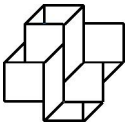


Laboratório Nacional de Computação Científica

Programa de Pós Graduação em Modelagem Computacional

Otimização de Funções Contínuas Usando Algoritmos Quânticos

Por

Pedro Carlos da Silva Lara

PETRÓPOLIS, RJ - BRASIL

MAIO DE 2015

OTIMIZAÇÃO DE FUNÇÕES CONTÍNUAS USANDO ALGORITMOS QUÂNTICOS

Pedro Carlos da Silva Lara

TESE SUBMETIDA AO CORPO DOCENTE DO LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM MODELAGEM COMPUTACIONAL

Aprovada por:

Prof. Renato Portugal, D.Sc

(Presidente)

Prof. Gilson Antonio Giraldi, D.Sc.

Prof. Carlile Campos Lavor, D.Sc.

Prof. Franklin de Lima Marquezino, D.Sc.

PETRÓPOLIS, RJ - BRASIL
MAIO DE 2015

Lara, Pedro Carlos da Silva

XXXX otimização de funções contínuas usando algoritmos quânticos / Pedro Carlos da Silva Lara. Petropólis, RJ. : Laboratório Nacional de Computação Científica, 2015.

xx, yy p. : il.; 29 cm

Orientador: Renato Portugal

Tese (D.Sc.) – Laboratório Nacional de Computação Científica, 2015.

1. Algoritmos de Busca. 2. Otimização Global. 3. Passeios Quânticos. 4. Computação Quântica. I. Portugal, Renato. II. LNCC/MCT. III. Título.

CDD XXX.XXX

Feliz aquele que transfere o que sabe e aprende o que ensina.

Cora Coralina

Dedicatória

A minha mãe, pelo apoio incondicional e pelo incentivo a busca de novos conhecimentos.

Agradecimentos

Agradeço ao meu orientador professor Renato Portugal e ao professor Carlile Campos Lavor pelo apoio crucial no desenvolvimento e elaboração desta tese.

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

OTIMIZAÇÃO DE FUNÇÕES CONTÍNUAS USANDO ALGORITMOS QUÂNTICOS

Pedro Carlos da Silva Lara

Maio , 2015

Orientador: Renato Portugal, D.Sc

Algoritmos de otimização são conhecidos por apresentarem uma vasta gama de aplicações em diversas áreas do conhecimento. Desta forma, qualquer melhoria no desempenho dos algoritmos de otimização gera grande impacto na resolução de diversos problemas. Neste sentido, este trabalho introduz a área de algoritmos quânticos para a otimização global (maximização/minimização) de funções contínuas através de diferentes métodos quânticos de busca e algoritmos clássicos de otimização local. Neste caso, a utilização de algoritmos quânticos de busca está diretamente associada ao desempenho com relação ao método clássico: usando um computador quântico pode-se encontrar um elemento em um banco de dados não-ordenado usando apenas $O(\sqrt{N})$ consultas.

Abstract of Thesis presented to LNCC/MCT as a partial fulfillment of the requirements for the degree of Doctor of Sciences (D.Sc.)

QUANTUM CONTINUOUS FUNCTION OPTIMIZATION ALGORITHMS

Pedro Carlos da Silva Lara

May, 2015

Advisor: Renato Portugal, D.Sc

Optimization algorithms are known to have a wide range of applications in various areas of knowledge. Thus, any improvement in the performance of optimization algorithms generate great impact in solving various problems. Thus, this work indroduces the area of quantum algorithms for global optimization (maximization/minimization) of continuous functions through different quantum search methods and classical local optimization algorithms. In this case, the use of search quantum algorithms is tied directly to performance with respect to the classical method: using a quantum computer can find an element in an unsorted database using only $O(\sqrt{N})$ queries.

Sumário

Lista de Figuras

Figura

Lista de Tabelas

Tabela

Lista de Algoritmos

Capítulo 1

Introdução

O trabalho seminal de Grover (1996) mostrou que através de um computador quântico pode-se encontrar um elemento em um banco de dados não-ordenado usando $O(\sqrt{N})$ consultas. Anteriormente ao trabalho de L. Grover, o Algoritmo de Shor (1994), teria despertado interesse da comunidade acadêmica por resolver eficientemente, a partir de um computador quântico, o Problema da Fatoração de Inteiros (PFI) e o Problema do Logaritmo Discreto (PLD). O PFI e o PLD atualmente são bases para a segurança de diversos esquemas e protocolos de criptografia tais como Diffie e Hellman (2006), RSA (Rivest et al. (1978)), métodos baseados em curvas elípticas (Koblitz (1987); Miller (1986)), entre outros. Resolver o PFI e o PLD eficientemente significa aposentar estes métodos criptográficos. No entanto, o Algoritmo de Grover possui uma aplicabilidade substancialmente maior: qualquer problema que teria que ser resolvido por uma busca linear teria um *speedup* quadrático usando um computador quântico. Assim, o trabalho de L. Grover despertou ainda mais o interesse em computação quântica exibindo uma aplicação geral de computadores quânticos e posteriormente foi comprovado o *lower bound* do algoritmo. Além disso o Algoritmo de Grover foi comprovadamente implementado com sucesso (Chuang et al. (1998)). Pouco tempo depois de publicado o trabalho de L. Grover, Boyer et al. (1996) generalizaram o problema para mais de um elemento procurado. Eles também abordaram o problema de quando não se conhece o número de elementos procurados e comprovaram a complexidade do problema

também tendo uma vasta importância prática. Desta forma, usando o conceito de amplificação de amplitude, o Algoritmo de Grover inspirou uma grande variedade de outros algoritmos quânticos.

Influenciados pelo trabalho Boyer et al. (1996) os autores Dürr e Høyer (1996) introduziram a área de minimização usando algoritmos quânticos. Posteriormente, Baritompa et al. (2005) corrigiram algumas constantes do trabalho de Dürr e Høyer (1996) e propôs um *framework* para a utilização do Algoritmo de Grover na otimização de uma lista discreta de valores não-ordenados.

Este trabalho descreve o primeiro algoritmo quântico para a otimização de funções contínuas Lara et al. (2014b) e uma versão que utiliza como ferramenta de busca algoritmos baseados em caminhadas quânticas. Neste caso, foi utilizada uma estratégia híbrida: uma parte do algoritmo executa um procedimento quântico e outra parte executa um método clássico.

Este trabalho está divido da seguinte forma: No Capítulo 2 faremos uma breve revisão bibliográfica do Algoritmo de Grover e a sua respectiva generalização. No Capítulo 3 iremos expor os principais algoritmos para o problema da minimização em uma lista finita não-ordenada. No Capítulo 4 trataremos a otimização global de funções contínuas através de algoritmos híbridos (clássico-quântico). No Capítulo 5 abordaremos o problema da otimização de funções contínuas usando algoritmos de busca baseados em passeios quânticos. Finalmente, no Capítulo 5 colocaremos nossas conclusões e trabalhos futuros.

Capítulo 2

O Algoritmo de Grover e sua Generalização

Considere o seguinte problema: seja $A[0,\ldots,N-1]$ uma tabela de elementos sem ordenação e queremos encontrar um elemento $x_0 \in A$. Na verdade, desejamos encontrar um índice $i_0 \in \{0,\ldots,N-1\}$ tal que $A[i_0] = x_0$. A restrição é que só exista um único elemento procurado x_0, ou seja, não existem repetições. Com um computador clássico tentaríamos, sem muitas variações, uma estratégia do tipo (Algoritmo 1).

Algoritmo 1: Busca clássica em uma tabela sem ordenação

Input: A tabela $A[0,\ldots,N-1]$ e o elemento procurado x_0.
Output: O índice i_0 tal que $A[i_0] = x_0$.

```
begin
    for i ∈ {0,...,N−1} do
        if A[i] = x_0 then
            return i;
```

Este algoritmo também é conhecido como algoritmo de força bruta ou busca linear. Como não existe nenhuma informação ou estrutura por trás de $A[0,\ldots,N-1]$ que possa acelerar a busca por i_0 o algoritmo testa todas as possibilidades até encontrá-lo. Obviamente este algoritmo possui complexidade $O(N)$ e naturalmente uma ordenação em $A[0,\ldots,N-1]$ tornaria a complexidade para o problema da busca $O(\log N)$. O leitor poderá perceber que se trata de um problema de busca

completamente geral. Ou seja, uma busca sem qualquer restrição em sua definição salvo o fato de existir apenas um elemento procurado x_0. Assim, este mesmo problema poderá ser reescrito e aplicado em diversas áreas, tais como: quebra de códigos criptográficos, otimização, programação linear, genética computacional entre outras. Desta forma, acelerar este algoritmo impacta diretamente em diversas áreas da ciência. L. Grover (1996) propôs um algoritmo de busca quântico cuja complexidade é $O(\sqrt{N})$. Além disso, foi demonstrado por Bennett et al. (1997) que mesmo com um computador quântico não conseguiríamos nada melhor que $\Omega(\sqrt{N})$ consultas para este tipo de problema. Este trabalho incipiente mostrou a aplicabilidade em problemas gerais da computação quântica.

Iremos indroduzir um formalismo que facilita a explanação do Algoritmo de Grover. Seja

$$f : \{0, \ldots, N-1\} \to \{0, 1\}$$

tal que

$$f(i) = \begin{cases} 1, \text{se } A[i] = x_0; \\ 0, \text{se } A[i] \neq x_0; \end{cases}$$

Onde x_0 é o elemento procurado na tabela $A[0, \ldots, N-1]$ e seja $N = 2^n$ para $n \geq 1$. A escolha de $N = 2^n$ não tira a generalidade do problema. Na literatura a função f é conhecida como *função oráculo*. A complexidade do algoritmo é dada pelo número de vezes que se consulta a função f. O algoritmo trabalha com dois registradores, o primeiro com n qubits, inicializado no estado $|00\ldots0\rangle$ e o segundo com 1 qubit inicializado no estado $|1\rangle$. A primeira parte do algoritmo consiste em aplicar o operador de Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

em cada qubit da entrada, ou seja, no primeiro registrador temos

$$|\psi\rangle = H^{\otimes n}|00\ldots0\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle,$$

e o segundo registrador temos

$$|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right).$$

Agora, precisamos de um operador unitário que atue identificando o elemento marcado utilizado a função f definida anteriormente. Para isso, podemos usar a seguinte definição

$$U_f|i\rangle|r\rangle = |i\rangle|f(i)\oplus r\rangle,$$

onde $|i\rangle$ é o estado quântico de n qubits e $|r\rangle$ é o estado de 1 qubit. O símbolo $\oplus$ representa a operação de OU-exclusivo. O operador U_f atua invertendo o segundo registrador caso o primeiro seja o elemento procurado. Caso o primeiro registrador de n qubits não seja o elemento marcado, então o operador U_f não modifica a entrada. Esta é uma receita bastante comum de ser utilizar uma função qualquer, neste caso f, através de um operador unitário. A ideia é aplicar o operador U_f em $|\psi\rangle|-\rangle$, assim

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}U_f(|i\rangle|-\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\frac{1}{\sqrt{2}}\Big(U_f(|i\rangle|0\rangle) - U_f(|i\rangle|1\rangle)\Big)$$

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\Big(\sum_{i=0,i\neq i_0}^{N-1}\frac{1}{\sqrt{2}}\Big(|i\rangle|0\rangle - |i\rangle|1\rangle\Big) - \frac{1}{\sqrt{2}}\Big(|i_0\rangle|0\rangle - |i_0\rangle|1\rangle\Big)\Big)$$

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{N}}\Big(\Big(\sum_{i=0,i\neq i_0}^{N-1}|i\rangle|-\rangle\Big) - |i_0\rangle|-\rangle\Big) = \frac{1}{\sqrt{N}}\Big(\sum_{i=0}^{N-1}(-1)^{f(i)}|i\rangle\Big)|-\rangle.$$

Ou seja, U_f é uma reflexão em torno do espaço ortogonal a $|i_0\rangle|-\rangle$. Assim o

operador U_f pode ser escrito como

$$U_f = I - 2|i_0, -\rangle\langle i_0, -|.$$

Em outras palavras,

$$U_f(|i\rangle|-\rangle) = \begin{cases} -|i\rangle|-\rangle, \text{se } i = i_0; \\ |i\rangle|-\rangle, \text{se } i \neq i_0; \end{cases}$$

Uma representação geométrica da aplicação de U_f em $|\psi\rangle|-\rangle$ pode ser vista na Figura C.1. O vetor $|i_0\rangle|-\rangle$ é o elemento marcado e $|i_0\rangle|-\rangle^\perp$ é ortogonal a $|i_0\rangle|-\rangle$. A aplicação de U_f reflete $|\psi\rangle|-\rangle$ com relação a $|i_0\rangle|-\rangle^\perp$.

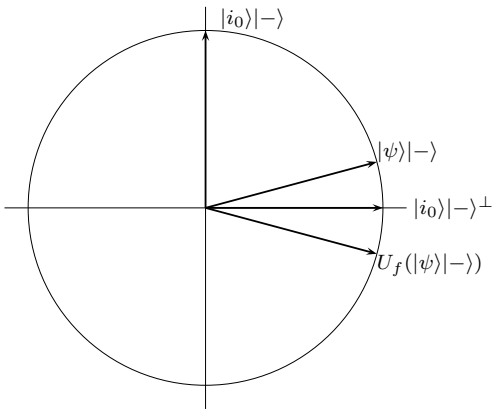


Figura 2.1: Representação geométrica do operador U_f.

Observe que após aplicação do operador U_f o segundo registrador se mantém invariante. O seu estado continua sendo $|-\rangle$, no entanto, não quer dizer que ele seja desnecessário. Por questão de conveniência iremos omitir o segundo registrador uma vez que ele sempre se manterá no estado $|-\rangle$. O operador U_f identifica o elemento marcado x_0 invertendo o sinal do estado. Mas isso ainda não é suficiente: se fizermos uma medida iremos obter o estado procurado com probabilidade

$$|\langle x_0|\psi\rangle|^2 = \Big(-\frac{1}{\sqrt{N}}\Big)^2 = \frac{1}{N}.$$

Neste caso todos os estados da base computacional possui a mesma amplitude de probabilidade igual a $\frac{1}{N}$.

O objetivo do Algoritmo de Grover é encontrar um elemento marcado. Para isso, utiliza um estado em superposição e aplica o conceito de paralelismo quântico (Nielsen e Chuang (2011)) para identificar o elemento marcado a partir do operador U_f. Observando a Figura C.1 tem-se que o estado $|\psi\rangle$ deve se aproximar do estado $|i_0\rangle$ a cada iteração para que o algoritmo seja bem sucedido. Agora considere a operação de reflexão do estado $U_f|\psi\rangle$ com relação ao vetor $|\psi\rangle$. Esta operação torna o ângulo entre o vetor $U_f|\psi\rangle$ e $|i_0\rangle$ menor (veja a Figura 2.2).

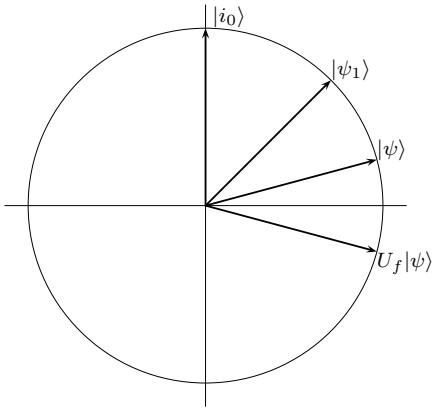


Figura 2.2: Representação geométrica do operador R_ψ.

O vetor $|\psi_1\rangle$ é o resultado desta nova reflexão em torno do vetor $|\psi\rangle$. Algebricamente podemos escrever esta operação como

$$R_\psi = 2|\psi\rangle\langle\psi| - I.$$

A aplicação de U_f em $|\psi\rangle$ pode ser escrita como:

$$U_f|\psi\rangle = |\psi\rangle - \frac{2}{\sqrt{N}}|i_0\rangle \tag{2.1}$$

Aplicando R_ψ na parte da esquerda da Equação 2.1 temos:

$$R_\psi\Big(|\psi\rangle - \frac{2}{\sqrt{N}}|i_0\rangle\Big) = (2|\psi\rangle\langle\psi| - I)\Big(|\psi\rangle - \frac{2}{\sqrt{N}}|i_0\rangle\Big)$$

$$R_\psi\Big(|\psi\rangle - \frac{2}{\sqrt{N}}|i_0\rangle\Big) = \frac{N-4}{N}|\psi\rangle + \frac{2}{\sqrt{N}}|i_0\rangle \tag{2.2}$$

Observe que o a replexão R_ψ atua apenas no primeiro registrador de n qubits. Chamaremos de rotação de Grover, a composição dos dois operadores vistos anteriormente, ou seja

$$G = (R_\psi \otimes I)U_f. \tag{2.3}$$

Assim o Algoritmo de Grover se resume a aplicar uma determinada quantidade de vezes o operador G ao estado $|\psi\rangle$. Após isso medir o estado do primeiro registador e obter o índice do elemento marcado.

Seja $|\psi_j\rangle = G^j|\psi\rangle$. De Boyer et al. (1996) temos que o estado $|\psi_j\rangle$ pode ser escrito como

$$|\psi_j\rangle = \sum_{i=0, i\neq i_0}^{N-1} l_j|i\rangle + k_j|i_0\rangle$$

onde

$$k_j = \sin((2j+1)\theta), \tag{2.4}$$

$$l_j = \frac{1}{\sqrt{N-1}}\cos((2j+1)\theta) \tag{2.5}$$

e $\sin^2\theta = \frac{1}{N}$. Agora precisamos determinar a quantidade de vezes que se aplica o operador G de Grover no estado $|\psi\rangle$. Ou seja, precisamos determinar o momento certo para a medida do primeiro registrador. Assim, queremos encontrar um m tal que $k_m = 1$. Usando a Equação 2.4 temos

$$1 = \sin((2m+1)\theta)$$

$$m = \frac{\pi}{4\theta} - \frac{1}{2}. \tag{2.6}$$

Lembrando que $\theta = \arcsin \frac{1}{\sqrt{N}}$ e usando o fato que para valores pequenos de x temos a seguinte aproximação $\sin x \simeq x$. Assim $\theta = \arcsin \frac{1}{\sqrt{N}} \simeq \frac{1}{\sqrt{N}}$. Substituindo θ em 2.6 temos

$$m = \frac{\pi}{4}\sqrt{N} - \frac{1}{2}. \tag{2.7}$$

Naturalmente precisamos de um número inteiro de iterações. Assim, bastaria tomar $m = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$. Substituindo $m = \frac{\pi}{4}\sqrt{N}$ na Equação 2.4 temos a seguinte relação:

$$\sin\Big(\Big(\frac{2\pi\sqrt{N}}{4} + 1\Big)\frac{1}{\sqrt{N}}\Big) = k_m$$

$$\sin\Big(\frac{\pi}{2} + \frac{1}{\sqrt{N}}\Big) = k_m.$$

Usando a relação trigonométrica $\sin(A + B) = \sin A \cos B + \sin B \cos A$ temos

$$\cos\Big(\frac{1}{\sqrt{N}}\Big) = k_m.$$

Para valores pequenos de x temos que $\cos x \simeq 1 - \frac{x^2}{2}$. Assim

$$1 - \frac{1}{2N} \simeq k_m.$$

Na verdade, a probabilidade de se obter i_0 após uma medida é dada por k_m^2. Assim esta probabilidade pode ser bem aproximada por $k_m^2 \simeq 1 - \frac{1}{N} + \frac{1}{4N^2}$. Em temos assintóticos podemos escrever

$$k_m^2 \simeq 1 - O\Big(\frac{1}{N}\Big).$$

Isto quer dizer que temos que aplicar $m = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ vezes o operador $G = (R_\psi \otimes I)U_f$ no estado $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|-\rangle$ para ter uma probabilidade de sucesso de $k_m^2 \simeq 1 - O\Big(\frac{1}{N}\Big)$ de encontrar o elemento marcado. Os passos para o Algoritmo de Grover é dado pelo Algoritmo 2.

