Hibridizando a metaheurística C-GRASP com o método de busca por padrões adaptativos para resolução de problemas de otimização global contínua

Tiago Maritan Ugulino de Araújo (UFPB, PB, Brasil) – tiagomaritan@lavid.ufpb.br
• UFPB – Campus I, Cidade Universitária, CEP: 58051-900, João Pessoa-PB
Lucídio dos Anjos Formiga Cabral (UFPB, PB, Brasil) – lucidio@de.ufpb.br
Roberto Quirino do Nascimento (UFPB, PB, Brasil) – quirino@de.ufpb.br

Recebido em: 20/08/08 Aprovado em: 06/10/08

Resumo

Recentemente, tem crescido, na literatura, o interesse em resolver problemas de otimização global contínua utilizando metaheurísticas. A Greedy Randomized Adaptive Search Procedure (GRASP) pertence a esse grupo de metaheurísticas. Recentemente, Hirsch et al. (2007) desenvolveram a primeira, e até então única, adaptação da metaheurística GRASP para o domínio contínuo, denominada C-GRASP (Continuous-GRASP). Posteriormente, Hirsch et al. (2008) desenvolveram um novo artigo, onde algumas melhorias no C-GRASP foram propostas. Nesse trabalho, introduziu-se o método EC-GRASP (Enhanced Continuous-GRASP), uma versão híbrida da metaheurística C-GRASP com o método de busca direcionado, Busca por Padrões Adaptativos (Adaptive Pattern Search - APS) para resolução de problemas de otimização global contínua. O EC-GRASP é um método simples que não utiliza cálculos de derivadas, tornando-o uma boa aproximação para resolução de problemas de otimização global contínua. A eficiência e robustez do método são avaliadas através da comparação do método EC-GRASP com a metaheurística C-GRASP proposta por Hirsch et al.(2007 e 2008). Para isso, um conjunto funções de testes multi-modal com mínimo global conhecido é utilizado. Os resultados computacionais atestam a eficiência e robustez do método.

Palavras-chave: *C-GRASP; Método de busca por padrões adaptativos; Otimização global contínua.*

Abstract

Recently, the interest in solving global continuous optimization problems using metaheuristics has grown in literature. Many of these metaheuristics were originally proposed for combinatorial problems and have been adapted to deal with continuous optimization problems. The Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic can be included in this group of metaheuristics. Recently, Hirsch et al. (2007) developed the first adaptation of GRASP metaheuristics for the continuous domain, called Continuous-Grasp (C-GRASP). Later, Hirsch et al. (2008) developed a new article and some improvements in C-GRASP were proposed. This work introduces the EC-GRASP (Enhanced Continuous-GRASP) method, a hybrid version composed of C-GRASP and the Adaptive Pattern Search (APS) method, a direct search method. The EC-GRASP is a simple method to implement and does not make use of derivate information. The efficiency and robustness of the method are evaluated by comparing EC-GRASP with C-GRASP proposed by Hirsch et al. (2007 e 2008). A set of test functions with global minimum known is used for this comparison. The computational results show the efficiency and robustness of the method.

Keywords: *C-GRASP; Adaptive Pattern Search; Global continuous optimization.*

GEPROS. Gestão da Produção, Operações e Sistemas – Ano 3, nº 4, Out-Dez/2008, p. 155-167

1. INTRODUÇÃO

O rápido crescimento no tamanho e na complexidade dos problemas encontrados na indústria, na ciência e na economia, aliado aos elevados índices de competitividade e o avanço tecnológico tem incentivado engenheiros, pesquisadores e economistas a obter soluções cada vez melhores e mais baratas para esses problemas. Esses fatores têm impulsionado um rápido desenvolvimento dos modelos e das técnicas de otimização ao longo dos últimos anos, tornando a otimização mais presente na indústria, no mercado financeiro e nas atividades governamentais, dentre outras.

De acordo com a natureza do problema, discreto ou contínuo, podemos dividir a otimização em dois campos de estudo: a otimização discreta e a otimização contínua. Enquanto a otimização discreta trata de problemas que possuem um conjunto de soluções finito ou enumerável, a otimização contínua lida com problemas cujo conjunto de soluções, a função objetivo e as restrições pertencem ao domínio contínuo. Nesse trabalho será explorada a otimização contínua.

Um problema de otimização global contínua tem por objetivo encontrar o valor das variáveis do problema que determinem o menor (mínimo) ou o maior (máximo) valor da função objetivo, onde várias soluções extremas podem existir. Matematicamente, a otimização global contínua procura uma solução $x^* \in S \subseteq \Re^n$, tal que $f(x^*) \leq f(x)$, $\forall x \in S$, onde S é uma região do $\Re^n$ e f é a função objetivo $f : S \rightarrow R$ A solução x^* é denominada mínimo global. Uma solução x' é denominada mínimo local, numa vizinhança local $S_0 \subseteq S$, se $f(x') \leq f(x)$, $\forall x \in S_0$.

Segundo Hedar e Fukushima (2006), recentemente, tem crescido, na literatura, o interesse em resolver problemas de otimização global contínua utilizando metaheurísticas. A maioria delas foram originalmente propostas para resolver problemas de otimização discreta e têm sido adaptadas para lidar com problemas de otimização contínua.

A metaheurística Greedy Randomized Adaptive Search Procedure (GRASP) pertence a esse grupo. A primeira, e até então única, adaptação da metaheurística GRASP para o domínio contínuo foi desenvolvida por Hirsch et al. (2007) e denominada de C-GRASP (Continuous-GRASP). Posteriormente, Hirsch et al. (2008) desenvolveram um novo artigo, onde algumas melhorias na metaheurística C-GRASP foram implementadas.

Contudo, em geral, essas metaheurísticas possuem uma convergência lenta, o que pode representar um elevado esforço computacional para encontrar a solução. Uma alternativa para contornar esse problema é a utilização de métodos de busca direcionada. De acordo com Hedar e Fukushima (2006), o papel dos métodos de busca direcionada é estabilizar a busca na vizinhança do mínimo local. Mais especificamente, ao invés de utilizar um método de busca aleatório na geração dos movimentos de vizinhança, utiliza-se os métodos de busca direcionado para explorá-la.

Nesse artigo, foi introduzida a metaheurística EC-GRASP (Enhanced Continuous GRASP), uma versão híbrida da metaheurística C-GRASP com o método de busca direcionada, Busca por Padrões Adaptativos (Adaptive Pattern Search – APS), para resolução de problemas de otimização global contínua.

O restante desse artigo é organizado da seguinte forma: Seção 2 - apresentação da metaheurística C-GRASP, proposta por Hirsch et al. (2008). Seção 3 - apresentação do método de busca direcionada APS utilizado no EC-GRASP. Seção 4 - apresentação do método proposto, o EC-GRASP. Seção 5 - apresentação dos experimentos computacionais. Seção 6 - apresentação das conclusões.

2. C-GRASP (CONTINUOUS GRASP)

A metaheurística GRASP, desenvolvida por Feo e Resende (1995), é um método iterativo composto por duas fases: uma de construção e uma de busca local. Na fase de construção, um conjunto de soluções de boa qualidade é gerado através de uma mistura de gulosidade e aleatoriedade. A fase de busca local tenta

melhorar a solução encontrada na fase de construção. Em cada iteração, uma solução é encontrada, sendo a melhor entre elas a solução final do problema.

Recentemente, Hirsch et al. (2007 e 2008) adaptaram a metaheurística GRASP para o domínio contínuo. Essa adaptação foi denominada de C-GRASP (Continuous-GRASP). O C-GRASP é um método simples, de fácil implementação e que não utiliza cálculos de derivada da função objetivo.

No C-GRASP, cada iteração inicia com uma discretização inicial do espaço de soluções e com um vetor solução x uniformemente distribuído sobre os limites inferior e superior do problema. Em seguida, os procedimentos de construção e busca local são chamados seqüencialmente, partindo da solução atual x. A principal diferença para o GRASP é que no C-GRASP a solução de saída do procedimento de busca local também é utilizada como entrada do procedimento de construção. À medida que o algoritmo progride, a grade de pontos discretos vai adaptativamente se tornando mais densa. A melhor solução encontrada em todas as iterações é a solução do problema. O pseudocódigo do C-GRASP é apresentado na figura 1.

```
procedimento C-GRASP (n, l, u, f(.), hs, he, ρlo)
1       f* ← ∞;
2       enquanto Critério de parada não satisfeito faça
3               x ← RandomUniforme(l, u);
4               h ← hs;
5               enquanto h ≥ he faça
6                       Impr_C ← false;
7                       Impr_L ← false;
8                       [x, Impr_C] ← Construção (x, f(.), h, n, Impr_C);
9                       [x, Impr_L] ← BuscaLocal (x, f(.), h, n, ρlo, Impr_L);
10                      se f(x) < f* então
11                              x* ← x;
12                              f* ← f(x);
13                      fim se;
14                      se Impr_C = false and Impr_L = false então
15                              h ← h/2_i /* toma a grade mais densa */
16                      fim se
17              fim enquanto
18      fim enquanto
19      retorne x*;
fim C-GRASP
```

Figura 1 – Pseudocódigo da metaheurística C-GRASP.

O C-GRASP possui, como parâmetros de entrada, a dimensão do problema n, os vetores l e u que representam os limites inferiores e superiores, respectivamente, a função objetivo f(.) e os parâmetros hs, he ρ_{lo}. Os parâmetros hs e he representam a densidade inicial e final da grade de discretização, respectivamente, e o parâmetro ρ_{lo} representa o percentual da vizinhança da solução atual que será visitada durante a fase de busca local.

O procedimento de construção inicia permitindo que todas as coordenadas de x possam mudar de valor (denominaremos de coordenadas não-fixas). A cada iteração, é realizada uma busca linear na direção de cada coordenada não-fixa de x com as outras n-1 coordenadas mantidas com os seus valores atuais. Uma Lista de Candidatos Restrita (LCR) com tamanho definido por $\alpha \in [0,1]$ é criada apenas com as coordenadas não-fixas. Um elemento da LCR é, então, escolhido aleatoriamente e essa coordenada é retirada da lista de coordenadas não-fixas. O procedimento é repetido até que o conjunto de coordenadas não-fixas esteja

vazio. O pseudocódigo do procedimento de construção do C-GRASP é apresentado na figura 2.

É interessante observar que o melhor candidato da LCR não é necessariamente o candidato escolhido, ou seja, a coordenada que será retirada do conjunto não-fixas. Utilizar um critério de aleatoriedade na decisão sobre qual das "boas" coordenadas escolher, ajuda o C-GRASP a examinar uma porção maior do espaço de busca e evita que o C-GRASP fique preso em mínimos locais.

Após a construção, aplica-se o procedimento de busca local que realiza uma busca na vizinhança da solução atual x. O método de busca local, então, determina em que pontos dessa vizinhança, se existirem, a função objetivo melhora. Se nenhuma solução melhor for encontrada, a solução atual é retornada. O procedimento de busca local é apresentado na figura 3.

```
procedimento Construção (x, f(.), h, n, Impr_C)
1       não-fixas ← [1,2,...,n];
2       α ← RandomUniforme(0,1);
3       Reuse ← false;
4       enquanto não-fixas ≠ ∅ faça
5               gMin ← -∞;
6               gMax ← +∞;
7               para i = 1,...,n faça
8                       se i ∈ não-fixas então
9                               se Reuse = false então
10                                      z_i BuscaLinear (x, f(.), h, n, i);
11                                      g_i ← f(x);
12                              fim se
13                              gMin > g_i então gMin ← g_i;
14                              gMax < g_i então gMax ← g_i;
15                      fim se
16              fim para
17              LCR ← ∅;
18              Threshold ← gMin + α (gMsx–gMin);
19              para i = 1,...,n faça
20                      se i ∈ não-fixas and g_i ≤ Threshold então
21                              LCR ← LCR ∪ {i};
22                      fim se
23              fim para
24              j ← Selecione aleatoriamene LCR;
25              se x_i ← z_i então
26                      Reuse ← true;
27              senão
28                      x_i ← z_i
29                      Reuse ← false;
30                      Impr_C ← true;
31              fim se
32              não-fixas ← não-fixas / {j};
33      fim enquanto;
34      retorne (x, Impr_C);
fim Construção
```

Figura 2 – Pseudocódigo do procedimento de construção do GRASP.

A vizinhança Bh(x) é definida como as projeções de todos os pontos da grade de discretização na hiper-esfera centrada em x com raio igual ao tamanho do passo de discretização atual h. Mais especificamente, seja a solução atual e h o parâmetro de discretização da grade de pontos. Definimos:

$$S_h(\bar{x}) = \{x \in S \mid l \leq x, \leq u, x = \bar{x} + \tau \,.\, h, \tau \in Z^n\}$$

$$B_h(\bar{x}) = \{x \in S \mid x = \bar{x} + h \,.\, (x' = \bar{x}) / \mid x' - \bar{x} \mid\mid, x' \in S_h(\bar{x}) \setminus \{\bar{x}\} \text{ e,}$$

O conjunto $S_h(\bar{x})$ corresponde ao conjunto de pontos em S que pertencem a grade de discretização. O conjunto $B_h(\bar{x})$ é o conjunto formado pela projeção dos pontos que pertencem a $S_h(\bar{x}) \setminus \{\bar{x}\}$ numa hiper-esfera de raio h centrada em $\bar{x}$. Maiores detalhes são apresentados por Hirsch et al. (2008).

```
procedimento BuscaLocal (x, f(.), h, n_e, ρ_lo, Impr_C)
1       x* ← x;
2       f* ← f(x);
3       NumPontosGrade ← ∏_(i-1)^n ⌈(u_i − l_i) / h⌉;
4       NumPontosParaExaminar ← ⌈ρ_lo . NumPontosGrade⌉
5       NumPontosExaminados ← 0;
6       enquanto NumPontosExaminados ≤ NumPontosParaExaminar
7               NumPontosExaminados ← NumPontosExaminados + 1;
8               x ← Selecione Aleatoriamene (B_h(x*));
9               se l ≤ x ≤ u and f(x) < f* então
10                      x* ← x;
11                      f* ← f(x);
12                      Impr_L = true;
13                      NumPontosExaminados ← 0;
14              fim se;
15      fim enquanto;
16      retorne (x, Impr_L);
fim Busca Local
```

Figura 3 – Pseudocódigo do procedimento de busca local do GRASP.

3. MÉTODO DE BUSCA POR PADRÕES ADAPTATIVOS (APS)

A idéia principal do Método de Busca por Padrões Adaptativos (Adaptive Pattern Search – APS), proposto por Hedar e Fukushima (2006), é baseada no método direção de descida aproximada (approximate descent direction – ADD) proposto por Hedar e Fukushima (2004). O método ADD é um procedimento livre de derivadas com uma grande habilidade de produzir direções de descidas. Essa direção é construída utilizando-se pontos ao redor da solução atual.

Mais especificamente, o APS constrói n direções padrões paralelas aos eixos coordenados, partindo de uma solução atual x, e gera n pontos $\{y_n\}_{i=1}^{n}$ ao longo dessas direções com o tamanho do passo de discretização h. A direção adaptativa v é computada utilizando esses n pontos, como segue.

$$v = \sum_{i=1}^{n} \omega_i u_i \text{, onde,}$$

$$\omega_i = \frac{\Delta f_i}{\sum_{j=1}^{n} |\Delta f_j|}, \quad i = 1,2,\text{K} \ ,\text{n},$$

$$u_i = -\frac{(y_i - x)}{|(y_i - x)|}, \quad i = 1,2,\text{K} \ ,\text{n},$$

$$\Delta f_i = f(y_i) - f(x), \ i = 1,2,\text{K} \ ,\text{n},$$

Uma contribuição do método EC-GRASP no método APS é que, após computar a direção v, o EC-GRASP aplica o método da seção áurea (Bazarra et al. (1989)) na direção v com comprimento inicial h e comprimento final he. A solução gerada pelo método da seção áurea é atribuída ao ponto yn+1. O método APS retorna o melhor dos pontos $\{y_n\}_{i=1}^{n}$.

No exemplo apresentado na figura 4, de duas dimensões (n=2), os dois pontos de vizinhança y1 e y2 são gerados na vizinhança da solução atual x, conforme figura 4 (a). Uma direção de descida aproximada v é computada utilizando os pontos y1 e y2. Se for definido que x é melhor que y1 e y2, então o vetor v é composto em direção aos vetores x- y1 e x- y2 com pesos inversamente proporcionais a $|f(x) - f(y_2)|$ e $|f(x) - f(y_1)|$, conforme figura 4 (b). Finalmente, aplica-se o método da seção áurea que gera o ponto y3, conforme figura 4 (c). A melhor das soluções y1, y2 e y3 é retornado como solução do método APS.

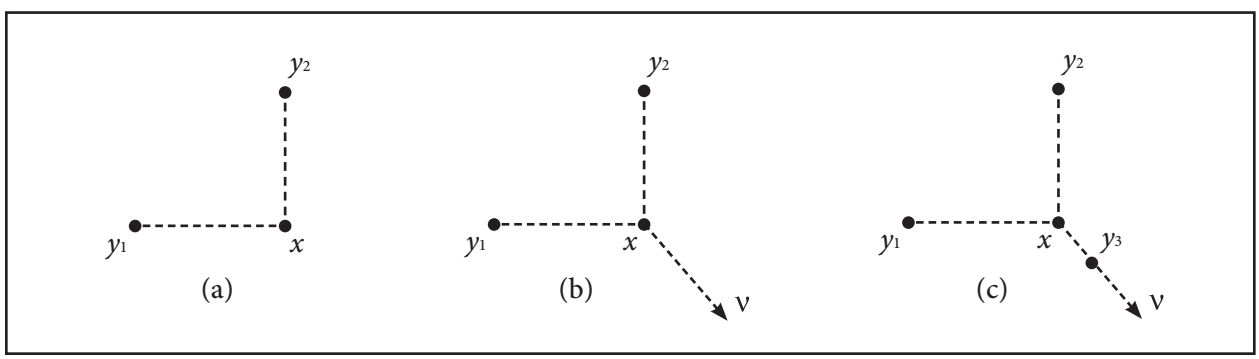

Figura 4 – Estratégia APS em duas dimensões.

4. EC-GRASP

Nessa seção foi apresentada a metaheurística EC-GRASP que é utilizada para resolução de problemas de otimização global contínua e sujeito a restrições nos limites do domínio (box constraints). Sem perda de generalidade, definimos o domínio S como sendo um hiper-retângulo $S = \{x = (x_1, x_2,..., x_n)\} \in \Re^n : l \le x \le u\}$, onde $l \in \Re^n$ e $u \in \Re^n$, tal que $u_i \ge l_i$ para $i = 1,..., n$. O problema considerado nesse artigo é encontrar $x^* = \arg\min \{ f(x) \mid l \le x \le u\}$, onde $f : \Re^n \to \Re$ e $l,x,u \in \Re^n$.

O EC-GRASP possui uma estrutura similar ao C-GRASP. Ele é um método iterativo composto por uma fase de construção gulosa e adaptativa e por uma fase de busca local. A solução retornada pelo método

é a melhor solução de todas as iterações. A solução de saída do procedimento de construção é utilizada como solução de entrada do procedimento de busca local e vice-versa. O EC-GRASP utiliza a mesmo procedimento de construção do C-GRASP, descrito na Seção 2. O pseudocódigo do procedimento EC-GRASP é apresentado na figura 5.

A principal diferença para o C-GRASP é que o EC-GRASP utiliza um novo procedimento de busca local baseado no método de busca direcionada APS e no conjunto vizinhança $B_h(x)$. Na próxima subseção detalharemos esse método.

Os parâmetros de entrada do EC-GRASP são n, l, u, $f(.)$, h_s e h_e e *MaxIters*. Os parâmetros n, l, u, $f(.)$, h_s e h_e são os mesmos parâmetros utilizados pelo C-GRASP e o parâmetro *MaxIters*, um parâmetro específico do EC-GRASP, representa quantas iterações sem melhora o procedimento de busca local deve explorar.

4.1. Busca Local

O método de busca local do EC-GRASP é um método de refinamento que explora um conjunto de vizinhanças diferentes aceitando, apenas, soluções melhores que a atual. Sempre que um conjunto vizinho não produz uma solução melhor que a solução atual, um outro conjunto vizinhança é explorado.

As vizinhanças do método de busca local são baseadas no conjunto $B_h(x)$ do C-GRASP e no método de Busca por Padrões Adaptativos (Adaptive Pattern Search – APS), descrito na Seção 3. O papel do método APS no procedimento de busca local é estabilizar a busca na vizinhança do mínimo local.

```
procedimento EC-GRASP (n, l, u, f(.), h_s, h_e, MaxIters)
1       f* ← ∞;
2       enquanto Critério de parada não satisfeito faça
3               x ← RandomUniforme(l, u);
4               h ← h_s;
5               enquanto h ≥ h_e faça
6                       Impr_C ← false;
7                       Impr_L ← false;
8                       [x, Impr_C] ← Construção(x, f(.), h, n, Impr_C);
9                       [x, Impr_L] ← BuscaLocal_APS (x, f(.), h, n, h_e, MaxIters, Impr_L);
10                      se f(x) < f* então
11                              x* ← x;
12                              f* ← f(x);
13                      fim se;
14                      se Impr_C = false and Impr_L = fase então
15                              h ← h/2; /* torna a grade mais densa */
16                      fim se;
17              fim enquanto;
18      fim enquanto;
19      retorne x*
fim C-GRASP
```

Figura 5 – Pseudocódigo do procedimento EC-GRASP.

Cada conjunto de vizinhança pode ser definido da seguinte forma: inicialmente é selecionado aleatoriamente uma solução x' que pertence a $B_h(x^*)$, onde h é o tamanho do passo de discretização do domínio e x^* é a melhor solução encontrada pelo procedimento de busca local. A partir disso, aplica-se o método de busca por padrões adaptativos (APS) em x', retornando o vizinho de x' com menor valor de função objetivo. O pseudocódigo do procedimento de busca local é apresentado na figura 6.

O procedimento BuscaLocal_APS possui, como parâmetros de entrada, a solução atual x, a função objetivo *f(.)*, o tamanho do passo de discretização atual h, a dimensão do problema n e os parâmetros he, MaxIters e ImprL. O parâmetro he representa o tamanho do passo de discretização final, utilizado no método da seção áurea do método APS. O parâmetro MaxIters representa quantas iterações sem melhora o procedimento de busca local deve explorar, ou seja, quantos conjunto vizinhança ele deve explorar, e o parâmetro ImprL indica se a solução retornada pelo procedimento BuscaLocal_APS é melhor ou não que a solução de entrada do procedimento.

O procedimento de busca local inicia, nas linhas 1 e 2, atribuindo a solução atual $x^* \in S \subseteq \Re^n$, a melhor solução da busca local x*. A linha 3 inicializa a variável NumIters que representa o número de iterações sem melhorar a solução x* com o valor zero. Partindo da solução atual x, no laço das linhas 4-14, o algoritmo seleciona o melhor vizinho obtido pelo algoritmo APS partindo da solução atual x com parâmetros h e he. Se essa solução retornada pelo procedimento APS for melhor que x* , linha 7, então a melhor solução da busca local x* é atualizada com o valor de x, linhas 8 e 9, a variável ImprL recebe o valor true, linha 10, e o parâmetro NumIters recebe o valor zero, linha 11. Na linha 13, seleciona-se uma solução aleatoriamente do conjunto $B_h(x)$ e a atribui à solução atual x. Essa operação deslocará a solução atual x para uma solução aleatória em $B_h(x)$, permitindo que o APS explore um outro conjunto vizinhança na próxima iteração do laço.

```
procedimento BuscaLocal_APS (x, f(.), h, n, h_e, MaxIters, Impr_L)
1       x* ← x;
2       f* ← f(x);
3       Numbers ← 0;
4       enquanto Numbers ≤ MaxIters faça
5               Numbers ← NumIters + 1;
6               x ← APS (x, h, h_e);
7               se l ≤ x ≤ u and f(x) ← f* então
8                       x* ← x;
9                       f* ← f(x);
10                      Impr_L ← true;
11                      NumIters ← 0;
12              fim se;
                /* Seleciona outro conjunto de vizinhança */
13              x ← SelecioneAleatoriamente(B_h(x*));
14      fim enquanto;
15      retorne (x, Impr_L);
fim BuscaLocal_APS
```

Figura 6 – Pseudocódigo do procedimento BuscaLocal_APS.

5. EXPERIMENTOS COMPUTACIONAIS

Nessa seção será descrito os experimentos computacionais realizados na metaheurística EC-GRASP. Inicialmente, far-se-á uma comparação do EC-GRASP com as versões do C-GRASP propostas por Hirsch et al. (2007 e 2008) para um conjunto de funções de teste padrão. Por questões de simplicidade, denominar-se-á a versão proposta em Hirsch et al. (2007) de "Antigo-CGRASP" e a versão proposta em Hirsch et al. (2007) de "Novo-CGRASP".

Os experimentos foram executados num notebook Dell Vostro 1000, com processador AMD-Athlon 64 X2 1,9 GHz e com 2 GB de memória RAM. O sistema operacional utilizado foi o Ubuntu Linux 8.0.4, kernel 2.6.24-16. A metaheurística EC-GRASP foi implementada utilizando a linguagem de programação C++ e o compilador GNU g++ 3.4.6. Os problemas testes utilizados nos experimentos computacionais estão listadas na tabela 1.

Tabela 1 – Problemas testes.

Problema Teste	Dimensões	Problema Teste	Dimensões
Branin (BR)	2	Rosenbrock (R10)	10
Goldstein e Price (GP)	2	Shekel(S4,5)	4
Shubert (SH)	2	Shekel(S4,7)	4
Hartmann(H3,4)	3	Shekel(S4,10)	4
Rosenbrock(R2)	2	Zakharov(Z5)	5
Rosenbrock (R5)	5	Zakharov(Z10)	10

Inicialmente, comparou-se a metaheurística EC-GRASP com as metaheurística "Antigo-CGRASP" e "Novo-CGRASP" para todos os problemas listados na tabela 1. Como esses problemas possuem mínimo global conhecido, as metaheurísticas foram executadas até que o valor atual da função objetivo f(x) se aproximasse significativamente do ótimo global f(x*). Assim como em Hedar e Fukushima (2002, 2003, 2004 e 2006), foi considerado que a solução atual x estaria significativamente próxima do ótimo global se ela obedecesse a seguinte inequação:

$$|f(x^*) - f(x)| \leq \varepsilon_1 |f(x^*)| + \varepsilon_2, \text{ onde}$$

$$\varepsilon_1 = 10^{-4} \; e \; \varepsilon_2 = 10^{-6}$$

Cada uma das variantes EC-GRASP, "Antigo-CGRASP" e "Novo-CGRASP", foi executado 100 vezes, utilizando uma semente do gerador de números aleatórios (seed) diferente para cada execução. Para o EC-GRASP, o parâmetro MaxIters utilizado foi 2n. Os parâmetros hs e he utilizados pelo EC-GRASP são apresentados na tabela 2.

Tabela 2 – Valor dos parâmetros do EC-GRASP.

Problema Teste	h_s	h_e	Problema Teste	h_s	h_e
BR	1.0	0.001	R_{10}	1.0	0.1
GP	1.0	1.0	$S_{4,5}$	1.0	0.5
SH	1.0	0.01	$S_{4,7}$	1.0	0.5
$H_{3,4}$	0.5	0.001	$S_{4,10}$	1.0	0.5
R_2	1.0	0.1	Z_5	1.0	0.5
R_5	1.0	0.1	Z_{10}	1.0	0.05

Tabela 3 – Comparação entre EC-GRASP e as metaheurísticas "Antigo-CGRASP" e "Novo_CGRASP".

Problema Teste	Antigo-CGRASP		Novo-CGRASP		EC-GRASP	
	Sucesso	Avalia. Fn.	Sucesso	Avalia. Fn.	Sucesso	Avalia. Fn.
BR	100	59.857	100	10.090	100	**8.735**
GP	100	29	100	53	100	84
SH	100	82.630	100	18.608	100	**12.990**
H3,4	100	20.743	100	1.719	100	9.441
R2	100	1.158.350	100	23.544	100	**5.038**
R5	100	6.205.503	100	182.520	100	**18.025**
R10	99	20.282.529	100	725.281	100	**78.375**
S4,5	100	5.545.982	100	9.274	100	**1.286**
S4,7	100	4.052.800	100	11.766	100	**1.739**
S4,10	100	4.701.358	100	17.612	100	**4.040**
Z5	100	959	100	12.467	100	**924**
Z10	100	3.607.653	100	2.297.937	100	**26.159**

A média dos resultados é apresentada na tabela 3 em relação ao percentual de vezes que o algoritmo encontrou uma solução significativamente próxima do ótimo global. Os dados para o "Novo-CGRASP" e para o "Antigo-CGRASP" foram retirados de Hirsch et al. (2007) e Hirsch et al. (2008), respectivamente.

Como destacado na tabela 3, na maioria dos problemas, houve uma diminuição significativa no número de avaliações da função objetivo, comprovando a eficiência do método. Apenas o problema teste R10 não apresentou sucesso 100, comprovando, também, a robustez do método.

6. CONCLUSÕES

Nesse artigo, foi apresentado o método EC-GRASP, uma versão híbrida da metaheurística C-GRASP com o método de busca direcionado APS para resolução de problemas de otimização global contínua. O EC-GRASP introduz um novo método de busca local baseado na vizinhança Bh(x) do C-GRASP e no método APS. Os experimentos computacionais realizados para um conjunto de problemas testes da literatura, mostram que o EC-GRASP diminuiu significativamente o número de avaliações da função objetivo em relação ao C-GRASP, sem nenhum impacto negativo no percentual de execuções em que o algoritmo converge.

7. REFERÊNCIAS BIBLIOGRÁFICAS

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. **Nonlinear Programming:** Theory and Algorithms. 2. ed. Estados Unidos: John Wiley & Sons, Inc., 1989.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, n. 6, p. 109–133, 1995.

HEDAR, A. R.; FUKUSHIMA, M. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. **Optimization Methods Software**, n. 17, p. 891–912, 2002.

HEDAR, A. R.; FUKUSHIMA, M. Minimizing multimodal functions by simplex coding genetic algorithms. **Optimization Methods Software**, n. 18, p. 265–282, 2003.

HEDAR, A. R.; FUKUSHIMA M. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, **Optimization Methods and Software**, n. 19, p. 291–308, 2004.

HEDAR, A. R.; FUKUSHIMA, M. Tabu search directed by direct search methods for. **European Journal Of Operational Research**, n. 170, p. 329-349, 2006.

HIRSCH, M. J.; MENESES, C. N.; PARDALOS, P. M.; RAGLE, M. A.; RESENDE M. G. C. A continuous GRASP to determine the relationship between drugs and adverse reactions. Data Mining, Systems Analysis e Optiimization in Medicine, **AIP Conference Proceedings**, n. 952, p. 106-121, 2008.

HIRSCH, M. J.; MENESES, C. N.; PARDALOS, P. M.; RESENDE, M. G. C. Global optimization by continuous grasp. **Optimization Letters**, n. 1, p. 201-212, 2007.

HIRSCH, M. J.; PARDALOS, P. M.; RESENDE, M. G. C. Speeding up continuous GRASP. **European Journal of Operational Research**, submitted. (site: http://www.research.att.com/~mgcr/papers.html, acessado em janeiro de 2008).

GEPROS. Gestão da Produção, