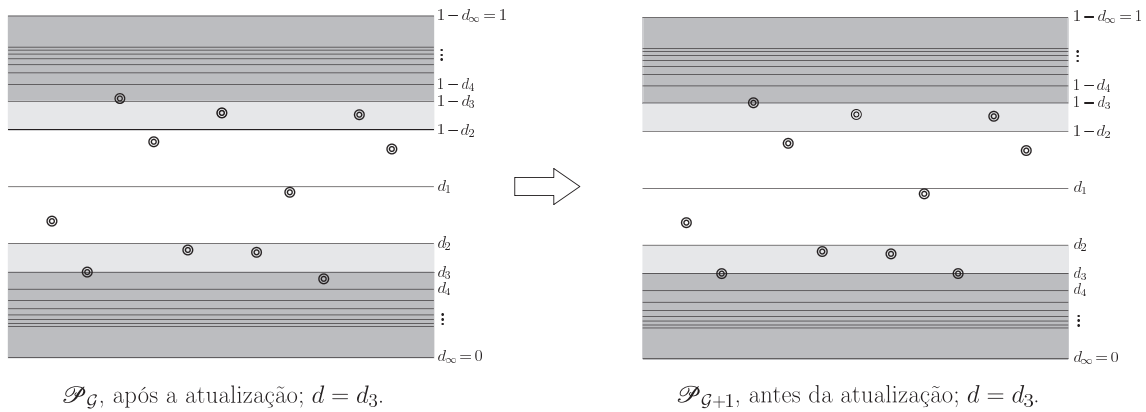


Figura 3.3: Mutaç o 2: Nesta situaç o, $d = d_3$ e, ap s a atualizaç o de \mathcal{P} , existem apenas 3 pontos na “ rea proibida” ( rea cinza escuro), o que n o   suficiente para a abertura da “porta 3”. Em compensa o, existem 7 pontos em  reas cinzas, o que permite que a “porta 2” permaneça aberta. Os pontos na “ rea proibida” s o recuados para a fronteira.

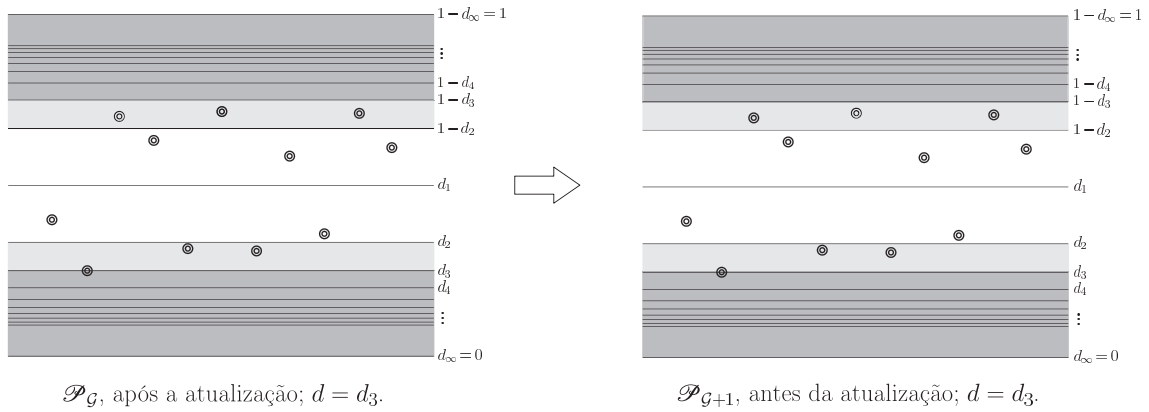


Figura 3.4: Mutaç o 3: Nesta situaç o, $d = d_3$ e, ap s a atualizaç o de \mathcal{P} , apenas um ponto toca a fronteira da “ rea proibida”. Os 6 pontos em  reas cinzas permitem que a “porta 2” permaneça aberta. Como o  nico ponto em “ rea proibida” encontra-se na fronteira, \mathcal{P} n o sofre modificaç o.

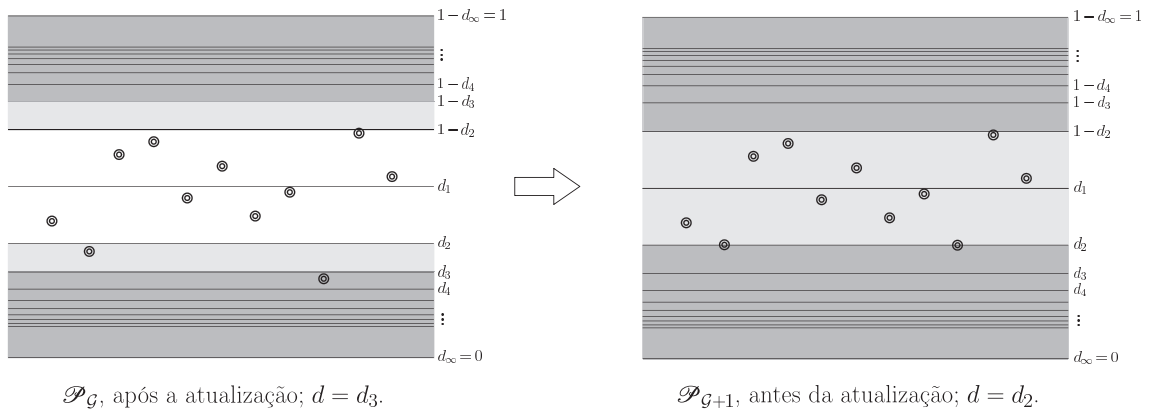


Figura 3.5: Mutaç o 4: Nesta situaç o, $d = d_3$ e, ap s a atualizaç o de \mathcal{P} , o  nico ponto na “ rea proibida” ( rea cinza escuro) n o   suficiente para a abrir a “porta 3”. Os 2 pontos em  reas cinzas n o s o suficientes para manter a “porta 2” aberta. A “porta 2”   fechada (d se torna d_2) e todos os pontos em  reas cinzas s o recuados para a nova fronteira (correspondente a $d = d_2$).

escuro) — c — e de pontos em “área proibida” e “área próxima” (cinza escuro e cinza claro) — c' . Estas são as linhas 17–24 da figura 3.1 na página 15.

Até este ponto o FPBIL é capaz de atingir resultados superiores ao do PBIL original, para \mathcal{P} suficientemente grande. Mas ainda é possível eliminar este último parâmetro e ao mesmo tempo tornar o FPBIL mais eficiente. Essa possibilidade será explorada na seção a seguir.

3.4 Eliminando o Parâmetro \mathcal{P}

A escolha do tamanho da população está relacionada à dificuldade do problema em mãos; um problema simples não necessita de uma população grande.

Existem vários aspectos de um problema que o torna mais difícil que outros. Alguns aspectos intrínsecos, como tamanho e topologia do espaço de busca \mathcal{B} ; outros introduzidos quando se utiliza o PBIL, como a escolha de $\mathcal{M}_{\mathcal{S}_{\mathcal{B}}}^n$.

O número total de elementos em $\check{\mathcal{H}}_n$ é 2^n , o que significa que mesmo existindo um computador que possa avaliar 10^{10} indivíduos por segundo (capacidade muito além da existente atualmente, para a maioria das aplicações práticas), uma busca exaustiva de $\check{\mathcal{H}}_n$ para $n \gtrsim 60$, apenas, necessitaria de, pelo menos, 100 anos, tempo que só os mais pacientes (e saudáveis) seriam capazes de esperar. O que o PBIL faz é direcionar a busca utilizando uma minúscula fração dos elementos de $\check{\mathcal{H}}_n$ em cada geração. Portanto é razoável empregar a relação

$$\mathcal{P} = 2^{\frac{n}{w}} \tag{3.34}$$

com $w > 1$.

Uma característica interessante do PBIL é que, diferentemente de outros algoritmos baseados em populações, torna-se muito simples modificar o tamanho da população em qualquer geração, bastando, para isso, modificar o número de indivíduos sorteados à partir de \mathcal{P} . Isso nos permite melhorar a eficiência do FPBIL.

Conforme as componentes de \mathcal{P} se aproximam de d (ou $1 - d$), a necessidade de manter o mesmo tamanho da população diminui. Após c componentes de \mathcal{P} atingirem esta condição, seguindo a lógica da equação (3.34) na página precedente, seria necessária uma população de apenas $2^{\frac{n-c}{w}}$ indivíduos para dar conta das $n - c$ componentes restantes. Entretanto, este número deve ser multiplicado por um fator f que garanta a reprodutibilidade das c componentes de \mathcal{P} , consideradas corretas.

A probabilidade de sortear um indivíduo com c componentes “corretas” é $(1 - d)^c$, o que significa que, em média, são necessários $1/(1 - d)^c$ sorteios para garantir a presença de um exemplar de tal indivíduo, mas às vezes é necessário um número maior de sorteios ($f = k \cdot 1/(1 - d)^c$, com $k > 1$).

O tamanho da população no FPBIL, em cada geração, depende de c e possui a seguinte expressão:

$$\mathcal{P}_c = \frac{k}{(1 - d)^c} \cdot 2^{\frac{n-c}{w}}. \quad (3.35)$$

A escolha do valor de k será examinada mais adiante.

O valor de d , em qualquer geração será $d_c = 1/(1 + c)$ e, portanto, a equação (3.35) pode ser reescrita como

$$\mathcal{P}_c = k \left(1 + \frac{1}{c}\right)^c 2^{\frac{n-c}{w}}. \quad (3.36)$$

A função $\epsilon(c) = (1 + 1/c)^c$ é interessante pois $\lim_{c \rightarrow 0} \epsilon(c) = 1$, $\lim_{c \rightarrow \infty} \epsilon(c) = e$ e $1 < \epsilon(c) < e$ para $0 < c < \infty$. Isto nos permite relacionar w com \mathcal{P}_0 simplesmente por $\mathcal{P}_0 = k \cdot 2^{\frac{n}{w}}$. Eliminando w e rearrumando a equação (3.36), chega-se, finalmente, a

$$\mathcal{P}_c = \epsilon(c) \cdot \mathcal{P}_0 \left(\frac{\mathcal{P}_0}{k}\right)^{-\frac{c}{n}}. \quad (3.37)$$

A curva de \mathcal{P}_c é esboçada na figura 3.6 na página seguinte, juntamente com a curva de $\epsilon(c)$ e com a curva que expressa o comportamento limite de \mathcal{P}_c : $e\mathcal{P}_0 (\mathcal{P}_0/k)^{-\frac{c}{n}}$.

Com relação a k , verifica-se experimentalmente, como na figura 3.7 na próxima página, que a distribuição da quantidade N de sorteios necessária para se conseguir

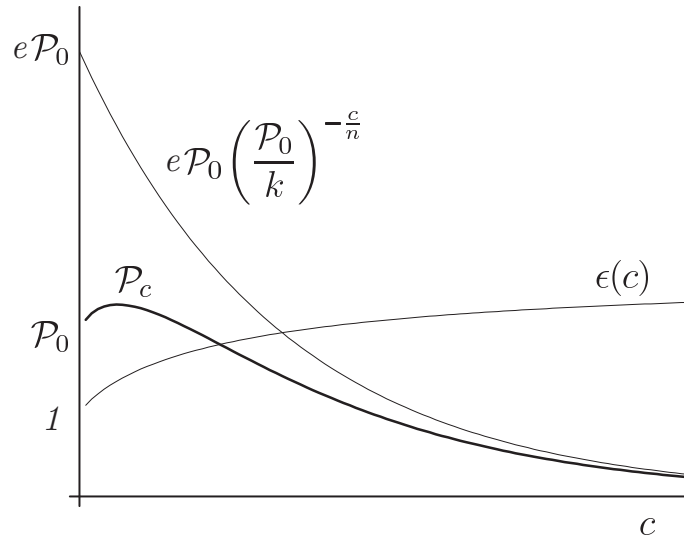


Figura 3.6: Comportamento de \mathcal{P}_c .

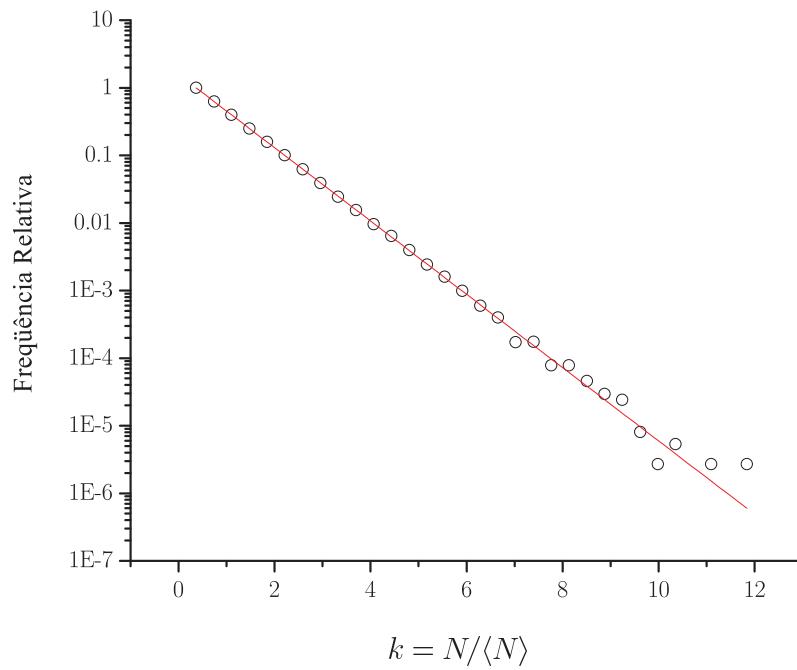


Figura 3.7: Distribuição de N .

um indivíduo com c componentes “corretas”, é uma exponencial decrescente, o que significa que a densidade de probabilidade $\rho(k)$ de se sortear um indivíduo nessas condições é, também, uma exponencial decrescente:

$$\rho(k) = ae^{-bk}, \quad (3.38)$$

onde $k = N/\langle N \rangle$ e $\langle N \rangle = 1/(1-d)^c$ é o número de sorteios, em média, necessários para gerar tal indivíduo.³ Resolvendo-se as equações

$$\int_0^{\infty} \rho(k) dk = 1; \quad (3.39)$$

$$\langle k \rangle = \int_0^{\infty} k \cdot \rho(k) dk = \left\langle \frac{N}{\langle N \rangle} \right\rangle = 1, \quad (3.40)$$

chega-se à conclusão que $a = b = 1$ e, portanto, a variância de $\rho(k)$ é

$$\sigma^2 = \int_0^{\infty} (k-1)^2 \cdot \rho(k) dk \quad (3.41)$$

$$= \int_0^{\infty} k^2 \cdot \rho(k) dk - 2 \int_0^{\infty} k \cdot \rho(k) dk + \int_0^{\infty} \rho(k) dk \quad (3.42)$$

$$= \int_0^{\infty} k^2 e^{-k} dk - 2 + 1. \quad (3.43)$$

Esta última integral pode ser resolvida por partes:

$$\int_0^{\infty} k^2 e^{-k} dk = -k^2 e^{-k} \Big|_0^{\infty} + 2 \int_0^{\infty} k e^{-k} dk \quad (3.44)$$

$$= 0 + 2, \quad (3.45)$$

o que significa que o desvio padrão σ também é 1.

Agora considere a integral

$$P(k) = \int_0^k \rho(\zeta) d\zeta. \quad (3.46)$$

Ela representa a probabilidade de se sortear um indivíduo com c componentes “cor-

³Apesar do número N ser um inteiro na prática, nada impede de se trabalhar com a variável contínua k e deixar o arredondamento para o final. Isso justifica o uso de densidades de probabilidade e integrais. O mesmo raciocínio se aplica ao cálculo de \mathcal{P}_c .

retas” em $N = k \langle N \rangle = k/(1 - d)^c$ tentativas — N é o fator que multiplica $2^{\frac{n-c}{w}}$ na equação (3.35) na página 28. Quanto maior o valor de k , menor a chance de falha. A tabela 3.1 mostra alguns valores de $P(k)$ para os primeiros múltiplos do desvio padrão σ .

Tabela 3.1: Valores de $P(k)$ para os 10 primeiros múltiplos de σ .

m	$P(m\sigma)$
1	0,6321205588
2	0,8646647168
3	0,9502129316
4	0,9816843611
5	0,9932620530
6	0,9975212478
7	0,9990881180
8	0,9996645374
9	0,9998765902
10	0,9999546001

Pode-se perceber que para $k = 7\sigma = 7$, a probabilidade de sortear um indivíduo com c componentes “corretas” é aproximadamente 99,9%. Portanto 7, o número da perfeição, será o valor padrão de k . Esta é a linha 09 da figura 3.1 na página 15.

Até agora, conseguimos aumentar a eficiência do FPBIL reduzindo o número de avaliações desnecessárias, mas ainda temos que lidar com \mathcal{P}_0 , que é agora o responsável por fornecer poder de busca ao FPBIL. Quanto mais difícil o problema a ser resolvido, maior este parâmetro deverá ser. Avaliar exatamente a dificuldade de um determinado problema e relacionar cada nível de dificuldade a um \mathcal{P}_0 específico não é uma tarefa fácil.

Primeiramente deve haver um equilíbrio entre o sucesso do algoritmo e o número de avaliações da aptidão — \mathcal{P}_0 pode ser superestimado, fazendo com que o algoritmo tenha sucesso, embora com um número de avaliações da aptidão desnecessariamente grande. O valor ótimo de \mathcal{P}_0 é o que possibilita que uma solução satisfatória seja encontrada com o mínimo de avaliações da aptidão; e a única forma de conhecer

esse valor exatamente é experimentalmente.

A solução adotada pelo FPBIL é bem genérica e fundamentada na seguinte observação. Em problemas simples, o número de componentes c de \mathcal{P} consideradas corretas cresce rapidamente e logo se aproxima de n . Quando o problema é mais difícil, c cresce mais lentamente, sofrendo flutuações. O FPBIL simplesmente associa o número de flutuações de c à dificuldade de um problema.

Neste ponto torna-se necessária uma definição precisa do que será chamado de flutuação em c . Será computada uma flutuação em c sempre que este não crescer ou decrescer diretamente, ou seja, sempre que c , como função de \mathcal{G} atingir um ponto de máximo, de mínimo ou simplesmente permanecer constante.

O FPBIL inicia com o menor \mathcal{P}_0 possível: $\mathcal{P}_0 = \mathcal{P}_n$, e, conforme as gerações passam, \mathcal{P}_0 é incrementado em 1 à cada flutuação sofrida por \mathcal{P} .

Pode parecer arriscado iniciar a busca com “o menor \mathcal{P}_0 possível”; também pode parecer pouco o incremento de \mathcal{P}_0 em apenas 1, contudo, observe a figura 3.8. Ela

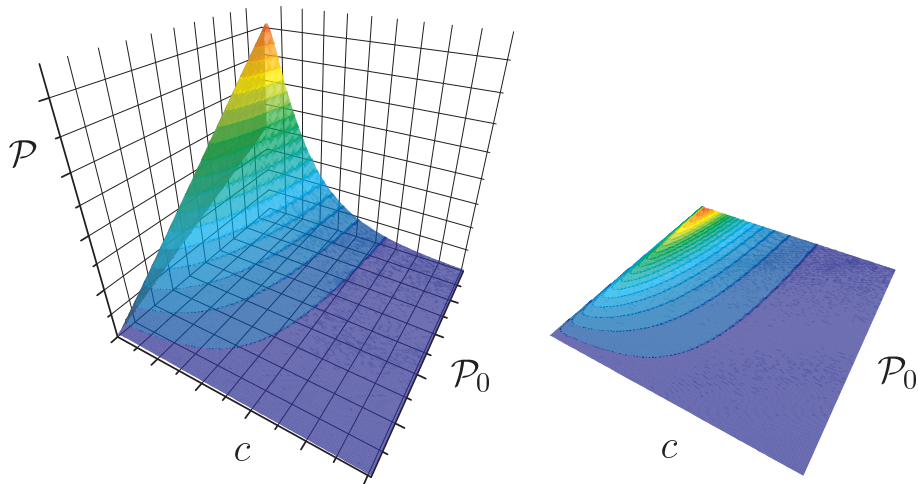


Figura 3.8: \mathcal{P} como função de c e \mathcal{P}_0 . Do lado direito são mostradas as curvas de nível correspondentes à superfície de \mathcal{P} do lado esquerdo, o que ajuda a visualizar que \mathcal{P} atravessa um máximo, quando c é pequeno.

mostra o comportamento de \mathcal{P} conforme c e \mathcal{P}_0 variam. A dependência de \mathcal{P} com c já foi evidenciada na figura 3.6 na página 29. Para facilitar a visualização de como

\mathcal{P} varia com \mathcal{P}_0 , podemos reescrever a equação (3.37) na página 28 como

$$\mathcal{P}_c(\mathcal{P}_0) = \epsilon(c)k^{\frac{c}{n}} \cdot \mathcal{P}_0^{\frac{n-c}{n}}, \quad (3.47)$$

ou seja, $\mathcal{P}_c(\mathcal{P}_0)$ varia como uma potência fracionária de \mathcal{P}_0 . O menor valor que o FPBIL admite para c é 2 (pois o maior valor permitido de d é $1/3$). O comportamento de \mathcal{P} , para valores pequenos de c , é próximo, então, de

$$\mathcal{P}_2(\mathcal{P}_0) = \epsilon(2)k^{\frac{2}{n}} \cdot \mathcal{P}_0^{\frac{n-2}{n}}, \quad (3.48)$$

que para $n \gg 2$ se aproxima da reta

$$\mathcal{P}(\mathcal{P}_0) = \lim_{n \rightarrow \infty} \mathcal{P}_2(\mathcal{P}_0) = \frac{9}{4}\mathcal{P}_0. \quad (3.49)$$

O outro extremo é quando $c \approx n$, quando $\mathcal{P}_c(\mathcal{P}_0) \approx k \cdot \epsilon(n) = \mathcal{P}_n$, independentemente de \mathcal{P}_0 . Para qualquer outro valor de c , $\mathcal{P}_c(\mathcal{P}_0)$ varia como uma raiz qualquer. Por exemplo, quando $c = \frac{2}{3}n$, $\mathcal{P}_{\frac{2}{3}n}(\mathcal{P}_0) = \epsilon(\frac{2}{3}n)k^{\frac{2}{3}} \cdot \sqrt[3]{\mathcal{P}_0}$ — uma raiz cúbica.