Algoritmo 2: Busca quântica em uma tabela sem ordenação

Input: A tabela $A[0, \ldots, N-1]$ e o elemento procurado x_0 onde $N = 2^n$.
Output: O índice i_0 tal que $A[i_0] = x_0$.
begin
  Criar os registradores $|00\ldots0\rangle|1\rangle$
  Aplicar Hadamard em cada qubit gerando $|\psi_0\rangle \leftarrow \frac{1}{\sqrt{N}}\Big(\sum_{i=0}^{N-1}|i\rangle\Big)|-\rangle$
  $m \leftarrow \lfloor \frac{\pi}{4}\sqrt{N}\} \rfloor$
  for $j \in \{1, \ldots, m\}$ **do**
    $|\psi_j\rangle \leftarrow U_f|\psi_{j-1}\rangle$;
    $|\psi_j\rangle \leftarrow R_{\psi_j}|\psi_j\rangle$;
  Medir $|\psi_m\rangle$ em i
  return i

2.1 Múltiplos Elementos Marcados

Pouco tempo depois da publicação do Algoritmo de Grover os autores Boyer et al. (1996) publicaram uma solução para o problema de múltiplos elementos marcados. Agora o problema poderá ser posto conforme o Algoritmo 3.

Aqui o conjunto dos elementos marcados será representado por $\mathcal{M}$ e sua cardinalidade é dada por M. Assim, existem M valores de i tais que $A[i]$ é um elemento marcado. A principio, iremos assumir que a cardinalidade M é conhecida a priori. Seja $\mathcal{A} = \{i|A[i] \in M\}$ e $\mathcal{B} = \{i|A[i] \notin M\}$ podemos rescrever o estado $|\psi\rangle$ como:

$$|\psi\rangle = \sum_{i\in\mathcal{A}} k|i\rangle + \sum_{i\in\mathcal{B}} l|i\rangle. \tag{2.8}$$

Assim, a relação de unitariedade $Mk^2 + (N-M)l^2 = 1$ é válida. A aplicação do operador de Grover, Equação 2.3, resulta na seguinte relação de recorrência

$$k_j = \frac{1}{\sqrt{M}}\sin((2j+1)\theta) \tag{2.9}$$

$$l_j = \frac{1}{\sqrt{N-M}}\cos((2j+1)\theta). \tag{2.10}$$

O ângulo θ é escolhido de forma que $\sin^2\theta = \frac{M}{N}$. Neste caso estamos consideranto o conhecimento prévio do número de elementos marcados M. Agora queremos encontrar um m tal que $l_m \simeq 0$. Assim como para o caso onde temos

Algoritmo 3: Busca clássica de múltiplos elementos marcados em uma tabela sem ordenação

Input: A tabela $A[0,\ldots,N-1]$ e os elementos procurados $\mathcal{M}=\{x_0,\ldots,x_{M-1}\}$.
Output: Um índice i tal que $A[i]=x_j$ para algum $j\in\{0,\ldots,M-1\}$.
begin
  for $i\in\{0,\ldots,N-1\}$ **do**
    if $A[i]\in\mathcal{M}$ **then**
      return i;

um elemento marcado, o valor que torma a Equação 2.10 igual a zero é dado por

$$m=\frac{\pi}{4\theta}-\frac{1}{2}. \tag{2.11}$$

Neste caso também tomamos um número inteiro para o valor de m. A análise segue conforme feita na seção anterior. Assim $m=\lfloor\frac{\pi}{4}\sqrt{\frac{N}{M}}\rfloor$ e a probabilidade de se encontrar pelo menos um elemento marcado é dada por $k_m^2=1-O\left(\frac{M}{N}\right)$. Se tivermos $M\ll N$ o valor $O\left(\frac{M}{N}\right)$ é desprezível.

2.2 Um caso especial quando $N=4M$

Obviamente quanto temos $N=4M$ um computador clássico encontraria eficientemente um elemento marcado. Mas não seria tão eficiente quanto o caso quântico. Veja o valor que l_j assume na primeira iteração

$$l_1=\frac{1}{\sqrt{3M}}\cos 3\theta$$

como $\sin^2\theta=\frac{M}{N}$ temos que $\theta=\pi/6$ temos que $l_1=0$. Ou seja, após uma iteração do operador de Grover certamente encontraremos um elemento marcado. Assim, ele é mais rápido que qualquer versão clássica probabilística para este problema.

2.3 Busca com o número de elementos marcados desconhecido

Nesta seção iremos abordar o problema da busca onde o valor de M é desconhecido. Para colocar o problema, imagine que tenhamos uma lista com $N=2^{16}$

elementos. Precisamos encontrar pelo menos um elemento marcado e não conhecemos quantos elementos marcados existem. Vamos supor que iremos aplicar $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor = 201$ vezes o operador de Grover. Se tivermos $M = 15$ elementos marcados a probabilidade de se encontrar um elemento, após uma medida é de apenas $0,0342$. A Figura 2.3 exibe a probabilidade de sucesso com relação ao número de aplicações do operador de Grover.

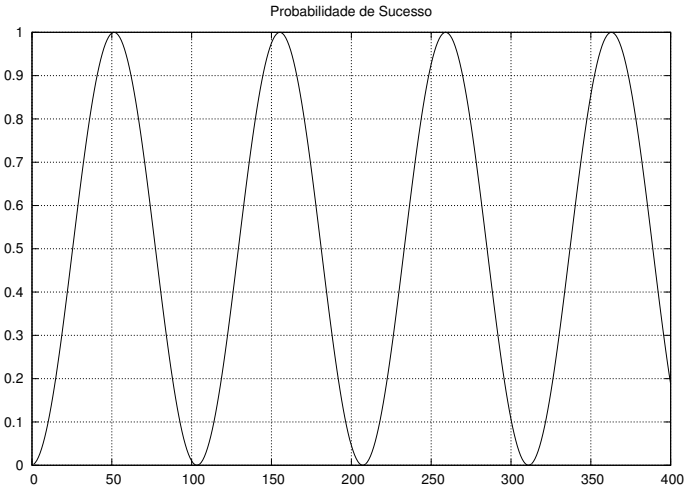


Figura 2.3: Probabilidade de sucesso com relação ao número de aplicações do operador de Grover usando $N = 2^{16}$ sendo 15 elementos marcados.

Para obter o algoritmo quando não conhecemos o valor de M iremos colocar o seguinte lema (Boyer et al. (1996)).

Lema 2.3.0.1 Seja M o valor desconhecido do número de elementos marcados e seja θ o ângulo tal que $\sin^2\theta = M/N$. Seja m um inteiro positivo e j um valor escolhido aleatoriamente no intervalor $[0, \ldots, m-1]$. Se observarmos o primeiro registrador após aplicar j vezes o operador de Grover no estado $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$ teremos a seguinte probabilidade de sucesso:

$$P_m = \frac{1}{2} - \frac{\sin(4m\theta)}{4m\sin(2\theta)}.$$

Em particular, quando $m \geq 1/\sin(2\theta)$ então $P_m \geq 1/4$.

Prova: A probabilidade de sucesso, após j aplicações do operador de Grover é dada

por $\sin^2((2j+1)\theta)$. A probabilidade de sucesso quando j é escolhido no intervalo $[0,\ldots,m-1]$ é dada pela média de todos possíveis valores de j, ou seja:

$$P_m = \frac{1}{m}\sum_{j=0}^{m-1}\sin^2((2j+1)\theta)$$

$$P_m = \frac{1}{2m}\sum_{j=0}^{m-1}1-\cos((2j+1)2\theta)$$

$$P_m = \frac{1}{2}-\frac{1}{2m}\sum_{j=0}^{m-1}\cos((2j+1)2\theta).$$

Usando a relação trigonométrica

$$\sum_{j=0}^{m-1}\cos((2j+1)2\theta) = \frac{\sin(4m\theta)}{2\sin 2\theta}$$

temos que

$$P_m = \frac{1}{2}-\frac{\sin(4m\theta)}{4m\sin 2\theta}.$$

Em particular, se $m \geq 1/\sin(2\theta)$ então

$$\frac{\sin(4m\theta)}{4m\sin 2\theta} \leq \frac{1}{4m\sin 2\theta} \leq \frac{1}{4}$$

■

O Algoritmo 4 busca um elemento marcado sem conhecer o valor M e possui complexidade $O\left(\sqrt{\frac{N}{M}}\right)$. A demonstração da complexidade do Algoritmo 4 pode ser encontrada no trabalho Boyer et al. (1996).

Algoritmo 4: Busca quântica de múltiplos elementos marcados em uma tabela sem ordenação sem conhecer o valor de M.

Input: A tabela $A[0, \ldots, N-1]$ e a função f (função oráculo).
Output: Um índice i tal que $A[i] = x_j$ para algum $j \in \{0, \ldots, M-1\}$.
begin
    $m \leftarrow 1$
    Escolha λ no intervalo $1 < \lambda < 4/3$
    while TRUE do
        Escolha aleatoriamente um inteiro j no intervalo $[0, \ldots, m-1]$
        Aplique j vezes o operador de grover no estado $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$
        Meça o estado em i
        if $f(i) = 1$ **then**
            return i
        $m \leftarrow \min(\lambda m, \sqrt{N})$

Capítulo 3

Otimização Quântica Discreta

Influenciados fortemente pelo trabalho Boyer et al. (1996) os autores Dürr e Høyer (1996) introduziram a área de otimização/minimização usando algoritmos quânticos. O Algoritmo Dürr-Høyer, como ficou conhecido, utiliza a generalização do algoritmo de Grover (BBHT) para encontrar um mínimo de uma lista finita não-ordenada. Posteriormente, o trabalho Baritompa et al. (2005) propôs um *framework* denominado GAS (Grover Adaptative Search) para a otimização global usando o Algoritmo de Grover. Eles também apresentaram um novo algoritmo, aqui será denominado BBW, que usa uma sequência previamente definida de rotações de Grover para executar a otimização global. Além disso, estes autores corrigiram os valores de algumas constantes apresentadas no trabalho Dürr e Høyer (1996). Posteriormente os autores Liu e Koehler (2010) apresentaram discreto ganho na modificação do Algoritmo BBW.

Classicamente, o problema da minimização em uma lista finita não-ordenada poderá ser resolvido da seguinte forma: seja L uma lista de N elementos cuja operação $<$ é bem definida. O Algoritmo 5 encontra o mínimo desta lista em $O(N)$.

O Algoritmo 5 procura pelo índice do menor elemento da lista L. Um problema semelhante e bastante pertinente seria busca por um elemento entre os M menores elementos. Neste caso a minimização possui complexidade $O(N/M)$.

Algoritmo 5: Minimização discreta clássica

Input: A lista L
Output: i_0 tal que $L[i_0] = \min\{L\}$.
begin
    $i_0 = 1$;
    $curr\_min = L[i_0]$;
    for $i = 2$ **to** N **do**
        if $L[i] < curr\_min$ **then**
            $i_0 = i$;
            $curr\_min = L[i]$;
    return i_0;

3.0.1 Grover Adaptative Search

Nesta seção será apresentado o algoritmo geral de otimização chamado de Grover Adaptative Search (GAS) introduzido por Bulger et al. (2003). Este algoritmo é caracterizado por requerer como entrada a sequência $\{r_1, r_2, \ldots, r_n, \ldots\}$ de número de rotações de Grover a cada iteração. Desta forma, o desafio do algoritmo é descrever um método adequado para encontrar a sequência de rotações de Grover.

Inicialmente o algoritmo escolhe aleatoriamente um elemento na lista L. Este elemento será chamado de limiar. A cada iteração i é aplicado r_i rotações de Grover marcando todos os elementos que estão abaixo do atual limiar. Veja a ilustração da Figura 3.1. Caso as rotações de Grover retornem algum elemento menor que o limiar, este deverá ser atualizado. O algoritmo termina quando uma determinada condição de parada é atingida. Estas ideias estão descritas no Algoritmo 6.

A função $random(a, b)$ gera um número aleatório no intervalo $[a, b]$. Uma vantagem de se trabalhar com este tipo de método geral é utilizar análises de convergência previamente desenvolvidas (veja Bulger et al. (2004); Bulger e Wood (1998); Wood et al. (2001); Zabinsky et al. (1995); Zabinsky e Smith (1992)).

3.0.2 Algoritmo Dürr-Høyer (DH)

O algoritmo pode ser visto como um caso especial do Algoritmo GAS onde as rotação vão crescendo exponencialmente no intervalo $\{0, \ldots, m\}$ por um fator

Algoritmo 6: Grover Adaptative Search

Input: A lista L e sequência $\{r_1, r_2, \ldots, r_n, \ldots\}$

Output: X tal que $L[X] = \min\{L\}$ com uma probabilidade previamente estabelecida.

begin

$X \leftarrow random(1, N)$;

$Y \leftarrow L[X]$;

for $i = 1, 2, \ldots$ até que a condição de parada seja atingida **do**

Use o Algoritmo BBHT com r_i rotações. A saída será o índice x;

$y \leftarrow L[x]$;

if $y < Y$ **then**

$X \leftarrow x$;

$Y \leftarrow y$;

return X;

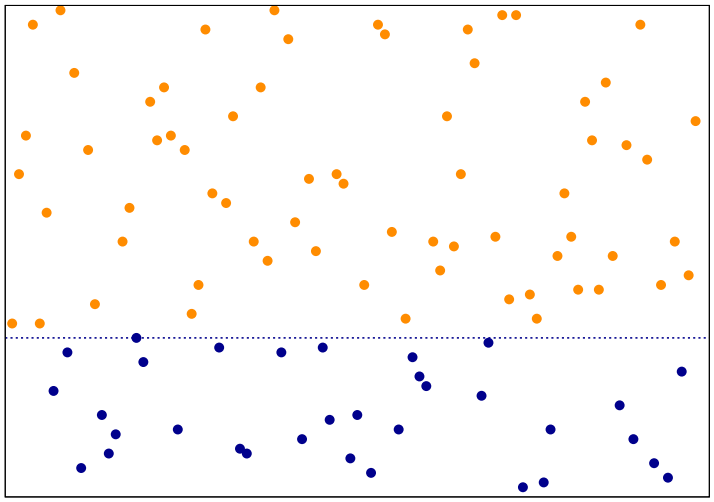


Figura 3.1: Elementos abaixo da linha tracejada (limiar) serão marcados para as rotações de Grover.

λ previamente escolhido. A cada iteração, o algoritmo tenta descobrir algum elemento abaixo do limiar. Se o algoritmo encontra o processo se reinicia fazendo $m = 1$. Caso contrário, o algoritmo aumenta o intervalo, multiplicando m por um fator λ, para a seleção aleatória do número de rotações. O Algoritmo 7 usa $22.5\sqrt{N} + 1.4 \log^2 N$ aplicações da rotação de Grover e precisa de $1.32\sqrt{N}$ rotações de Grover para encontrar um novo limiar no caso médio. Não existe a possibilidade, usando um computador quântico, da busca (não-ordenada) seja mais rápida que $O(\sqrt{N})$. Isto foi provado por Bennett et al. (1997) e Zalka (1999). Desta forma, a única esperança é reduzir o prefator.

As Figuras 3.2 e 3.3 confirmam o valor de $\lambda = 1.34$ sugerido por Baritompa et al. (2005) para busca de 1% e 0.2% dos melhores elementos respectivamente. Para esta simulação usamos uma lista de 1024 valores inteiros aleatórios. Rodamos o Algoritmo DH 100000 vezes e extraímos a média. A medida de *effort* apresentada nos gráficos é determinada pela quantidade de rotações de Grover e medidas quânticas acumuladas. Já o eixo y exibe a probabilidade de sucesso para o algoritmo. Os gráficos das Figuras 3.2 e 3.3 são conhecidos como *performance graphs* (gráficos de desempenho) (Baritompa et al. (2005)).

Algoritmo 7: Algoritmo Dürr-Høyer

Input: A lista L e sequência $\{r_1, r_2, \ldots, r_n, \ldots\}$
Output: X tal que $L[X] = \min\{L\}$ com uma probabilidade próxima de 1.
begin
    $m \leftarrow 1$;
    Escolha um valor para o parâmetro λ;
    $X \leftarrow random(1, N)$;
    $Y \leftarrow L[X]$;
    for $i = 1, 2, \ldots$ até que a condição de parada seja atingida **do**
        $r \leftarrow random(0, \lceil m - 1 \rceil)$;
        Use o Algoritmo BBHT com r rotações. A saída será o índice x;
        if $y < Y$ **then**
            $X \leftarrow x$;
            $Y \leftarrow y$;
            $m \leftarrow 1$;
        else
            $m \leftarrow \lambda m$;
    return X;

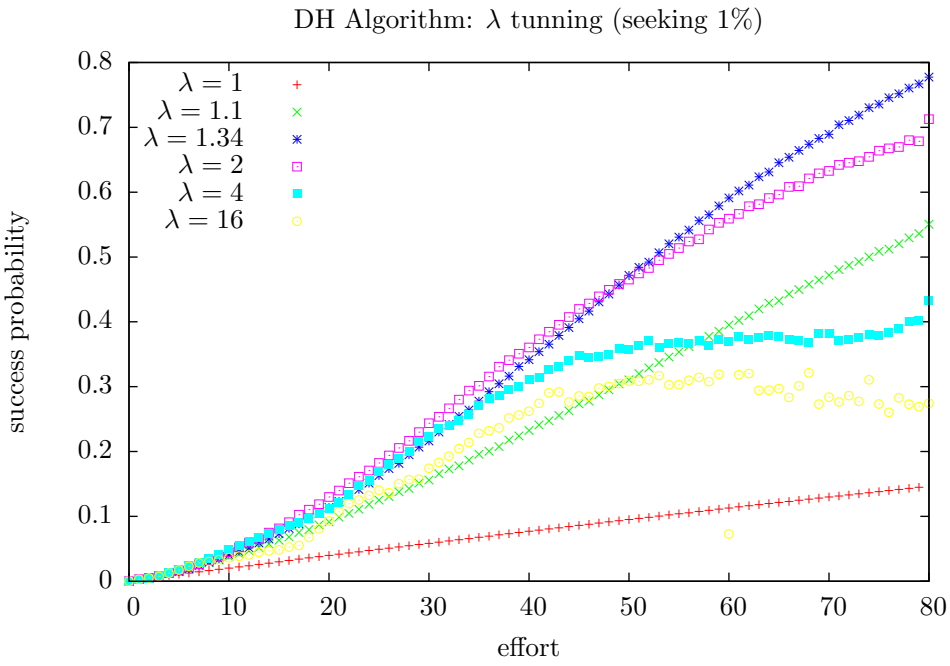


Figura 3.2: Simulação do Algoritmo DH com diferentes valores de λ. Neste caso, estamos buscando por 1% dos melhores elementos. Confirmando o valor de $\lambda = 1.34$ reportado em Baritompa et al. (2005)

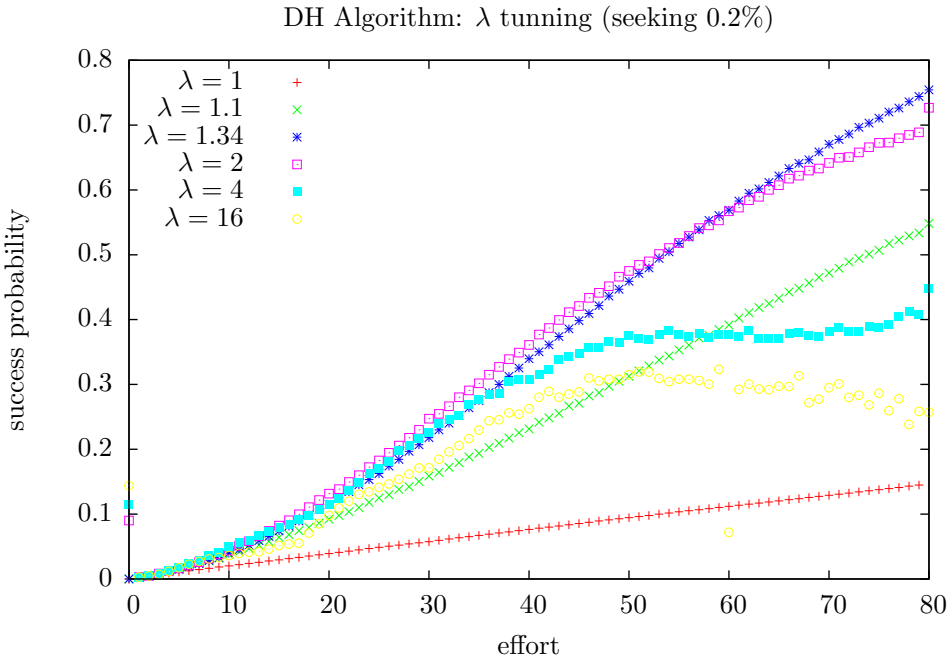


Figura 3.3: Simulação do Algoritmo DH com diferentes valores de λ. Busca por pelo menos um dos 0.02% melhores elementos. Confirmando o valor de $\lambda = 1.34$ reportado em Baritompa et al. (2005)

3.0.3 Algoritmo BBW

Os autores Baritompa et al. (2005) propuseram um método estático baseado em cadeias de Markov não-homogêneas para obter o número de rotações a cada iteração do Algoritmo 6 (GAS). Este processo está descrito no Algoritmo 8. Além disso, o algoritmo usa a função $g_r(t)$ que retorna a probabilidade de sucesso no Algoritmo BBHT.

