Identificação da maior elipse com excentricidade prescrita inscrita em um polígono não convexo através do *Continuous Grasp*

Victor Billy da Silva [1]
Marcus Ritt [2]
João Batista da Paz Carvalho [3]
Marcos José Brusso [4]
Juliano Tonezer da Silva [4]

Resumo: Este trabalho apresenta um algoritmo heurístico, baseado no *Continuous Grasp*, que busca encontrar a maior elipse, de excentricidade prescrita, inscrita no interior de um polígono não convexo e um estudo de caso da eficiência desta abordagem. Primeiramente, descreve-se o método *Continuous Grasp* e a formulação matemática do problema de otimização global. Após, é descrito o algoritmo implementado. Por fim, relata-se os resultados obtidos através de uma avaliação experimental.

***Palavras-chave*:** Continuos Grasp. Otimização. Polígono não convexo.

Abstract: *This paper presents a heuristic algorithm based on Continuous Grasp to finding the largest ellipse, with prescribed eccentricity, inscribed in a non-convex polygon and a case study of the efficiency of this approach. We describe Continuous Grasp, present a mathematical formulation of the global optimization problem and report the results of an experimental evaluation.*

***Keywords*:** *Continuous Grasp. Optimization. Non-convex polygon.*

1 Introdução

Problemas de otimização global buscam encontrar o máximo ou o mínimo de uma função tipicamente multimodal sobre um domínio discreto ou contínuo [1]. Por definição, um problema de otimização global visa determinar o máximo, ou o mínimo, global $x^* \in S \subseteq R^n$ tal que $f(x^*) \geq f(x), \forall x \in S$, sendo S uma região de R^n e $f : S \to R$.

O projeto 3D-Gemas estuda, entre outros problemas, o de maximizar o aproveitamento volumétrico de gemas de pedras preciosas utilizadas para lapidação. Como exemplo, pode-se citar o software Otimizador 3D Gemas, ferramenta computacional integrante da tecnologia 3D Gemas [4], com o objetivo de encontrar para cada gema, digitalizada tridimensionalmente, qual o modelo virtual de lapidação e o posicionamento do mesmo que resulte no maior aproveitamento volumétrico destas.

Recentemente, Hirsch et al. (2006), propuseram a meta-heurística *Continuous Grasp* (**C-Grasp**) para solucionar problemas contínuos de otimização global. O C-Grasp gera soluções, repetidamente, utilizando um algoritmo guloso construtivo, aplicando a essas soluções uma busca local estocástica visando encontrar soluções de boa qualidade. Uma vantagem desse método é que não são necessárias informações da derivada da função objetivo.

[1]Centro Tecnológico de Pedras Gemas e Joias do RS, Soledade (RS) - Brasil
{victorbilly.silva@gmail.br}
[2]Instituto de Informática, UFRGS, Porto Alegre (RS) - Brasil
{marcus.ritt@inf.ufrgs.br}
[3]Instituto de Matemática Pura e Aplicada, UFRGS, Porto Alegre (RS) - Brasil
{carvalho@mat.ufrgs.br}
[4]Curso de Ciência da Computação, UPF, Campus 1 - BR 285 - Passo Fundo (RS) - Brasil
{brusso,tonezer@upf.br}

http://dx.doi.org/10.5335/rbca.2012.2078

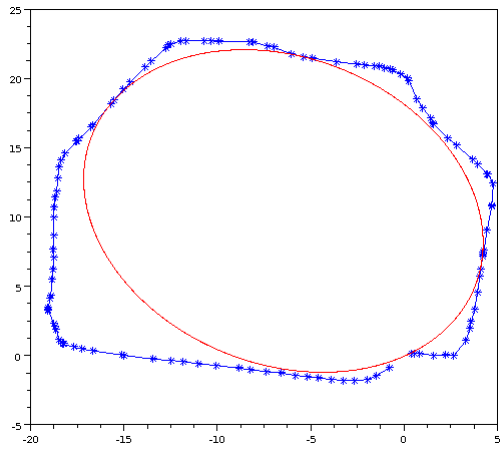


Figura 1: Exemplo de uma instância (azul) do problema e da solução ótima.

Atualmente, o software Otimizador 3D Gemas usa algoritmos genéticos como procedimento principal de otimização. Este trabalho tem como objetivo avaliar a meta-heurística C-Grasp como método alternativo. Para isso, foi avaliado o algoritmo proposto em [1], em um estudo de caso. O problema selecionado busca encontrar a elipse de maior área, com excentricidade prescrita, inscrita no interior de um polígono arbitrário. Esse problema é uma versão bidimensional de otimização solucionado pelo software otimizador 3D Gemas, servindo, então, como bom parâmetro de estudo para avaliar o C-Grasp.

No que concerne à organização do trabalho, na seção seguinte apresentamos a definição e formulação matemática do problema proposto. Na seção 3 o método C-Grasp é descrito. Na seção 4 os resultados computacionais são discutidos e, por fim, na seção 5 são aferidas as considerações finais sobre este trabalho.

2 Formulação Matemática do problema

Dados os pontos (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) de um polígono P no plano, nosso objetivo é encontrar a maior elipse com excentricidade prescrita contida no interior de P. Sendo a excentricidade da elipse fixa uma restrição do nosso problema [4], e sabendo que a elipse possui semieixos maior e menor denotados por a_0, b_0. A maior elipse contida na área determinada por esse polígono é definida por quatro parâmetros a serem identificados: escala α, coordenadas do centro da elipse x_c, y_c e ângulo de giro θ.

A Figura 1 exemplifica o problema apresentado. O contorno representa o polígono P e a elipse representa uma instância de solução obtida através do C-Grasp.

2.1 Problema de otimização

Sendo a elipse formada por parâmetros (x_c, y_c, θ) e sendo α a escala sobre os semi-eixos maior e menor da elipse (a_0, b_0), o problema de otimização global tem como objetivo encontrar os parâmetros (x_c^*, y_c^*, θ^*) que maximizam $\alpha(x_c, y_c, \theta)$ sujeitos às restrições

$$S(x_c, y_c, \theta, \alpha) = \{(x, y) \mid x = x_c + \alpha a_0 \cos\theta, y = y_c + \alpha b_0 \sin\theta, \theta \in [0, 2\pi]\} \subseteq P \quad (1)$$

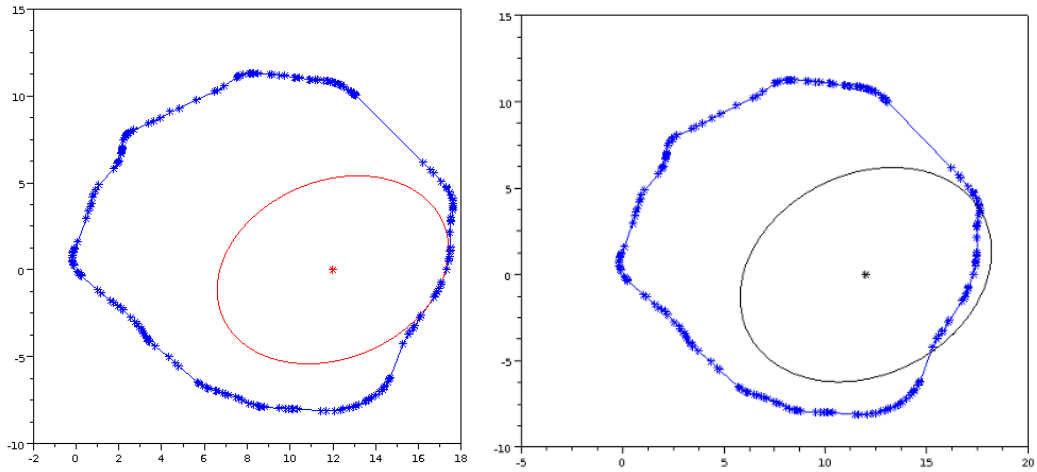
A restrição (1) garante que a elipse está no interior do polígono P. Assim, temos estabelecido um "problema de otimização global com restrições". Métodos para solução computacional desse problema existem na literatura [3]. Pela razão que nossa função objetivo não é diferenciável, o vetor gradiente $\nabla\alpha(x_c, y_c, \theta)$ em geral sequer está definido (mesmo se esse não fosse o caso, não é óbvio como calculá-lo analiticamente) e, portanto, uma extensa lista de métodos baseados no gradiente [3] não pode ser usada. Devemos, então, aplicar métodos de otimização que não usem derivadas, mas que tenham razoável convergência global, ao contrário do conhecido método de coordenadas cíclicas ou melhoramentos deste [3]. O método Continuous Grasp [2] apresenta tais características

necessárias para a resolução do problema, e nesse contexto será analisado no presente trabalho. Já a formulação da função objetivo, que gera soluções viáveis para este problema de otimização, é apresentada a seguir.

2.2 Estratégia computacional para calcular a função objetivo

A função objetivo modelada matematicamente para essa aplicação visa encontrar soluções viáveis para o problema de otimização proposto, isto é, dados os parâmetros (x_c, y_c, θ), sendo (x_c, y_c) as coordenadas do centro da elipse, θ o ângulo de giro da elipse, centrada em seu sistema de coordenadas locais, a função objetivo deve encontrar o valor máximo para α tal que a equação (1) seja satisfeita.

A Figura 2(a) traz um exemplo de uma solução viável para dados parâmetros, (x_c, y_c, θ) enquanto a Figura 2(b) demonstra uma solução não viável para o mesmo conjunto de parâmetros.



(a) Solução viável (b) Solução inviável

Figura 2: Exemplo de uma solução viável e de uma solução inviável

A estratégia computacional adotada para avaliar a função objetivo consiste em buscar linearmente (método de bissecção simples) na direção da componente α, em um dado intervalo $[l, u]$, onde l é o limitante inferior e u um limitante superior para α. A busca por bissecção usa um método de penalização que verifica se a elipse formada por dados parâmetros, mais uma escala α, está totalmente inscrita em um polígono P. Caso isso não ocorra a solução é penalizada em função da área da elipse no exterior do polígono. Na Figura 3 podemos observar, ainda para o mesmo trio de parâmetros citados acima, o comportamento da função objetivo, utilizando-se da função penalizadora, para valores de α entre 0.1 e 0.6. O máximo global ocorre com $\alpha = 0.47$ e para valores maiores ocorre a penalização, como por exemplo para $\alpha = 0.5$.

O comportamento da função penalizadora justifica a utilização do método de bissecção simples na busca unidimensional executada pela função objetivo, pois esta garante que só existe um máximo global para essa função em questão $(\alpha \approx 0.47)$ e para qualquer valor $\alpha > 0.47$ o valor da função é negativo, ou seja, há penalização, pois a elipse é parcialmente exterior ao polígono.

Ainda, para tornar a função objetivo mais explícita, exemplificaremos o método de penalização formulado.

2.3 Função penalizadora

Conforme visto, a função penalizadora atribui penalidades para um conjunto de parâmetros e mais uma escala α. Para isso, é necessário verificar se a elipse S é interior a P, ou seja, se nenhum dos segmentos de P a intercepta.

Visto que P não é necessariamente convexo, um modo robusto de averiguar se S é interior a P consiste em percorrer uma lista de pontos $\{\bar{x_j}, \bar{y_j}\}$ ótimos (segundo critério de ortogonalidade), distribuídos ao longo das

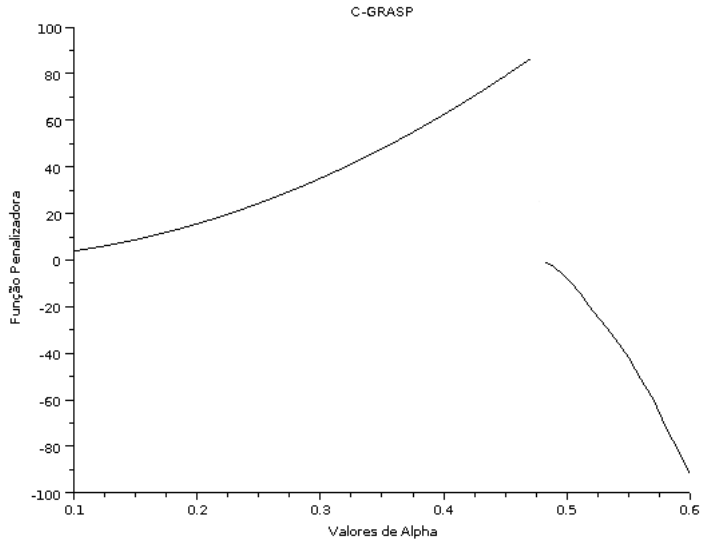


Figura 3: Gráfico da função objetivo

arestas de P. Após, transformamos cada um desses pontos para o sistema de coordenadas locais da elipse S. Sendo (x'_j, y'_j) a representação de $(\bar{x_j}, \bar{y_j})$, no sistema de coordenadas locais de S, sua transformação se dá por meio das equações