Quando c é pequeno, independentemente da geração corrente, significa que \mathcal{P} está próximo de \mathcal{P}_0 e o FPBIL ainda não tomou uma decisão firme de qual direção seguir. Portanto, quanto mais tempo (gerações) permanecer esta situação, maior o número de flutuações sofridas por c e maior a taxa com que \mathcal{P} cresce ($\mathcal{P} \approx \frac{9}{4}\mathcal{P}_0$) — o FPBIL está procurando o parâmetro \mathcal{P}_0 de acordo com a dificuldade do problema.

Imagine agora um ponto qualquer na superfície $\mathcal{P}_c(\mathcal{P}_0)$ da figura 3.8 na página anterior. Cinco situações podem acontecer:

c pode crescer diretamente Neste caso, não existe flutuação em c e, portanto, \mathcal{P}_0 não varia. Como consequência \mathcal{P} decresce exponencialmente (base \mathcal{P}_0) rumo à solução. Este é o caso ideal.

c pode sofrer flutuações sem, em média, variar Neste caso apenas \mathcal{P}_0 cresce. Como consequência \mathcal{P} cresce como uma raiz qualquer de \mathcal{P}_0 . Uma população maior ajudará o FPBIL a decidir que direção \mathcal{P} deverá seguir. Entretanto, se

esta situação se mantiver, pode significar que o algoritmo convergiu prematuramente.

c pode decrescer diretamente Neste caso também não existe flutuação em c . Como consequência \mathcal{P} cresce exponencialmente (base \mathcal{P}_0). O FPBIL está retornando de um caminho errado e uma população crescendo nesta taxa ajudará o FPBIL a corrigir o erro. Evidentemente, esta situação não perdura e acaba por cair na próxima situação.

c pode decrescer em média, sofrendo flutuações Neste caso, c decresce enquanto \mathcal{P}_0 cresce. Como consequência \mathcal{P} continua crescendo.

c pode crescer em média, sofrendo flutuações Neste último caso, c e \mathcal{P}_0 crescem. O que acontece com \mathcal{P} vai depender dos valores de c e \mathcal{P}_0 e da frequência das flutuações em c .

Como pode ser observado, o único risco que ocorre é se c crescer com \mathcal{P} indo na direção errada. Entretanto, o FPBIL possui mecanismos para minimizar essa possibilidade, a começar pela mutação que impede que c cresça desenfreadamente, mesmo para pequenas populações. Entretanto, não existe como garantir que o algoritmo, até este ponto, não convirja prematuramente.

O último recurso do FPBIL é retornar com \mathcal{P} para o centro de \mathcal{H}_n e com d para $d_2 = 1/3$ sempre que for detectada a convergência prematura.

Considere

$$\langle c \rangle_{\mathcal{G}} = \frac{1}{\mathcal{G} - \mathcal{G}_i + 1} \sum_{k=\mathcal{G}_i}^{\mathcal{G}} c_k, \quad (3.50)$$

que é a média dos valores de c desde a geração \mathcal{G}_i até a geração corrente \mathcal{G} . \mathcal{G}_i é a geração em que \mathcal{P} e d são reinicializados. Inicialmente $\mathcal{G}_i = 0$ e $\langle c \rangle_{\mathcal{G}}$ cresce conforme c cresce. À medida que aumentam as flutuações em c , este cresce mais lentamente, o mesmo acontecendo com $\langle c \rangle_{\mathcal{G}}$. Quando acontece de \mathcal{P} tomar um caminho errado,

c praticamente pára de crescer e o algoritmo pode nunca encontrar uma solução satisfatória.

Baseando-se nisso, o FPBIL monitora constantemente o valor de

$$\Delta\langle c \rangle_{\mathcal{G}} = \langle c \rangle_{\mathcal{G}} - \langle c \rangle_{\mathcal{G}-1}, \quad (3.51)$$

a variação de $\langle c \rangle_{\mathcal{G}}$. Toda vez que $\Delta\langle c \rangle_{\mathcal{G}}$ atinge valores próximos de 0, significa que possivelmente o FPBIL ficou preso em um ótimo local. Embora seja sempre possível escapar, na maioria das vezes a solução procurada é encontrada mais rapidamente reiniciando-se \mathcal{P} e d .

Essa alteração dos valores de \mathcal{P} e d poderia ser considerada uma espécie de mutação, entretanto é radical demais. A real intenção por de trás dessa operação é reiniciar o FPBIL com cada vez maior poder de busca.

Quando, no início de uma geração \mathcal{G}_k qualquer, verifica-se que $\Delta\langle c \rangle_{\mathcal{G}} < 1\%$ (um pouco mais que $\frac{1}{2}^\circ$ de inclinação), o FPBIL é reiniciado com uma importantíssima diferença: \mathcal{P}_0 será maior do que quando o algoritmo inicializou pela última vez. Conseqüentemente, \mathcal{P} dá um salto — graças à combinação de \mathcal{P}_0 grande (inalterado) com c pequeno (agora igual a 2) — fazendo com que seja cada vez menos provável que o FPBIL erre o caminho. \mathcal{G}_i torna-se \mathcal{G}_k e o algoritmo continua — mais forte.

Utiliza-se $\langle c \rangle_{\mathcal{G}}$ — e não simplesmente c , por exemplo — por este variar mais suavemente. Iniciando o FPBIL com $\mathcal{P}_0 = \mathcal{P}_n$ e incrementando \mathcal{P}_0 em 1 à cada flutuação de c são medidas que visam apenas contribuir com a eficiência do algoritmo.

4 *Comparação Entre os Algoritmos*



finalidade deste capítulo é explicitar as diferenças na execução dos algoritmos PBIL original e FPBIL, apresentados nos capítulos 2 e 3. Também serão examinados os algoritmos que serão chamados de PBIL* e FPBIL*.

O PBIL* é simplesmente o PBIL com população igual a 1.000, mantendo o restante dos parâmetros inalterados. O FPBIL* é o FPBIL sem reinicializações e com \mathcal{P}_0 fixo, igual a 1.000.

A comparação entre o FPBIL e o FPBIL*, em particular, mede até que ponto as reinicializações são realmente importantes ou se são um desperdício de computação e o FPBIL, ao invés, deveria adotar a escolha do parâmetro \mathcal{P}_0 .

Esses 4 algoritmos serão testados em 3 problemas de classes diferentes. São eles: O problema quatro picos, que é multimodal, altamente deceptivo e os indivíduos deste não necessitam ser decodificados, já que são avaliados diretamente sobre \mathcal{H}_n . O problema banana, que é um problema clássico de minimização numérica conhecido por dificultar a execução de algoritmos de otimização baseados em gradiente. E o Problema do Caixeiro Viajante (PCV) Rykel48, que é um problema combinatório multimodal que exige um tipo diferente de representação denominado *random keys*, que automaticamente incorpora as restrições impostas pelo problema.

O problema PCV Rykel48 tem uma importância especial neste trabalho, uma vez

que o problema da recarga nuclear, tratado no próximo capítulo, pertence a mesma classe deste e, portanto, as mesmas técnicas e estratégias devem causar respostas semelhantes em ambos.

4.1 Considerações Iniciais

Antes de discutir as peculiaridades de cada problema teste, existem questões comuns que se aplicam a todos os casos as quais serão sucintamente consideradas.

4.1.1 Código de Gray

No FPBIL (e no PBIL), uma região \mathcal{S}_B de interesse do espaço de busca é representada por meio de vetores binários, os elementos de $\check{\mathcal{H}}_n$, significando que a decodificação dos indivíduos — escolha de $\mathcal{M}_{\mathcal{S}_B}^n$ — tem influência sobre o desempenho do algoritmo.

Por exemplo, considere a função aptidão \mathcal{A}_1 definida na tabela 4.1. Pode-se perceber que o indivíduo mais apto é $\mathcal{I}^\dagger = 1000$. Imagine (em uma situação hipotética) que, com a exceção dele, todos os indivíduos são sorteados uma única vez. A partir da equação (3.3) na página 16 conclui-se que $\mathcal{P}' = (0,048; 0,884; 0,694; 0,599)$ e podemos calcular $\mathcal{P}'(\mathcal{I})$, que é a probabilidade do indivíduo \mathcal{I} ser sorteado à partir de \mathcal{P}' , bem como a razão entre $\mathcal{P}'(\mathcal{I})$ e $\mathcal{P}_0(\mathcal{I})$, sendo $\mathcal{P}_0(\mathcal{I})$ a probabilidade de se sortear \mathcal{I} ao acaso.

Compare esses valores com os correspondentes aos da função aptidão \mathcal{A}_2 definida na tabela 4.2 na página 39. Da mesma forma, $\mathcal{I}^\dagger = 1100$ não participa da atualização do vetor de probabilidades e todos os outros indivíduos são sorteados uma única vez resultando em $\mathcal{P}' = (0,048; 0,878; 0,394; 0,476)$.

As duas funções aptidão são evidentemente diferentes. Entretanto, ambas podem ser funções aptidão do mesmo problema: o de encontrar o valor de $x \in [0, 8] \cap \mathbb{N}$ que maximiza a função $f(x) = x^2$ em que, a fim de evitar x fora do seu domínio,

Tabela 4.1: Definição da função aptidão \mathcal{A}_1 e os valores de $\mathcal{P}'(\mathcal{I})$ e de $\mathcal{P}'(\mathcal{I})/\mathcal{P}_0(\mathcal{I})$, onde \mathcal{P}' é o vetor de probabilidades gerado a partir de todos os indivíduos, com exceção de \mathcal{I}^\dagger , admitindo que cada indivíduo é sorteado uma única vez.

\mathcal{I}	$\mathcal{A}_1(\mathcal{I})$	$\mathcal{P}'(\mathcal{I})$	$\mathcal{P}'(\mathcal{I})/\mathcal{P}_0(\mathcal{I})$
0000	0	0,01353	0,216
0001	1	0,02018	0,323
0010	4	0,03067	0,491
0011	9	0,04575	0,732
0100	16	0,10348	1,656
0101	25	0,15435	2,470
0110	36	0,23456	3,753
0111	49	0,34985	5,598
1000	64	0,00068	0,011
1001	1	0,00101	0,016
1010	1	0,00153	0,024
1011	1	0,00229	0,036
1100	1	0,00517	0,083
1101	1	0,00772	0,123
1110	1	0,01173	0,188
1111	1	0,01749	0,280

Tabela 4.2: Definição da função aptidão \mathcal{A}_2 e os valores de $\mathcal{P}'(\mathcal{I})$ e de $\mathcal{P}'(\mathcal{I})/\mathcal{P}_0(\mathcal{I})$, onde \mathcal{P}' é o vetor de probabilidades gerado a partir de todos os indivíduos, com exceção de \mathcal{I}^\dagger , admitindo que cada indivíduo é sorteado uma única vez.

\mathcal{I}	$\mathcal{A}_2(\mathcal{I})$	$\mathcal{P}'(\mathcal{I})$	$\mathcal{P}'(\mathcal{I})/\mathcal{P}_0(\mathcal{I})$
0000	0	0,0370	0,592
0001	1	0,0336	0,538
0011	4	0,0219	0,350
0010	9	0,0241	0,386
0110	16	0,1727	2,764
0111	25	0,1570	2,512
0101	36	0,2410	3,855
0100	49	0,2650	4,241
1100	64	0,0132	0,212
1101	1	0,0120	0,193
1111	1	0,0078	0,126
1110	1	0,0086	0,138
1010	1	0,0012	0,019
1011	1	0,0011	0,018
1001	1	0,0017	0,027
1000	1	0,0018	0,030

penaliza-se f da seguinte forma:

$$f(x) = \begin{cases} x^2, & \text{se } x \leq 8; \\ 1, & \text{se } x > 8. \end{cases} \quad (4.1)$$

A única diferença é que no primeiro caso, $\mathcal{M}_{\mathcal{S}_B}^n$ é a transformação natural de números na base 2 para os correspondentes números na base 10, enquanto que no segundo caso $\mathcal{M}_{\mathcal{S}_B}^n$ transforma a partir de vetores binários utilizando o código de Gray¹ [16]. Essa diferença aparentemente sem importância é a responsável por $\mathcal{P}'(\mathcal{I}^\dagger)$ ser, aproximadamente, 20 vezes maior com a utilização do código de Gray, o que é bem significativo.

O motivo para uma diferença tão acentuada resulta do fato de a função aptidão na tabela 4.1 na página 38 ter sido construída sob medida para falhar. Observe que \mathcal{I}^\dagger difere do segundo melhor indivíduo em todos os bits. A consequência dessa “descontinuidade em $\tilde{\mathcal{H}}_n$ ” é que $\mathcal{P}'(\mathcal{I}^\dagger)$ é menor mesmo que $\mathcal{P}'(\mathcal{I}^-)$, correspondente ao pior indivíduo. A vantagem do código de Gray é que este representa números consecutivos como vetores binários que diferem sempre por apenas um bit.

A seqüência Γ_n do código de Gray com n bits pode ser construída recursivamente a partir de

$$\Gamma_0 = \epsilon; \quad (4.2)$$

$$\Gamma_{k+1} = 0\Gamma_k, 1\Gamma_k^R, \quad (4.3)$$

onde ϵ é a “seqüência” vazia, $0\Gamma_k$ é a seqüência anterior precedida do bit 0 e $1\Gamma_k^R$ é a seqüência anterior, em ordem reversa, precedida do bit 1.

¹O código de Gray recebe seu nome de Frank Gray, um físico que ficou famoso por ajudar a planejar o método de transmissão de sinais de TV a cores. Gray criou o código para transmissão analógica de dados digitais.

Por exemplo, para construir Γ_3 seguem-se os seguintes passos:

$$\Gamma_0 = \epsilon; \tag{4.4}$$

$$\Gamma_1 = 0, 1; \tag{4.5}$$

$$\Gamma_2 = 00, 01, 11, 10; \tag{4.6}$$

$$\Gamma_3 = 000, 001, 011, 010, 110, 111, 101, 100. \tag{4.7}$$

A primeira coluna da tabela 4.2 na página 39 corresponde a Γ_4 que é construída facilmente a partir de Γ_3 acima.

Testes [5] realizados comparando o desempenho do PBIL utilizando a codificação binária natural e o código de Gray mostram que a utilização do código de Gray aumentou o desempenho do algoritmo em 100% dos casos. Assim, todos os problemas tratados neste trabalho utilizarão o código de Gray.

4.1.2 Minimizando o Impacto sobre a Escolha da Aptidão

Embora o FPBIL seja livre de parâmetros, este ainda depende da forma exata da aptidão. Existem várias formas funcionais para \mathcal{A} capazes de determinar a mesma ordem $\mathcal{A}(\mathcal{J}^-) \leq \mathcal{A}(\mathcal{J}_i) \leq \mathcal{A}(\mathcal{J}_j) \leq \dots \leq \mathcal{A}(\mathcal{J}^+)$ e cada uma delas pode gerar \mathcal{D} e ξ diferentes, o que resultaria em comportamentos igualmente diferentes. Considere a análise das equações (3.8) e (3.9) na página 17 em dois exemplos simples.

1. Com a transformação $\mathcal{A}'_i = f \cdot \mathcal{A}_i$ ($f \in \mathbb{R}$), temos que $\mathcal{D}' = \mathcal{D}$ e $\xi' = \xi$, ou seja, a multiplicação da aptidão por um fator constante, não altera em nada o comportamento do FPBIL.
2. Já com a transformação $\mathcal{A}'_i = t + \mathcal{A}_i$ ($t \in \mathbb{R}$); $\mathcal{D}' = \mathcal{D}$, porém

$$\xi' = \frac{\langle \mathcal{A} \rangle}{\langle \mathcal{A} \rangle + t} \xi, \tag{4.8}$$

significando que se $t \gg \langle \mathcal{A} \rangle$ então $\xi' \approx 0$, ou seja, a adição da aptidão com um

termo constante, altera a intensidade dos passos do FPBIL, podendo tornar o FPBIL impraticável para grandes valores de t .

O item 2 sugere que uma boa prática pode ser o emprego da aptidão $\mathcal{A}'_i = \mathcal{A}_i - \mathcal{A}(\mathcal{I}^-)$, garantindo que ξ nunca será menor que o necessário.

A equação (3.26) na página 21 mostra que a escolha de aptidões com valores exageradamente grandes podem resultar em ξ muito próximos de zero, quando $\delta \rightarrow 0$. Isso geralmente ocorre quando \mathcal{P} se aproxima de um vértice de \mathcal{H}_n , atrasando demasiadamente a convergência do FPBIL. Essa observação desencoraja o uso da transformação $\mathcal{A}'_i = (\mathcal{A}_i)^n$ ($n \in \mathbb{N}$) para valores grandes de n , como tentativa de aumentar as contribuições dos indivíduos $\mathcal{A}(\mathcal{I}^-)$ e $\mathcal{A}(\mathcal{I}^+)$ na equação (3.3) na página 16.

Seguindo tais recomendações, adotou-se um procedimento genérico para se construir a aptidão — baseado no procedimento empregado por Koza no algoritmo da programação genética [12] — descrito a seguir.

Chama-se de aptidão bruta, \mathcal{A}_b , a quantidade natural do problema que se deseja maximizar ou minimizar, também chamada de função objetivo.

Por exemplo, se o objetivo de um problema é maximizar uma função $f(x)$,

$$\mathcal{A}_b(\mathcal{I}_i) = f(x(\mathcal{I}_i)), \quad (4.9)$$

com $x(\mathcal{I}_i) = \mathcal{M}_{\mathcal{S}_B}^n(\mathcal{I}_i)$. Se em um outro problema o objetivo é minimizar o tempo de execução de um processo,

$$\mathcal{A}_b(\mathcal{I}_i) = t(\mathcal{I}_i), \quad (4.10)$$

com $t(\mathcal{I}_i) = \mathcal{M}_{\mathcal{S}_B}^n(\mathcal{I}_i)$, onde $t(\mathcal{I}_i)$ é o tempo medido de um processo executado com os parâmetros codificados em \mathcal{I}_i .

A partir da aptidão bruta, constrói-se a aptidão padrão, \mathcal{A}_p , que possui a característica de $0 \leq \mathcal{A}_p(\mathcal{I}_a) < \mathcal{A}_p(\mathcal{I}_b)$ sempre que \mathcal{I}_a é melhor que \mathcal{I}_b , e, de preferência,

com $\mathcal{A}_p(\mathcal{I}^\dagger) = 0$. A aptidão padrão construída a partir da equação (4.9) na página anterior pode ser então

$$\mathcal{A}_p(\mathcal{I}_i) = \mathcal{A}_{\text{máx}} - \mathcal{A}_b(\mathcal{I}_i), \quad (4.11)$$

onde $\mathcal{A}_{\text{máx}}$ é uma estimativa de $\mathcal{A}(\mathcal{I}^\dagger)$. Uma outra possibilidade seria

$$\mathcal{A}_p(\mathcal{I}_i) = \frac{1}{\mathcal{A}_b(\mathcal{I}_i)}, \quad (4.12)$$

desde que $\mathcal{A}_b(\mathcal{I}_i) > 0, \forall i$. Já para a equação (4.10) na página precedente, a aptidão padrão pode ser simplesmente

$$\mathcal{A}_p(\mathcal{I}_i) = \mathcal{A}_b(\mathcal{I}_i). \quad (4.13)$$

Com a aptidão padrão, calcula-se a aptidão ajustada, \mathcal{A}_a , a partir de

$$\mathcal{A}_a(\mathcal{I}_i) = \frac{1}{1 + \mathcal{A}_p(\mathcal{I}_i)}. \quad (4.14)$$

O comportamento da aptidão ajustada é ilustrado na figura 4.1. Pode-se per-

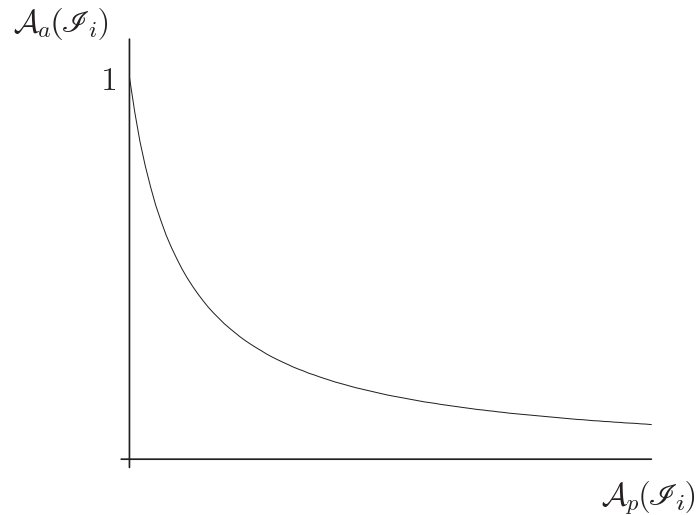


Figura 4.1: Comportamento da aptidão ajustada.

ceber que \mathcal{A}_a possui algumas propriedades interessantes. Em primeiro lugar, \mathcal{A}_a

elimina o problema de ξ chegar muito próximo de zero, pois está limitada a $[0, 1]$, independentemente da escolha de \mathcal{A}_b . Em segundo lugar, a medida que os indivíduos se tornam melhores ($\mathcal{A}_p(\mathcal{S}_i) \rightarrow 0$), a diferença na aptidão de indivíduos próximos torna-se mais acentuada, garantindo competitividade até o final.

Seguindo a recomendação de se ter $\mathcal{A}'_i = \mathcal{A}_i - \mathcal{A}(\mathcal{S}^-)$, a função aptidão utilizada em todos os problemas deste trabalho será

$$\mathcal{A}(\mathcal{S}_i) = \begin{cases} \mathcal{A}_a(\mathcal{S}_i) - \mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-), & \text{se } \mathcal{A}_a(\mathcal{S}_i) - \mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-) \geq 0; \\ 0, & \text{se } \mathcal{A}_a(\mathcal{S}_i) - \mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-) < 0, \end{cases} \quad (4.15)$$

onde $\mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-)$ é a aptidão ajustada do pior indivíduo da geração anterior.

A razão para se utilizar $\mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-)$ é que, para encontrar $\mathcal{A}(\mathcal{S}^-)$, é necessário avaliar todos os indivíduos da geração, o que implica que, para seguir à risca a recomendação de se ter $\mathcal{A}'_i = \mathcal{A}_i - \mathcal{A}(\mathcal{S}^-)$, ou todos os indivíduos devem ser avaliados duas vezes por geração ou todos os indivíduos de uma geração devem ser armazenados em uma estrutura de dados. A solução adotada, além de econômica, não agride demasiadamente a recomendação original, visto que geralmente $\mathcal{A}_a(\mathcal{S}_{\mathcal{G}-1}^-) \approx \mathcal{A}_a(\mathcal{S}^-)$.

4.1.3 Procedimento Geral

Como o FPBIL e o FPBIL* possuem populações de tamanho variável (e o PBIL e o PBIL* não), as comparações não foram feitas de geração em geração, e sim depois de um mesmo número de avaliações da aptidão. O número de avaliações da aptidão após \mathcal{G} gerações é ${}^0\mathcal{P} + {}^1\mathcal{P} + \dots + {}^{\mathcal{G}}\mathcal{P}$, que corresponde à soma dos tamanhos das populações de cada geração.

Todos os algoritmos foram executados até atingir 1.000.000 avaliações da aptidão e, a fim de estimar o melhor valor encontrado por um algoritmo após um determinado número de avaliações da aptidão, os melhores valores ao final de cada geração são computados e, então, por meio de *splines*, o valor no ponto desejado é interpolado.

Em todos os testes, são interpolados 10.000 pontos igualmente espaçados — tanto em escala linear como em logarítmica — a partir dos quais são calculados as médias e os desvios. Adotou-se o critério de que para 0 avaliações da aptidão (1, para a escala logarítmica), o melhor valor correspondente deve ser igual ao melhor valor encontrado ao final da geração 0 — razão pela qual, nos gráficos, os valores para o número de avaliações entre 0 (ou 1) e o tamanho da população da geração 0 praticamente não variam.

Nos problemas banana e quatro picos, as médias e os desvios apresentados nos gráficos são calculados a partir de 100 execuções independentes de cada algoritmo. Para o problema PCV Rykel48, os algoritmos foram executados até que o desvio relativo máximo fosse de 1%.

4.2 Problemas Testes

4.2.1 Quatro Picos

Considere as duas funções definidas em $\check{\mathcal{H}}_{100}$:

$$Z(\mathcal{S}) = \text{número de 0's contíguos em } \mathcal{S} \text{ terminando na posição 100}; \quad (4.16)$$

$$U(\mathcal{S}) = \text{número de 1's contíguos em } \mathcal{S} \text{ iniciando na posição 1}; \quad (4.17)$$

onde, por exemplo, $U(011 \cdots 111) = 0$, $U(111 \cdots 111) = 100$, $Z(111 \cdots 110) = 1$ e $Z(000 \cdots 010) = 1$. Considere também a função prêmio

$$P(Z(\mathcal{S}), U(\mathcal{S}); T) = \begin{cases} 100 + T, & \text{se } Z(\mathcal{S}) \geq T \text{ e } U(\mathcal{S}) \geq T; \\ 0, & \text{caso contrário.} \end{cases} \quad (4.18)$$

definida em $\{0, 1, 2, \dots, 100\}^2 \times \{0, 1, 2, \dots, 50\}$. No problema quatro picos, o objetivo é maximizar a função

$$Q_T(\mathcal{S}) = \text{máx}\{Z(\mathcal{S}), U(\mathcal{S})\} + P(Z(\mathcal{S}), U(\mathcal{S}); T). \quad (4.19)$$

Observando o gráfico de Q_T na figura 4.2, percebe-se que o problema quatro picos é altamente enganoso. Existem duas regiões. Uma premiada, correspondente

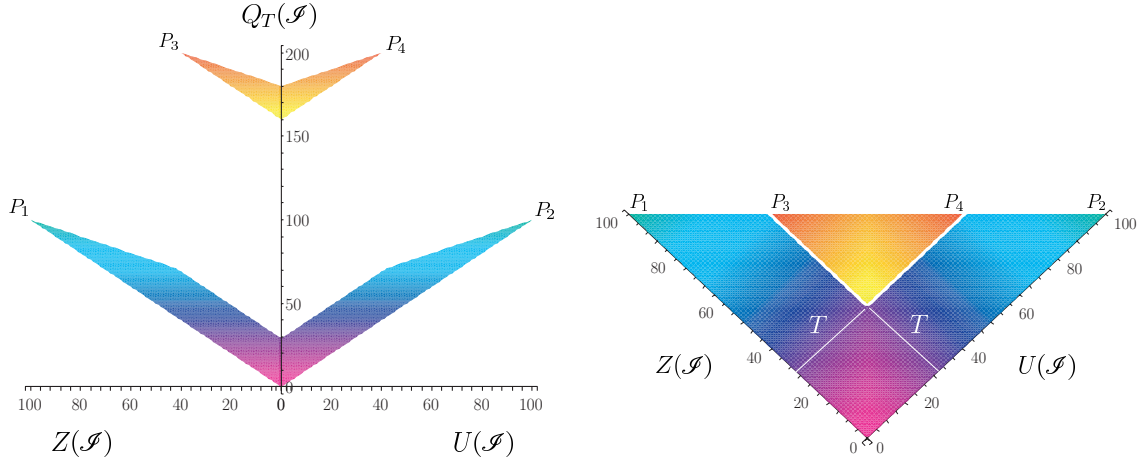


Figura 4.2: Gráfico de $Q_T(\mathcal{S})$, função objetivo do problema quatro picos.

à superfície de cima, e outra não premiada, correspondendo à superfície de baixo. Nenhum ponto da região não premiada (que aumenta com T) fornece qualquer indicação da existência do prêmio, dando a falsa impressão da existência de apenas os picos P_1 e P_2 — que correspondem a $Q_T(\mathcal{S}) = 100$ — enquanto existem ainda os picos P_3 e P_4 — correspondendo a $Q_T(\mathcal{S}) = 200$, os ótimos globais.

No problema quatro picos, existe um total de $2^{100} = 1.267.650.600.228.229.401.496.703.205.376$ possíveis indivíduos. Uma vez que os primeiros indivíduos da região premiada são encontrados, alcançar o pico P_3 (ou P_4) torna-se trivial. A dificuldade do problema quatro picos depende do valor de T — dificuldade esta que aumenta com T . No limite em que $T = 0$, a região não premiada desaparece, juntamente com os picos P_1 e P_2 . No outro extremo, quando $T = 50$, a região premiada colapsa em um único ponto: $P_3 \equiv P_4$. Verifica-se, sem muita dificuldade, que a probabilidade de se sortear ao acaso um indivíduo da região premiada é $1/2^{2T}$, tornando

$$D(T) \equiv 2^{2T} \quad (4.20)$$

uma medida da dificuldade do problema de acordo com o parâmetro T .

Conclui-se, então, que $D(T + 1) = 4 \cdot D(T)$, ou seja, a dificuldade quadruplica quando T é acrescido de 1. Todos os testes do problema quatro picos, realizados neste trabalho tiveram $T = 30$, correspondendo a uma dificuldade 20 vezes maior que a máxima dificuldade utilizada em [7], quando, dentre um total de 25 execuções, o PBIL convergiu prematuramente (para um ótimo local) 20 vezes e os algoritmos genéticos, entre 22 e 25 vezes.

4.2.1.1 Representação

A forma como o problema quatro picos foi definido não permite muita liberdade na escolha de representações distintas. Cada indivíduo é um vetor de 100 bits e, como Q_T é avaliada diretamente sobre $\check{\mathcal{H}}_{100}$, o código de Gray não é aplicável (nem nenhum outro).

4.2.1.2 Aptidões Bruta e Padrão

A aptidão bruta do problema quatro picos utilizada será simplesmente o valor de $Q_T(\mathcal{I})$:

$$\mathcal{A}_b(\mathcal{I}_i) = Q_T(\mathcal{I}_i). \quad (4.21)$$

Como se trata de um problema de maximização e $\mathcal{A}_p(\mathcal{I}^\dagger) = 200$, a aptidão padrão utilizada será

$$\mathcal{A}_p(\mathcal{I}_i) = 200 - Q_T(\mathcal{I}_i). \quad (4.22)$$

4.2.1.3 Resultados

A figura 4.3 na página seguinte mostra o resultado da comparação entre o FPBIL e o PBIL Original, onde estão plotadas as médias das melhores aptidões de cada execução para cada algoritmo. Das 100 execuções, o PBIL não conseguiu alcançar a região premiada em nenhuma delas, enquanto que o FPBIL a alcançou todas, tendo

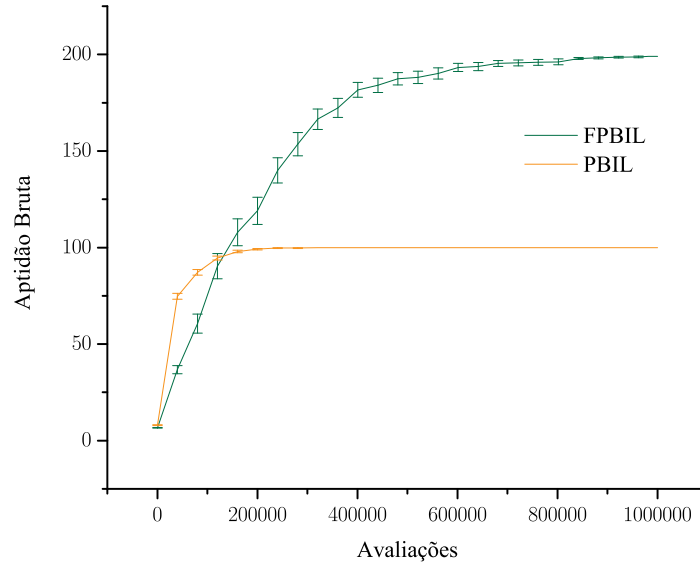


Figura 4.3: Comparação entre o FPBIL e o PBIL.

como pior resultado $\mathcal{A}_b(\mathcal{I}_i) = 178$.

Em seguida, o PBIL* é comparado com o FPBIL*. O resultado é mostrado na figura 4.4 na próxima página, que indica um comportamento semelhante.

Comparando o FPBIL com o FPBIL*, o FPBIL se mostra superior, conforme indica a figura 4.5 na página seguinte. Uma diferença importante entre o FPBIL e o FPBIL* é que o FPBIL* falha em alcançar a região premiada em 4 das 100 execuções, tendo como pior resultado $\mathcal{A}_b(\mathcal{I}_i) = 65$, não atingindo sequer os picos P_1 e P_2 da região não premiada.

O PBIL contra o PBIL*, mostrado na figura 4.6 na página 50 indica que existe pouca diferença entre os dois, embora a figura 4.7 na página 50 indique o curioso fato de que o aumento da população do PBIL não necessariamente implica o aumento do desempenho para um número fixo de avaliações da função aptidão.

No problema quatro picos, a observação da evolução do vetor de probabilidade

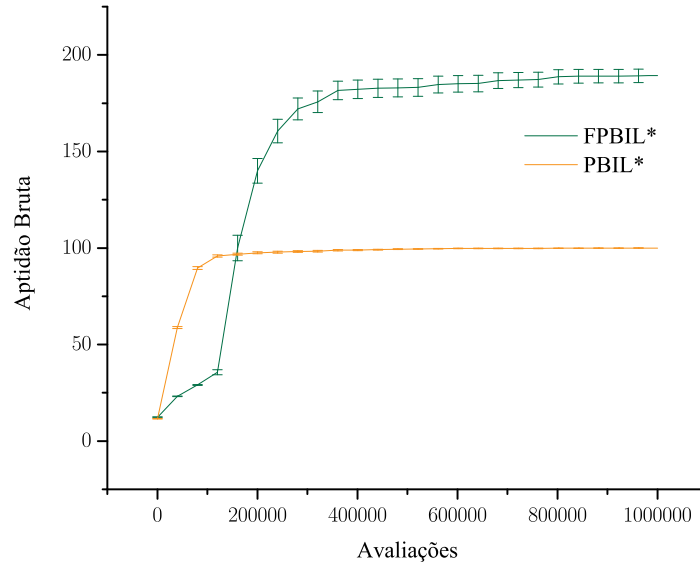


Figura 4.4: Comparação entre o FPBIL* e o PBIL*.

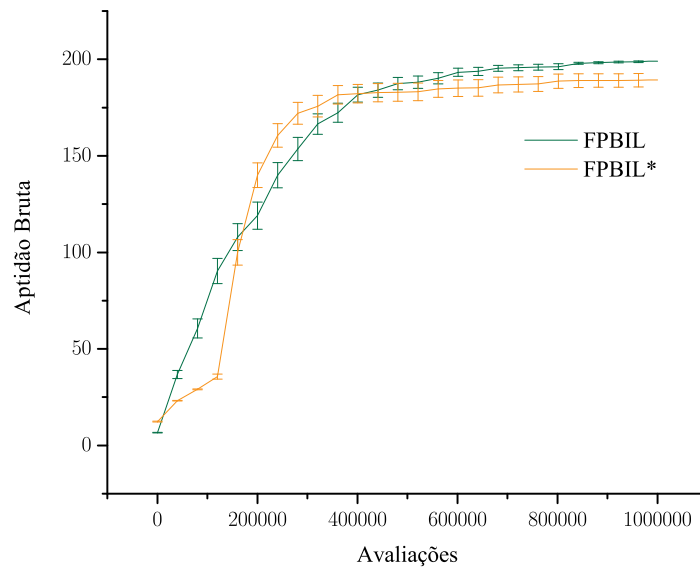


Figura 4.5: Comparação entre o FPBIL e o FPBIL*.

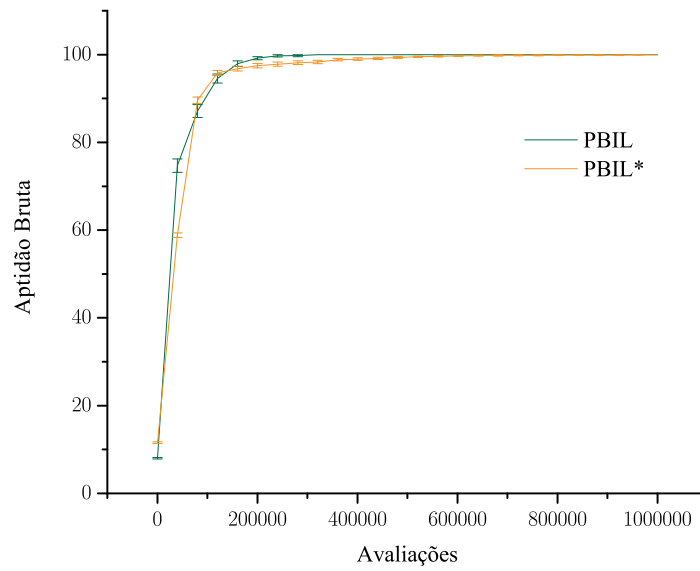


Figura 4.6: Comparação entre o PBIL e o PBIL*.

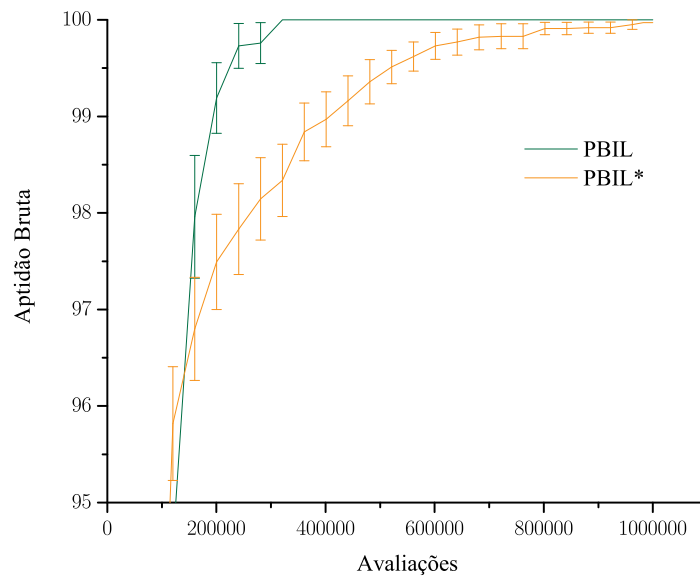


Figura 4.7: Detalhe do PBIL contra o PBIL*.

fornece uma visão muito peculiar sobre o funcionamento dos algoritmos. A figura 4.8, por exemplo, ilustra uma execução típica do FPBIL. Pode-se observar claramente

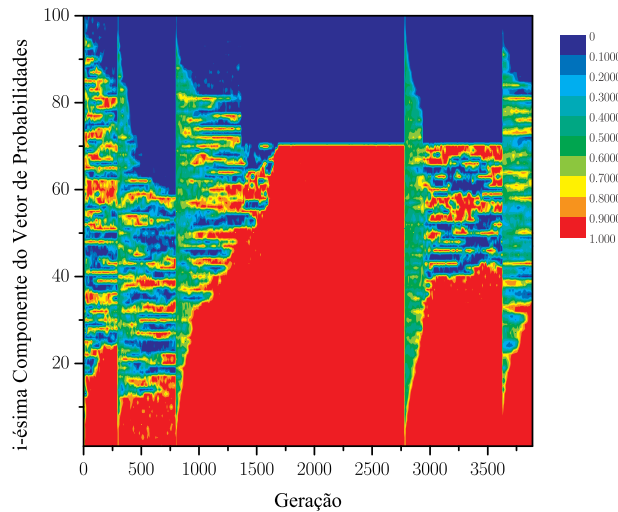


Figura 4.8: Evolução típica do vetor de probabilidades do FPBIL.

que durante as 1.000.000 avaliações da aptidão houve 4 reinicializações. Depois da segunda reinicialização, por volta da 2000^a geração, o FPBIL claramente atingiu o ótimo global. Já o PBIL, conforme mostra a figura 4.9 na página seguinte, converge, por volta da 2000^a geração, para P_2 . Também é interessante notar a ocorrência da mutação no PBIL. A região branca indica que a correspondente componente do vetor de probabilidades é exatamente 1. As várias manchas vermelhas ocorrem quando a mutação atinge uma determinada componente, fazendo com que esta se aproxime do valor 0.5.