Algoritmo 8: Algoritmo de obtenção da sequência de rotações para o Algoritmo BBW

Output: Sequência $\mathcal{R}$ com as rotações de Grover para o Algoritmo BBW
begin
  $\mathcal{R} \leftarrow ()$;
  $u(y) \leftarrow y$;
  for $i = 1, 2, \ldots$ **do**
    $E_u \leftarrow 1 - \int_0^1 u dy$;
    $b' \leftarrow 0$;
    for $r = 0, 1, \ldots$ **until** $E_u/(r+1) \leq 2b'$ **do**
      $v \leftarrow u + y \int_y^1 \frac{g_r(t)}{t} du(t)$;
      $E_v \leftarrow 1 - \int_0^1 v dy$;
      $b \leftarrow (E_u - E_v)/(r+1)$;
      if $b > b'$ **then**
        $r' \leftarrow r$;
        $b' \leftarrow b$;
        $v' \leftarrow v$;
    $u \leftarrow v'$;
    $\mathcal{R} \leftarrow \mathcal{R} \cup r'$;

Os 33 primeiros valores gerados pelo Algoritmo 8 são (Baritompa et al. (2005)):

$$0, 0, 0, 1, 1, 0, 1, 1, 2, 1, 2, 3, 1, 4, 5, 1, 6, 2, 7,$$

$$9, 11, 13, 16, 5, 20, 24, 28, 34, 2, 41, 49, 4, 60.$$

As comparações de desempenho serão vistas no Capítulo 4 onde será introduzido um novo método de otimização para funções contínuas.

Capítulo 4

Otimização Quântica Contínua

Neste capítulo começaremos a tratar o principal tema desta tese: a otimização de funções contínuas usando algoritmos quânticos e clássicos. Inicialmente o problema da otimização usando algoritmos quânticos foi abordado por Dürr-Høyer tendo como base uma lista discreta de valores não-ordenados. Posteriormente, Baritompa et al. (2005) corrigiu alguns problemas no trabalho de Dürr-Høyer e propôs um método universal para a otimização usando o Algoritmo de Grover. Neste método o desafio está em gerar uma sequência de inteiros que definirá o número de aplicações de Grover.

Este capítulo é uma compilação das publicações Lara et al. (2012) e Lara et al. (2014b) e aborda otimização de funções contínuas usando algoritmos híbridos: clássico e quântico.

4.1 Otimização Global usando algoritmos Clássico-Quânticos

A ideia central do algoritmo é combinar um otimizador local clássico junto com o Algoritmo BBHT quântico através de uma estratégia chamada *Multistart* descrita no Algoritmo 9 (veja a referência Hendrix e Tóth (2010)).

Algoritmo 9: Algoritmo Multistart

Input: A lista L e sequência $\{r_1, r_2, \ldots, r_n, \ldots\}$
Output: X tal que $L[X] = \min\{L\}$ com uma probabilidade próxima de 1.
begin
    $y* \leftarrow \infty$;
    for $i = 1, 2, \ldots$ **até que a condição de parada seja atingida do**
        Gerar aleatoriamente x_i no donínio de f;
        $x' \leftarrow LS(x_i)$;
        if $f(x') < y*$ **then**
            $x* \leftarrow x'$;
            $y' \leftarrow f(x')$;
    return $x*$;

Algoritmo 10: Otimização Global Híbrida

begin
    Gerar x' aleatoriamente no intervalo $\{0, \cdots, N-1\}$;
    $x_0 \leftarrow LS(x')$;
    $y_0 = f(x_0)$;
    $m = \leftarrow 1$;
    $\lambda \leftarrow 1.34$ (conforme sugerido por Baritompa et al. (2005));
    for $i = 1, 2, \cdots$, **até que a condição de parada seja atingida do**
        Defina $M_i = \{x \in \{0, \cdots, N-1\} | f(x) < y_{i-1}\}$;
        Gerar r_i aleatoriamente no intervalo $\{0, \cdots, \lceil m - 1 \rceil\}$;
        Aplicar r_i rotações de Grover;
        Medir quanticamente o estado e seja $x' \in \{0, \cdots, N-1\}$ a saída da medida;
        if $x' \in M_i$ **then**
            $x_i \leftarrow LS(x')$;
            $y_i \leftarrow f(x_i)$;
        else
            $x_i \leftarrow x_{i-1}$;
            $y_i \leftarrow f(x_i)$;
            $m \leftarrow \min\{\lambda\, m, \sqrt{N}\}$;
    return Último valor de x_i;

No Algoritmo 9 definimos $LS(x) : \mathbb{R}^n \to \mathbb{R}$ como um procedimento que recebe um x valor no domínio da função f e retorna a aproximação de um mínimo local. O procedimento $LS(x)$ não precisa necessariamente usar a derivada de $f(x)$, por exemplo, para o procedimento $LS(x)$ pode-se utilizar o método de otimização local Nelder–Mead (Nelder e Mead (1965a)). Agora, se $f(x)$ é diferenciável pode-se utilizar o Método de Newton (Quarteroni et al. (2000)) no procedimento $LS(x)$.

O Algoritmo 9 gera um valor aleatório no domínio de $f(x)$ a cada iteração. Já o Algoritmo 10 (Lara et al. (2014b)) usa o Algoritmo de Grover (BBHT) para encontrar algum valor abaixo do limiar e escapar de mínimos locais. O Algoritmo de Grover recebe como entrada uma lista discreta de valores. Desta forma, para se adequar ao Algoritmo BBHT, temos que discretizar a função $f(x)$ em um domínio previamente definido. A discretização da função $f(x)$ estará associada a um erro ϵ e isto determinará o tamanho da entrada para o Algoritmo de Grover.

Seja $f : \mathcal{D}_{a,b} \to \mathbb{R}$ onde $\mathcal{D}_{a,b} \subset \mathbb{R}^n$ e $\mathcal{D}_{a,b} = \{(x_1, \ldots, x_n) | a \leq x_i \leq b \text{ e } x_i \in \mathbb{R}\}$ e sendo ϵ o erro máximo imposto pela discretização. Assim, o tamanho da lista que será passado para o Algoritmo de Grover será determinado por

$$N = \left(\frac{b-a}{\epsilon} \right)^n . \tag{4.1}$$

A estratégia para definir o número de rotações para o Algoritmo de Grover foi baseada na estratégia descrita no Algoritmo DH porém, com uma diferença importante. A variável m define o intervalo onde serão gerados as rotações de Grover r_i. Este intervalo cresce exponencialmente usando o fator $\lambda = 1.34$ (conforme sugerido por Baritompa et al. (2005)). Existe uma diferença básica entre a estratégia de gerar os valores de r_i usada Algoritmo DH e no Algoritmo 10: no Algoritmo DH quando o Algoritmo de Grover encontra um elemento abaixo do limiar o valor de m é reinicializado para $m = 1$. No Algoritmo 10 isto não ocorre devido ao fato que ao encontrar um elemento abaixo do limiar o algoritmo precisa, no caso geral,

de mais rotações. Ao fazer $m = 1$ o número de rotações será, no caso geral, menor que o número de rotações utilizado para encontrar um elemento abaixo do limiar. O trabalho de Dürr-Høyer parece ter sido fortemente influenciado pelo trabalho BBHT. O Algoritmo DH utiliza como subrotina o Algoritmo BBHT, desta forma, como o Algoritmo BBHT começa (corretamente) fazendo $m = 1$ esta instrução se manteve no Algoritmo DH.

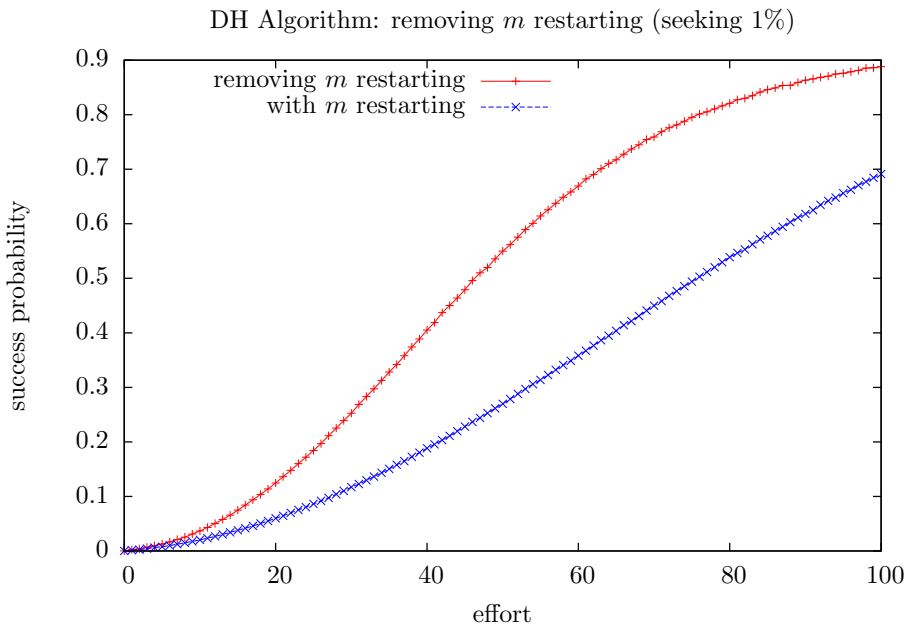


Figura 4.1: Melhoria ao remover a reinicialização de m no Algoritmo DH.

A Figura 4.1 exibe a significante melhoria de eficiência no Algoritmo DH ao se remover a reinicialização de m. Desta forma, o Algoritmo 10 utiliza a mesma estratégia do Algoritmo DH, no entanto, sem reinicializar m.

4.1.1 Condição de Parada

Um ponto fundamental na descrição do Algoritmo 10 é definir a condição de parada. Neste caso, ela será determinada pelo número de avaliações da função objetivo f durante a parte quântica n_1 e durante a parte clássica n_2. No Algoritmo DH, o algoritmo termina imediatamente depois que o número de avaliações da função objetivo fica maior que $2.46\sqrt{N}$. No caso do Algoritmo 10 temos que

destinar um peso diferenciado para a busca local devido sua eficiência. Assim temos a seguite expressão para a condição de parada

$$n_1 + \frac{\sqrt{N}}{\log^n N} n_2 > 2.46\sqrt{N}.$$

Neste caso estamos estimando que a busca local encontre um mínimo local em $O(\log^n N)$. O peso de $\frac{\sqrt{N}}{\log^n N}$ está associado a essa estimativa. Esta condição de parada será avaliada na Seção Resultados Computacionais.

4.2 Resultados Computacionais

Nesta Seção iremos exibir comparações entre o Algoritmo Dürr-Høyer (DH), Algoritmo Baritompa et. al (BBW) e o Algoritmo 10, que chamaremos de LPL (iniciais dos nomes dos autores). Usaremos como *framework* para a comparação os gráficos de desempenho (Hendrix e Klepper (2000)) e tabelas com o número de avaliações da função objetivo. Os gráficos de desempenho medem o esforço computacional contra a probabilidade de sucesso de um determinado médoto. Neste caso, o esforço computacional é dado pelo número de chamadas a função objetivo (número de rotações de Grover) adicionado ao número de medidas quânticas realizadas. Para os experimentos computacionais utilizamos as seguintes funções objetivo e seus respectivos domínios de n variáveis:

(1) Neumaier

$$f(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} (x_i - 1)^2 - \sum_{i=1}^{n-1} x_i x_{i-1},\ 0 \leq x_i \leq 4$$

(2) Griewank

$$f(x_0, \ldots, x_{n-1}) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 - \prod_{i=0}^{n-1} \cos\left(\frac{x_i}{\sqrt{i+1}}\right) + 1,\ -40 \leq x_i \leq 40$$

(3) Shekel

$$f(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{m-1} \frac{1}{c_i + \sum_{j=0}^{n-1}(x_j - a_{ji})^2}, \ -1 \leq x_i \leq 1$$

(4) Rosenbrock

$$f(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-2}(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2, \ -30 \leq x_i \leq 30$$

(5) Michalewicz

$$f(x_0, \ldots, x_{n-1}) = -\sum_{i=0}^{n-1} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right), \ 0 \leq x_i \leq 10$$

(6) Dejong

$$f(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} x_i^2, \ -5.12 \leq x_i \leq 5.12$$

(7) Ackley

$$f(x_0, \ldots, x_{n-1}) = -20 \exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=0}^{n-1}\cos(2\pi x_i)\right) + 20 + \exp(1),$$

$$-15 \leq x_i \leq 20$$

(8) Schwefel

$$f(x_0, \ldots, x_{n-1}) = -\sum_{i=0}^{n-1} x_i \sin\left(\sqrt{|x_i|}\right), \ -20 \leq x_i \leq 20$$

(9) Rastrigin

$$f(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right), \ -5.12 \leq x_i \leq 5.12$$

(10) Raydan

$$f(x_0, \ldots, x_{n-1}) = -\sum_{i=0}^{n-1} \frac{(i+1)}{10} \left(\exp(x_i) - x_i\right), \ -5.12 \leq x_i \leq 5.12$$

As funções foram escolhidas por serem consideradas difíceis de otimizar e possuírem n variáveis. A subfigura à direita da Figura 4.2 exibe a função objetivo Griewank de uma variável e o gráfico da esquerda mostra os mínimos locais da função Griewank.

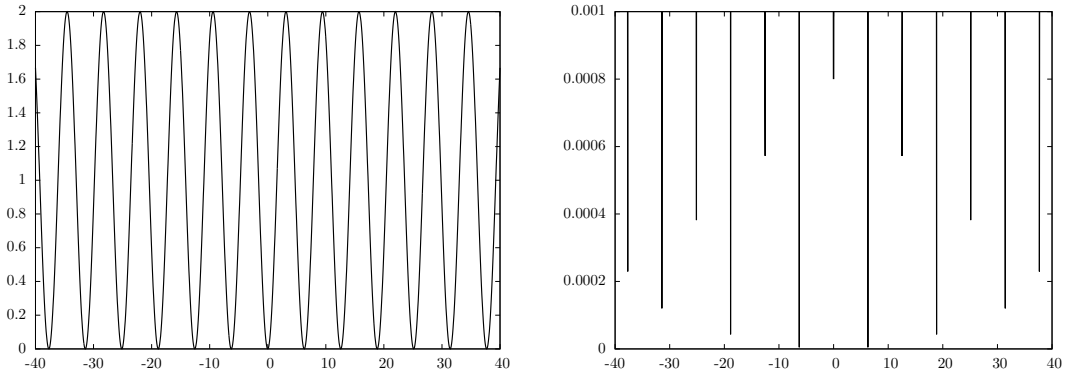


Figura 4.2: À esquerda a função objetivo Griewank com uma variável. À direita a disposição dos mínimos locais, neste caso, só pode ser observada após uma ampliação da região próxima de $y = 0$.

Para implementar os testes utilizamos a biblioteca NLopt que é livre e *open-source* distribuída sobre licença GNU e escrita em linguagem `C`. Esta biblioteca é usada para otimização não linear e possui uma variedade de minimizadores locais. Assim, a biblioteca NLOpt pode ser usada para otimização local ou global. Para os resultados computacionais utilizamos as seguintes rotinas de minimização local da biblioteca NLOpt:

- LBFGS - Este método é baseado na atualização de métricas de usando a recorrência de Strang do método BFGS com baixo armazenamento (Nocedal (1980); Liu e Nocedal (1989)).
- TNEWT - Este método é baseado no Algoritmo de Newton truncado para otimizações de larga escala (Facchinei et al. (2002)).

- MMA - Esta rotina é baseada no Algoritmo *Method-of-Moving-Asymptotes* para a busca local baseada em gradiente, incluindo restrições de inequações não-lineares (Svanberg (2002)).

- COBYLA - Este algoritmo é uma rotina de otimização com restrição usando algoritmo de aproximação linear para otimização sem derivada com restrições de equações e inequações não-lineares (Powell (1998)).

- NMEAD - Esta rotina usa o método simplex para otimização de funções (Nelder-Mead simplex algorithm) (Nelder e Mead (1965b); Richardson e Kuester (1973)).

- AGL - Esta rotina é baseada no método Augmented Lagrangian Algorithm com restrições gerais (Lewis e Torczon (2002); Conn et al. (1991)).

- BOBYQA - Este método executa otimização sem o uso de derivadas e com restrições de borda usando um método interativo para aproximação quadrática da função objetivo (Powell (2009)).

- CCSA - Este método é baseado no Algoritmo Conservative Convex Separable Approximation que é uma variação do Algoritmo MMA (Svanberg (2002)).

Para cada Algoritmo (DH, BBW e LPL) discretizamos as funções objetivo da seguinte maneira: criamos uma lista L mapeando o domímio de cada função objetivo no intervalo $\{0, \ldots, N\}$ onde $N = 2048$ para uma variável, $N = 2048^2$ para duas variáveis e $N = 256^3$ para três variáveis. Assim, computamos o número de avaliações da função objetivo até que eles encontrem o mínimo global correto. Executamos o mesmo processo 100 vezes e extraímos a média. Outro objetivo dos resultados computacionais foi testar a condição de parada para o Algoritmo LPL.

As Tabelas 4.1, 4.2 e 4.3 exibem o número de avaliações de cada função objetivo usando cada método de minimização clássica local para uma, duas e três

variáveis respectivamente. Os valores em azul exibem o melhor algoritmo de minimização local para uma determinada função objetivo e os valores em vermelho os piores resultados. Os algoritmos com asterisco fazem o uso da derivada. Outros resultados poderão ser vistos no Apêndice B.

Para uma variável, o melhor método de minimização local foi o LBFGS. Já para duas variáveis o método que mais se destacou foi o BOBYQA. E para três varíaveis, discretamente, o método BOBYQA teve o melhor desempenho. De maneira geral os algoritmos com os piores desempenhos foram os métodos NMEAD e SPB que foi marcado em vermelho 15 e 9 vezes, respectivamente num total de 30 testes. No entanto, o mesmo método NMEAD também foi marcado como melhor desempenho (azul) em 4 de 30 testes. Os métodos AGL e CSS ficaram próximo do comportamento mediano: o algorimo CSS foi marcado apenas uma vez como melhor e o algoritmo AGL somente foi marcado uma vez como pior. No caso geral é difícil determinar o método com o melhor desempenho de forma que cada uma das funções apresentam diferentes desafios para a minimização. Além disso também não é possível concluir, através dos resultados, se o uso da derivada é mais adequado.

A Tabela 4.4 exibe a comparação entre os Algoritmos DH, BBW e LPL do número de avaliações da função objetivo para cada algoritmo de minimização e cada função objetivo. Para o método LPL tomamos os valores marcados em azul nas Tabelas 4.1, 4.2 e 4.3, ou seja, os algoritmos de minimização local clássico que obtiveram o melhor desempenho. Os valores em azul exibem o melhor algoritmo de minimização local para uma determinada função objetivo e os valores em vermelho os piores resultados. Na Tabela 4.4, os algoritmos com asterisco fazem o uso da derivada.