$$x'_j = cos(\theta)(\bar{x_j} - x_c) + sen(\theta)(\bar{y_j} - y_c); \tag{2}$$

$$y'_j = -sen(\theta)(\bar{x_j} - x_c) + cos(\theta)(\bar{y_j} - y_c); \tag{3}$$

Finalmente, o ponto (x'_j, y'_j) é exterior à fronteira da elipse canônica se e somente se

$$\frac{x'_j}{a^2} + \frac{y'_j}{b^2} > 1 \tag{4}$$

Assim, se para todos os pontos ótimos a equação (4) for satisfeita, a elipse S é interior a P, e o valor da função penalizadora nesse caso é α. Caso contrário, se a elipse S possui partes exteriores a P, um valor fp de penalização é estipulado da seguinte maneira:

$$fp = \sum_{j \in n} g(c_j(x_j)) \tag{5}$$

onde

$$g(c_j(x_j)) = max(1 - \frac{x'_j}{a^2} + \frac{y'_j}{b^2},\ 0). \tag{6}$$

Dessa forma, com a formulação da função objetivo, utilizando penalidades, pode-se, então, implementar o método C-Grasp, que será relatado a seguir.

3 Método C-Grasp

C-Grasp é uma meta-heurística que tem como objetivo solucionar problemas contínuos de otimização global [1]. C-Grasp é um método multipartida que consiste, basicamente, em efetuar dois passos:

1. uma fase de construção que combina técnicas gulosas e randomizadas, para gerar soluções iniciais custo-eficientes;

2. uma busca local, visando melhorar as soluções construídas na fase anterior, enquanto um dado critério de parada não for satisfeito. Geralmente se estabelece o número de iterações como critério. Nessa seção serão apresentadas as principais características desse método.

3.1 Funcionamento do C-Grasp

C-Grasp visa otimizar globalmente (minimizar ou maximizar) uma função objetivo $f(\cdot)$ com domínio de dimensão n em um universo representado pelos vetores l e u. Tem-se ainda os parâmetros h_s, h_e , p_{lo}, sendo que h_s, h_e definem, respectivamente, o início e o fim da densidade da grade de discretização do problema, ao passo que p_{lo} define a porção da vizinhança nas "redondezas" da solução atual, onde será realizada a busca local.

C-Grasp é um procedimento multipartida, sendo executado até que um dado critério de parada seja satisfeito. Cada vez que o critério de parada não é satisfeito, uma nova iteração C-Grasp acontece, inicializando, primeiramente, uma solução x através da escolha randômica em um domínio uniformemente distribuído em $\mathbb{R}^n$. Após isso, são executados os procedimentos de construção e de busca local. Por fim, se o valor da função objetivo $f(x)$, retornada do procedimento de busca local, for melhor que o valor da atual melhor solução $f(x^*)$, a melhor solução $f(x)$ e o conjunto de soluções x são atualizados. Nas próximas seções serão descritas, em detalhes, as fases de construção e de busca local.

3.2 Fase de construção

A fase de construção busca – coordenada a coordenada do vetor de solução x, combinando características gulosas, aleatórias e adaptativas –, uma melhora na solução atual gerada aleatoriamente no início do C-Grasp. Nessa fase todas as coordenadas do vetor de solução x, ou seja, todos os parâmetros que se buscam encontrar, são considerados livres e, conforme a solução for construída, essas coordenadas vão sendo fixadas. Um pseudocódigo exemplificando a fase de construção é apresentado na sequência.

Sucintamente, o método consiste de um laço exterior que percorre cada coordenada i, ainda não fixada, do vetor x, e executa uma busca unidimensional (*line search*) na direção dessa coordenada i, sendo que as demais $n-1$ coordenadas permanecem sem alterar seus valores atuais. Posteriormente, uma lista restrita de candidatos *LRC* é preenchida com as coordenadas i ainda não fixadas. A *LRC* recebe, para cada coordenada, um valor de função objetivo que não é maior do que um limite acima do menor valor de função objetivo encontrado, para todas as coordenadas. Este limite estabelecido é um valor μ, escolhido aleatoriamente entre 0 e 1, e cada coordenada i só será inserida na lista *LRC* se seu valor estiver entre o limite μ do intervalo da distância, entre o maior e menor valor de função objetivo já calculados. Explicitamente isso significa que para $\mu = 0$ o método irá escolher gulosamente a coordenada com o menor valor de função objetivo, e para $\mu = 1$ irá escolher alguma coordenada aleatoriamente.