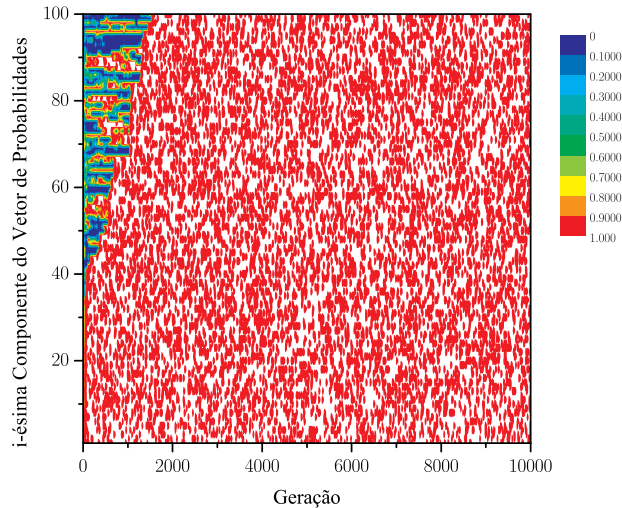


Figura 4.9: Evolução típica do vetor de probabilidades do PBIL.

4.2.2 Banana

O problema banana consiste em minimizar a função de *Rosenbrock* [17]:

$$B(x, y) = 100 (y - x^2)^2 + (1 - x)^2. \quad (4.23)$$

A partir da simples observação da expressão dessa equação conclui-se sem dificuldade que o mínimo de $B(x, y)$ ocorre para $(x, y) = (1, 1)$. Também não é difícil mostrar analiticamente que esse é o único ponto em que $B(x, y)$ se torna invariante. Entretanto, observando o gráfico de $B(x, y)$ na figura 4.10 na próxima página, torna-se impossível chegar à mesma conclusão.

É um tanto evidente a existência do vale localizado em $y = x^2$, mas visualizar o ponto exato do vale onde $B(x, y)$ é mínimo não é nada simples. A dificuldade para tal visualização deve-se ao fator 100 que multiplica apenas $(y - x^2)^2$, deixando o termo $(1 - x)^2$ de fora. Somente quando observado em escala logarítmica, como na figura 4.11 na página 54 é que torna-se aparente a região onde o mínimo está

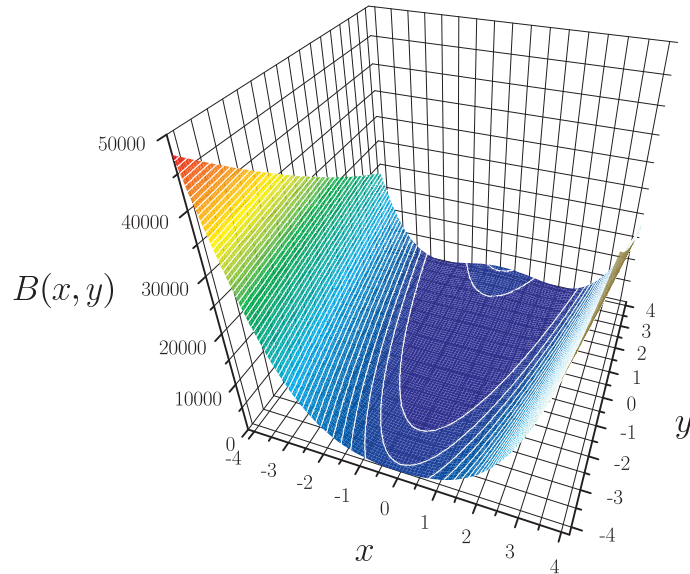


Figura 4.10: Gráfico de $B(x, y)$, função objetivo do problema banana.

localizado. A linha branca é uma curva de nível que evidencia o formato de banana, que dá nome ao problema.

A função de *Rosenbrock* é classicamente utilizada para testar algoritmos de otimização, justamente pela dificuldade que esta impõe, especialmente para os algoritmos baseados em gradiente.

4.2.2.1 Representação

O conjunto $\mathcal{S}_B \subset \mathbb{R}$ que será codificado em vetores binários, para a utilização dos algoritmos, é $[-4,194.304; 4,194.304]^2$, com uma granulação de 0,000.001 em ambas as variáveis x e y . Isto significa que cada variável necessita de 23 bits para representar \mathcal{S}_B , resultando num total de $2^{46} = 70.368.744.177.664$ possibilidades.

Mais formalmente, tem-se, com $n = 2 \cdot 23 = 46$,

$$\mathcal{M}_{\mathcal{S}_B}^n(\mathcal{I}) = (x, y), \quad (4.24)$$

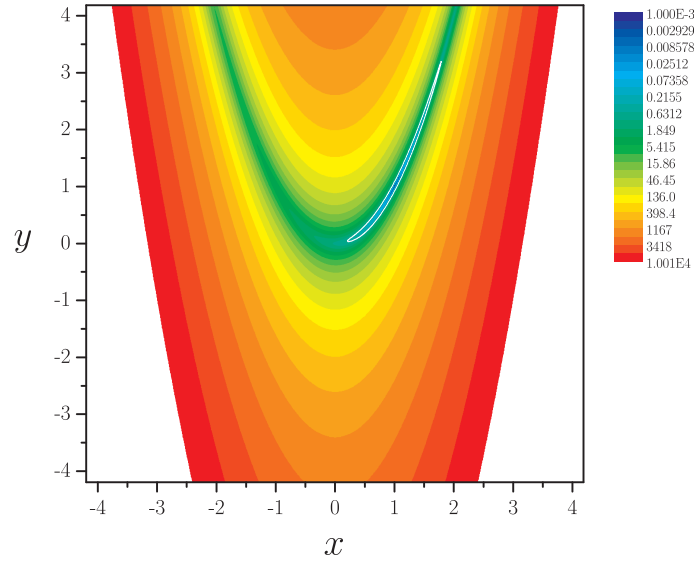


Figura 4.11: Curvas de nível de $\log_{10} B(x, y)$. A curva de cor branca, em forma de banana, destaca a região azul em que se encontra o ponto de mínimo.

com

$$x = -4,194.304 + \frac{G_1(\mathcal{I})}{1.000.000}; \quad (4.25)$$

$$y = -4,194.304 + \frac{G_2(\mathcal{I})}{1.000.000}, \quad (4.26)$$

onde $G_1(\mathcal{I})$ é a decodificação da primeira metade de \mathcal{I} e $G_2(\mathcal{I})$, da segunda, ambas utilizando o código de Gray.

4.2.2.2 Aptidões Bruta e Padrão

A aptidão bruta do problema banana utilizada neste trabalho é simplesmente o valor de $B(x, y)$:

$$\mathcal{A}_b(\mathcal{I}_i) = B(x, y) = B \circ \mathcal{M}_{S_B}^n(\mathcal{I}_i). \quad (4.27)$$

Como se trata de um problema de minimização e $\mathcal{A}_p(\mathcal{I}^\dagger) = 0$, a aptidão padrão

utilizada será a própria aptidão bruta:

$$\mathcal{A}_p(\mathcal{I}_i) = B \circ \mathcal{M}_{\mathcal{S}_B}^n(\mathcal{I}_i). \quad (4.28)$$

4.2.2.3 Resultados

Comparando o FPBIL com o PBIL original, na busca pelo mínimo, no problema banana, vê-se que a vantagem inicial do PBIL é superada em muito nas últimas avaliações da aptidão (por um fator de aproximadamente 10^6), como mostra a figura 4.12. O PBIL fica estagnado logo depois da 2000^a avaliação da aptidão, mas o valores

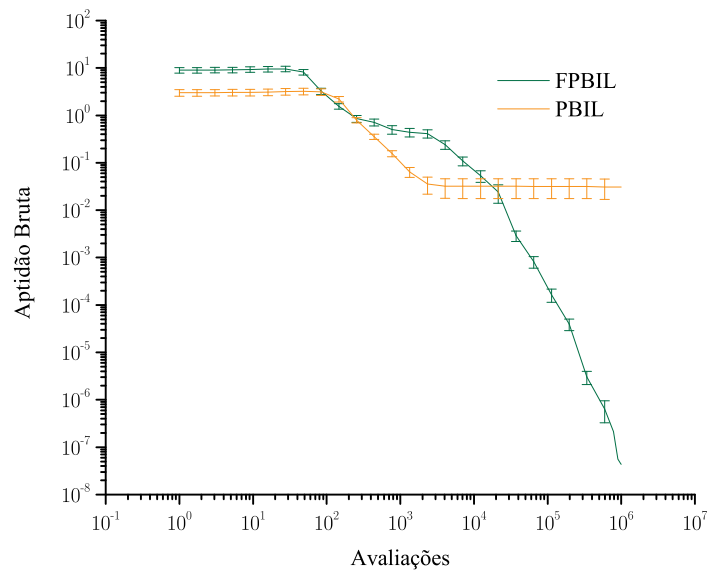


Figura 4.12: Comparação entre o FPBIL e o PBIL.

$B(x, y)$ encontrados pelo FPBIL continuam decrescendo a uma taxa contínua até o final.

O FPBIL*, porém, parece não diferir muito do PBIL*, conforme ilustra a figura 4.13 na página seguinte. Apenas uma leve vantagem do PBIL*. Entretanto, novamente, o PBIL* logo fica estagnado, enquanto o FPBIL* demonstra possuir ainda

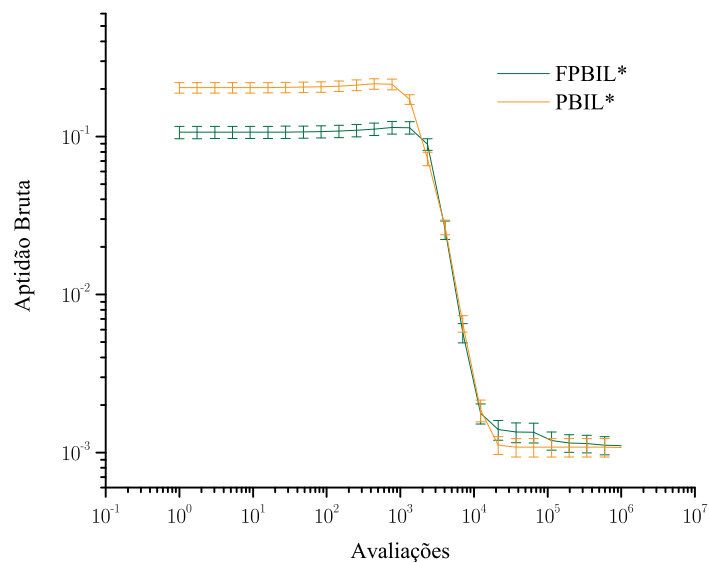


Figura 4.13: Comparação entre o FPBIL* e o PBIL*.

um potencial para encontrar uma solução melhor. A figura 4.14 na próxima página mostra esse detalhe.

O FPBIL também se mostra superior frente ao FPBIL* no final da execução, como se vê na figura 4.15 na página seguinte.

Por último, o PBIL é comparado com o PBIL*, indicando que, neste caso, o aumento da população fez a diferença, ainda que irrisória (comparada com a diferença entre o PBIL e o FPBIL), como mostra a figura 4.16 na página 58.

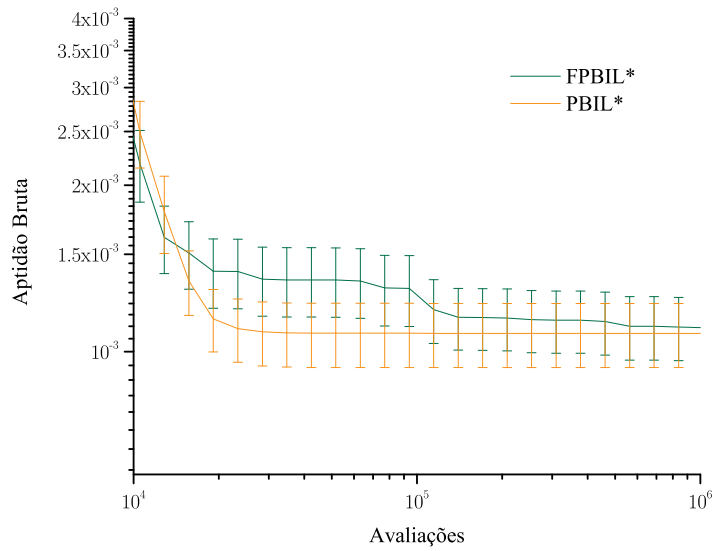


Figura 4.14: Detalhe do FPBIL* contra o PBIL*.

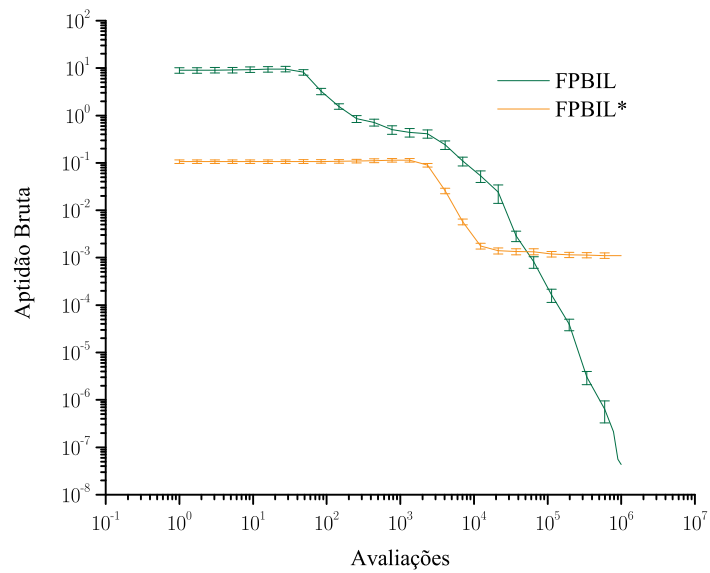


Figura 4.15: Comparação entre o FPBIL e o FPBIL*.

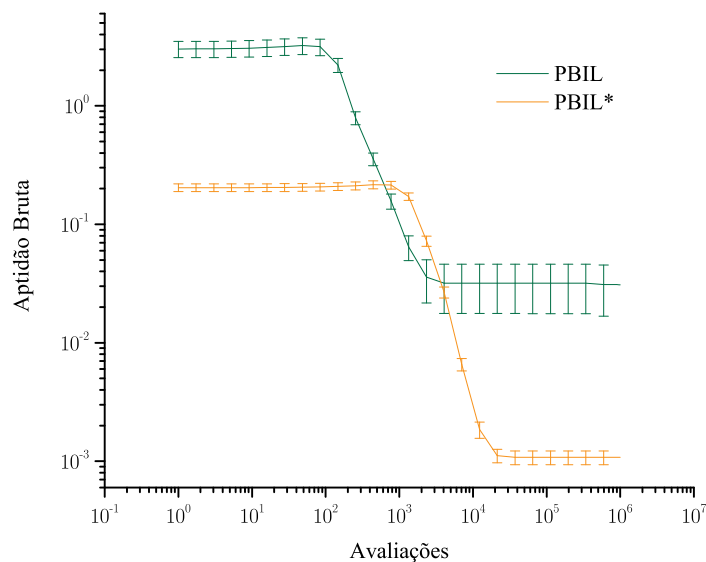


Figura 4.16: Comparação entre o PBIL e o PBIL*.

4.2.3 PCV Rykel48

Um caixeiro viajante deve percorrer N cidades, retornando ao final à cidade de origem, de modo que nenhuma cidade seja visitada mais de uma vez. Existem várias rotas possíveis (para $N > 2$); na realidade há muitas — com o número de rotas igual a N — pois o número de rotas é $(N - 1)!$ (fatorial, não ponto de exclamação!). O problema do caixeiro viajante (PCV) consiste em encontrar a rota mais curta.

O PCV é um problema da classe NP, significando que não existe hoje² um algoritmo de ordem polinomial que o resolva. A classe NP pode ser considerada uma classe de complexidade computacional intermediária entre as classes P e EXP, pois apenas a quantidade descomunal de combinações é responsável pela demanda de tempo [19]; a avaliação de cada combinação geralmente é a parte mais fácil.

²Pode ser que nunca venha a existir tal algoritmo. Quem solucionar essa questão será dono de US\$ 1,000,000.00 do *Clay Mathematics Institute*, já que o problema P versus NP é um dos problemas do milênio [18].

Rykel48 [20] é um PCV assimétrico de 48 cidades resultando num total de 258.623.241.511.168.180.642.964.355.153.611.979.969.197.632.389.120.000.000.000 rotas possíveis. Em um PCV assimétrico a distância de uma cidade A até uma outra B pode ser diferente da distância de B até A, talvez modelando vias de uma única mão. Embora os PCVs simétricos e assimétricos compartilhem o mesmo número de rotas, para o mesmo valor de N , a assimetria embaralha a topologia do espaço de busca, resultando em PCVs mais complexos.

Uma diferença importante do PCV Rykel48 para os problemas anteriores é que a restrição de que nenhuma cidade seja visitada mais de uma vez impede que as rotas sejam codificadas em vetores binários de forma direta. As rotas devem ser representadas de uma forma indireta.

4.2.3.1 Representação *Random Keys*

Considere um PCV com 3 cidades, A, B e C. Três objetos necessitam de 2 bits para serem representados em vetores binários, como mostra a tabela 4.3, e uma rota

Tabela 4.3: Representação binária de 3 cidades.

Representação Binária	Cidade
00	A
01	B
11	C
10	◇

pode ser representada por um vetor binário de 6 bits. A tabela 4.4 fornece as duas rotas válidas possíveis. No caso de PCV simétrico estas duas teriam o mesmo

Tabela 4.4: Representação binária das possíveis rotas.

\mathcal{I}	Rota
000111	A→B→C
001101	A→C→B

comprimento.

O problema surge quando se tenta sortear uma rota a partir de um vetor de probabilidades. A tabela 4.5 ilustra algumas possibilidades, a maioria delas inválidas.

Tabela 4.5: Possíveis rotas. A maioria delas inválidas.

\mathcal{I}	Rota
000000	A→A→A
001100	A→C→A
000110	A→B→◇
101010	◇→◇→◇
111010	C→◇→◇
011100	B→C→A
010001	B→A→B
110001	C→A→B

A estratégia da representação *random keys* [21] é representar não as cidades (ou quaisquer que sejam os objetos), mas números — a chave — que, quando ordenados, determinam a ordem em que as cidades devem ser arranjadas em uma rota³. Em caso de empate, a ordem natural prevalece.

Os mesmos indivíduos da tabela 4.5 podem agora gerar rotas sempre válidas. A tabela 4.6 na página seguinte mostra como. Primeiramente cada indivíduo é decodificado (utilizando o código de Gray, neste exemplo) em uma seqüência, do tamanho do número de cidades participantes, de números inteiros — os dentes da chave —, cada um associado a uma cidade de acordo com a ordem natural das cidades. Em seguida, os dentes das chaves são ordenados. Como conseqüência, as cidades associadas aos dentes de chave são juntamente reordenadas, gerando rotas válidas.

O preço que se paga com a utilização da representação *random keys* é a perda da bijetividade entre $\tilde{\mathcal{H}}_n$ e o conjunto das rotas válidas, o que pode dificultar o

³O modelo de listas [22] é outra forma de representação desse tipo mas, como verificado em [6], é menos eficaz que o *random keys*.

Tabela 4.6: Rotas válidas utilizando a representação *random keys*.

\mathcal{I}	Chave	Chave ordenada	Rota
000000	$0_A, 0_B, 0_C$	$0_A, 0_B, 0_C$	A→B→C
001100	$0_A, 2_B, 0_C$	$0_A, 0_C, 2_B$	A→C→B
000110	$0_A, 1_B, 3_C$	$0_A, 1_B, 3_C$	A→B→C
101010	$3_A, 3_B, 3_C$	$3_A, 3_B, 3_C$	A→B→C
111010	$2_A, 3_B, 3_C$	$2_A, 3_B, 3_C$	A→B→C
011100	$1_A, 2_B, 0_C$	$0_C, 1_A, 2_B$	C→A→B
010001	$1_A, 0_B, 1_C$	$0_B, 1_A, 1_C$	B→A→C
110001	$2_A, 0_B, 1_C$	$0_B, 1_C, 2_A$	B→C→A

aprendizado; bem como a ocorrência de empates, que reforçam sempre a ordem natural das cidades.

Uma outra questão é que o número de bits representando cada dente da chave pode ser qualquer um. Quanto menor o número de bits, maior o número de empates; quanto maior o número de bits, maior a perda de bijetividade e maior o tamanho de \mathcal{P} . Intuitivamente, um número de bits de valor intermediário deve ser a melhor escolha.

Para o PCV Rykel48, neste trabalho, serão testados dentes de chave com respectivamente 6, 9 e 12 bits, resultando em buscas sobre $\check{\mathcal{H}}_{288}$, $\check{\mathcal{H}}_{432}$ e $\check{\mathcal{H}}_{576}$, respectivamente.

4.2.3.2 Aptidões Bruta e Padrão

A aptidão bruta do PCV Rykel48 utilizada neste trabalho é simplesmente o comprimento C_i de cada rota correspondente ao indivíduo \mathcal{I}_i :

$$\mathcal{A}_b(\mathcal{I}_i) = C_i. \quad (4.29)$$

Como se trata de um problema de minimização, poder-se-ia ter $\mathcal{A}_p(\mathcal{I}_i) = \mathcal{A}_b(\mathcal{I}_i)$. Porém, como $\mathcal{A}_p(\mathcal{I}^\dagger) = 14.422 \neq 0$, a aptidão padrão utilizada será

$$\mathcal{A}_p(\mathcal{I}_i) = C_i - 14.422. \quad (4.30)$$

4.2.3.3 Resultados

O primeiro teste realizado com o PCV Rykel48 foi o de determinar um valor razoável para o número de bits codificando cada dente da chave. Como são 48 cidades, o menor número de bits por dente capaz de gerar uma chave sem empates é 6 — com 5 bits é possível gerar apenas $2^5 = 32$ dentes diferentes. A figura 4.17 mostra como o desempenho do FPBIL é influenciado por essa grandeza para 6 (mínimo), 12 (dobro do mínimo) e 9 (um valor intermediário) bits por chave. Verifica-se que até cerca de 10^3 avaliações da função aptidão, os três casos possuem

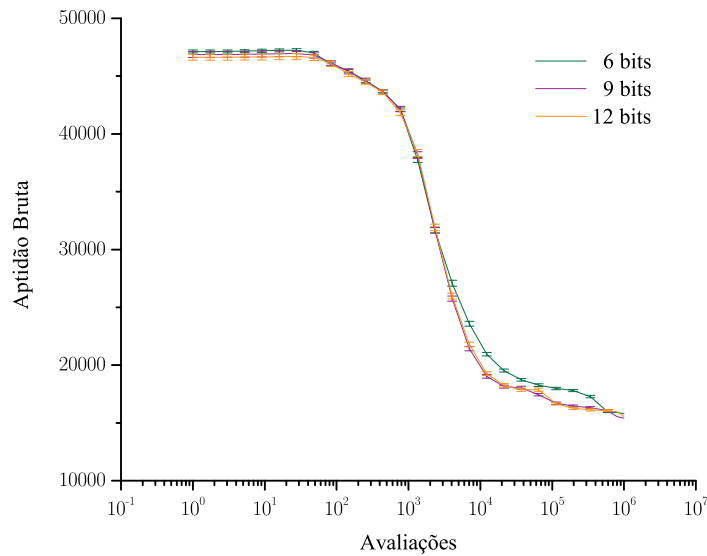


Figura 4.17: Comparação entre os algoritmos FPBIL para representações *random keys* de, respectivamente, 6, 9 e 12 bits por dente de chave.

o mesmo comportamento. A figura 4.18 na página seguinte ilustra os mesmos dados

para os comprimentos de rota menores que 20.000. O pior desempenho certamente

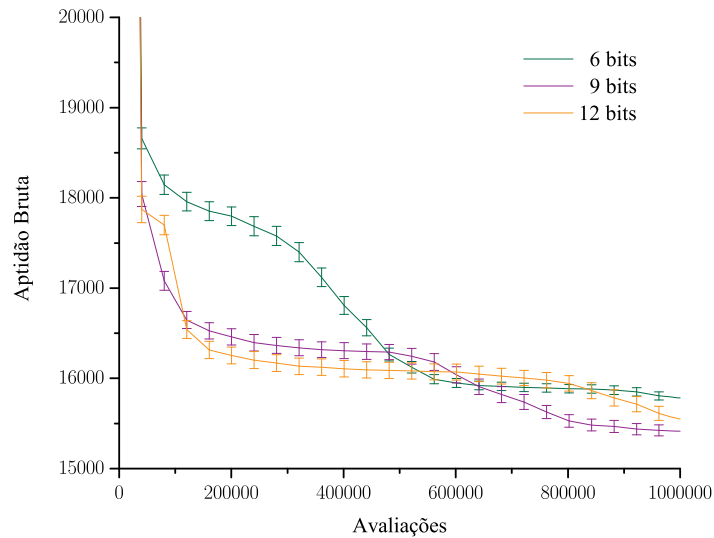


Figura 4.18: Detalhe da figura 4.17 na página anterior.

ocorre para 6 bits. Com 12 bits o FPBIL é um pouco melhor que com 9 até cerca de 500.000 avaliações, quando o FPBIL com 9 bits passa a dominar. Esta situação prevalece até o final, embora o FPBIL com 12 bits apresente um potencial de queda razoável nas últimas avaliações.

O mesmo teste foi realizado com o PBIL. A figura 4.19 na próxima página mostra o resultado. Até cerca de 10^4 avaliações da aptidão, o desempenho independe do número de bits por dente das chaves. A figura 4.20 na página seguinte mostra mais claramente os mesmos dados para os comprimentos de rotas menores que 20.000. Observa-se, mais uma vez, que 6 bits por dente de chave produz o pior desempenho. Entretanto, o PBIL com 12 bits parece manter uma leve vantagem em comparação com o PBIL com 9 bits, embora a diferença entre esses casos seja menor que os correspondentes desvios padrão.

A partir dos testes anteriores, parece razoável utilizar 9 bits por dente das chaves, para as comparações entre o PBIL original e o FPBIL. É o que mostra a figura 4.21.

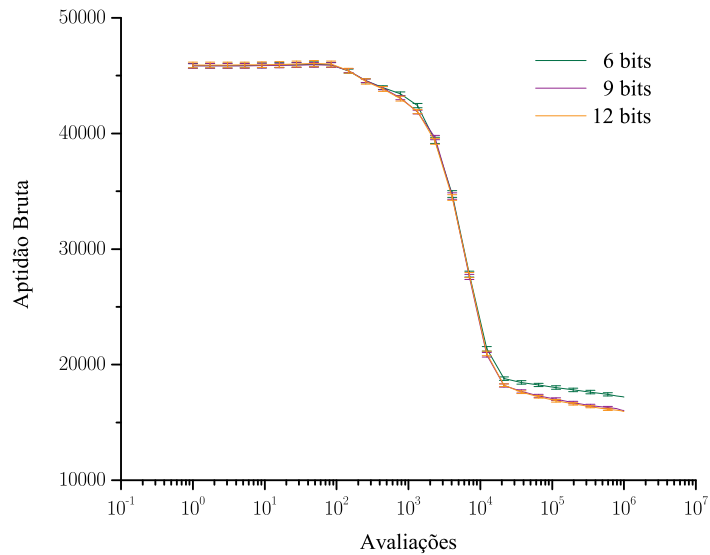


Figura 4.19: Comparação entre os algoritmos PBIL para representações *random keys* de, respectivamente, 6, 9 e 12 bits por dente de chave.

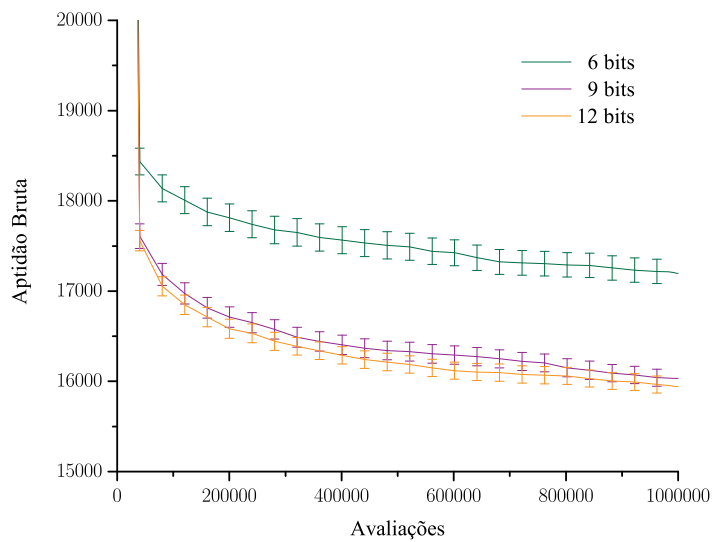


Figura 4.20: Detalhe da figura 4.19.

Visto em detalhe na figura 4.22 na próxima página, o FPBIL mantém a vantagem

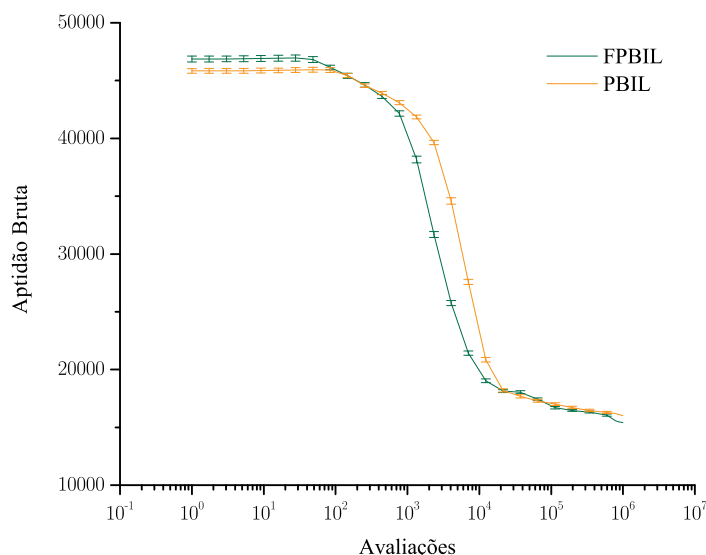


Figura 4.21: Comparação entre o FPBIL e o PBIL.

durante quase toda a execução dos algoritmos, especialmente nas últimas 500.000 avaliações.

A figura 4.23 na página seguinte mostra os valores mínimos e máximos ao final de um determinado número de execuções dos algoritmos. A rota mais curta encontrada pelo FPBIL foi 14.674, apenas 1,75% maior que o ótimo global. Nota-se que o PBIL apresenta uma maior dispersão em torno da média.

Neste ponto, vale ressaltar que a rota mais curta do PCV Rykel48, cujo comprimento vale 14.422, não é facilmente encontrada por nenhum algoritmo de busca de propósito geral. Por exemplo, os algoritmos genéticos só alcançam valores próximos a 16.500 [6] e os algoritmos baseados em colônias de formigas, desenhados especificamente para encontrar as menores rotas, alcançam o valor ótimo somente quando processados em paralelo, mesmo assim, apenas quando auxiliados de heurísticas [23]. A figura 4.23 na próxima página mostra que o PBIL é capaz de alcançar

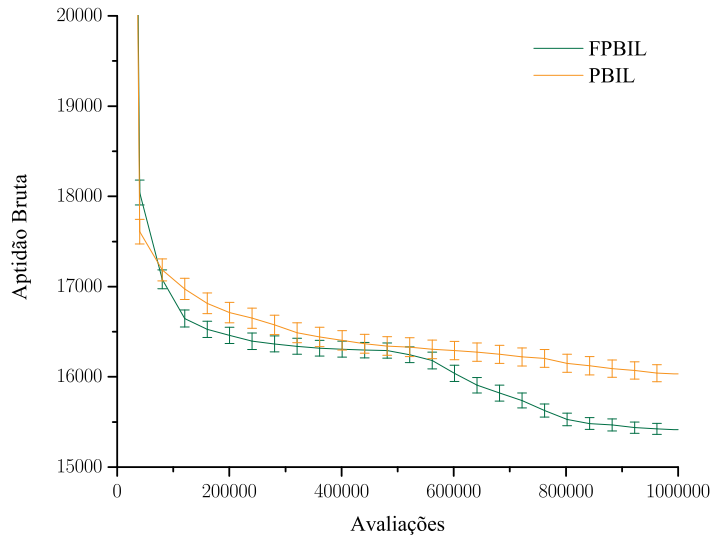


Figura 4.22: Detalhe do FPBIL contra o PBIL.

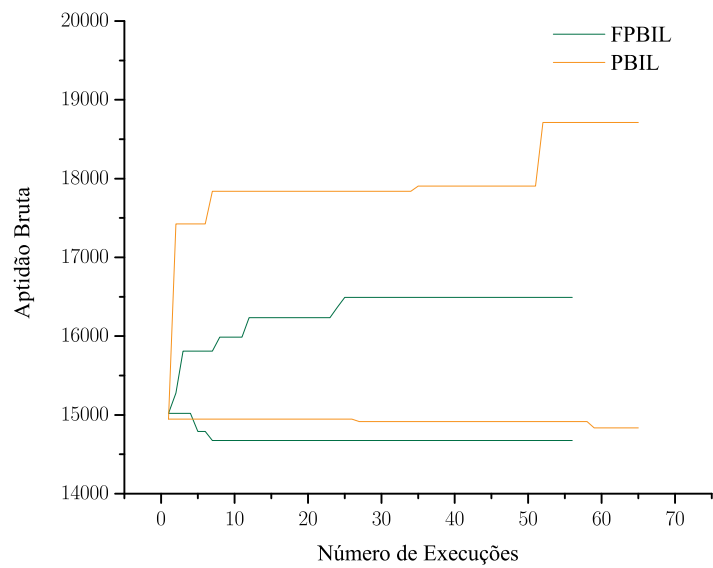


Figura 4.23: Valores mínimos e máximos ao final de um número de execuções.

valores pouco abaixo de 15.000. O fato de o FPBIL encontrar rotas com comprimento de 14.674 é uma conquista considerável.

Em seguida, o PBIL* é comparado com o FPBIL*. As figuras 4.24 e 4.25 apresentam os resultados. O FPBIL* é superior, embora perca nas primeiras avaliações.

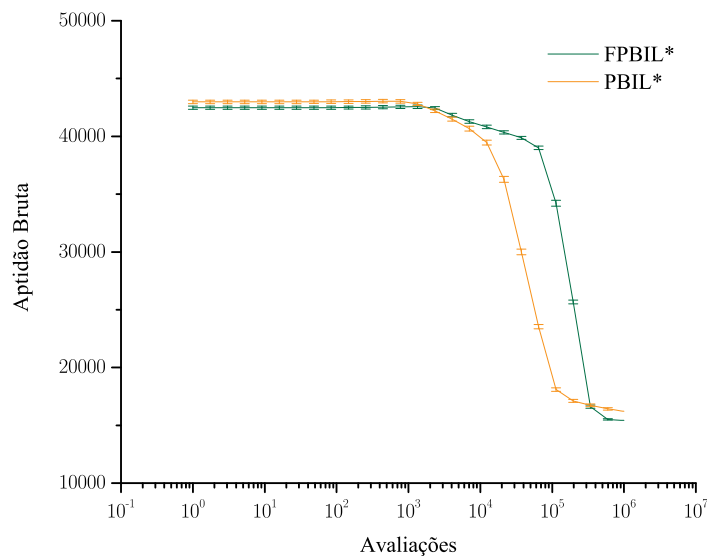


Figura 4.24: Comparação entre o FPBIL* e o PBIL*.

Finalmente, os algoritmos FPBIL e FPBIL* são comparados juntamente com o PBIL e PBIL*. As figuras 4.26 e 4.27 indicam que, embora o FPBIL perca para um número intermediário de avaliações, este é melhor para até cerca de 300.000 avaliações e empata com o FPBIL* no final de 10⁶ avaliações. A figura 4.28 na página 69 confirma o empate e até mostra uma tendência do FPBIL em passar para a liderança.

A comparação entre o PBIL e o PBIL*, mostrado nas figuras 4.29 e 4.30, confirma que o PBIL original não se torna necessariamente mais eficiente com o aumento da população.

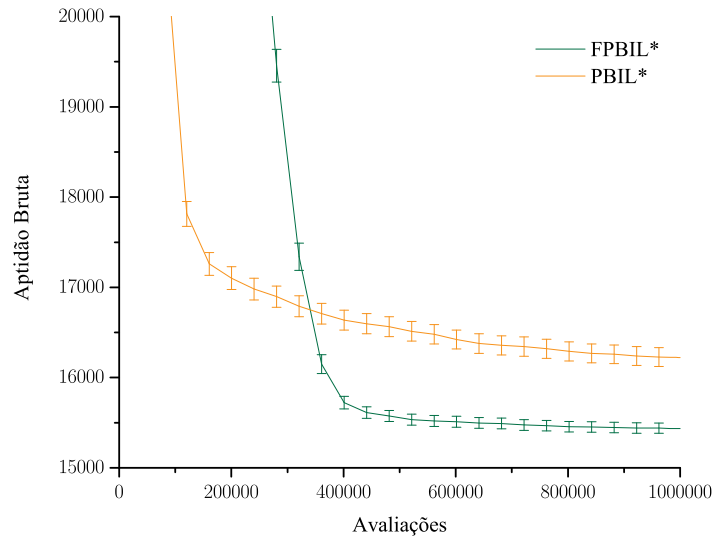


Figura 4.25: Detalhe do FPBIL* contra o PBIL*.

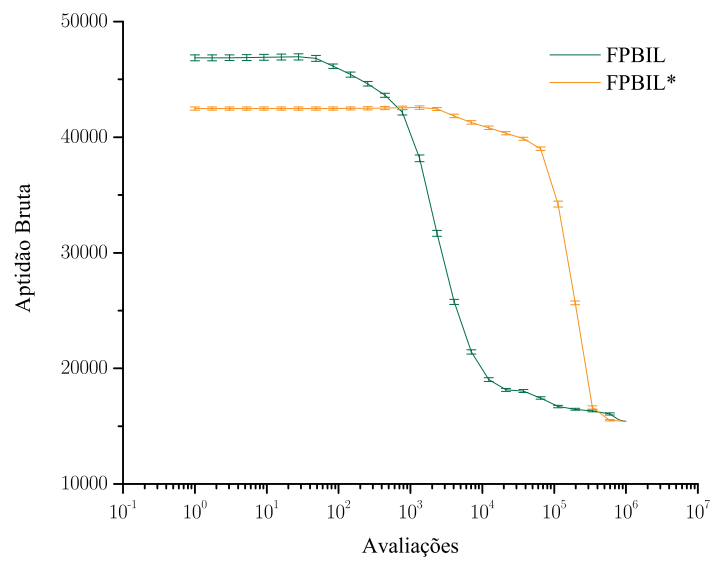


Figura 4.26: Comparação entre o FPBIL e o FPBIL*.

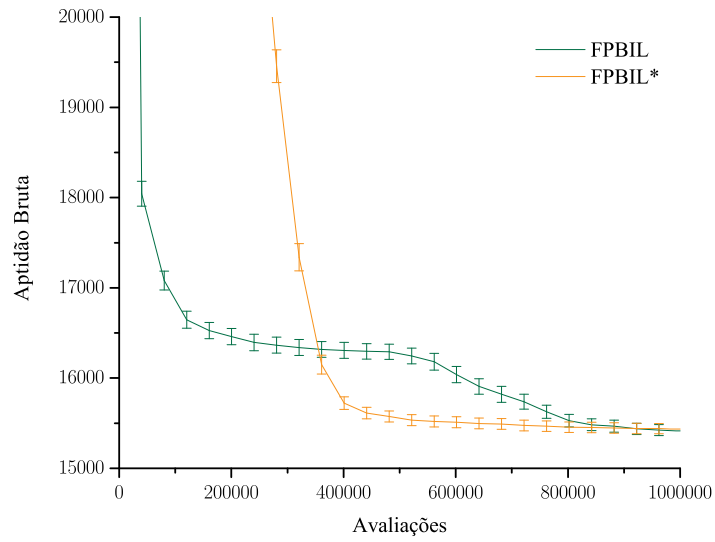


Figura 4.27: Detalhe do FPBIL contra o FPBIL*.

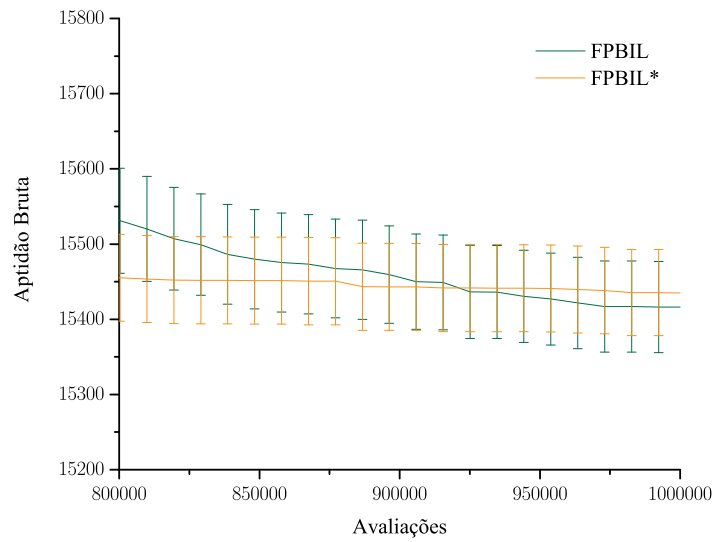


Figura 4.28: Detalhe da figura 4.27.

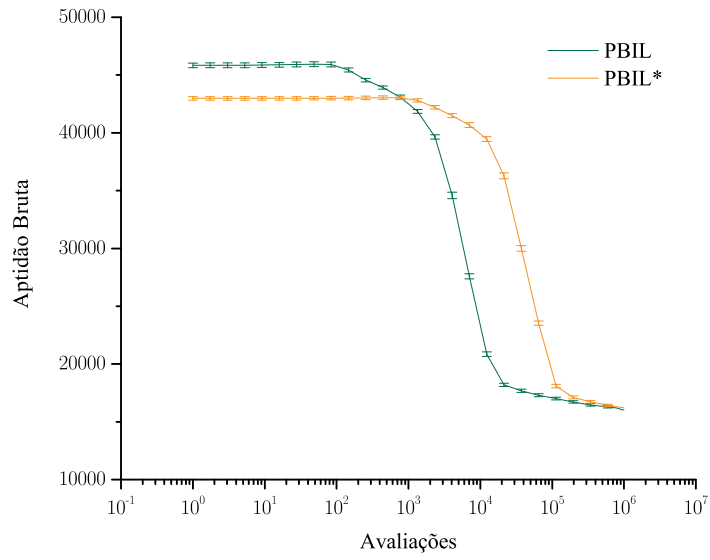


Figura 4.29: Comparação entre o PBIL e o PBIL*.

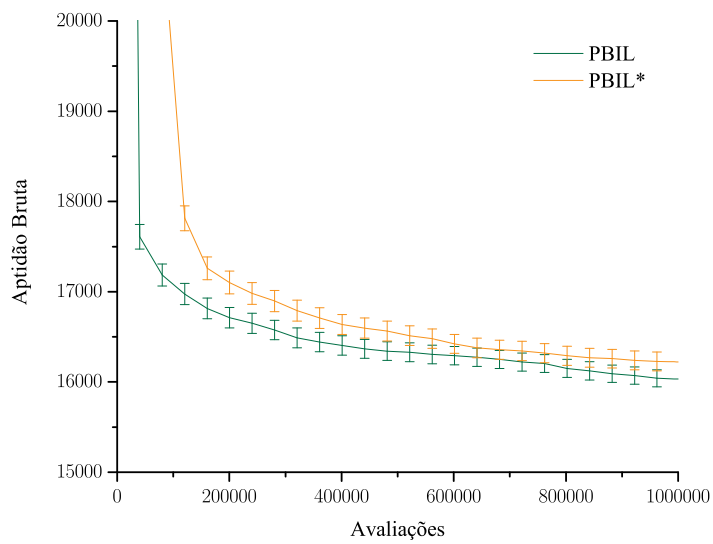



Figura 4.30: Detalhe do PBIL contra o PBIL*.

Tem-se, portanto, que em todos os testes realizados, ficou comprovado o desempenho superior do FPBIL, evidenciando o grande potencial deste, na solução de problemas de classes variadas. No próximo capítulo, o FPBIL será aplicado ao problema da recarga nuclear.

5 *FPBIL Aplicado ao Problema da Recarga Nuclear*

M dos problemas de grande interesse da engenharia nuclear diz respeito à recarga nuclear. Trata-se de uma questão que, como se verá oportunamente, apresenta várias sutilezas e grande complexidade. O problema da recarga nuclear será utilizado como último teste para o FPBIL neste trabalho, e colocará à prova o seu desempenho. Este capítulo descreve como o FPBIL pode ser aplicado ao problema e o compara, ao final, com as mais atuais e bem-sucedidas técnicas de otimização.

5.1 Descrição do Problema

Iniciada a operação da usina, a concentração de material físsil (^{235}U) dos elementos combustíveis começa a decrescer. Ao fim de um período de tempo, denominado ciclo de operação, não é mais possível manter o reator crítico produzindo energia à potência nominal. Os elementos combustíveis com baixas concentrações de ^{235}U são substituídos por elementos combustíveis novos. Estes, juntamente com os demais elementos combustíveis remanescentes do ciclo anterior, compõem o núcleo do ciclo subsequente.

Chama-se otimização de recargas o processo de determinação do posicionamento dos elementos combustíveis no reator, satisfazendo restrições técnicas e as impostas pela própria geometria do núcleo, que torna melhor algum aspecto do funcionamento

da usina. Neste trabalho, a recarga será otimizada para maximizar o comprimento do ciclo 7 de Angra 1.

5.2 O Núcleo de Angra 1 e suas Simetrias

O reator nuclear de Angra 1 é um reator a água pressurizada (PWR) que utiliza urânio enriquecido como combustível. Pastilhas de UO_2 são acondicionadas no interior do revestimento, um tubo de Zircaloy-4 — uma liga de zircônio, estanho, ferro e cromo — projetado para impedir o contato do UO_2 e dos produtos de fissão com a água. Após a introdução das pastilhas, o revestimento é pressurizado com hélio e selado, passando a constituir o que se denomina vareta combustível.

No núcleo do reator de Angra 1, há 28.435 varetas, distribuídas em 121 elementos combustíveis. Cada um desses contém 235 varetas combustíveis, que são sustentadas por uma estrutura composta de 20 tubos guias de barras de controle, 1 tubo de instrumentação nuclear, 8 grades espaçadoras e 1 bocal em cada extremidade. A figura 5.1 na próxima página mostra um esquema do núcleo de Angra 1.

Os eixos principais P_1 e P_2 dividem o núcleo em quatro quadrantes. Esses, juntamente com os eixos diagonais D_1 e D_2 , dividem o núcleo em oito octantes. Pode-se verificar que, geometricamente, o núcleo do reator é simétrico sob reflexões em relação a cada um desses eixos. Conseqüentemente, a posição de cada elemento combustível localizado fora dos eixos possui outras 7 posições simétricas. Um elemento combustível localizado em uma dessas posições é chamado de elemento de octeto. A posição de cada elemento combustível localizado sobre apenas um dos eixos possui outras 3 posições simétricas e cada um desses elementos combustíveis é chamado de elemento de quarteto. Chama-se de elemento central o elemento combustível que ocupa a posição sobre os 4 eixos simultaneamente. A figura 5.2 na página 75 destaca o elemento central, os elementos de quarteto e os elementos de octeto de um octante de núcleo e fornece um sistema de coordenadas para localizá-los.

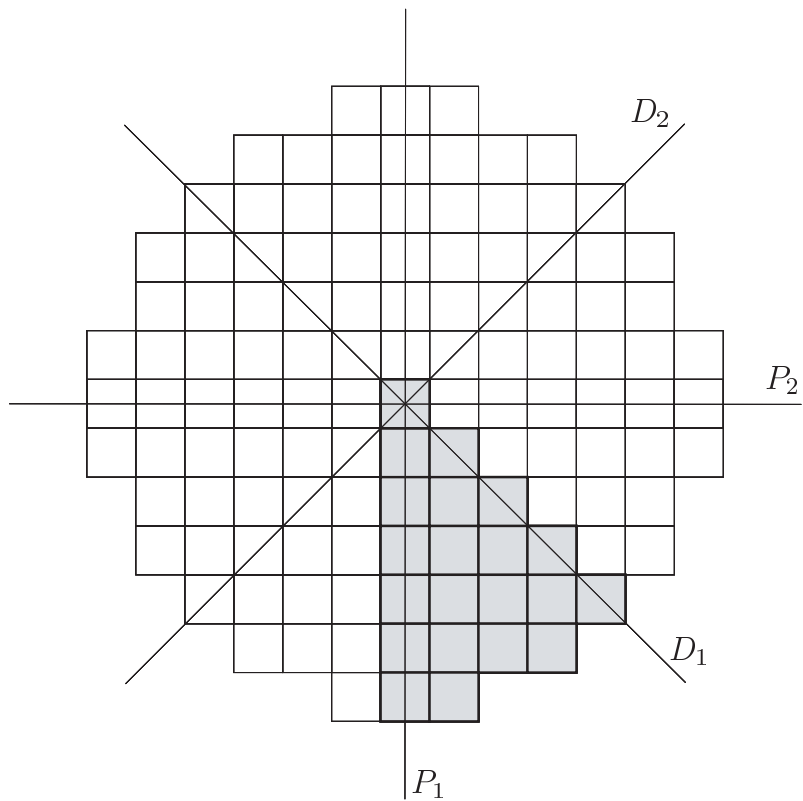


Figura 5.1: Esquema do núcleo de Angra 1. Os quadrados representam os 121 elementos combustíveis. Em destaque se encontra 1/8 de núcleo, determinado pelos eixos principais (P_1, P_2) e diagonais (D_1, D_2).

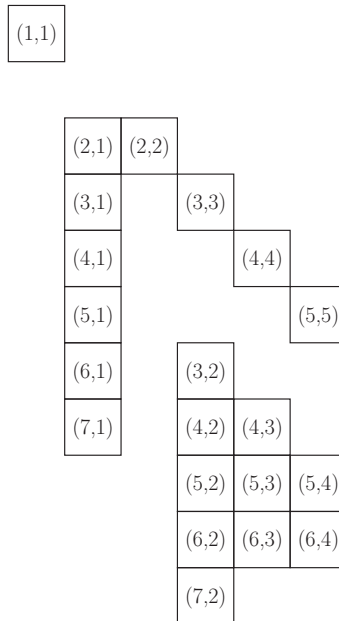


Figura 5.2: Elementos central, de quarteto e de octeto de um oitavo de núcleo e sistema de coordenadas para localizá-los.

Neste trabalho, o núcleo do reator será arranjado com simetria de 1/8 de núcleo, significando que o arranjo de um octante será copiado nos demais, com elementos combustíveis de mesmo tipo ocupando posições simétricas entre si.

5.3 Metodologia

A exemplo dos trabalhos [9, 23–25] com os quais o FPBIL será comparado adiante, a concentração crítica de Boro C_B (na etapa de queima do combustível nuclear quando o Xenônio entra em equilíbrio) será o objeto de maximização do FPBIL, o que corresponde [24] a maximizar o comprimento do ciclo. Todavia, existem restrições que precisam ser respeitadas.

A razão entre a densidade de potência linear máxima e a densidade de potência linear média, no plano horizontal do núcleo do reator onde ocorre o pico local de potência é denominada fator de pico de potência radial do núcleo (F_{XY}). As especi-

ficações técnicas das centrais nucleares restringem o valor máximo de F_{XY} . No caso de Angra 1, por exemplo, F_{XY} deve ser, no máximo, 1,435.

Uma outra restrição é imposta pela geometria do núcleo. Com simetria de 1/8 de núcleo, um elemento de quarteto não pode trocar de lugar com um elemento de octeto, simplesmente porque os números de posições simétricas de elementos de quarteto e de octeto são diferentes. Da mesma forma, com simetria de 1/8 de núcleo, o elemento central não pode mudar de lugar.

É evidente que um reator nuclear possui uma complexidade muito maior do que poderia ser descrita neste texto, de modo que existem outras restrições que não serão consideradas aqui — até porque os trabalhos com os quais o FPBIL será comparado utilizam somente as mesmas duas restrições já citadas.

Neste trabalho, a restrição geométrica é resolvida automaticamente pela representação *random keys* de cada arranjo, enquanto a restrição técnica imposta a F_{XY} é solucionada a partir da penalização da aptidão. Cada vetor binário é decodificado em uma configuração de núcleo que é, então, processada pelo RECNOd, um programa que simula um ciclo de operação de Angra 1.

5.3.1 RECNOd

A motivação para o desenvolvimento do RECNOd [24] foi a busca por um código de física de reatores bidimensional, a dois grupos de energia, mais rápido do que o ANC [26], de modo a reduzir os custos computacionais dos sistemas de otimização de recargas.

O RECNOd é um código nodal (de malha grossa) inspirado no trabalho de Montagnini [27], que utiliza uma variação do método de expansão de fluxo, criado por Langenbuch [28] no final dos anos 70. Segundo Montagnini, o seu código nodal é capaz de produzir erros máximos na distribuição de potência inferiores a 1,5%.

Uma vez que o RECNOd não calcula o fator de pico de potência radial F_{XY} ,

este parâmetro foi substituído por p_{rm} — a máxima potência média relativa dos elementos combustíveis. Devido às discrepâncias entre as distribuições de potência calculadas pelo RECNOD e pelo ANC, o valor 1,395 é adotado como o valor limiar de p_{rm} — devendo corresponder a 1,435 como limiar de F_{XY} .

O RECNOD recebe como entrada a configuração de 1/8 de núcleo de uma forma equivalente a mostrada na tabela 5.1. O “Tipo” do elemento combustível é o número

Tabela 5.1: Formato (equivalente) de entrada das configurações de núcleo usado pelo RECNOD.

	P_i	P_{i-1}	Tipo
Central	(1, 1)	(1, 1)	4
Quartetos	(2, 1)	(3, 1)	4
	(3, 1)	(2, 2)	5
	(4, 1)	(7, 1)	2
	(5, 1)	(3, 3)	4
	(6, 1)	(6, 1)	2
	(7, 1)	(4, 4)	1
	(2, 2)	(5, 1)	4
	(3, 3)	(5, 5)	2
	(4, 4)	(2, 1)	6
	(5, 5)	(4, 1)	4
Octetos	(3, 2)	(5, 3)	6
	(4, 2)	(5, 2)	4
	(5, 2)	(5, 4)	5
	(6, 2)	(7, 2)	5
	(7, 2)	(4, 3)	6
	(4, 3)	(3, 2)	4
	(5, 3)	(6, 4)	2
	(6, 3)	(6, 2)	3
	(5, 4)	(6, 3)	3
	(6, 4)	(4, 2)	4

que o identifica na biblioteca de dados nucleares do RECNOD. P_i é a posição que o elemento combustível, que ocupava a posição P_{i-1} no ciclo anterior, do tipo es-

pecificado, ocupa no ciclo em análise (as posições são determinadas utilizando-se o sistema de coordenadas da figura 5.2 na página 75).

Como saída, o RECNOD informa a concentração crítica de Boro C_B e as potências médias relativas dos elementos combustíveis — de onde se obtém p_{rm} — a partir dos quais a aptidão bruta é construída.

5.3.2 Representação das Configurações Nucleares

Conforme explicado na seção 5.3 na página 75, admitindo-se uma simetria de 1/8 de núcleo, o elemento central não muda de lugar e os elementos de quartetos e de octetos não podem trocar posições entre si. O resultado é um total de $10! \cdot 10! = 13.168.189.440.000$ possíveis configurações de núcleo.

Essas restrições são resolvidas com o simples uso da representação *random keys*. Considere as duas 10-uplas ordenadas

$$Q = ((2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (2, 2), (3, 3), (4, 4), (5, 5)); \quad (5.1)$$

$$O = ((3, 2), (4, 2), (5, 2), (6, 2), (7, 2), (4, 3), (5, 3), (6, 3), (5, 4), (6, 4)). \quad (5.2)$$

Elas estabelecem uma ordem sobre as posições dos elementos de quarteto e de octeto. Essa ordem será usada tanto pelo *random keys* como para o preenchimento do núcleo. O exemplo a seguir ilustra o procedimento.

Q e O possuem 10 elementos, cada. O menor número de bits por dente de chave que torna possível uma representação *random keys* sem empate é 4, valor adotado neste exemplo.

Considere um indivíduo \mathcal{I} , sorteado a partir de \mathcal{P} , formado pela concatenação de $\mathcal{I}|Q$ e $\mathcal{I}|O$:

$$\mathcal{I} = \mathcal{I}|Q \oplus \mathcal{I}|O, \quad (5.3)$$

com

$$\mathcal{I}|Q = 1000001111010100111111100101010100101100; \quad (5.4)$$

$$\mathcal{I}|O = 1010101111110110111101100100110100001100 \quad (5.5)$$

representando, respectivamente, os elementos de quarteto e de octeto. A tabela 5.2 mostra como se constrói uma reordenação dos elementos de quarteto (sempre sem o risco de repetições).

Tabela 5.2: *Random keys* aplicado a $\mathcal{I}|Q$, a primeira parte de \mathcal{I} , responsável pela representação dos elementos de quarteto.

$\mathcal{I} Q$	1000	0011	1101	0100	1111	1110	0101	0101	0010	1100
C	15	2	9	7	10	11	6	6	3	8
Q	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)	(2, 2)	(3, 3)	(4, 4)	(5, 5)
$\pi_{\triangleleft}(C)$	2	3	6	6	7	8	9	10	11	15
$\pi_{\triangleleft}(Q)$	(3, 1)	(4, 4)	(2, 2)	(3, 3)	(5, 1)	(5, 5)	(4, 1)	(6, 1)	(7, 1)	(2, 1)

A chave C surge da decodificação de cada um dos seus 10 dentes de 4 bits (utilizando o código de Gray, neste caso). A ordem dos dentes de C é, então, relacionada com a de Q . Por fim, os elementos de Q são reordenados a partir de π_{\triangleleft} , a permutação que arranja os dentes de C em ordem crescente. O mesmo é feito para O , na tabela 5.3 na próxima página.

A partir de $\pi_{\triangleleft}(Q)$ e $\pi_{\triangleleft}(O)$, a entrada de dados do RECNOd é construída da seguinte maneira: o elemento combustível que ocupava a posição correspondente ao primeiro elemento de $\pi_{\triangleleft}(Q)$ (o elemento combustível do tipo 4 que ocupava a posição (3, 1), ver tabela 5.1 na página 77) passa a ocupar a posição correspondente ao primeiro elemento de Q (a posição (2, 1)). Esse primeiro passo gera a linha mostrada na tabela 5.4 na próxima página.

O mesmo ocorre para um elemento de octeto. O elemento combustível que

Tabela 5.3: *Random keys* aplicado a $\mathcal{S}|O$, a segunda parte de \mathcal{S} , responsável pela representação dos elementos de octeto.

$\mathcal{S} O$	1010	1011	1111	0110	1111	0110	0100	1101	0000	1100
C	12	13	10	4	10	4	7	9	0	8
O	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)	(4, 3)	(5, 3)	(6, 3)	(5, 4)	(6, 4)
$\pi_{\triangleleft}(C)$	0	4	4	7	8	9	10	10	12	13
$\pi_{\triangleleft}(O)$	(5, 4)	(6, 2)	(4, 3)	(5, 3)	(6, 4)	(6, 3)	(5, 2)	(7, 2)	(3, 2)	(4, 2)

Tabela 5.4: Uma linha (de quarteto) da entrada de dados do RECNOD.

P_i	P_{i-1}	Tipo
(2, 1)	(3, 1)	4

ocupava a posição correspondente ao primeiro elemento de $\pi_{\triangleleft}(O)$ (o elemento combustível do tipo 5 que ocupava a posição (5, 4), ver tabela 5.1 na página 77) passa a ocupar a posição correspondente ao primeiro elemento de O (a posição (3, 2)). Esse passo gera a linha da tabela 5.5.

Tabela 5.5: Uma linha (de octeto) da entrada de dados do RECNOD.

P_i	P_{i-1}	Tipo
(3, 2)	(5, 4)	5

Como o *random keys* é aplicado separadamente em $\mathcal{S}|Q$ e $\mathcal{S}|O$, nunca um elemento de quarteto será trocado por um de octeto. A restrição geométrica é garantida.

A ordem estabelecida em Q e O não precisa ser a das equações (5.4) e (5.5) e uma ordem diferente pode afetar o desempenho do FPBIL (ou de qualquer outro algoritmo de busca). O motivo é que as escolhas de Q e O determinam o quanto uma configuração nuclear se torna diferente como consequência da alteração de uns

Tabela 5.6: Nova entrada das configurações de núcleo para o RECNOd.

	P_i	P_{i-1}	Tipo
Central	(1, 1)	(1, 1)	4
Quartetos	(2, 1)	(3, 1)	4
	(3, 1)	(4, 4)	1
	(4, 1)	(2, 2)	5
	(5, 1)	(3, 3)	4
	(6, 1)	(5, 1)	4
	(7, 1)	(5, 5)	2
	(2, 2)	(4, 1)	4
	(3, 3)	(6, 1)	2
	(4, 4)	(7, 1)	2
	(5, 5)	(2, 1)	6
Octetos	(3, 2)	(5, 4)	5
	(4, 2)	(6, 2)	3
	(5, 2)	(4, 3)	6
	(6, 2)	(5, 3)	6
	(7, 2)	(6, 4)	2
	(4, 3)	(6, 3)	3
	(5, 3)	(5, 2)	4
	(6, 3)	(7, 2)	5
	(5, 4)	(3, 2)	4
	(6, 4)	(4, 2)	4

poucos bits em \mathcal{J} . Dependendo das escolhas de Q ou O , um elemento combustível que está “funcionando” bem próximo ao núcleo pode ser deslocado para a periferia pela modificação de um único bit. Também existe a possibilidade de grupamentos de elementos combustíveis, que “funcionam” bem quando juntos, serem desfeitos, também, por uma pequena modificação em \mathcal{J} .

Dessa forma, a ordem escolhida em Q e O tem o poder de tornar o espaço de busca \mathcal{B} mais (ou menos) suave. A busca de ordenações em Q e O que torna \mathcal{B} o mais suave possível é um problema relacionado muito interessante e muito mais complexo do que aparenta. Neste trabalho, esse problema não será considerado e Q e O serão os mesmos das equações (5.4) e (5.5).

5.4 Resultados

O RECNOD leva um tempo da ordem de 1 segundo para simular a queima do combustível nuclear a partir de cada configuração de núcleo. Mesmo parecendo pouco, com centenas de milhares de simulações (uma para cada avaliação da aptidão), a execução do FPBIL pode durar por vários dias. Por conseqüência, apenas uns poucos testes puderam ser realizados. A seguir são descritos três destes testes que se destacaram.

No primeiro teste, foram utilizados 6 bits por dente de chave na representação *random keys*, resultando em indivíduos com 120 bits. A aptidão bruta utilizada foi

$$\mathcal{A}_b(\mathcal{J}) = \begin{cases} p_{rm}(\mathcal{J}) & \text{se } p_{rm}(\mathcal{J}) > 1,395; \\ \frac{1}{C_B(\mathcal{J})} & \text{se } p_{rm}(\mathcal{J}) \leq 1,395, \end{cases} \quad (5.6)$$

a mesma utilizada em [23]. Pode-se perceber que sempre que $p_{rm}(\mathcal{J}) > 1,395$, existe uma penalização, já que os valores de $C_B(\mathcal{J})$ estão em torno de 1.000. Como

o objetivo é minimizar $\mathcal{A}_b(\mathcal{S})$, a aptidão padrão será igual à bruta,

$$\mathcal{A}_p(\mathcal{S}) = \mathcal{A}_b(\mathcal{S}). \quad (5.7)$$

A evolução da aptidão bruta é mostrada na figura 5.3. Como pode ser verificado,

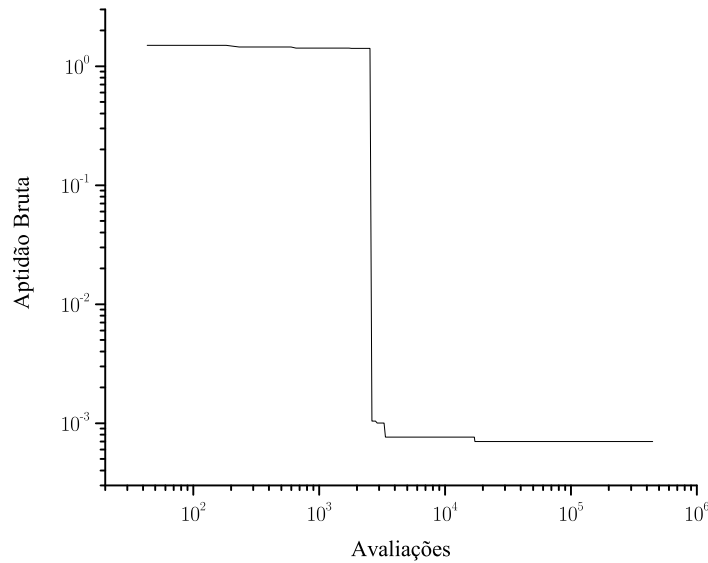


Figura 5.3: Evolução de $\mathcal{A}_b(\mathcal{S})$ para o teste 1.

a primeira configuração de núcleo não penalizada surgiu um pouco depois de 2.000 avaliações da aptidão e a melhor solução surge um pouco antes de 200.000 avaliações. A figura 5.4 na próxima página mostra as melhores soluções ao final de cada geração. A linha tracejada representa o limiar de 1.395. Todas as soluções acima dessa linha são penalizadas. Nota-se que são poucas as soluções que se destacam. A tabela 5.7 na página seguinte lista as soluções (não dominadas) não penalizadas com $C_B(\mathcal{S}) \geq 1.300$, com destaque para a melhor solução.

Como pode ser verificado na equação (5.6) na página precedente, a penalização funciona somente na direção de $p_{rm}(\mathcal{S})$ e é muito acentuada — um salto da ordem de 1.000.

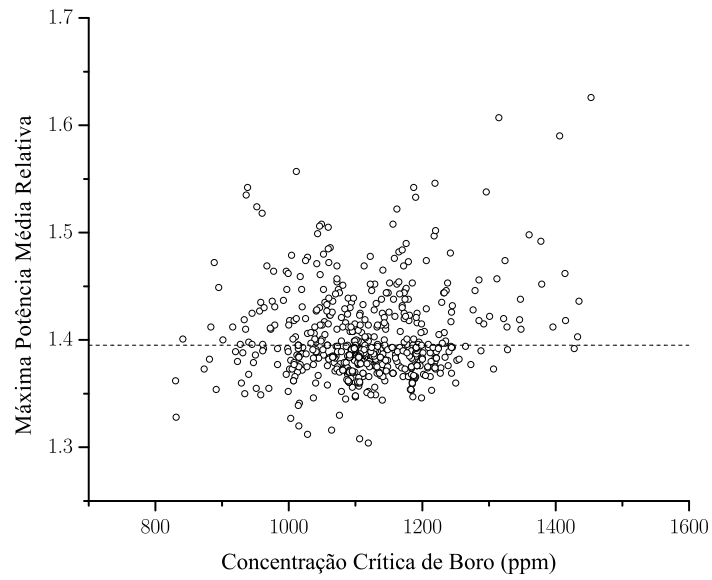


Figura 5.4: Melhores soluções ao final de cada geração do teste 1.

Tabela 5.7: Soluções não penalizadas do teste 1 com $C_B(\mathcal{I}) \geq 1.300$.

Geração	$C_B(\text{ppm})$	p_{rm}
47	1307	1,373
145	1428	1,392
850	1328	1,391

O segundo teste utilizou uma aptidão bruta diferente:

$$\mathcal{A}_b(\mathcal{J}) = \begin{cases} C_B(\mathcal{J}) & \text{se } p_{rm}(\mathcal{J}) \leq 1,395; \\ \frac{C_B(\mathcal{J})}{2} e^{-\frac{p_{rm}(\mathcal{J}) - p_{lim}}{\rho}} & \text{se } p_{rm}(\mathcal{J}) > 1,395, \end{cases} \quad (5.8)$$

com $p_{lim} = 1,395$ e $\rho = p_{lim}/2$. Com isso, a descontinuidade entre a região penalizada e a não penalizada é sempre $C_B(\mathcal{J})/2$ e a penalização se dá, também, no sentido de aumentar o valor de $C_B(\mathcal{J})$. A aptidão padrão utilizada foi

$$\mathcal{A}_p(\mathcal{J}) = 15.000 - \mathcal{A}_b(\mathcal{J}). \quad (5.9)$$

Dessa vez, foram utilizados 4 bits por dente de chave, resultando em indivíduos com 80 bits. A evolução da aptidão bruta é ilustrada na figura 5.5 e a figura 5.6 na

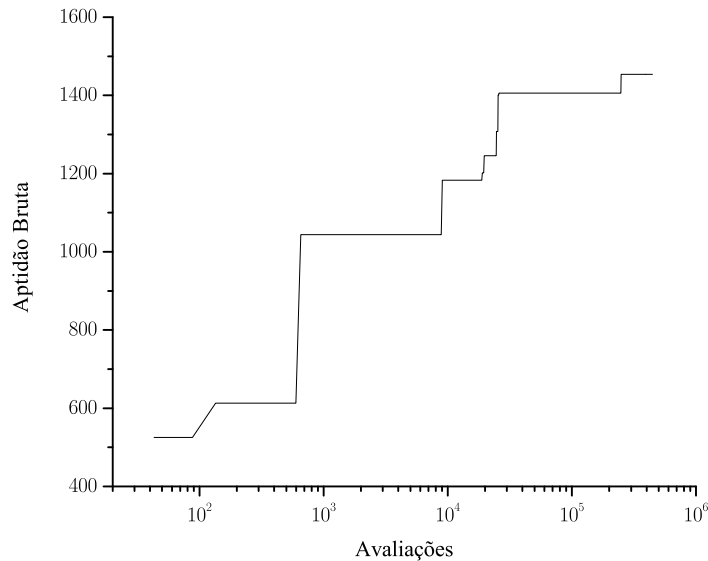


Figura 5.5: Evolução de $\mathcal{A}_b(\mathcal{J})$ para o teste 2.

próxima página mostra as melhores soluções ao final de cada geração. A distribuição das soluções é visivelmente diferente da do teste anterior. A tabela 5.8 na página 87 lista as soluções (não dominadas) não penalizadas com $C_B(\mathcal{J}) \geq 1.300$, onde a

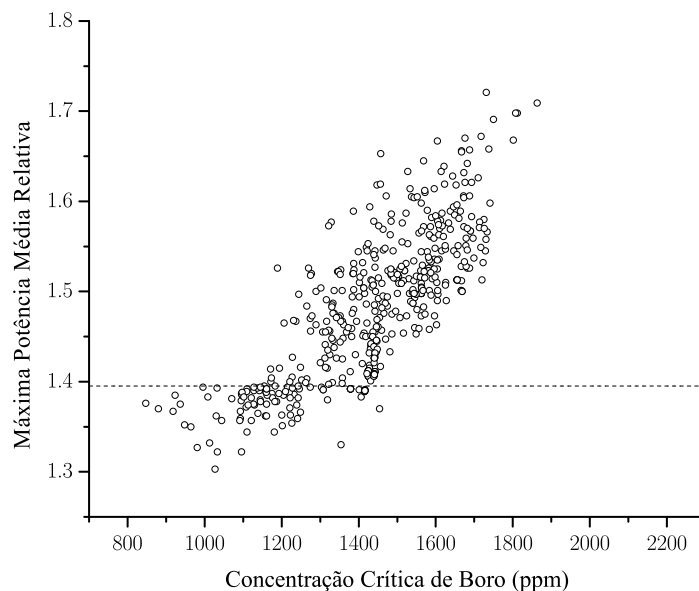


Figura 5.6: Melhores soluções ao final de cada geração do teste 2.

melhor solução é destacada.

No terceiro e último teste apresentado neste trabalho, encontrou-se a melhor solução com $p_{rm}(\mathcal{S}) \leq 1,395$. Para tanto, utilizou-se a mesma aptidão e o mesmo número de bits por dente de chave do teste anterior. A evolução da aptidão bruta é ilustrada na figura 5.7 e a figura 5.8 mostra as melhores soluções ao final de cada geração.

Observa-se que nestes dois últimos testes, em particular, a aptidão bruta apresenta uma tendência a continuar crescendo, significando que possíveis melhores soluções podem ainda ser alcançadas com o uso do FPBIL. A tabela 5.9 na página 88 apresenta as soluções (não dominadas) não penalizadas com $C_B(\mathcal{S}) \geq 1.300$, com destaque para a melhor solução.

A título de curiosidade, a melhor configuração de núcleo encontrada pelo FPBIL é mostrada na figura 5.9, onde os números correspondem aos tipos da tabela 5.1.

Tabela 5.8: Soluções não penalizadas do teste 2 com $C_B(\mathcal{I}) \geq 1.300$.

Geração	$C_B(\text{ppm})$	p_{rm}
198	1401	1,391
202	1406	1,383
203	1319	1,380
210	1303	1,394
234	1379	1,392
805	1454	1,370
822	1354	1,330
963	1416	1,390
968	1415	1,389

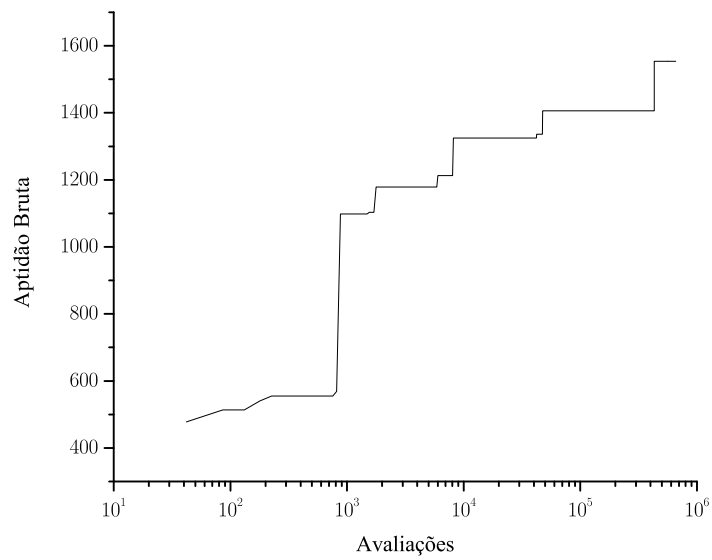


Figura 5.7: Evolução de $\mathcal{A}_b(\mathcal{I})$ para o teste 3.

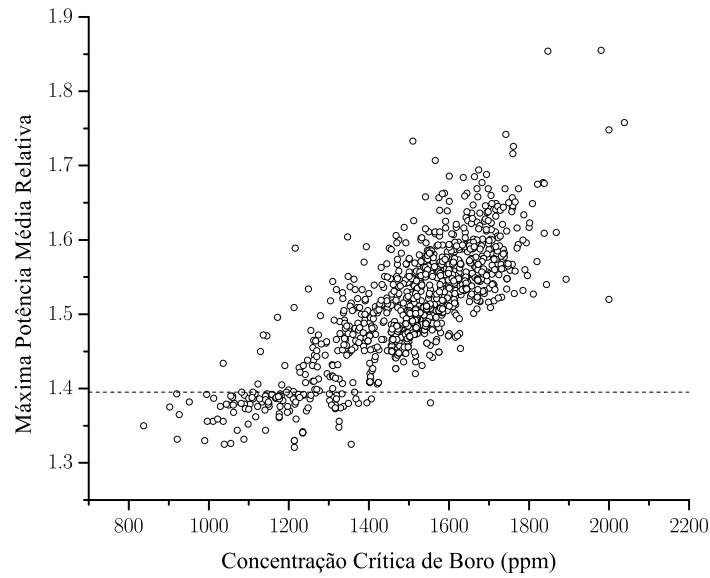


Figura 5.8: Melhores soluções ao final de cada geração do teste 3.

Tabela 5.9: Soluções não penalizadas do teste 3 com $C_B(\mathcal{I}) \geq 1.300$.

Geração	$C_B(\text{ppm})$	p_{rm}
90	1325	1,348
231	1336	1,376
249	1406	1,386
272	1364	1,395
285	1356	1,325
354	1333	1,393
686	1374	1,380
745	1329	1,374
760	1326	1,356
763	1310	1,388
798	1316	1,373
858	1554	1,381
987	1302	1,384
1078	1307	1,378

4					
4	4				
4	5	2			
2	2	3	1		
2	3	6	4	4	
5	6	4	4		
6	5				

Figura 5.9: A melhor configuração de núcleo encontrada pelo FPBIL.

Entretanto, as sutilezas de tal configuração podem ser melhor percebidas somente quando o núcleo é mostrado por inteiro, como na figura 5.10 na página seguinte. Uma característica clara desta configuração é que os elementos combustíveis dos tipos 4, 5 e 6 são posicionados no interior e na periferia do núcleo, enquanto que os de tipos 1, 2 e 3 preenchem a região intermediária.

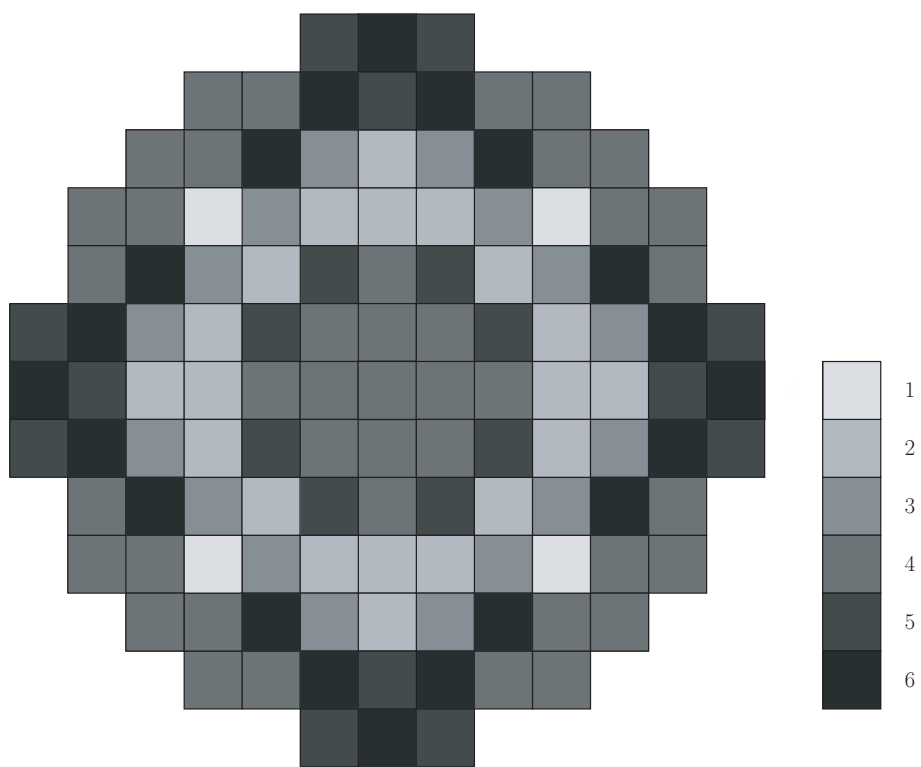


Figura 5.10: Núcleo construído a partir da figura 5.9.

5.5 Comparação com Outros Trabalhos

O problema da recarga nuclear, mais especificamente a maximização do ciclo 7 de Angra 1, foi alvo de diferentes técnicas de otimização, tornando-se um *benchmark* (a nível nacional). Esta seção destina-se a comparar o desempenho do FPBIL, frente a essas outras técnicas. A tabela 5.10 mostra os dados e os correspondentes valores

Tabela 5.10: Comparação do desempenho de diferentes técnicas aplicadas à maximização do ciclo 7 de Angra 1.

Ano	Autor	C_B (ppm)	p_{rm}^a	Técnica	Heur.	Avaliações
–	–	955	1,345	Manual [24]	–	–
2000	Chapot [24]	1026	1,390	AG	não	4.000
2002	Machado, L. [9]	1297	1,384	ANT-Q	sim	200
2005	Machado, M. [25]	1242	1,361	PBIL_N	não	6.000
2005	Machado, M. [25]	1305	1,349	PBIL_MO	não	10.000
2005	Lima [23]	1424	1,386	RCCA	sim	329.000
2006	Presente Trabalho	1554	1,381	FPBIL	não	430.364

^a F_{XY} , para a otimização manual.

de C_B e p_{rm} são plotados na figura 5.11 na página seguinte.

Esses resultados são os melhores encontrados com as respectivas técnicas pelos seus autores, de modo que as escolhas das funções objetivo e dos eventuais parâmetros correspondentes às diferentes técnicas não tem necessariamente relações entre si.

Uma característica interessante da solução encontrada pelo FPBIL é que, apesar do valor elevado de C_B , p_{rm} encontra-se razoavelmente abaixo de 1,395. Já a solução encontrada pelo PBIL_MO (PBIL Multi_Objeto) é a que encontra, como esperado, um valor relativamente alto de C_B ao mesmo tempo em que mantém p_{rm} bem mais baixo. As soluções encontradas pelo PBIL_MO são na realidade frentes de Pareto [25]. O resultado apresentado correspondendo a essa técnica é apenas o ponto da frente de Pareto abaixo da linha $p_{rm} = 1,395$ com maior valor de C_B .

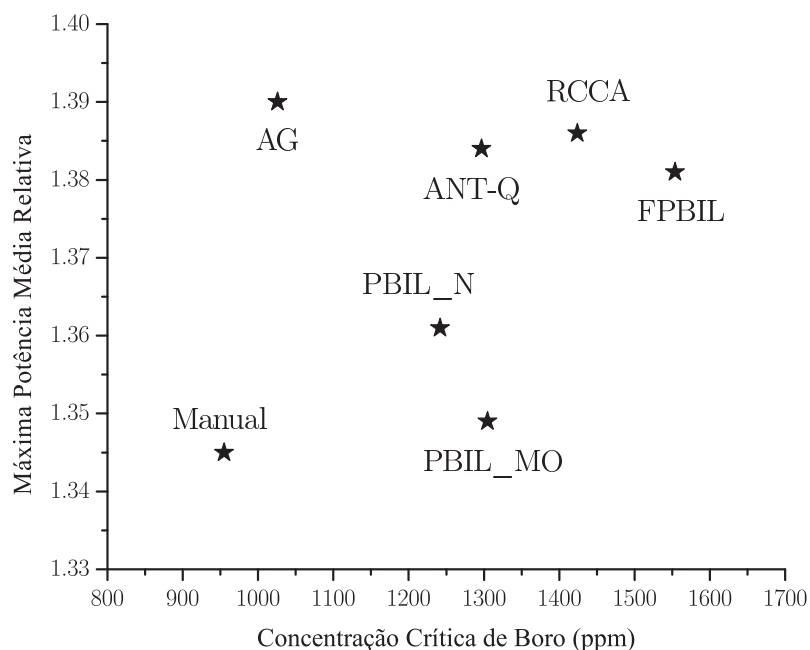


Figura 5.11: Melhor resultado das diferentes técnicas.

Outra informação relevante com relação a maximização do ciclo 7 de Angra 1 é a quantidade de Dias Efetivos à Plena Potência,

$$1 \text{ DEPP} = 625 \text{ MWe} \cdot 24 \text{ h} = 15.000 \text{ MWeh}, \quad (5.10)$$

que é a energia produzida em 24h com o reator funcionando à potência nominal, 625MWe. Angra 1 consome aproximadamente 4ppm de boro solúvel a cada DEPP [24] e, portanto, é possível computar durante quantos DEPP Angra 1 consegue produzir energia para cada configuração nuclear diferente. Nesta mesma linha, também é possível calcular o valor arrecadado com a venda de energia de um DEPP, bastando, para tanto, multiplicar 1DEPP por

$$T = \text{R\$ } 98,64/\text{MWeh}, \quad (5.11)$$

que é a tarifa estabelecida pela ANEEL para o ano de 2006 [29]. Tem-se, portanto

que

$$1 \text{ DEPP} \mapsto T \cdot \text{DEPP} = \text{R\$ } 1.479.600,00, \quad (5.12)$$


ou seja, cada dia, a mais, de geração de energia que uma configuração de núcleo atinge, corresponde a um ganho de R\$1.479.600,00. A tabela 5.11 apresenta o número de DEPP, a mais, em relação a otimização manual e a receita, a mais, correspondente.

Tabela 5.11: DEEP e Receita, a mais, em relação a otimização manual.

Técnica	DEPP – DEPP _{Manual}	$T \cdot (\text{DEPP} - \text{DEPP}_{\text{Manual}})$ (em R\$)
Manual	0,00	0,00
AG	17,75	26.262.900,00
ANT-Q	85,50	126.505.800,00
PBIL_N	71,75	106.161.300,00
PBIL_MO	87,50	129.465.000,00
RCCA	117,25	173.483.100,00
FPBIL	149,75	221.570.100,00

A solução encontrada pelo FPBIL corresponde a 149,75 DEEP a mais, comparando com a otimização manual — realizada por especialistas — o que representa cerca de 5 meses à plena potência. Comparando com o segundo melhor resultado, o FPBIL garante 32,5 DEEP a mais, uma economia de R\$ 48.087.000,00. Se a comparação for feita com o PBIL_MO (PBIL Multi_Objetivo), o segundo melhor resultado sem uso de heurísticas, o FPBIL obtém 62,25 DEEP a mais, economizando R\$ 92.105.100,00.

6 *Conclusão*

 ODE-SE afirmar, em conclusão a este trabalho, que o FPBIL é um algoritmo evolucionário, competitivo com as melhores técnicas atuais de otimização, compacto, relativamente modesto no uso de recursos computacionais — especialmente memória —, bem fundamentado, eficiente, robusto, auto-adaptável, simples e sem parâmetros.

O bom desempenho do FPBIL em problemas combinatórios ficou evidente com o problema da recarga nuclear, avaliado e discutido no capítulo 5. Além disso, os testes realizados no capítulo 4, mostram que o FPBIL também se destaca em problemas numéricos e deceptivos.

Com relação ao algoritmo em si, o FPBIL é conceitualmente simples por ser intuitivo e não requer conhecimento muito sofisticado; é compacto no sentido de que pode ser programado com poucas linhas de código; e utiliza pouca memória, uma vez que não há necessidade de armazenar os indivíduos de uma população em alguma estrutura de dados.

A forma radicalmente diferente da mutação observada no FPBIL é fundamentada na distribuição de probabilidades inerente ao próprio vetor de probabilidades, sendo, este último, atualizado com a utilização de toda a informação disponível em cada geração. Tais modificações possibilitam ao FPBIL adquirir características auto-ajustáveis — como o mecanismo de população de tamanho variável — que tornam o algoritmo mais eficiente e mais robusto. Eficiente no sentido de que encontra soluções em menos tempo; robusto, significando que possui mais recursos para escapar

de ótimos locais.

Com relação ao fato de ser o FPBIL livre de parâmetros, é importante enfatizar que se pode optar por abrir mão das reinicializações e ter \mathcal{P}_0 como único parâmetro. Esta pode ser uma alternativa plausível, mas sujeita a eventuais riscos. Por exemplo, um problema pode ser mais fácil do que aparenta, causando desperdício de tempo; ou mais difícil, resultando no fracasso da busca. A questão é que o processo de ajuste de qualquer parâmetro implica em um tempo precioso de processamento que, somado ao tempo de execução propriamente dito, conduz à utilização de mais tempo que quaisquer tentativas do FPBIL, tal como proposto.

Com relação especificamente ao problema da recarga nuclear, existem questões ainda abertas que, quando solucionadas, devem possibilitar uma busca mais profunda no correspondente espaço de busca. Uma dessas questões já foi considerada no capítulo 5 e diz respeito à determinação da ordem em que os elementos de quarteto e de octeto devem ser inseridos no núcleo do reator, a partir da configuração estabelecida pela representação *random keys*, de modo a suavizar ao máximo o espaço de busca.

Uma outra questão, também tratada no capítulo 5, está relacionada diretamente à representação *random keys*, mais especificamente no que se refere ao número ideal de bits por dente de chave. Deve haver um número ótimo, já que um número pequeno de bits por dente de chave resulta em grande número de empates, enquanto que um número grande desses bits destrói uma significativa parte da bijetividade entre \mathcal{S}_B e $\check{\mathcal{H}}_n$, criando uma grande região de planície. Ambos os extremos mencionados dificultam a busca. Para agravar ainda mais a determinação do valor ideal do número de bits por dente de chave, vale lembrar que diferentes valores dessa grandeza produzem \mathcal{H}_n de tamanhos diferentes, o que também influencia na dificuldade da busca por uma solução do problema.

Finalmente, uma outra questão importante a ser considerada é avaliar até que ponto a utilização do código de Gray é mais vantajosa. Certamente os resultados

obtidos com o código de Gray são superiores, quando comparados aos obtidos com a representação binária convencional. Todavia, intuitivamente, a representação binária convencional deveria ser a melhor, uma vez que essa faz uma distinção muito clara entre números pequenos e grandes — os grandes começam com 1's e os pequenos, com 0's.

Uma possibilidade é que a representação binária convencional faça diferença no início do algoritmo, quando os primeiros e últimos elementos podem ser separados rapidamente; enquanto que o código de Gray faça diferença no final do algoritmo, possibilitando um ajuste fino na ordem dos elementos, com a alteração de poucos bits. Isto sugere que o processo de busca pode ser aperfeiçoado utilizando-se simultaneamente os dois tipos de codificação.

Com a proposição do FPBIL, espera-se ter contribuído para questões teóricas e práticas pertinentes, apresentando melhorias técnicas viáveis com uma contrapartida econômica considerável, tanto em relação a seu custo quanto a seu benefício.

Existem, entretanto, avanços que podem, ainda, ser incorporados ao FPBIL. Depois de escapar de ótimos locais, o FPBIL costuma se aproximar mais lentamente do ótimo global que outros algoritmos — como, por exemplo, o PBIL original. Considerando o processo como um todo, o FPBIL leva vantagem, mas talvez seja possível combinar o FPBIL com algum algoritmo de busca rápida, resultando em um algoritmo ainda mais eficiente.

Outras melhorias, além das já citadas, podem surgir construindo-se um FPBIL multiobjetivo — adaptando-se as técnicas de [25] — ou, até mesmo, um FPBIL paralelo — com base nas técnicas de [23]. Pode-se, ainda, tentar incorporar algum tipo de heurística ao FPBIL — talvez alguma descrita em [23], para o problema específico da recarga nuclear. Trabalhos nessas direções provam que, com o emprego dessas técnicas, costuma-se gerar soluções melhores.

Há ainda muito o que se fazer e pesquisar no que se refere aos sistemas de otimização. Devido a restrições de tempo e escopo, esse trabalho apenas propicia uma

breve discussão no campo dos algoritmos não parametrizados e reforça a necessidade e a validade de se continuar estudando o assunto.

Referências

- [1] GOLDBERG, D. E. *GENETIC ALGORITHMS in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [2] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Segunda Edição. Massachusetts: MIT Press, 1992.
- [3] DARWIN, C. R. *On The Origin of Species by Means of Natural Selection*. London: John Murray, 1859.
- [4] BALUJA, S. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Pittsburgh, PA, 1994. Disponível em: <<http://citeseer.ist.psu.edu/baluja94population.html>>.
- [5] BALUJA, S. *An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics*. [S.l.], 1995. Disponível em: <<http://citeseer.ist.psu.edu/baluja95empirical.html>>.
- [6] MACHADO, M. D. *Um Novo Algoritmo Evolucionário com Aprendizado LVQ para a Otimização de Problemas Combinatórios como a Recarga de Reatores Nucleares*. Dissertação (Mestrado) — COPPE/UFRJ, Rio de Janeiro, 1999.
- [7] BALUJA, S.; CARUANA, R. Removing the Genetics from the Standard Genetic Algorithm. In: PRIEDITIS, A.; RUSSEL, S. (Ed.). *The Int. Conf. on Machine Learning 1995*. San Mateo, CA: Morgan Kaufmann Publishers, 1995. p. 38–46. Disponível em: <<http://citeseer.ist.psu.edu/baluja95removing.html>>.
- [8] BALUJA, S.; DAVIES, S. Fast Probabilistic Modeling for Combinatorial Optimization. In: *AAAI/IAAI*. [s.n.], 1998. p. 469–476. Disponível em: <citeseer.ist.psu.edu/baluja98fast.html>.
- [9] MACHADO, L.; SCHIRRU, R. The Ant-Q algorithm applied to the nuclear reload problem. *Annals of Nuclear Energy*, v. 29, n. 12, p. 1455–1470, 2002.

- [10] KENNEDY, J.; EBERHART, R. Particle Swarm Optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1995. p. 1942–1945.
- [11] BANZHAF, W. et al. *Genetic Programming - An Introduction*. San Francisco: Morgan Kaufmann Publishers, Inc., 1998.
- [12] KOZA, J. R. *Genetic Programming - On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press, 1992.
- [13] JANIKOW, C. Z.; WEESE, S. W. de. *CGP lil-gp 2.1;1.02 User's Manual*. [S.l.].
- [14] CALDAS, G. H. F.; SCHIRRU, R. Híbrido de programação genética com variante de PBIL. In: *2005 International Nuclear Atlantic Conference - INAC 2005*. [S.l.: s.n.], 2005.
- [15] JUELS, A.; BALUJA, S.; SINCLAIR, A. *The Equilibrium Genetic Algorithm and the Role of Crossover*. Disponível em: <<http://citeseer.ist.psu.edu/juels93equilibrium.html>>.
- [16] KNUTH, D. E. *The Art of Computer Programming*. Pre-fascicle 2A. Stanford: Addison-Wesley, 2002.
- [17] GILL, P. E.; MURRAY, W.; WRIGHT, M. H. *Practical Optimization*. San Diego: Academic Press, 1981.
- [18] DEVLIN, K. *Os Problemas do Milênio*. Rio de Janeiro: Record, 2004.
- [19] LEWIS, H. R.; PAPADIMITRIOU, C. H. *Elementos de teoria da Computação*. Segunda edição. Porto Alegre: Bookman, 2000.
- [20] TSPLIB - A Library of Traveling Salesman Problem And Related Problem Instances. Disponível em: <<http://nhse.cs.rice.edu/softlib/catalog/tsplib/tsp/>>.
- [21] BEAN, J. C. “Genetic Algorithms and Random Keys for Sequencing and Optimization”. *ORSA Journal on Computing*, v. 6, n. 2, 1994.
- [22] SCHIRRU, R.; PEREIRA, C. M. N. A. [S.l.].
- [23] LIMA, A. M. M. de. *Recarga de Reatores Nucleares Utilizando Redes Conectivas de Colônias Artificiais*. Tese (Doutorado) — COPPE/UFRJ, Rio de Janeiro, 2005.
- [24] CHAPOT, J. L. C. *Otimização Automática de Recargas de Reatores a Água Pressurizada Utilizando Algoritmos Genéticos*. Tese (Doutorado) — COPPE/UFRJ, Rio de Janeiro, 2000.

- [25] MACHADO, M. D. *Algoritmo Evolucionário PBIL Multi_Objetivo Aplicado ao Problema da Recarga de Reatores Nucleares*. Tese (Doutorado) — COPPE/UFRJ, Rio de Janeiro, 2005.
- [26] LIU, Y. S. et al. *ANC: A Westinghouse Advanced Nodal Computer Code*. [S.l.], 1985.
- [27] MONTAGNINI, B. et al. “A well-Balanced Coarse Mesh Flux Expansion Method”. *Annals of Nuclear Energy*, v. 21, n. 11, p. 45–53, 1994.
- [28] LANGENBUCH, S.; MAURER, W.; WERNER, W. “Coarse Mesh Flux Expansion Method for Analysis of Space-Time Effects in Large Water Reactor Cores”. *Nuclear Science and Engineering*, n. 63, p. 437–456, 1977.
- [29] Disponível em: <<http://www.aneel.gov.br>>.

Índice Remissivo

- $\#$, 8
 α , 12
 Cuidado no ajuste de, 22
 \mathcal{A} , 11
 \mathcal{A}_a , 43
 \mathcal{A}_b , 42
 \mathcal{A}_i , 17
 $\langle \mathcal{A} \rangle$, 17
 $\varnothing \langle \mathcal{A} \rangle$, 19
 \mathcal{A}_p , 42
 β , 12
 Cuidado no ajuste de, 22
 \mathcal{B} , 5
 c , 23
 Flutuação em, 32
 c' , 24
 C_B , 75
 $\Delta \langle c \rangle_G$, 35
 $\langle c \rangle_G$, 34
 $\mathbf{d}^{\mathcal{I}}$, 7
 d , 23
 Ótimo, 24
 d_c , 24
 $\frac{\partial \mathcal{P}_G}{\partial \mathcal{G}}$, 18
 δ , 21
 δ_r , 21
 \mathcal{D} , 17
 $d_k^{\mathcal{I}}$, 7
 $D.M.$, 13
 F_{XY} , 75
 \mathcal{G} , 11
 γ , 13
 \mathcal{H}_n , 5
 $\check{\mathcal{H}}_n$, 5
 $\check{\mathcal{H}}$
 Ordem de, 8
 I_k , 5
 \mathcal{I} , 5
 \mathcal{I}^+ , 11
 \mathcal{I}^- , 11
 \mathcal{I}^\dagger , 11
 $\mathcal{I}[k]$, 5
 k , 28
 $\mathcal{M}_{\mathcal{S}_B}^n$, 5
 \mathcal{O} , 7
 p_k , 7
 \mathcal{P} , 9
 \mathcal{P}_0 , 28
 \mathcal{P}_c , 28
 p_{rm} , 77
 \mathcal{P} , 7
 \mathcal{P}_0 , 9
 $\mathcal{P}(\mathcal{I})$, 8
 $\mathcal{P}[k]$, 7
 \mathcal{S}_B , 5
 $\vartheta(\mathcal{P}_G)$, 22
 $\mathcal{V}_{\mathcal{H}_n}$, 7
 ξ , 18, 21
 Algoritmos
 Evolucionários, 1
 Genéticos, 5, 9
 ANC, 76

ANT-Q, 1
 Aptidão, 11
 Ajustada, 43
 Propriedades, 43
 Bruta, 42
 padrão, 42
 Procedimento para construção da,
 42
 Banana, o problema, 52–56
 Código
 de Gray, 37–41
 RECNOD, 76
 Central
 Elemento, 73
 Chave, 60
 Ciclo de operação, 72
 Concentração crítica de Boro, 75
 Dente de chave, 60
 DEPP, 92
 Desvio absoluto médio, 21
 Diagonais
 Eixos, 73
 Dias Efetivos à Plena Potência, 92
 Eixos
 Diagonais, 73
 Principais, 73
 Elemento
 de octeto, 73
 de quarteto, 73
 Central, 73
 Elementos combustíveis, 73
 Espaço de Busca, 5
 Estrutura do PBIL, 5
 Evolucionários, algoritmos, 1
 Fator de pico de potência radial do nú-
 cleo, 75
 Flutuação em c , 32
 FPBIL, 2, 14–35
 Atualização do vetor de probabili-
 dades, 16, 22
 Tamanho da população no, 28
 Velocidade no, 19, 20
 FPBIL*, 36
 Função
 Aptidão, 11
 de *Rosenbrock*, 52
 Genéticos
 Algoritmos, 5, 9
 Geração, 11
 Gray
 Código de, 37–41
 Hiper-cubo, 5
 Hiperplanos, 7
 Indivíduo
 Ótimo, 11
 Mais apto, 11
 Menos apto, 11
 Indivíduos, 11
 Máxima potência média relativa, 77
 Motivação para o trabalho, 2
 Mutação, 12
 Função da, 22
 No FPBIL, 23
 Núcleo de Angra 1
 Simetrias, 73
 Octantes, 73
 Octeto
 Elemento de, 73
 Ordem de $\tilde{\mathcal{H}}$, 8
 PBIL, 1, 5–13

Atualização do vetor de probabilidades, 12
 Estrutura, 5
 Mutação, 12
 Original, 9
 Terminologia, 9
 Velocidade no, 18
 PBIL*, 36
 PCV Rykel48, 58–67
 População, 11
 Escolha do tamanho da, 27
 Potência média relativa, Máxima, 77
 Principais
 Eixos, 73
 Problema
 Banana, 52–56
 da recarga nuclear, 72–93
 PCV Rykel48, 58–67
 Quatro picos, 45–51
 PSO, 1
 Quadrantes, 73
 Quarteto
 Elemento de, 73
 Quatro picos, 45–51
Random keys, 59–61
 Chave, 60
 Dente de chave, 60
 Recarga nuclear
 Problema da, 72–93
 RECNOD, 76
 Representação *random keys*, 59–61
 Chave, 60
 Dente de chave, 60
Rosenbrock
 Função de, 52
 Simetrias
 do núcleo de Angra 1, 73
 Terminologia do PBIL, 9
 Vareta combustível, 73
 Velocidade, 18