O método proposto se mostrou melhor que os Algoritmos DH e BBW para a grande maioria dos testes. Somente em três funções objetivo o Algoritmo LPL foi menos eficiente que o os Algortimos DH e BBW.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	**9.00**	17.52	33.00	11.00	193.5	**236.2**	34.00	11.00	32.67
Griewank	58.63	92.77	104.0	99.21	**252.5**	214.5	108.8	**52.61**	108.6
Shekel	12.00	15.00	32.67	11.81	**5.05**	11.37	**33.66**	11.00	33.00
Rosenbrock	**84.86**	110.5	153.8	129.7	**3220.4**	207.9	159.1	101.7	165.5
Michalewicz	55.93	92.71	85.75	131.2	**185.7**	153.6	89.46	**39.06**	91.82
Dejong	**9.00**	15.00	32.67	84.74	133.7	**179.7**	33.66	11.00	33.00
Ackley	90.39	106.3	129.6	136.6	**212.8**	210.3	141.6	**44.49**	145.4
Schwefel	9.00	12.00	**66.26**	21.95	**4.00**	25.15	34.00	19.57	33.00
Rastrigin	75.75	91.58	**33.00**	70.12	**244.9**	236.6	76.09	36.51	89.57
Raydan	**25.87**	43.97	33.00	55.94	**191.8**	178.6	34.00	28.89	36.73

Tabela 4.1: Número de avaliações da função objetivo adicionado ao número de medidas quânticas (*effort*) para o método híbrido usando uma variável. Em vermelho o pior método de minimização local e em azul o melhor para cada função teste. Os métodos que usam derivada estão marcados com *.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	1126	1671	1434	**22.00**	8943	**9771**	1204	122.8	1213
Griewank	1583	1441	1290	612.0	**9285**	9077	1012	**412.2**	1036
Shekel	2643	**2785**	1575	22.00	**8.00**	22.01	1824	21.78	1933
Rosenbrock	**217.3**	770.2	1333	5762	7614	**9707**	1300	1086	1155
Michalewicz	1707	1881	1409	3887	**10323**	9841	1283	**390.4**	1258
Dejong	1534	1316	1379	22.00	6174	**7356**	1477	**21.78**	1433
Ackley	2385	2083	1487	1958	**7611**	7232	2001	**638.0**	2822
Schwefel	**2101**	1795	1422	356.7	**53.71**	1502	1306	70.32	1475
Rastrigin	1390	1219	1413	1360	**7695**	6180	1265	**191.4**	1153
Raydan	1718	1726	1445	467.0	9430	**10211**	1540	**324.7**	1651

Tabela 4.2: Número de avaliações da função objetivo adicionado ao número de medidas quânticas (*effort*) para o método LPL usando duas variáveis. Em vermelho o pior método de minimização local e em azul o melhor para cada função teste. Os métodos que usam derivada estão marcados com *.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	1789	4467	3622	4147	6099	**6805**	4080	**1390**	3215
Griewank	2932	2172	1500	1128	11392	**12001**	3179	**711.4**	938.7
Shekel	4775	**5137**	1900	24.00	**12.00**	24.00	1300	24.00	1742
Rosenbrock	**1776**	1847	1898	3795	18074	**18199**	2592	4018	2531
Michalewicz	8051	4843	–	4037	**14779**	9944	2209	3873	**1468**
Dejong	2604	2685	1582	1005	**9735**	6608	1953	**23.76**	1338
Ackley	–	**918.3**	1674	2944	**16455**	15680	2390	1098	1249
Schwefel	6988	**7549**	557.0	**141.7**	1100	209.3	883.2	172.4	1619
Rastrigin	2578	**555.7**	1779	1638	**11386**	9408	1365	925.1	1180
Raydan	2826	2908	1816	**726.9**	**17283**	16469	2081	780.1	1628

Tabela 4.3: Número de avaliações da função objetivo adicionado ao número de medidas quânticas (*effort*) para o método LPL usando três variáveis. Em vermelho o pior método de minimização local e em azul o melhor para cada função teste. Os métodos que usam derivada estão marcados com *.

Função Teste	**Uma variável**			**Duas variáveis**			**Três variáveis**		
	Minimizador Local								
	LPL	BBW	DH	LPL	BBW	DH	LPL	BBW	DH
Neumaier	**9.00**	97.50	**112.4**	**22.00**	**960.8**	694.0	1390	**371.3**	**13660**
Griewank	**52.61**	70.21	**88.44**	**412.2**	865.1	**1119**	**711.4**	884.4	**2643**
Shekel	**5.05**	99.04	**113.6**	**8.00**	944.0	**4588**	**12.00**	528.0	**18143**
Rosenbrock	**84.86**	87.68	**124.3**	**217.3**	852.8	**11017**	**1776**	–	**4580**
Michalewicz	**39.06**	90.13	**107.7**	**390.4**	739.3	**2794**	1468	**685.5**	**17334**
Dejong	**9.00**	75.00	**97.50**	**21.78**	826.6	**3056**	**23.76**	826.9	**5886**
Ackley	**44.49**	102.0	**114.4**	**638.0**	805.7	**875.0**	918.3	**617.0**	**4260**
Schwefel	**4.00**	88.38	**124.2**	**53.71**	871.1	**3675**	**141.7**	685.5	**13475**
Rastrigin	**33.00**	74.48	**99.45**	**191.4**	792.2	**3510**	**555.7**	967.0	**1010**
Raydan	**25.87**	93.24	**115.2**	**324.7**	609.6	**10063**	**726.9**	774.0	**4948**

Tabela 4.4: Comparação entre os os Algoritmos DH, BBW e LPL.

Os gráficos das Figuras 4.3, 4.4 e 4.5 comparam os Algoritmos DH, BBW e LPL através dos gráficos de desempenho (*performance graphs*) para 1, 2 e 3 variáveis respectivamente. Para estes testes utilizamos a função teste Griewank. Neste caso, o desempenho do Algoritmo LPL foi superior ao DH e BBW. Além disso, para este teste, o desempenho do Algoritmo LPL se tornou mais evidente a medida que o número de variáveis cresce.

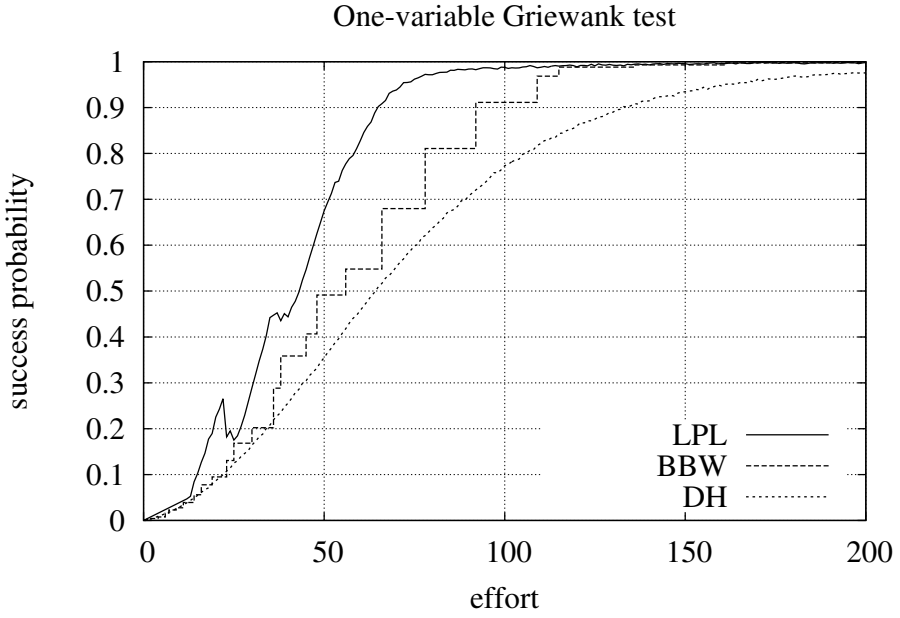


Figura 4.3: Comparação entre os Algoritmos DH, BBW e LPL usando a função objetivo Griewank para 1 variável.

Um outro objetivo desta Seção é avaliar computacionalmente a condição de parada proposta

$$n_1 + \frac{\sqrt{N}}{\log^n N} n_2 > 22.5\sqrt{N}.$$

Onde n_1 é o número de avaliações da função objetivo pelo Algoritmo de Grover (parte quântica) e n_2 é o número de avaliações da função objetivo pelo minizador local (parte clássica). Desta forma, executamos o algoritmo 100 vezes para cada função teste e contamos o número de vezes que o algoritmo definitivamente encontra o mínimo global. A Tabela 4.5 exibe o percentual de sucesso ao rodar o Algoritmo LPL para 1, 2 e 3 variáveis.

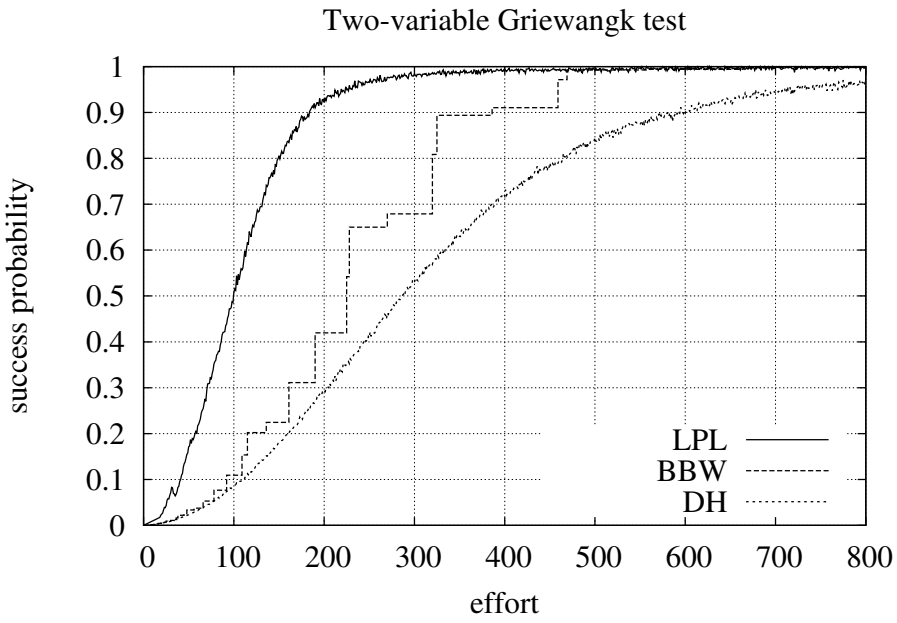


Figura 4.4: Comparação entre os Algoritmos DH, BBW e LPL usando a função objetivo Griewank para 2 variável.

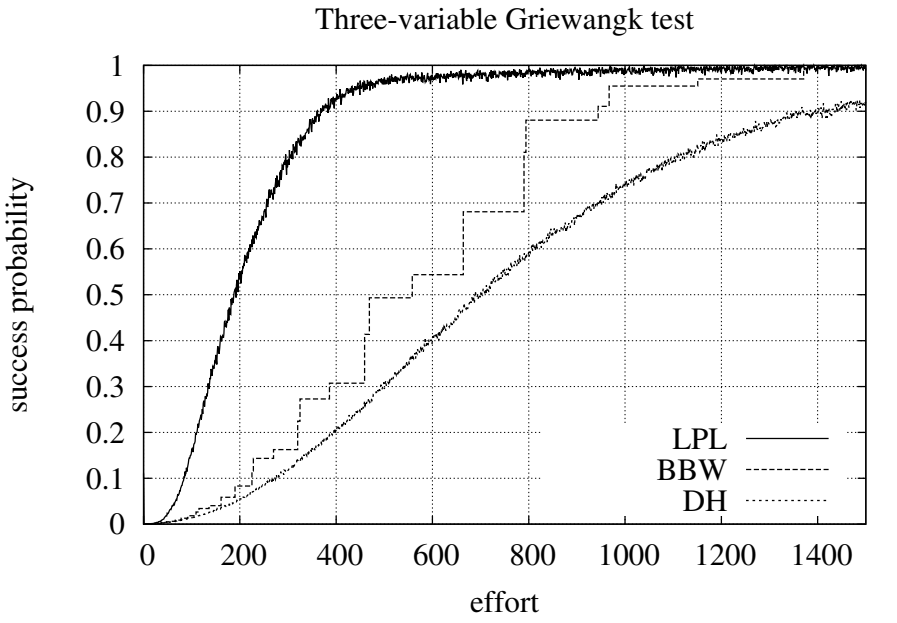


Figura 4.5: Comparação entre os Algoritmos DH, BBW e LPL usando a função objetivo Griewank para 3 variável.

A taxa de sucesso foi discretamente melhor para 2 e 3 variáveis. Isso pode indicar algum *bias* com relação ao número de variáveis. Um outro problema é que

Função Teste	**Uma variável**	**Duas variáveis**	**Três variáveis**
Neumaier	1.00	1.00	1.00
Griewank	0.87	1.00	1.00
Shekel	0.87	0.96	1.00
Rosenbrock	0.99	1.00	0.99
Michalewicz	1.00	1.00	0.99
Dejong	0.97	0.99	1.00
Ackley	1.00	1.00	1.00
Schwefel	0.98	1.00	1.00
Rastrigin	1.00	1.00	1.00
Raydan	1.00	1.00	1.00
Média	0.96	0.99	0.99

Tabela 4.5: Taxa de sucesso para o Algoritmo LPL.

cada algoritmo de minimização clássico local apresenta uma taxa de convergência específica. De qualquer forma, como a taxa de sucesso se manteve próximo de 1.00 a escolha da condição de parada pareceu adequada.

Capítulo 5

Otimização Usando Passeios Quânticos

Neste capítulo iremos tratar da otimização global de funções contínuas usando como ferramenta algoritmos de busca que utilizam passeios quânticos. A ideia central é substituir, no Algoritmo 10, o método BBHT pelo algoritmo de busca usando passeios aleatórios a tempo discreto.

5.1 Passeios Quânticos

O conceito de passeio quântico foi introduzido por Aharonov et al. (1993) sendo uma generalização do passeio aleatório clássico na reta, posteriormente, foi generalizado por Aharonov et al. (2001) para grafos arbitrários. No passeio aleatório clássico, uma partícula se move para uma direção com uma determinada probabilidade. Em contrapartida, no passeio quântico, são utilizados conceitos de superposição de um estado quântico permitindo explorar múltiplos caminhos concomitantemente. Este fato permite que o desvio padrão da dinâmica seja maior que o clássico (veja Figura 5.1), assim, em alguns casos, obtém-se a aplicação eficiente em alguns algoritmos. É o caso da versão quântica do Algoritmo *Element Distinctness* proposto por Ambainis (2004). O problema associado ao *Element Distinctness* é verificar se todos os elementos de uma lista são distintos. Neste caso, o algoritmo quântico ótimo utiliza como ferramenta passeios quânticos. Passeios quânticos também são promissores aplicados em avaliações de *Game Trees* binárias (Farhi et al. (2008)) e avaliações de formulas booleanas (Ambainis et al.

(2010)). Pode-se também avaliar a dinâmica do passeio em diferentes estruturas como grafos completos (Santos e Portugal (2010)), hipercubos (Marquezino et al. (2008)) e fractais (Lara et al. (2013)). Para uma referência em passeios quânticos veja o livro Portugal (2013).

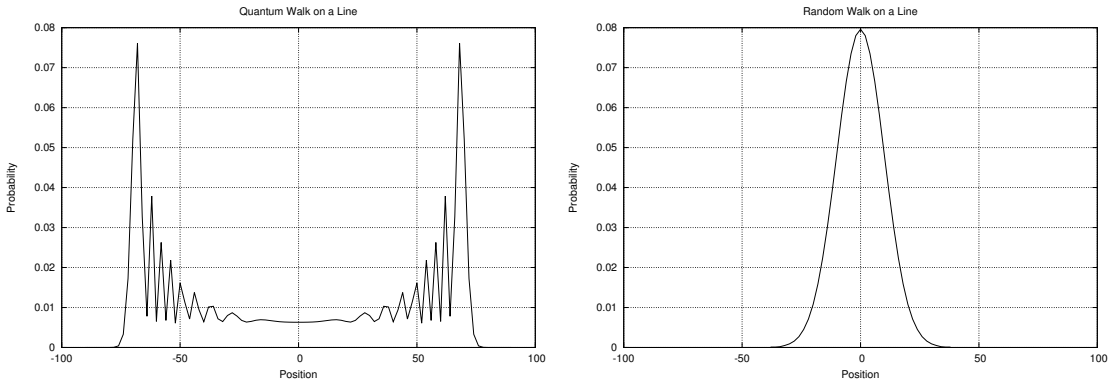


Figura 5.1: Diferença do passeio quântico (esquerda) e passeio aleatório (direita). O desvio padrão/variância é maior no passeio quântico. A Figura da esquerda foi gerada usando o software Hiperwalk (Lara et al. (2014a), veja o Apêndice D) .

Atualmente passeios quânticos têm se mostrado uma importante área na computação quântica e possui dois principais segmentos: passeios quânticos de tempo contínuo e passeios quânticos de tempo discreto. No caso de tempo discreto, podemos subdividir em dois modelos: passeios com e sem moeda.

Neste trabalho iremos explorar a busca usando passeios quânticos em tempo discreto e com moeda. Nos passeios quânticos em tempo discreto temos um estado $|p\rangle$ no espaço de Hilbert que representa a posição da partícula. No caso do passeio com moeda, temos um segundo vetor $|c\rangle$ no espaço de Hilbert que representa o estado da moeda. A moeda, neste caso, é responsável por estabelecer a direção da partícula. Por exemplo, se o passeio está definido na reta, o estado da moeda pertence a $\mathcal{H}^2$. Neste caso, a partícula poderá se deslocar somente para duas possíveis direções: esquerda e direita. Podemos definir a seguinte dinâmica para o passeio: se o valor da moeda for $|0\rangle$ o operador de deslocamento movimenta a partícula para a esquerda, se for $|1\rangle$ ele movimenta para a direita.

Assim, a representação do estado para o passeio quântico pode ser obtida

pela composição via produto tensorial

$$|\psi\rangle = |c\rangle \otimes |p\rangle.$$

Já a evolução do passeio quântico é caracterizada por um operador unitário U. No caso do passeio quântico em tempo discreto usando moeda o operador pode ser decomposto em dois operadores: O operador que atua na moeda (*Coin Operator*) e o operador que faz o deslocamento no grafo (*Shift Operator*). No caso do passeio na reta, o operador moeda é uma matriz unitária 2×2. A Figura 5.1 (direita) foi obtida usando como moeda o operador de Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

A Figura 5.2 exibe a distribuição assimétrica de probabilidade do estado após 100 aplicações do operador U utilizando a moeda de Fourier

$$F = \frac{1}{\sqrt{2}} \begin{bmatrix} -i & 1 \\ 1 & -i \end{bmatrix},$$

e a condição inicial $|\psi\rangle = |0\rangle \otimes |0\rangle$. O operador de deslocamento é responsável pelo deslocamento da partícula. No caso da reta, o movimento se dá em duas direções. Desta forma, o operador pode ser definido como

$$S|c\rangle|p\rangle = \begin{cases} |c\rangle|p-1\rangle, \text{se } c = 0 \\ |c\rangle|p+1\rangle, \text{se } c = 1 \end{cases}$$

Assim, o operador U que define o passeio com moeda na reta é dado por

$$U = S \cdot (C \otimes I) \tag{5.1}$$

Observe que o operador C só atua no espaço da moeda (isso justifica o termo $C \otimes I$). Para malhas do tipo 2D a construção é análoga: temos um estado que

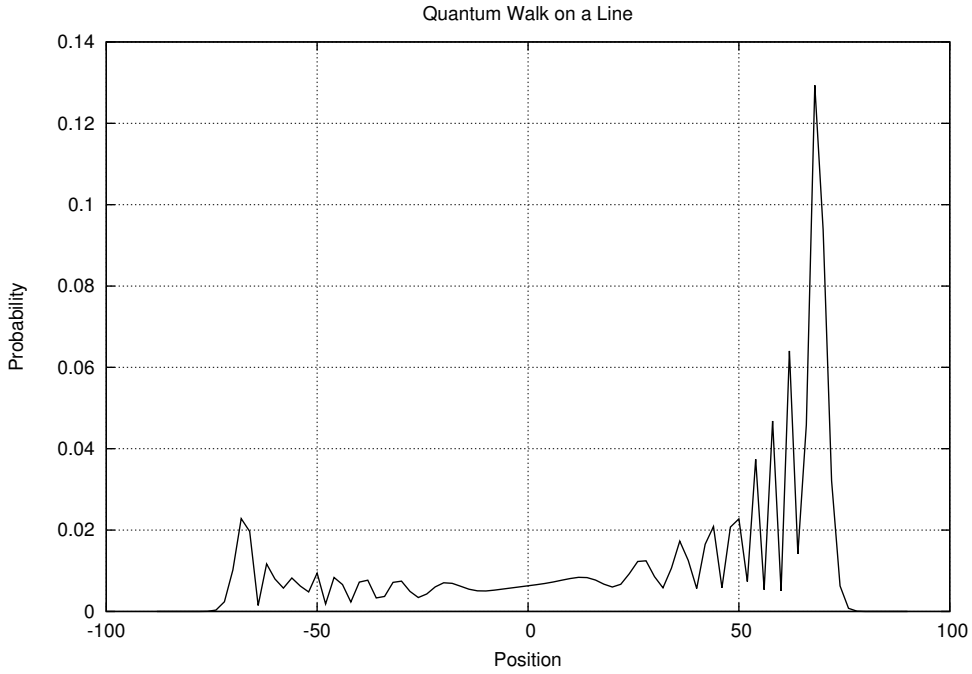


Figura 5.2: Distribuição de probabilidade para $T = 100$ usando a moeda de Fourier.

descreve o passeio é dado por $|c\rangle|x, y\rangle$ onde $|c\rangle$ pertence ao espaço de Hilbert $\mathcal{H}^4$ de forma que passa a ter 4 direções possíveis na malha regular e o vetor posição é descrito por $|x, y\rangle$ que representa um vértice na malha.

5.2 Busca usando Passeios Quânticos

Iremos apenas considerar a busca usando passeios quânticos (Shenvi et al. (2003)) em malhas bidimensionais com condições de contorno periódicas. Neste trabalho, usaremos o Algortimo de busca AKR (Ambainis et al. (2005)) na malha 2D que funciona marcando o elemento a ser buscado da seguinte forma: seja i_0 (neste caso, i_0 é um vértice (x, y)) o elemento procurado. Para todos os elementos tal que $i \neq i_0$ iremos aplicar o seguinte operador

$$U^\star = S \cdot (G \otimes I),$$

onde G é a matriz de Grover dada por

$$G = 2|D\rangle\langle D| - I,$$

$|D\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} |i\rangle$. O operador S é operador de deslocamento que usando o conceito de *flip − flop*: inverte a direção da moeda a cada aplicação. Sem utilizar o *flip-flop* Ambainis et al. (2005) mostrou que o Algoritmo AKR não teria nenhum ganho sobre o método clássico. Caso seja o elemento i_0 precisamos aplicar um operador diferenciado:

$$U^* = S \cdot (-I \otimes I).$$

Ou seja, no vértice i_0 usaremos como operador moeda o $-I$. Em termos de dinâmica, o uso do $-I$ irá funcionar como um atrator, aumentando periodicamente a probabilidade no elemento marcado.

A diferença entre aplicar o operador no elemento marcado é a moeda. Desta forma, o operador moeda que será aplicado em toda malha poderá ser escrito como:

$$C' = -I \otimes |i_0\rangle\langle i_0| + G \otimes (I - |i_0\rangle\langle i_0|)$$

Assim o operador de busca usando passeios quânticos poderá ser descrito como

$$U' = S \cdot C'$$

As Figuras 5.3 e 5.4 exibem a dinâmica da probabilidade do elemento marcado em função do número de aplicações do U' em uma malha bidimensional 40×40. Como se trata de uma busca usando caminhadas a tempo discreto, o eixo T nas figuras indica o número de aplicações do operador U, ou seja, o número de vezes que a função objetivo foi utilizada. Desta forma, este T está correlacionado ao *effort* visto na Seção anterior. Neste caso, o importante é conhecer o primeiro ponto de máximo. Os experimentos foram realizados usando a linguagem Neblina (veja o Apêndice C). É possível mostrar que este o algoritmo encontra o elemento

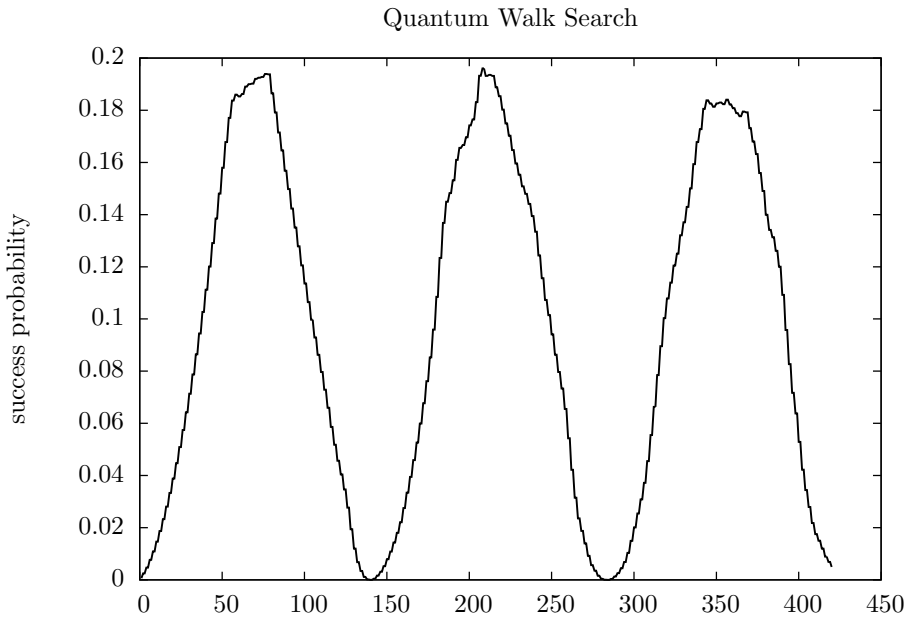


Figura 5.3: Probabilidade do elemento marcado com relação ao número de aplicações do operador U' para uma malha 40×40.

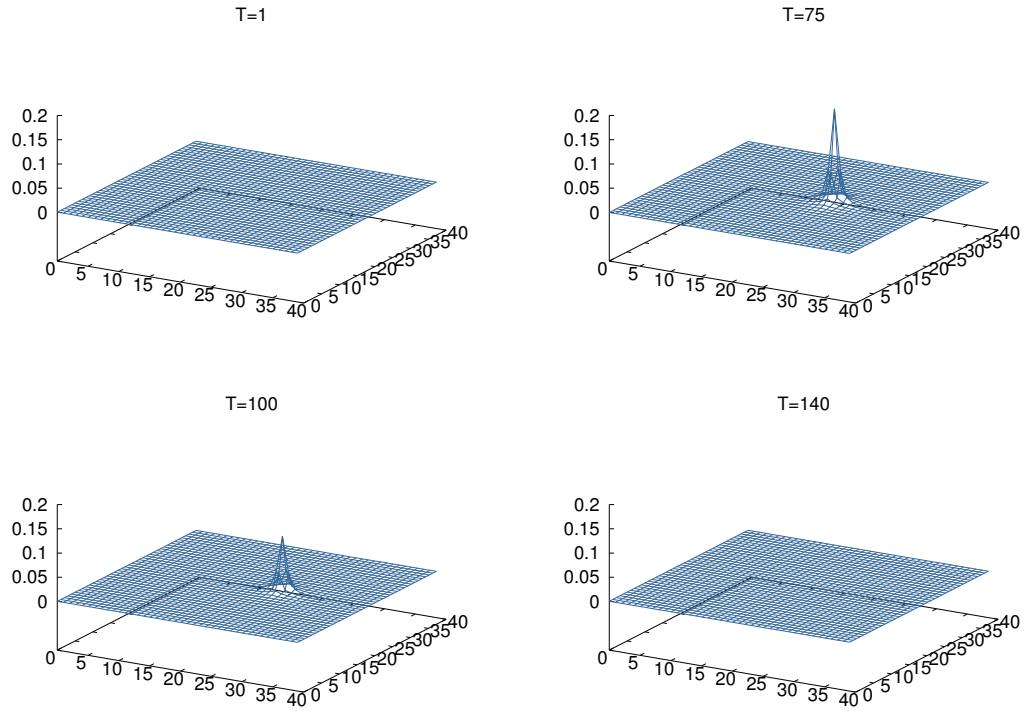


Figura 5.4: Busca do elemento marcado em uma malha bidimensional 40×40. Próximo de $t = 75$ a probabilidade do elemento marcado atinge o máximo. Com $t = 140$ a dinâmica recomeça. O valor de t indica a quantidade de vezes que a função objetivo foi avaliada.

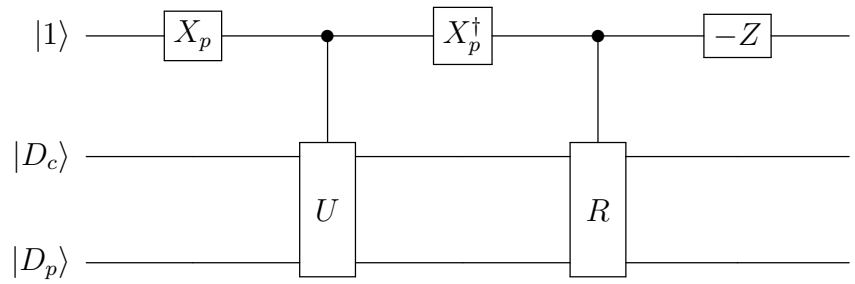


Figura 5.5: Melhoria na busca usando proposta por A. Tulsi

marcado em $O(\log N\sqrt{N})$ (Portugal (2013)).

Tulsi (2008) melhorou o sistema ao adicionar um qbit de controle no Algoritmo de busca AKR de forma que o custo computacional fique $O(\sqrt{N \log N})$. Para isso precisamos reescrever o operador de busca como

$$U' = UR \tag{5.2}$$

Onde U é determinado pela Equação 5.1 e a reflexão R é dada por

$$R = I - 2|D_c\rangle\langle D_c| \otimes |i_0\rangle\langle i_0|$$

A Figura 5.6 exibe a melhoria após aplicação da modificação de Tulsi no Algoritmo AKR. O circuito quântico da Figura 5.5 descreve os passos para a implementação da proposta de Tulsi onde

$$X_p = \begin{bmatrix} \cos(\sqrt{p}) & \sin(\sqrt{p}) \\ -\sin(\sqrt{p}) & \cos(\sqrt{p}) \end{bmatrix}$$

e p é a probabilidade máxima observada na Figura 5.3.

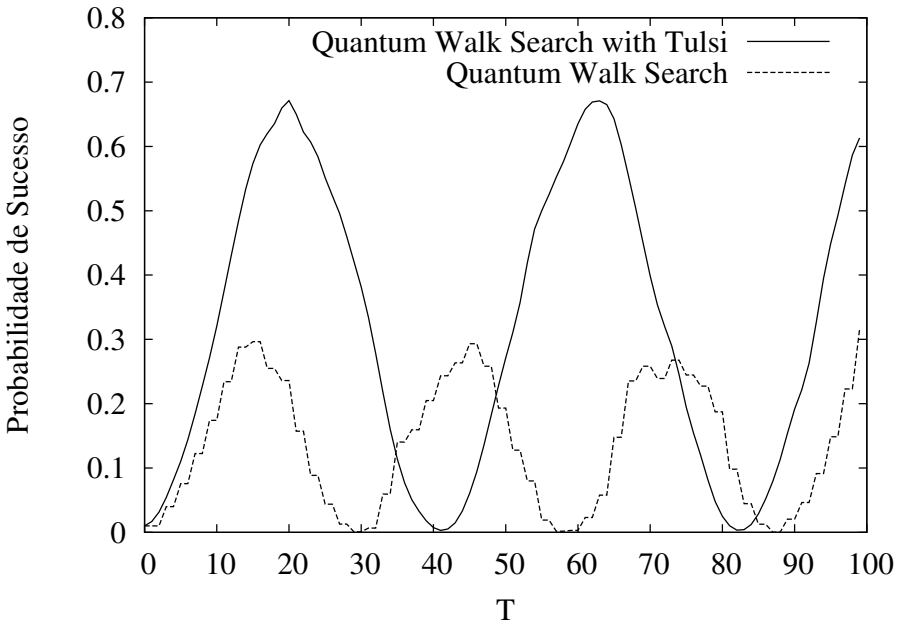


Figura 5.6: Busca do elemento marcado em uma malha bidimensional 10×10 usando a modificação de Tulsi.

5.3 Busca Usando Passeios Quânticos com Mais de um Elemento Marcado

Todos os algoritmos de otimização vistos aqui (DH, BBW e LPL) utilizam a busca utilizando o Algoritmo BBHT. A ideia aqui é substituir o Algoritmo BBHT pela busca usando passeios quânticos na malha bidimensional. A priori o uso do Algoritmo AKR não é adequada: a complexidade para a busca usando passeios quânticos é $O(\sqrt{N \log N})$ contra $O(\sqrt{N})$ do Algoritmo BBHT. A complexidade $O(\sqrt{N \log N})$ foi obtida para apenas um elemento marcado. Apesar de possuir uma grande variedade de aplicações, o caso onde temos mais de um elemento marcado não possui, até o presente momento, resultados analíticos. Não sabemos por exemplo qual é a complexidade do algoritmo. Mesmo sem resultados analíticos podemos simular alguns experimentos. Neste caso, o algoritmo será baseado no ARK com a modificação de Tulsi (chamaremos de AKR-Tulsi) usando o seguinte operador de evolução:

$$U' = UR$$

onde U é determinado pela Equação 5.1 e R é dado por

$$R = I - 2|D_c\rangle\langle D_c| \otimes \left(\sum_{i=0}^{M-1} |i_j\rangle\langle i_j| \right),$$

e $\{i_0, \ldots, i_{M-1}\}$ são os M elementos marcados na malha bidimensional. A condição inicial é dada por

$$|\psi\rangle = |D_c\rangle|D_p\rangle.$$

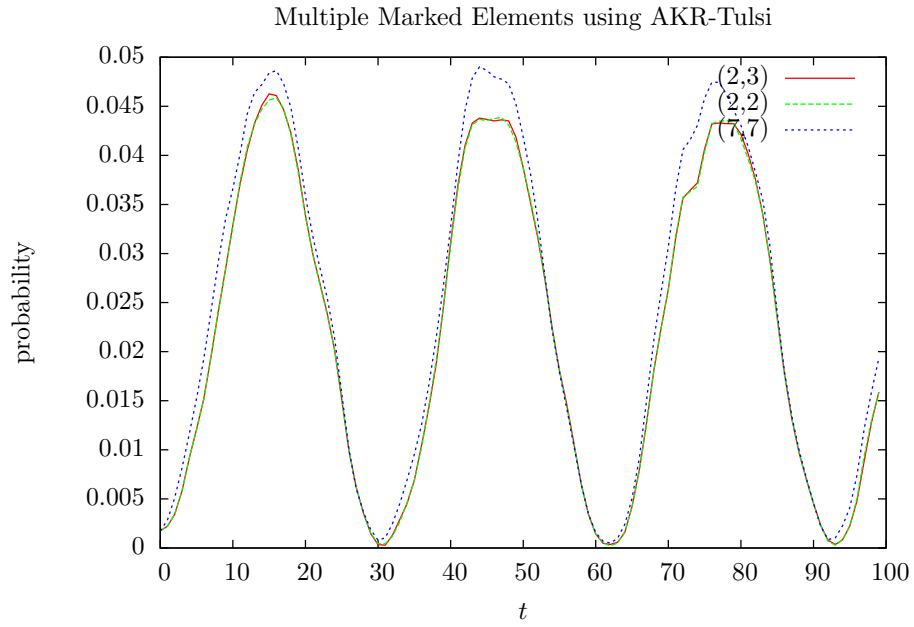


Figura 5.7: Foram marcados os elementos $(2,2)$, $(2,3)$ e $(7,7)$ em uma malha bidimensional 12×12. O vétice $(7,7)$ apresentou uma probabilidade maior que os elementos $(2,2)$ e $(2,3)$.

A Figura 5.9 exibe as probabilidades para cada um dos elementos marcados usando uma malha 12×12. A primeira coisa que observamos é a presença de uma conservação de probabilidade baseada nas distâncias. Ou seja, os pontos próximos $(2,2)$ e $(2,3)$ apresentaram uma probabilidade menor que o ponto isolado $(7,7)$. Agora, se for marcado além destes pontos o vértice $(7,8)$ temos seguinte distribuição das probabilidades (Figura 5.8).

Outra questão observada ao se marcar múltiplos elementos acontece quando um determinado ponto marcado é circundado por outros pontos marcados, este

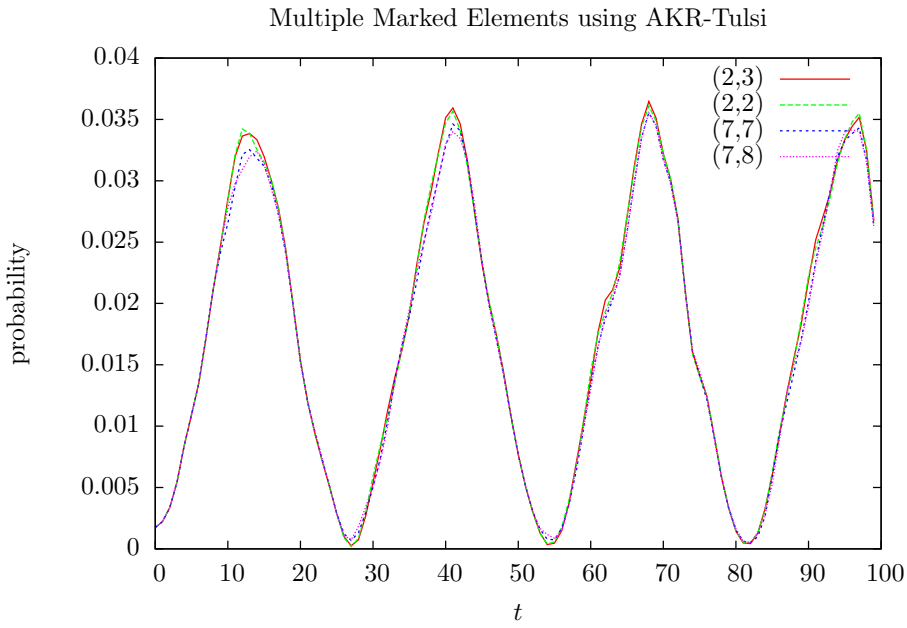


Figura 5.8: Inserção do elemento marcado $(7, 8)$. Os gráficos ficam distribuídos de maneira mais balanceada.

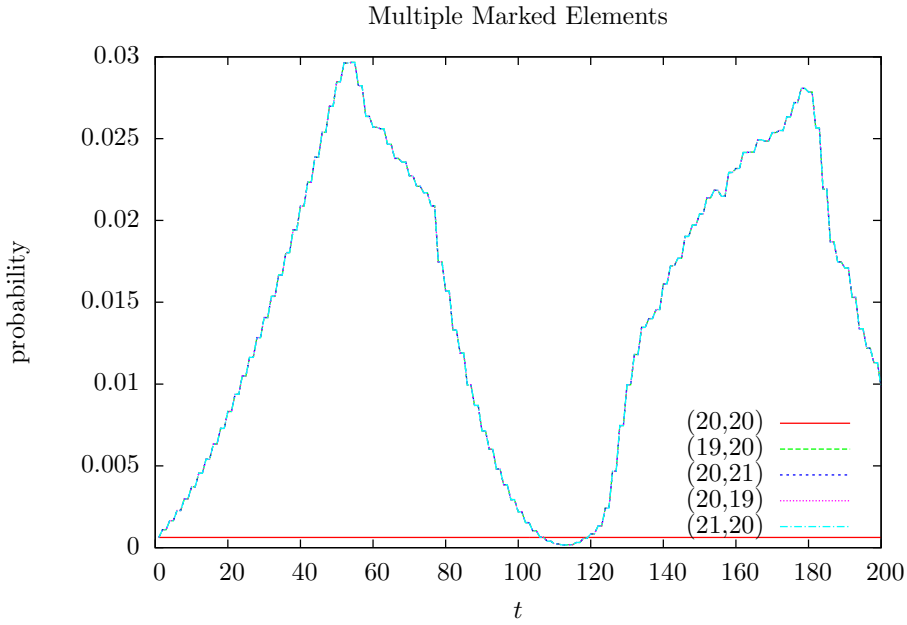


Figura 5.9: Probabilidade associada a cada ponto marcado. O ponto central $(20, 20)$ ficou com probabilidade constante próxima de zero. Os outros pontos apresentam a mesma curva de probabilidade.

ponto central terá a probabilidade constante próxima de zero. Veja a Figura 5.9.

Função Teste	Otim. AKR-Tulsi	LPL
Neumaier	32.38	46.77
Griewank	26.83	42.44
Shekel	11.12	11.37
Rosenbrock	28.52	45.91
Michalewicz	32.55	36.31
Dejong	29.10	47.52
Ackley	28.42	32.29
Schwefel	31.28	31.84
Rastrigin	29.62	35.25
Raydan	29.55	50.34

Tabela 5.1: Comparação entre o número médio de avaliações entre o Algoritmo LPL e o Algoritmo de Otimização que usa AKR-Tulsi

5.4 Otimização usando o AKR-Tulsi com múltiplos elementos marcados

Nesta seção será considerado os métodos de minimização locais, funções objetivo de 2 variáveis vistas na Seção 3. Assim como na Seção 3, discretizamos a função objetivo e associamos a cada ponto discretizado do domínio de f a um vértice da malha bidimensional. A dinâmica foi a mesma do Algoritmo 10: cada iteração crie a malha e marque todos elementos abaixo de um limiar, após isso, aplique r_i vezes o operador U (Equação 5.2) e faça uma medida.

A Tabela 5.1 exibe uma comparação entre o Algoritmo de otimização usando o AKR-Tulsi e o Algoritmo LPL. Neste caso rodamos cada algoritmo 200 vezes e extraímos a média de avaliações da função objetivo até encontrar um mínimo. Para todas as funções objetivo o método de otimização que usa o AKR-Tulsi se mostrou mais eficiente mesmo que discretamente, no caso da função objetivo Shekel. Veja os resultados completos do experimento computacional no Apêndice A.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho descreveu e comparou algoritmos híbridos para a otimização de funções contínuas. Inicialmente utilizamos o Algoritmo quântico BBHT para escapar de mínimos locais e métodos de minimização locais para acelerar a busca. Posteriormente foi substituído o algoritmo BBHT pela busca usando o Algoritmo AKR-Tulsi. Descrevemos uma condição de parada baseado no ganho da minimização local. Esta condição foi avaliada computacionamente e se mostrou adequada com taxa de sucesso próxima de 1. O Algoritmo LPL foi comparado com os Algoritmos de otimização discreta BBW e DH. O método proposto se mostrou, no caso geral mais eficiente que os Algoritmos BBW e DH. Este resultado era, de certa forma, esperado. O Algoritmo LPL usa o fato da função ser contínua e toma proveito através do uso de métodos eficientes de minimização local. No entanto, a comparação com os métodos BBW e DH foi razoável no sentido que o Algoritmo LPL foi o primeiro método quântico na literatura a atuar com funções contínuas. Desta forma, as referências naturais eram os métod BBW e DH que foram propostos para listas discretas. Outros autores publicaram métodos de otimização de listas discretas, e.g., Liu e Koehler (2012) descreveram métodos baseados no Algoritmo BBW apresentando ganhos discretos com relação ao próprio BBW.

Na Seção 4 propomos um Algoritmo para a otimização de funções discretas usando o Algoritmo de busca AKR-Tulsi. Apesar de não existir resultados analíticos para este caso (não é conhecido o custo computacional para este tipo de busca)

os resultados se mostraram promissores comparados ao Algoritmo LPL: em todos os testes o Algoritmo de Otimização AKR-Tulsi obteve melhor desempenho.

6.1 Trabalhos Futuros

Os resultados positivos obtidos com o uso da busca usando passeios quânticos induzem os trabalhos futuros em dois segmentos: analítico e computacional. Na parte analítica, seria importante estabelecer o custo computacional em função do número de elementos marcados. Pode-se começar atacando o problema quando os elementos marcados estão dispostos de maneira equidistantes. Isto faz com que os pontos marcados tenham a mesma probabilidade. A partir disto, pode-se pensar no caso onde temos a disposição dos pontos marcados de maneira aleatória. Neste ponto é importante procurar compreender a conservação de probabilidade que há quando se clusteriza pontos na malha. Em todos os casos, o problema analítico parece ser um problema difícil.

Na parte computacional, pode-se começar a explorar o comportamento de alguns parâmetros, por exemplo o valor p da matriz X_p usada na modificação de Tulsi, com o objetivo de encontrar um valor ótimo. Mesmo sem resultados analíticos, um outro trabalho poderia ser definir e avaliar computacionalmente uma condição de parada para este algoritmo. Computacionamelnte podemos caracterizar quais funções objetivo são mais interessantes ao utilizar a busca AKR-Tulsi. Talvez a conservação de probabilidade possa trazer benefícios em casos espefíficos.

Referências Bibliográficas

Dorit Aharonov, Andris Ambainis, Julia Kempe, e Umesh Vazirani. Quantum walks on graphs. In: **Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing**, STOC '01, páginas 50–59, New York, NY, USA, 2001. ACM. ISBN 1-58113-349-9. URL http://doi.acm.org/10.1145/380752.380758.

Y. Aharonov, L. Davidovich, e N. Zagury. Quantum random walks. **Phys. Rev. A**, 48:1687–1690, Aug 1993. URL http://link.aps.org/doi/10.1103/PhysRevA.48.1687.

Eric Allen, David Chase, Joe Hallett, Victor Luchangco, Jan-Willem Maessen, Sukyoung Ryu, Guy L. Steele Jr., e Sam Tobin-Hochstadt. The Fortress Language Specification. Relatório técnico, Sun Microsystems, Inc., 2007. URL http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf.

A. Ambainis. Quantum walk algorithm for element distinctness. In: **Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on**, páginas 22–31, Oct 2004.

A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, e S. Zhang. Any and-or formula of size n can be evaluated in time $n^{1/2+O(1)}$ on a quantum computer. **SIAM J. Comput.**, 39(6):2513–2530, Abril 2010. ISSN 0097-5397. URL http://dx.doi.org/10.1137/080712167.

Andris Ambainis, Julia Kempe, e Alexander Rivosh. Coins make quantum walks faster. In: **Proceedings of the Sixteenth Annual ACM-SIAM Sympo-**

sium on Discrete Algorithms, SODA '05, páginas 1099–1108, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0-89871-585-7. URL http://dl.acm.org/citation.cfm?id=1070432.1070590.

William Baritompa, David W. Bulger, e Graham R. Wood. Grover's quantum algorithm applied to global optimization. **SIAM Journal on Optimization**, 15(4):1170–1184, 2005. URL http://dx.doi.org/10.1137/040605072.

Charles H. Bennett, Ethan Bernstein, Gilles Brassard, e Umesh Vazirani. Strengths and weaknesses of quantum computing. **SIAM J. Comput.**, 26(5):1510–1523, Outubro 1997. ISSN 0097-5397. URL http://dx.doi.org/10.1137/S0097539796300933.

Michel Boyer, Gilles Brassard, Peter Hoyer, e Alain Tapp. 1996. URL arXiv:quant-ph/9605034.

D. W. Bulger, D. Alexander, W. P. Baritompa, G. R. Wood, e Z. B. Zabinsky. Expected hitting times for backtracking adaptive search. **Optimization**, 53(2): 189–202, 2004.

David W. Bulger, William Baritompa, e Graham R. Wood. Implementing pure adaptive search with grover's quantum algorithm. **Journal of Optimization Theory and Applications**, 116(3):517–529, 2003.

David W. Bulger e Graham R. Wood. Hesitant adaptive search for global optimisation. **Math. Program.**, 81:89–102, 1998.

B.L. Chamberlain, D. Callahan, e H.P. Zima. Parallel programmability and the chapel language. **Int. J. High Perform. Comput. Appl.**, 21(3):291–312, Augosto 2007. ISSN 1094-3420. URL http://dx.doi.org/10.1177/1094342007078442.

Isaac L. Chuang, Neil Gershenfeld, e Mark Kubinec. Experimental Implementation of Fast Quantum Searching. **Physical Review Letters**, 80(15):3408+, Abril 1998. URL http://dx.doi.org/10.1103/PhysRevLett.80.3408.

Andrew R. Conn, Nicholas I. M. Gould, e Philippe L. Toint. A globally convergent augmented langrangian algorithm for optimization with general constraints and simple bounds. **SIAM Journal on Numerical Analysis**, 28(2):pp. 545–572, 1991. ISSN 00361429. URL http://www.jstor.org/stable/2157828.

W. Diffie e M. Hellman. New directions in cryptography. **IEEE Trans. Inf. Theor.**, 22(6):644–654, Setembro 2006. ISSN 0018-9448. URL http://dx.doi.org/10.1109/TIT.1976.1055638.

Christoph Dürr e Peter Høyer. 1996. URL http://arxiv.org/abs/quant-ph/9607014.

Francisco Facchinei, Stefano Lucidi, e Laura Palagi. A truncated newton algorithm for large scale box constrained optimization. **SIAM J. on Optimization**, 12(4):1100–1125, Abril 2002. ISSN 1052-6234. URL http://dx.doi.org/10.1137/S1052623499359890.

Edward Farhi, Jeffrey Goldstone, e Sam Gutmann. A quantum algorithm for the hamiltonian nand tree. **Theory of Computing**, 4(8):169–190, 2008. URL http://www.theoryofcomputing.org/articles/v004a008.

Lov K. Grover. A fast quantum mechanical algorithm for database search. In: **Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing**, STOC '96, páginas 212–219, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5. URL http://doi.acm.org/10.1145/237814.237866.

Eligius MT Hendrix e Olivier Klepper. On uniform covering, adaptive random search and raspberries. **Journal of Global Optimization**, 18(2):143–163, 2000.

Eligius MT Hendrix e B Tóth. **Introduction to nonlinear and global optimization**, volume 37. Springer Optimization and Its Applications, Springer Science+Business Media, New York, NY, USA, 2010. ISBN 978-0-387-88670-1.

Khronos OpenCL Working Group. **The OpenCL Specification, version 1.2**, November 2011 2011. URL www.khronos.org/registry/cl/specs/opencl-1.2.pdf.

Neal Koblitz. Elliptic curve cryptosystems. **Mathematics of Computation**, 48 (177):203–209, Janeiro 1987. ISSN 0025-5718.

Pedro Lara, Carlile Lavor, e Renato Portugal. Global optimization workshop 2012, gow 2012, natal-rn. In: **3rd Conference of Computational Interdisciplinary Sciences. CCIS 2014, Proceedings.**, páginas 89–92, June 2012.

Pedro Lara, Aaron Leão, e Renato Portugal. Simulation of quantum walks using hpc. In: **3rd Conference of Computational Interdisciplinary Sciences. CCIS 2014, Proceedings.**, páginas 226–237, Sept 2014a.

Pedro Lara, Renato Portugal, e Stefan Boettcher. Quantum walks on sierpinski gaskets. **International Journal of Quantum Information**, 11(08):1350069, 2013. URL http://www.worldscientific.com/doi/abs/10.1142/S021974991350069X.

Pedro Lara, Renato Portugal, e Carlile Lavor. A new hybrid classical-quantum algorithm for continuous global optimization problems. **J. of Global Optimization**, 60(2):317–331, Outubro 2014b. ISSN 0925-5001. URL http://dx.doi.org/10.1007/s10898-013-0112-8.

Robert Michael Lewis e Virginia Torczon. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. **SIAM Journal on Optimization**, 12(4):1075–1089, 2002.

Dong C. Liu e Jorge Nocedal. On the limited memory bfgs method for large scale optimization. 45:503–528, 1989.

Yipeng Liu e Gary J. Koehler. Using modifications to grover's search algorithm for quantum global optimization. **European Jour-**

nal of Operational Research, 207(2):620–632, 2010. URL http://EconPapers.repec.org/RePEc:eee:ejores:v:207:y:2010:i:2:p:620-632.

Yipeng Liu e Gary J. Koehler. A hybrid method for quantum global optimization. **Journal of Global Optimization**, 52(3):607–626, 2012. ISSN 0925-5001.

F. L. Marquezino, R. Portugal, G. Abal, e R. Donangelo. Mixing times in quantum walks on the hypercube. **Phys. Rev. A**, 77:042312, Apr 2008. URL http://link.aps.org/doi/10.1103/PhysRevA.77.042312.

Victor S Miller. Use of elliptic curves in cryptography. In: **Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85**, páginas 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc. ISBN 0-387-16463-4. URL http://dl.acm.org/citation.cfm?id=18262.25413.

J. A. Nelder e R. Mead. A simplex method for function minimization. **The Computer Journal**, 7(4):308–313, 1965a. URL http://comjnl.oxfordjournals.org/content/7/4/308.abstract.

J. A. Nelder e R. Mead. A simplex method for function minimization. **Computer Journal**, 7:308–313, 1965b.

Michael A. Nielsen e Isaac L. Chuang. **Quantum Computation and Quantum Information: 10th Anniversary Edition**. Cambridge University Press, New York, NY, USA, 10th edição, 2011. ISBN 1107002176, 9781107002173.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. **Mathematics of computation**, 35(151):773–782, 1980.

R. Portugal. **Quantum Walks and Search Algorithms**. Springer Science+Business Media, New York, NY, USA, 2013. ISBN 978-1-4614-6336-8.

M. J. D. Powell. Direct search algorithms for optimization calculations. **Acta Numerica**, 7:287–336, 1998. URL http://dx.doi.org/10.1017/S0962492900002841.

M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Augosto 2009.

A. Quarteroni, R. Sacco, e F. Saleri. **Numerical Mathematics**. Texts in applied mathematics. Springer, 2000. ISBN 9780387989594.

Joel A. Richardson e J. L. Kuester. The complex method for constrained optimization [e4] (algorithm 454). **Commun. ACM**, 16(8):487–489, 1973.

R.L. Rivest, A. Shamir, e L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, 21:120–126, 1978.

Raqueline Azevedo Medeiros Santos e Renato Portugal. Quantum hitting time on the complete graph. **International Journal of Quantum Information**, 08(05):881–894, 2010. URL `http://www.worldscientific.com/doi/abs/10.1142/S0219749910006605`.

Vijay Saraswat, Bard Bloom, Igor Peshansky, Olivier Tardieu, e David Grove. X10 language specification. Relatório técnico, IBM, October 2012. URL `http://x10.sourceforge.net/documentation/languagespec/x10-latest.pdf`.

Neil Shenvi, Julia Kempe, e K. Birgitta Whaley. Quantum random-walk search algorithm. **Phys. Rev. A**, 67:052307, May 2003. URL `http://link.aps.org/doi/10.1103/PhysRevA.67.052307`.

P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In: **Proceedings of the 35th Annual Symposium on Foundations of Computer Science**, SFCS '94, páginas 124–134, Washington, DC, USA, 1994. IEEE Computer Society. ISBN 0-8186-6580-7. URL `http://dx.doi.org/10.1109/SFCS.1994.365700`.

Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. **SIAM Journal on Optimization**, 12:555–573, 2002.

Avatar Tulsi. Faster quantum-walk algorithm for the two-dimensional spatial search. **Phys. Rev. A**, 78:012310, Jul 2008. URL http://link.aps.org/doi/10.1103/PhysRevA.78.012310.

Graham R. Wood, Zelda B. Zabinsky, e Birna P. Kristinsdottir. Hesitant adaptive search: the distribution of the number of iterations to convergence. **Math. Program.**, 89(3):479–486, 2001. URL http://dx.doi.org/10.1007/PL00011410.

Zelda B. Zabinsky e Robert L. Smith. Pure adaptive search in global optimization. **Math. Program.**, 53:323–338, 1992.

Zelda B. Zabinsky, Graham R. Wood, Mike A. Steel, e William Baritompa. Pure adaptive search for finite global optimization. **Math. Program.**, 69:443–448, 1995.

Christof Zalka. Grover's quantum searching algorithm is optimal. **Phys. Rev. A**, 60:2746–2751, Oct 1999. URL http://link.aps.org/doi/10.1103/PhysRevA.60.2746.

Apêndice A

Resultados Computacionais para a Otimização usando AKR-Tulsi

Este apêndice exibe, em maiores detalhes, os experimentos computacionais do Capítulo 4. Os valores das tabelas mostram o número de avaliações da função objetivo. Entre parênteses o desvio padrão das amostras. Para melhor vizualização, após as tabelas seguem figuras comparativas com o conteúdo das tabelas.

Função de teste: Neumaier		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	35.82(20.94)	50.09(12.71)
SBPLX	32.38(20.71)	49.09(15.72)
AUGLAG	–	–
MMA	51.00(0.00)	51.00(0.00)
CCSAQ	51.00(0.00)	–
BOBYQA	42.90(20.46)	48.68(14.56)
NELDERMEAD	38.40(21.98)	47.68(16.23)
TLPLTON	75.17(30.81)	79.85(30.09)
LBFGS	36.83(17.83)	46.77(16.55)

Função de teste: Griewank		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	29.58(15.61)	43.98(12.84)
SBPLX	26.83(16.68)	44.25(15.75)
AUGLAG	52.00(0.00)	52.00(0.00)
MMA	–	–
CCSAQ	–	51.00(0.00)
BOBYQA	28.89(15.30)	44.38(14.91)
NELDERMEAD	31.24(20.22)	42.44(16.49)
TLPLTON	59.00(1.00)	58.40(5.28)
LBFGS	45.20(8.89)	46.96(12.97)

Função de teste: Sherkel		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	11.32(3.49)	16.06(13.73)
SBPLX	13.48(8.53)	13.76(6.42)
AUGLAG	83.34(27.48)	96.31(23.27)
MMA	84.57(26.26)	89.79(24.64)
CCSAQ	83.65(25.98)	92.21(24.05)
BOBYQA	11.12(1.39)	12.33(7.25)
NELDERMEAD	11.61(8.41)	11.37(6.58)
TLPLTON	54.42(20.26)	61.62(14.97)
LBFGS	47.99(18.21)	53.79(16.09)

Função de teste: Rosembrock		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	44.13(20.70)	50.84(14.75)
SBPLX	28.52(19.57)	50.12(13.23)
AUGLAG	52.00(0.00)	52.00(0.00)
MMA	51.00(0.00)	51.00(0.00)
CCSAQ	79.00(28.04)	68.33(24.51)
BOBYQA	38.10(25.26)	46.67(18.92)
NELDERMEAD	39.10(20.62)	49.68(14.07)
TLPLTON	57.30(19.06)	56.50(17.37)
LBFGS	35.29(14.59)	45.91(16.29)

Função de teste: Michalewicz		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	41.07(22.92)	36.31(15.79)
SBPLX	41.31(24.90)	44.55(17.31)
AUGLAG	91.78(24.84)	101.28(17.03)
MMA	89.64(24.71)	98.00(20.44)
CCSAQ	90.25(25.89)	99.34(17.71)
BOBYQA	38.18(22.48)	42.16(19.90)
NELDERMEAD	32.55(20.33)	43.12(17.04)
TLPLTON	87.50(23.58)	94.05(20.31)
LBFGS	69.06(29.90)	80.97(20.02)

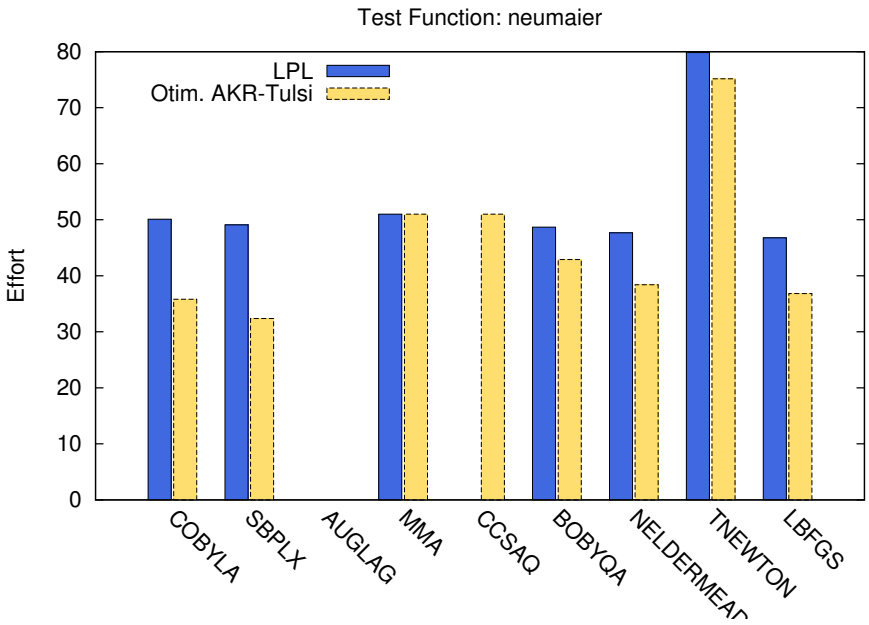
Função de teste: Dejong		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	46.29(19.68)	49.99(10.96)
SBPLX	35.65(18.24)	47.57(13.90)
AUGLAG	–	52.00(0.00)
MMA	51.00(0.00)	–
CCSAQ	61.60(21.20)	51.00(0.00)
BOBYQA	29.20(19.78)	47.52(17.15)
NELDERMEAD	33.00(18.20)	49.16(15.27)
TLPLTON	57.33(31.28)	63.43(19.38)
LBFGS	43.00(16.01)	51.46(13.32)

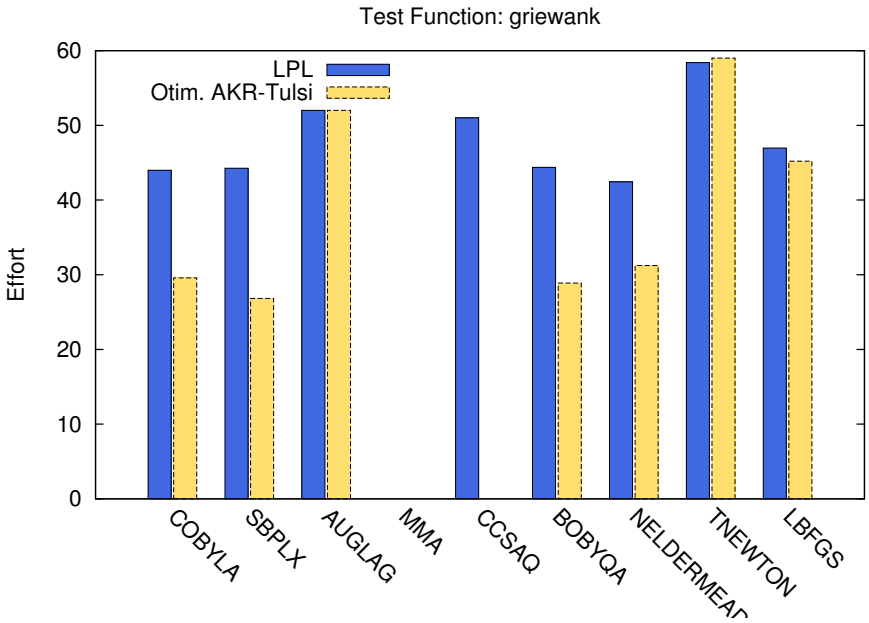
Função de teste: Ackley		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	38.01(23.42)	32.29(16.43)
SBPLX	28.42(19.76)	45.09(17.53)
AUGLAG	85.27(26.64)	93.32(24.27)
MMA	80.75(27.34)	96.07(21.70)
CCSAQ	85.13(26.09)	90.32(24.00)
BOBYQA	35.30(23.67)	32.86(15.80)
NELDERMEAD	30.50(14.47)	47.98(17.21)
TLPLTON	94.50(25.34)	103.07(17.02)
LBFGS	80.12(24.75)	93.46(17.38)

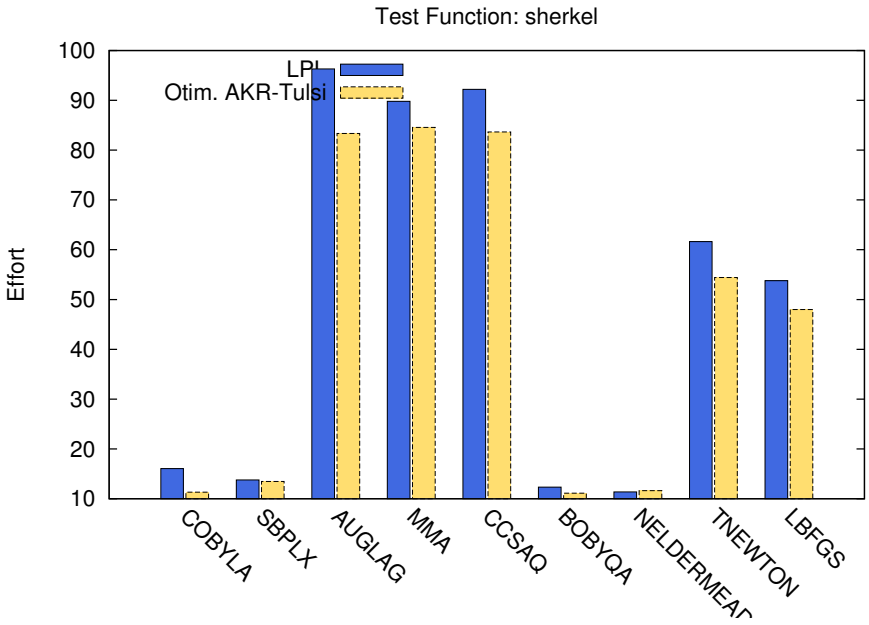
Função de teste: Schwefel		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	36.95(21.60)	31.84(12.16)
SBPLX	40.65(23.13)	35.68(13.05)
AUGLAG	96.08(21.19)	96.26(22.37)
MMA	82.83(27.00)	98.62(18.46)
CCSAQ	90.03(24.39)	96.45(21.02)
BOBYQA	31.28(22.32)	28.02(11.76)
NELDERMEAD	39.68(22.10)	37.70(15.23)
TLPLTON	59.13(24.73)	63.74(19.80)
LBFGS	59.44(22.98)	54.05(18.23)

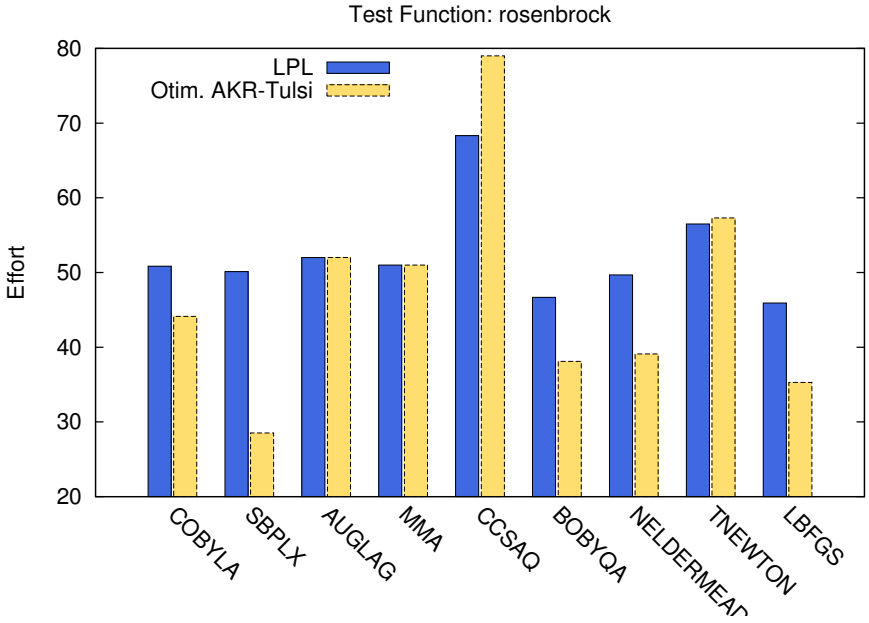
Função de teste: Rastrigin		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	38.33(20.73)	48.16(13.53)
SBPLX	31.19(17.37)	48.85(14.31)
AUGLAG	52.00(0.00)	52.00(0.00)
MMA	51.00(0.00)	51.00(0.00)
CCSAQ	67.67(23.57)	79.86(25.00)
BOBYQA	29.62(18.11)	39.30(16.59)
NELDERMEAD	37.98(21.85)	46.96(14.78)
TLPLTON	32.00(26.00)	35.25(17.83)
LBFGS	56.92(31.29)	48.10(24.50)

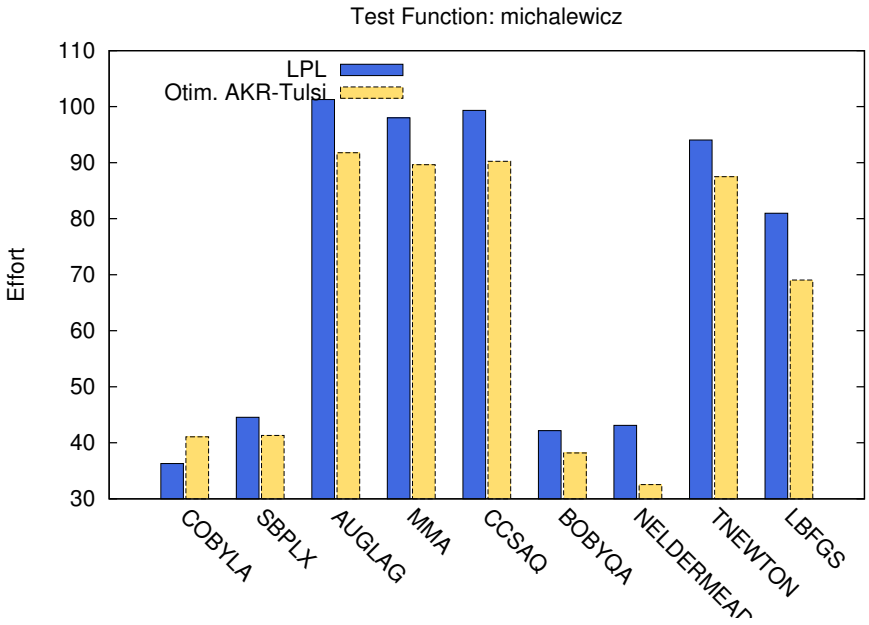
Função de teste: Raydan		
Local Search	Otim. AKR-Tulsi	LPL
COBYLA	47.28(15.28)	52.44(17.45)
SBPLX	31.37(19.45)	48.76(15.95)
AUGLAG	–	52.00(0.00)
MMA	51.00(0.00)	51.00(0.00)
CCSAQ	51.00(0.00)	–
BOBYQA	29.55(16.81)	54.17(14.96)
NELDERMEAD	30.79(22.12)	50.34(15.41)
TLPLTON	84.33(27.27)	102.06(21.55)
LBFGS	67.52(22.42)	74.71(25.54)

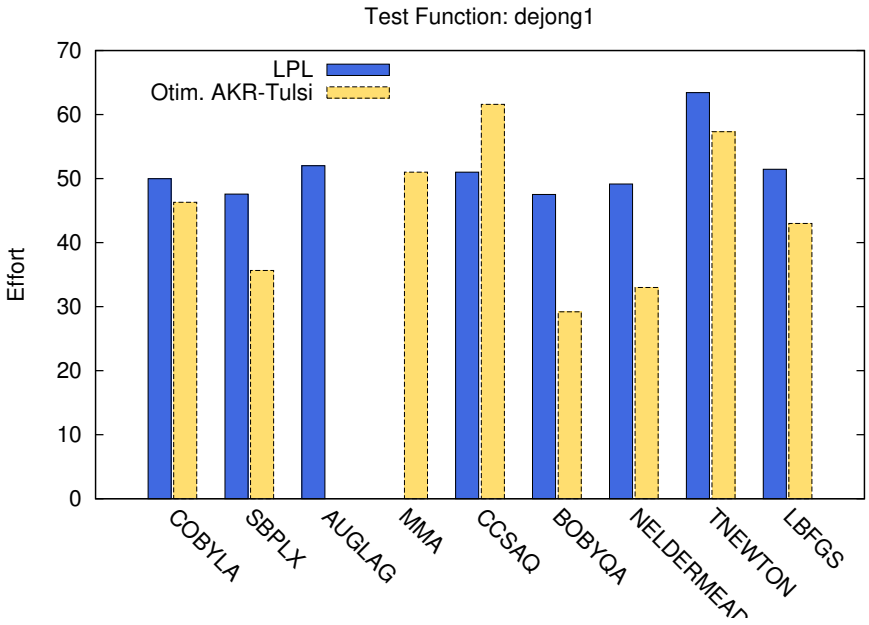


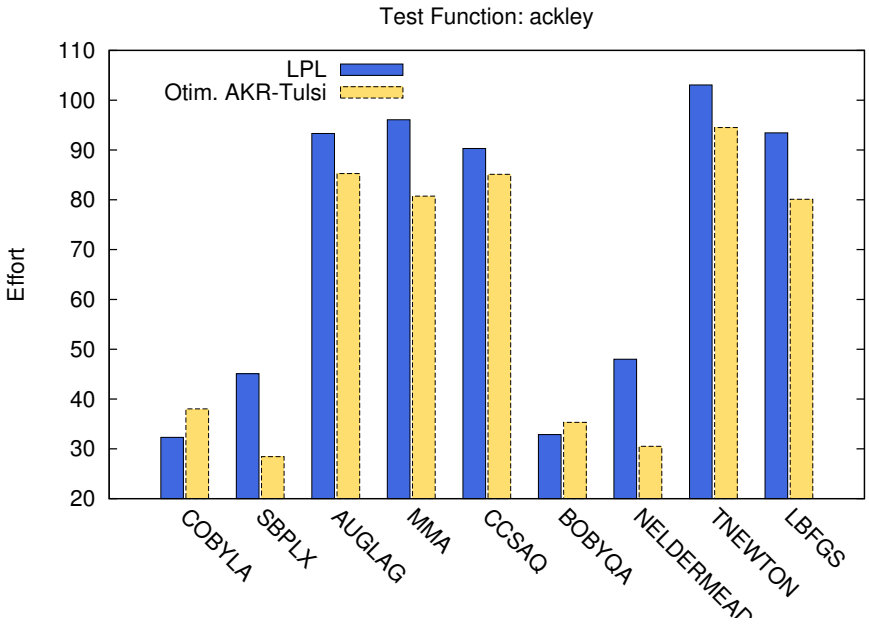

Test Function: griewank
LPL
Otim. AKR-Tulsi
Effort
60
50
40
30
20
10
0
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS

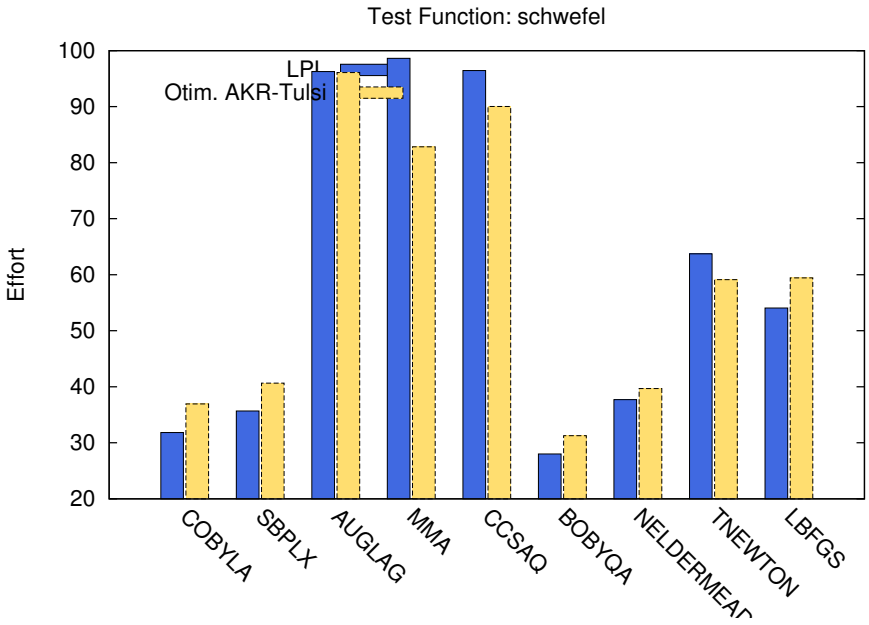

Test Function: sherkel
LPL
Otim. AKR-Tulsi
Effort
100
90
80
70
60
50
40
30
20
10
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS

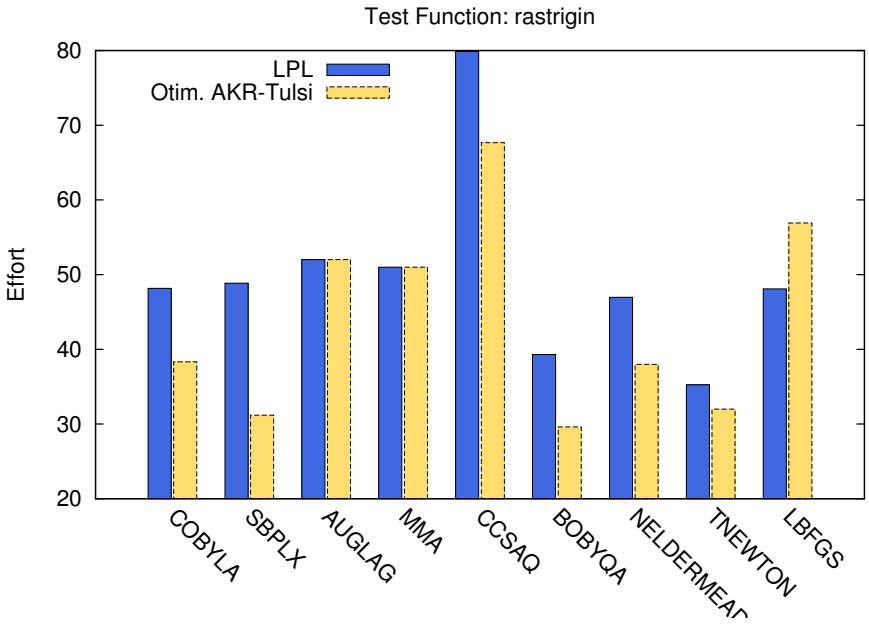

Test Function: rosenbrock
LPL
Otim. AKR-Tulsi
Effort
80
70
60
50
40
30
20
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS

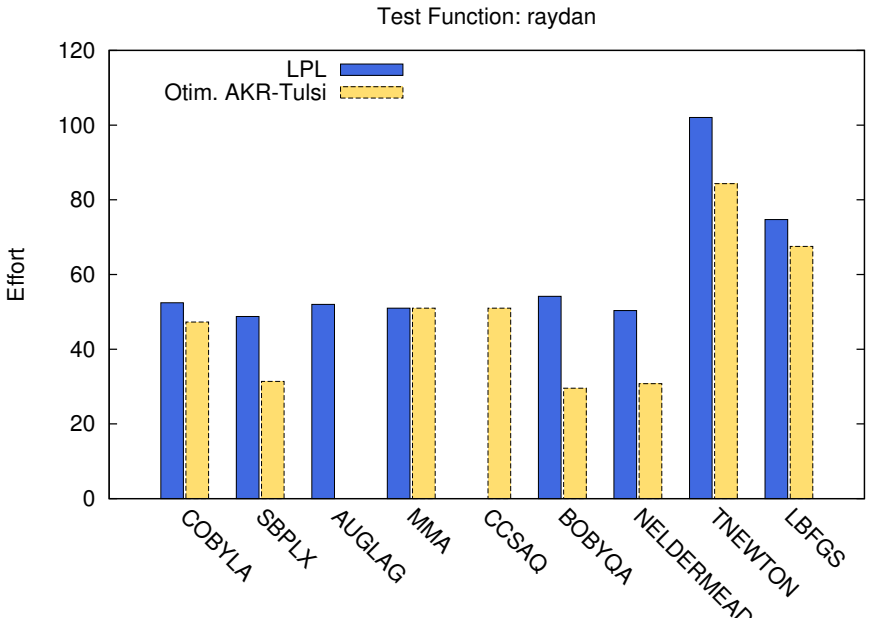

Test Function: michalewicz
LPL
Otim. AKR-Tulsi
Effort
110
100
90
80
70
60
50
40
30
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS


Test Function: dejong1
LPL
Otim. AKR-Tulsi
Effort
70
60
50
40
30
20
10
0
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS


Test Function: ackley
LPL
Otim. AKR-Tulsi
Effort
110
100
90
80
70
60
50
40
30
20
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS


Test Function: schwefel
LPL
Otim. AKR-Tulsi
Effort
100
90
80
70
60
50
40
30
20
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS


Test Function: rastrigin
LPL
Otim. AKR-Tulsi
Effort
80
70
60
50
40
30
20
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS


Test Function: raydan
LPL
Otim. AKR-Tulsi
Effort
120
100
80
60
40
20
0
COBYLA
SBPLX
AUGLAG
MMA
CCSAQ
BOBYQA
NELDERMEAD
TNEWTON
LBFGS

Apêndice B

Resultados computacionais para o Algoritmo LPL

As Tabelas B.1, B.2 e B.3 exibem o desvio padrão das amostras do número de avaliações da função objetico no Algoritmo LPL para 1, 2 e 3 variáveis respectivamente. As Tabelas B.4, B.5 e B.6 exibem as taxas de sucesso para o Algoritmo LPL para 1, 2 e 3 variáveis respectivamente. Em vermelho o método que teve o pior desempenho, em azul o método que teve o melhor desempenho.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	**0.00**	3.55	0.00	0.00	150.4	**210.3**	0.00	0.00	3.27
Griewank	22.05	38.36	40.63	90.67	**212.7**	181.5	37.47	**21.55**	41.86
Shekel	**0.00**	0.00	3.27	**9.28**	2.82	2.60	3.37	0.00	0.00
Rosenbrock	**35.08**	45.33	52.07	96.28	**190.4**	146.1	55.62	52.44	50.81
Michalewicz	29.42	39.70	35.52	107.4	**136.3**	103.4	36.53	**23.93**	38.09
Dejong	**0.00**	0.00	3.27	62.03	128.4	**139.2**	3.37	0.00	0.00
Ackley	**48.83**	54.81	54.75	91.30	**182.4**	171.6	54.63	50.23	52.63
Schwefel	**0.00**	0.00	**26.05**	13.99	0.00	24.04	0.00	8.44	0.00
Rastrigin	28.23	32.11	**0.00**	40.89	**216.2**	207.0	34.35	17.76	34.00
Raydan	5.88	21.42	**0.00**	27.22	**177.9**	143.3	0.00	34.48	15.96

Tabela B.1: Desvio padrão dos experimentos do Algoritmo LPL para uma uma variável.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	1240	3163	849.5	**0.00**	**9011**	8935	712.7	452.5	602.9
Griewank	2757	1394	633.2	475.0	9369	**10116**	531.5	**271.4**	412.8
Shekel	**2736**	2692	1076	**0.00**	0.00	3.17	1305	2.18	1531
Rosenbrock	**307.4**	751.4	1258	6865	**8471**	7946	883.1	1713	714.2
Michalewicz	1817	2190	900.0	3552	**9537**	9376	670.2	**615.7**	658.8
Dejong	2625	917.5	909.1	**0.00**	7359	**7635**	1042	2.18	817.4
Ackley	1693	1778	1017	4867	**7990**	6736	1177	**918.7**	2156
Schwefel	4390	2498	1041	2038	**23.94**	**4840**	1030	182.6	851.1
Rastrigin	1006	825.5	856.9	1567	**8918**	6952	922.0	**110.3**	679.0
Raydan	2823	2181	791.9	**493.0**	**9429**	8425	1068	535.3	1336

Tabela B.2: Desvio padrão dos experimentos do Algoritmo LPL para duas variáveis.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	**1421**	5056	3018	2847	**8923**	5652	3628	2173	2425
Griewank	3866	2720	801.1	1460	**15369**	15037	4247	699.2	**695.4**
Shekel	6247	**7481**	1599	**0.00**	0.00	0.00	843.6	0.00	1154
Rosenbrock	2196	**1306**	1390	3521	**17103**	15874	1806	3026	1931
Michalewicz	5928	2770	–	4859	**13681**	8277	2209	4610	**1468**
Dejong	5084	3581	1082	843.7	**14373**	9545	1241	**2.38**	857.6
Ackley	–	1299	1674	3644	**15637**	14966	1797	1654	**1026**
Schwefel	8141	**9863**	557.0	**343.5**	2164	645.1	550.4	870.5	1155
Rastrigin	2024	785.8	1266	1727	12457	**12655**	868.4	**636.2**	798.6
Raydan	4042	3778	1325	**778.6**	15591	**16436**	1393	877.8	1058

Tabela B.3: Desvio padrão dos experimentos do Algoritmo LPL para três variáveis.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	**1.00**	**1.00**	**1.00**	0.37	**0.10**	0.12	**1.00**	**1.00**	**1.00**
Griewank	0.85	0.59	0.42	0.83	0.06	**0.04**	0.42	**0.87**	0.40
Shekel	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Rosenbrock	**0.87**	0.50	0.25	0.44	0.13	**0.10**	0.25	0.27	0.15
Michalewicz	0.82	0.46	0.68	0.21	**0.08**	0.08	0.68	**0.99**	0.58
Dejong	**1.00**	**1.00**	**1.00**	0.63	0.22	**0.16**	**1.00**	**1.00**	**1.00**
Ackley	0.57	0.44	0.32	0.23	0.13	**0.10**	0.32	**0.97**	0.34
Schwefel	0.91	0.84	**0.79**	**1.00**	**1.00**	**1.00**	0.79	**1.00**	0.88
Rastrigin	0.52	0.53	0.76	0.51	0.03	**0.02**	0.76	**0.98**	0.59
Raydan	**1.00**	**1.00**	**1.00**	0.87	0.11	**0.08**	**1.00**	**1.00**	**1.00**
AVERAGE	0.854	0.736	0.722	0.609	0.286	**0.27**	0.722	**0.908**	0.694

Tabela B.4: Taxa de sucesso para o Algoritmo LPL usando uma variável.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	0.96	0.98	0.92	0.86	0.37	**0.28**	0.93	**1.00**	0.98
Griewank	**1.00**	0.94	0.81	**1.00**	**0.39**	0.39	0.84	**1.00**	0.86
Shekel	0.95	0.94	0.74	**1.00**	**1.00**	**1.00**	**0.70**	**1.00**	0.77
Rosenbrock	0.90	0.93	0.95	0.65	0.35	**0.30**	**0.96**	0.95	0.95
Michalewicz	0.96	0.97	0.87	0.98	0.42	**0.36**	0.86	**1.00**	0.90
Dejong	0.99	0.97	0.93	**1.00**	0.72	**0.52**	0.93	**1.00**	0.93
Ackley	0.71	0.46	0.68	**0.99**	**0.33**	0.37	0.79	**0.99**	0.81
Schwefel	0.97	0.95	0.85	**1.00**	*__1.00__	0.94	0.87	**1.00**	**0.84**
Rastrigin	0.94	0.78	0.88	**1.00**	0.51	**0.41**	0.86	**1.00**	0.91
Raydan	0.99	0.99	0.94	**1.00**	0.34	**0.29**	0.90	**1.00**	0.94
AVERAGE	0.937	0.891	0.857	0.948	0.543	**0.486**	0.864	**0.994**	0.889

Tabela B.5: Taxa de sucesso para o Algoritmo LPL usando duas variáveis.

Função Teste	**Minimizador Local**								
	LBFGS*	TNEWT*	MMA*	COB	NMEA	SBP	AGL*	BOB	CCS*
Neumaier	0.94	**0.80**	0.88	**1.00**	0.96	0.91	0.89	0.98	0.90
Griewank	0.95	0.96	0.94	**1.00**	**0.57**	0.60	0.92	**1.00**	0.95
Shekel	0.95	0.92	0.93	**1.00**	**1.00**	**1.00**	0.92	**1.00**	**0.91**
Rosenbrock	0.69	0.75	0.85	0.72	0.43	**0.40**	0.89	**0.99**	0.88
Michalewicz	0.79	0.86	0.78	0.77	**0.59**	0.68	0.77	**0.99**	0.80
Dejong	0.99	0.99	0.97	**1.00**	**0.84**	0.86	0.99	**1.00**	0.92
Ackley	0.77	0.68	0.89	0.96	**0.46**	0.70	0.89	**1.00**	0.83
Schwefel	0.92	0.88	0.81	0.99	**1.00**	0.99	**0.73**	**1.00**	0.84
Rastrigin	0.78	0.87	0.95	**1.00**	**0.59**	0.64	0.94	**1.00**	0.96
Raydan	0.94	0.94	0.96	0.99	**0.39**	0.47	0.92	**1.00**	0.93
AVERAGE	0.872	0.865	0.896	0.943	**0.683**	0.725	0.886	**0.996**	0.892

Tabela B.6: Taxa de sucesso para o Algoritmo LPL usando três variáveis.

Apêndice C

Linguagem de Programação Neblina

Neblina é uma linguagem de programação[1] científica cujo objetivo é criar uma camada transparente para a programação usando aceleradores (GPU, CPU multicore,...) aumentando a produtividade através de uma linguagem simples e concisa. Para isto, usamos o padrão OpenCL (Khronos OpenCL Working Group (2011)) como ferramenta básica para o paralelismo.

Melhorar a produtividade no desenvolvimento de aplicações paralelas é um desafio atual de pesquisa. Podemos citar um projeto denominado High Productivity Computing Systems (HPCS) do departamento norte-americano Defense Advanced Research Projects Agency (DARPA) cujo o objetivo era produzir a nova geração de computação de alto-desempenho e de alta-produtividade. Neste contexto surgiram as seguintes linguagens de programação X10 (Saraswat et al. (2012)), Fortress (Allen et al. (2007)) e Chepel (Chamberlain et al. (2007)).

A Figura C.2 exibe um exemplo de código para o experimento de Cadeias de Markov discretas. O paralelismo implícito é aplicado nas funções `init` (linha 7 e 8) e na função `mat_mulvec` (linha 11). Foi realizada uma implementação em linguagem `C` para comparar o desempenho e produtividade com o código Neblina da Figura C.2. O programa em `C` usou aproximadamente 60 linhas de código (5 vezes mais que o programa em Neblina). O desempenho poderá ser comparado na Tabela C.1. Os valores entre parênteses exibem o *speedup* com relação a imple-

[1] www.lncc.br/∼pcslara/neblina

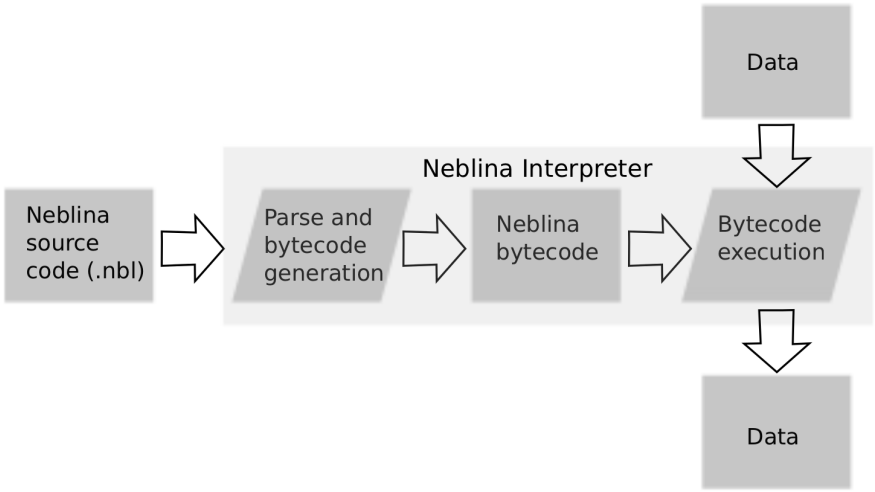


Figura C.1: Representação geométrica do operador U_f.

mentação sequencial em `C`. Neste caso, o código Neblina ficou até 70 vezes mais rápido que o sequencial na GPU Nvidia GTX-580.

```
1  using io
2  def main()
3     n = toint( args[2] )
4     T = toint( args[3] )
5     M = float[n,n]
6     p = float[n]
7     init( p, 0.0 )
8     init( M, 1.0/n )
9     p[1] = 1.0
10    for i = 1 : T
11       p = mat_mulvec( M, p )
12    end
13 end
```

Figura C.2: Discrete Markov Chain implementation written in Neblina.

Para alguns tipos de matrizes M onde é possível gerar o elemento M_{ij} dinamicamente a linguagem Neblina permite gerar os elementos por demanda. Assim não é preciso armazenar memória para os elementos da matriz. Desta forma, é possível trabalhar com matrizes realmente grandes. No caso do Algoritmo de Grover, por exemplo, é possível simular, em tempo razoável, a busca com até 2^{20} elementos. Caso armazenássemos esta matriz em disco seria necessário na ordem de 8TB.

Dimensão de $\mathbf{p}(t)$	**C (s)**	**Neblina (s)**					
	CPU (1 core)	CPU (16 cores)		GPU C2050		GPU GTX-580	
1000	5.36	2.24	(2.4x)	0.45	(11.9x)	0.32	(16.7x)
2000	21.66	4.91	(4.4x)	0.95	(22.8x)	0.64	(33.8x)
3000	48.69	8.38	(5.8x)	1.79	(27.2x)	1.10	(44.3x)
4000	86.56	12.46	(6.9x)	2.54	(34.1x)	1.66	(52.1x)
5000	135.14	17.26	(7.8x)	4.07	(33.2x)	2.39	(56.5x)
6000	194.61	25.15	(7.7x)	4.98	(39.1x)	3.18	(61.2x)
7000	264.36	35.19	(7.5x)	7.35	(36.0x)	4.24	(62.3x)
8000	346.29	44.95	(7.7x)	8.23	(42.1x)	5.35	(64.7x)
9000	437.14	55.96	(7.8x)	11.77	(37.1x)	6.63	(65.9x)
10000	540.52	68.91	(7.8x)	12.39	(43.6x)	7.88	(68.6x)
11000	653.78	82.93	(7.9x)	17.16	(38.1x)	9.62	(68.0x)
12000	778.74	99.02	(7.9x)	17.41	(44.7x)	11.07	(70.3x)
13000	913.29	115.25	(7.9x)	23.72	(38.5x)	13.17	(69.3x)
14000	1058.37	134.15	(7.9x)	23.30	(45.4x)	15.09	(70.1x)
15000	1215.55	154.81	(7.8x)	31.40	(38.7x)	17.31	(70.2x)

Tabela C.1: Comparação do experimento de Cadeias de Markov Discretas entre Neblina e C (sequencial, usando gcc) para $T = 1000$.

A Figura C.3 exibe uma implementação do Algoritmo de Grover usando esta técnica. Para isso é preciso descrever um trecho de código para gerar os elementos da matriz. Este código deverá ser fornecido em OpenCL (veja linhas 10, 11 e 12 da Figura C.3). Assim, quando a matriz é criada (linha 13) é gerado e compilado um kernel em OpenCL para esta matriz.

```
1    using math
2    using io
3    def main()
4        n = toint( args[2] )
5        N = 1 << n
6        psi = complex[N]
7        for i = 1 : N
8            psi[i] = c(1/sqrt(N), 0.0)
9        end
10       setup_u = "real c1 = "+ 2.0/N +";
11       re = (j==5)?((i==j)?(1-c1):-c1):((i== j)?c1-1:c1);
12       im = 0;"
13       U = rmatrix( setup_u, N, N )
14       for i = 1 : (3.1415/4)*sqrt(N)
15           psi = mat_mulvec( U, psi )
16       end
17       p = vec_conj( psi )
18       println( p )
19   end
```

Figura C.3: Implementação do Algoritmo de Grover em Neblina

Apêndice D

Hiperwalk: Simulador Paralelo de Passeio Aleatório Quântico

Hiperwalk (Lara et al. (2014a)) é um simulador de passeios quânticos livre e de código aberto que usa a linguagem de programação Neblina para explorar recursos como GPU, CPU multicore e outros dispositivos como Intel Xeon Phi com o objetivo de acelerar as simulações. Atualmente está em desenvolvimento no Grupo de Computação Quântica do LNCC e além do Neblina o Hiperwalk utiliza os seguintes softwares:

- python: responsável por preparar os dados de entrada, chamar o Neblina com os parametros definidos e interpretar os resultados.
- gnuplot: gera os gráficos de desvio padrão, média e dinâmica do passeio.
- numpy (biblioteca): fornece facilidades ao trabalhar com matrizes e vetores com python.

Atualmente o hiperwalk é capaz de simular os seguintes modelos de passeios quânticos discretos: 1D e 2D com moeda 1D e 2D sem moeda (modelo de Falk). Além disso, o simulador permite ao usuário definir livremente o operador de difusão através da opção `CUSTOM`. O programa também permite gerar animações para a visualização da dinâmica (veja Figura D.1).

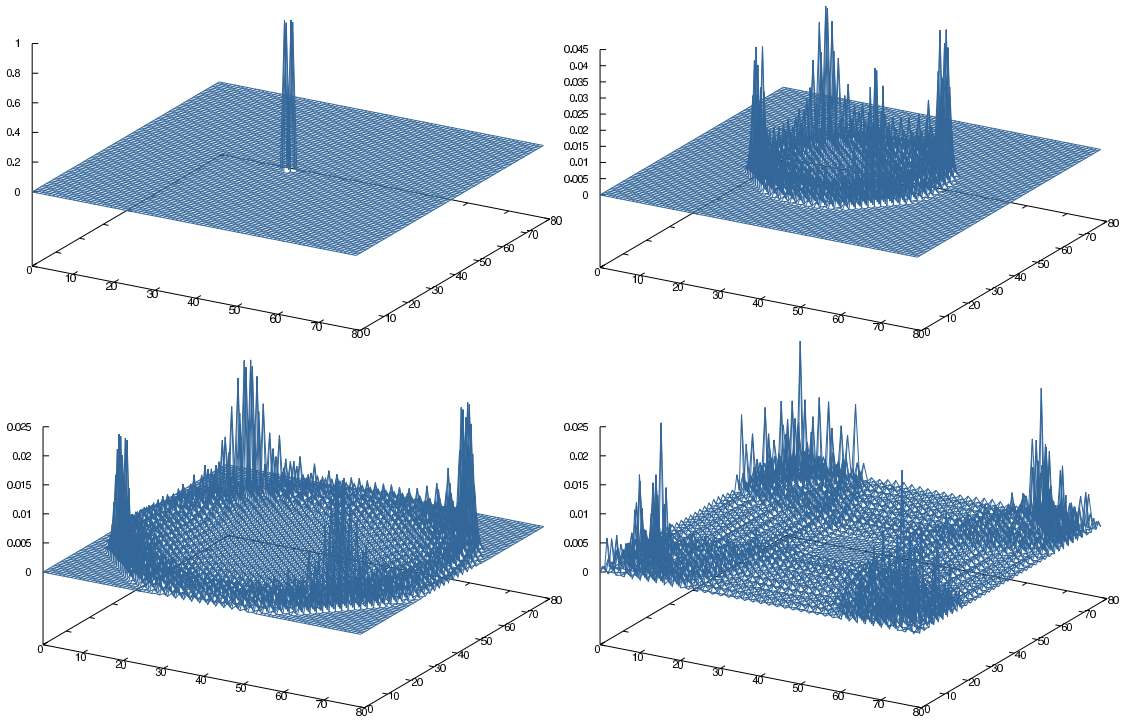


Figura D.1: Comportamento da onda na malha bidimensional 80×80. Neste caso é utilizado a moeada de Grover. O caminhante parte do meio do *grid* com condição inicial da moeda igual a $\frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$.