Após, uma coordenada j da *LRC* é eleita randomicamente, e essa coordenada j é, então, fixada. Assim, todas as coordenadas em x serão fixadas, uma a uma, repetindo esse procedimento. Quando todas as coordenadas estiverem fixadas, o novo conjunto solução x é retornado, encerrando essa fase.

3.3 Fase de busca local

O procedimento de busca local tem como objetivo melhorar a solução x decorrente da fase de construção. Para isso, verifica-se na vizinhança de x^* se existe um x', tal que o valor da função $f(x')$ seja melhor que $f(x^*)$. A busca local examina um número de pontos *PontosParaExaminar* – porção p_{lo} da vizinhança da solução x^* – contidos em uma grade S_h, que tem sua discretização inicial em h_s e final em h_e. O algoritmo a seguir apresenta

um pseudocódigo deste procedimento.

```
   Data: x, funçãoObjetivo
   Result: houveMelhora, x
 1 coordenadasLivres← {1, 2, ..., n};
 2 μ ← rand(0, 1);
 3 usarNovamente ← falso;
 4 while livres ≠ 0 do
 5     gMin ← +∞;
 6     gMax ← −∞;
 7     for i ← 1 to n do
 8         if i ∈ livres then
 9             if usarNovamente = falso then
10                 z_i ← buscaLinear(x, funcaoObjetivo);
11                 g_i ← f(x*_i);
12             end if
13         end if
14     end for
15     LRC ← 0 ;
16     Limiar ← gMin + mu · (gMax-gMin);
17     for i=1 to n do
18         if i ∈ coordenadasLivres and g_i ≥ Limiar then
19             LRC ← LRC ⋃ i ;
20         end if
21     end for
22     j ← selecionaELementoRandomicamente(LRC);
23     if x_j = z_j then
24         usarNovamente = verdadeiro;
25     end if
26     else
27         x_j ← z_j ;
28         usarNovamente = falso;
29         houveMelhora ← verdadeiro;
30     end if
31     coordenadasLivres ← coordenadasLivres \j;
32 end while
```

Algoritmo 1: Pseudo-Código da fase de Construção do C-GRASP

```
   Data: x, funçãoObjetivo
   Result: x, houveMelhora
 1 x*← x;
 2 f*← f(x);
 3 NumPontosNaGrade ← ∑_{1≤i≤n} ⌈(u_i − l_i)/h⌉ ;
 4 PontosParaExaminar ← ⌈p_lo · NumPontosNaGrade⌉ ;
 5 PontosExaminados ← 0 ;
 6 while PontosExaminados ≠ PontosParaExaminar do
 7     PontosExaminados ← PontosExaminados + 1;
 8     x ← selecionaElementoRandomicamente(B_h(x*));
 9     if l ≤ x ≤ u and f(x) > f* then
10         x*← x;
11         f*← f(x);
12         houveMelhora ← verdadeiro;
13         PontosExaminados ← 0;
14     end if
15 end while
```

Algoritmo 2: Pseudo-Código da fase de Busca Local do C-GRASP

Um laço externo é repetido *PontosParaExaminar* vezes, sendo que interno a esse laço um ponto x' é selecionado aleatoriamente em um conjunto B_h. Os pontos em B_h são calculados por meio da projeção dos pontos da grade S_h, em uma hyperesfera, com raio de tamanho h, centralizada em x^*. Se o ponto x' eleito pertence ao domínio $[l, u]$, isto é, $l \leq x' \leq u$, e $f(x') < f(x^*)$, então, os valores de x^* são atualizados para x'.

Por fim, a implementação do algoritmo para o presente trabalho foi baseada no pseudocódigo do Continuous Grasp, descrito em [1], que contém mais detalhes sobre o algoritmo.

4 Resultados experimentais

A métrica utilizada para avaliar o método C-Grasp consistiu em aplicar o método em um conjunto de quatro polígonos não convexos. Para esse conjunto de polígonos, observou-se o número de avaliações da função objetivo, o tempo desprendido para encontrar a maior elipse, e o valor da escala α dessa elipse final. Utilizou-se diferentes parâmetros para o C-Grasp. Esta seção apresenta os resultados obtidos para esses testes.

4.1 Ambiente de testes

Todos os testes foram executados com um Intel Core i7 CPU 2.80 GHz 8 MB cache e 12 GB de memória, usando o sistema operacional Ubuntu 9. O algoritmo foi implementado na linguagem de programação C++ e compilado com GNU g++ 4.4.1. Além disso, utilizou-se o algoritmo Mersenne Twister [6] para a geração de números aleatórios, distribuídos uniformemente.

4.2 Resultados numéricos

Conforme visto no método C-Grasp, a informação necessita de três parâmetros para sua execução, h_s, h_e, p_{lo}, e, além disso, necessita que seja definido um critério de parada. O parâmetro h_s foi invariante para todos os testes efetuados, tendo como valor $h_s = 1.0$, e o critério de parada adotado foi o número de iterações ($maxIt$). Adicionou-se uma restrição ao parãmetro p_{lo}, que tem seu valor fixo em 0.7, limitando seu número máximo. Foram efetuados, variando esses parâmetros, quatro testes:

- (a) $h_e = 0.1, min(p_{lo}, 100)$
- (b) $h_e = 0.1, min(p_{lo}, 500)$
- (c) $h_e = 0.1, min(p_{lo}, 1000)$
- (d) $h_e = 0.01, min(p_{lo}, 100)$

Para os testes (a), (b) e (c) avaliou-se também o número máximo de iterações $maxIt$ (critério de parada), testando com valores para $maxIt$: 1, 2, 5, 10. Para o teste (d) utilizou-se $maxIt$=2.

Utilizou-se ainda, para o C-Grasp, como domínio para a função objetivo

$$x_{min} \leq x_c \leq x_{max}, \text{onde } [x_{min}, x_{max}] \subset P, \tag{7}$$

$$y_{min} \leq y_c \leq y_{max}, \text{onde } [y_{min}, y_{max}] \subset P, \tag{8}$$

$$0 \leq \theta < 2\pi. \tag{9}$$

Os valores utilizados como semieixos para a elipse foram $a_0 = 12.432$ e $b_0 = 10.0$, onde a representa o semieixo maior e b o semieixo menor, sendo a excentricidade obtida pela relação entre esses eixos.

As tabelas a seguir demonstram os resultados obtidos, sendo esses a média de 100 execuções do C-Grasp para cada teste. Estas possuem a seguinte nomenclatura: a coluna *n* representa a quantidade de vértices de cada polígono; o tempo é expresso em segundos (coluna *t(s)*); a coluna α apresenta a média final do resultado do C-Grasp; a coluna *D.P.* (α) representa o desvio padrão da população de testes; e a coluna *f(x)* traz o valor médio de avaliações da função objetivo. A Tabela 1 contém os dados do teste (a); a Tabela 2, do teste (b); a Tabela 3 do teste (c), e a Tabela 4, do teste (d).

Os dados apresentados nas tabelas mostram coerência nos resultados, sendo que o desvio padrão para todos os testes é relativamente baixo, e o valor final α do C-Grasp ficou com duas casas decimais exatamente iguais para todos os testes (exceto um valor 0.4198 na Tabela 1). O que significa que o algoritmo implementado converge

Tabela 1: Dados do teste (a)

Teste (a) - hs:1.0, he:0.1, maxPlo:100																
	MaxIter: 1				MaxIter: 2				MaxIter: 5				MaxIter: 10			
n	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**
212	0,23	0,7414	0,0027	104	0,44	0,7424	0,0005	208	1,03	0,7428	0,0005	500	2,07	**0,7431**	**0,0005**	1016
392	0,37	0,4198	0,0031	95	0,73	0,4212	0,0009	191	1,82	0,4218	0,0005	480	3,59	**0,4220**	**0,0003**	953
23	0,05	0,6781	0,0090	102	0,07	0,6831	0,0050	203	0,15	0,6853	0,0007	504	0,28	**0,6856**	**0,0005**	1007
122	0,15	0,6919	0,0027	106	0,27	0,6933	0,0011	213	0,66	0,6939	0,0006	535	1,30	**0,6942**	**0,0004**	1072

Tabela 2: Dados do teste (b)

Teste (b) - hs:1.0, he:0.1: maxPlo:500																
	MaxIter: 1				MaxIter: 2				MaxIter: 5				MaxIter: 10			
n	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**
212	0,36	0,7423	0,0006	105	0,70	0,7424	0,0005	210	1,73	0,7429	0,0005	523	3,42	**0,7433**	**0,0004**	1042
392	0,65	0,4213	0,0014	102	1,28	0,4218	0,0004	203	3,19	0,4221	0,0003	508	6,30	**0,4222**	**0,0002**	1012
23	0,06	0,6797	0,0078	104	0,11	0,6833	0,0053	209	0,25	0,6854	0,0005	525	0,47	**0,6857**	**0,0003**	1032
122	0,24	0,6926	0,0024	111	0,46	0,6936	0,0010	223	1,12	0,6941	0,0004	560	2,22	**0,6943**	**0,0004**	1113

seguramente para a solução global, ou em suas proximidades, e não há divergência de respostas (o que pode acontecer em métodos heurísticos).

Analisando os dados nas tabelas correspondentes, pode-se observar, como é esperado, que quanto maior o número de iterações – critério de parada $maxIt$ – maior é o grau de confiabilidade da resposta (menor o desvio padrão), maior a média da solução α e, como consequência negativa, maior o tempo de execução.

O parâmetro p_{lo} interfere diretamente no tempo de execução do método; no entanto, aumentando seu valor, nem sempre corresponde ao ganho em confiabilidade ou em média da resposta α. Isso pode ser observado comparando-se os dados apresentados nas tabelas, com o parâmetro $p_{lo} = 500$ x $p_{lo} = 1000$, os valores do desvio padrão são os mesmos para as duas variações de teste (salvo a quantidade de casas decimais especificadas = 10^3). O ganho na média final α é observado apenas na quarta casa decimal, não representando um aumento significativo na resposta.

O teste (d) avalia a influência do parâmetro h_e no tempo e na confiabilidade da resposta, aumentando a densidade da grade de discretização do domínio com $h_e = 0.01$. É possível observar que esse teste apresentou um melhor resultado (maior média α) para três dos quatro polígonos testados, com uma confiabilidade (desvio padrão relativamente baixo) boa. Isso indica que o parâmetro h_e interage diretamente na qualidade da resposta e com um h_e elevado pode-se usar um número baixo de iterações (*numIt=2*), nivelando o tempo gasto de processamento.

Os dados apresentados comprovam a eficácia do C-Grasp. Apesar disso, um método simples e totalmente aleatório é introduzido na próxima subseção em comparativo ao C-Grasp. Esse método deve comprovar a qualidade do C-Grasp quanto ao uso do tempo de processamento, demonstrando que mesmo sendo randômico a meta-heurística C-Grasp converge para uma solução global, utilizando eficazmente o tempo desprendido na sua execução.

4.3 Comparando o C-Grasp com um método aleatório

Em comparativo ao método C-Grasp utilizado para este trabalho, fizemos a implementação de um método completamente aleatório. Esse método gera, a cada iteração, três valores aleatórios para os parâmetros x_c, y_c, θ e, gulosamente, armazena o maior valor de α resultante da função objetivo cada vez que essa é avaliada. Utilizou-se como critério de parada o número máximo de avaliações à função objetivo *maxAvaliações = 350*, sendo esse valor

Tabela 3: Dados do teste (c)

Teste (c) - hs:1.0, he:0.1: maxPlo:1000																
	MaxIter: 1				MaxIter: 2				MaxIter: 5				MaxIter: 10			
n	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**	**t(s)**	α	**D.P.(α)**	**f(x)**
212	0,50	0,7422	0,0005	104	1,00	0,7426	0,0006	208	2,45	0,7430	0,0005	518	4,96	**0,7433**	**0,0004**	1046
392	1,02	0,4215	0,0013	107	2,00	0,4219	0,0004	214	4,91	0,4222	0,0003	523	9,83	**0,4224**	**0,0002**	1049
23	0,09	0,6786	0,0084	107	0,15	0,6836	0,0050	209	0,35	0,6853	0,0017	523	0,69	**0,6857**	**0,0003**	1061
122	0,35	0,6927	0,0023	113	0,68	0,6937	0,0011	225	1,62	0,6942	0,0004	556	3,28	**0,6944**	**0,0004**	1122

Tabela 4: Dados do teste (d)

Teste (d) - hs:1.0, he:0.01, maxPlo:100					
		MaxIter: 2			
Pol	**n**	**t(s)**	α	**D.P.**(α)	**f(x)**
pol1	212	1,24	0,7446	0,0001	326
pol2	392	1,93	0,4226	0,0005	304
pol3	23	0,17	0,6842	0,0054	318
pol4	122	0,72	0,6957	0,0002	336

a média (nas 100 execuções anteriores) de avaliações da função objetivo – necessário no teste (d) para que o C-Grasp alcance o fim de sua iteração. A Tabela 5 traz os resultados do método aleatório implementado, utilizando para as colunas a mesma nomenclatura das tabelas anteriores.

Tabela 5: Dados do teste com o algoritmo randômico

Teste randômico					
		maxAvaliações: 350			
Pol	**n**	**t(s)**	α	**D.P.**(α)	**f(x)**
pol1	212	0,47	0,6781	0,0140	350
pol2	392	0,70	0,3804	0,0269	350
pol3	23	0,07	0,6256	0,0048	350
pol4	122	0,27	0,6243	0,0102	350

O algoritmo randômico realiza a tarefa em tempo consideravelmente menor. No entanto, o valor médio de α é bem menor que o valor atingido pelo C-Grasp, se comparado à qualquer um dos testes. Além disso, o desvio padrão do algoritmo aleatório é, também, relativamente maior que o valor obtido pelo C-Grasp, demonstrando a instabilidade do método completamente aleatório. Nesse sentido, pela análise dos dados gerados por esse algoritmo simples, é possível contrastar os dados com os obtidos pelos C-GRASP e concluir que esse método faz uso de suas técnicas, não apenas se utiliza de soluções totalmente aleatórias.

5 Considerações finais e trabalhos futuros

Neste trabalho apresentou-se a implementação de um algoritmo heurístico baseado no método C-GRASP [1] como solução ao problema de encontrar a maior elipse, com excentricidade prescrita, inscrita no interior de um dado polígono não convexo. Por meio deste estudo de caso mediu-se a potencialidade do C-Grasp e comparou-se também o seu desempenho com um método totalmente aleatório. Os dados apresentados demonstram a potencialidade desse método e sua eficácia na resolução do problema, pois o método convergiu seguramente para o máximo global, em todos os testes efetuados em um tempo satisfatório.

Almeja-se como trabalho futuro otimizar o tempo gasto no algoritmo atual implementado e contrastar os resultados obtidos com a meta-heurística C-Grasp com outros métodos de otimização global, tais como algoritmo genético, Hook and Jeeves [3], e avaliar qual o melhor método para solucionar esse problema. Por fim, sendo o C-Grasp a melhor alternativa para solucionar o problema, pretende-se implementar um algoritmo para encontrar o maior elipsoide num poliedro não convexo, utilizando-se desse método para o software Otimizador 3D-Gemas em comparativo à atual versão que utiliza algoritmos genéticos.

6 Agradecimentos

Agradecimento à Fapergs pelo apoio ao projeto 3D Gemas-II por meio do edital Fapergs 003/2009 ARD, com vigência até fev. 2011, prorrogado até jul. 2011.

Referências

[1] HIRSCH, M. J. et al. Speeding up Continuous GRASP. *European Journal of Operational Research*. v. 205, p. 507-521, 2010.

[2] HIRSCH, M. J. et al. Global optimization by continuous GRASP. *Optimization Letters*, 2006.

[3] BAZARRA M.; SHERALI, H.; Shetty, C. *Nonlinear programming, theory and algorithms*. Third Edition, Toronto: John-Wiley. 1993.

[4] BRUSSO, M. J. et al. Tecnologia 3D Gemas: otimização do aproveitamento de gemas coradas digitalizadas tridimensionalmente. In: *Tecnologias para o setor de gemas, joias e mineração*. Porto Alegre: IGEO/UFRGS, 2010, p. 40-52. 2010.

[5] FIOREST, M. B. *Algoritmos genéticos na otimização da lapidação de gemas coradas.* Trabalho de conclusão de curso - Instituto de Ciências Exatas e da Terra. Universidade de Passo Fundo, 2010.

[6] WAGNER, R. (2009) *Mersenne Twister Random Number Generator.* Disponível em: <http://pastebin.com/J92wAxXV>. Acesso em: 5 jun. 2011.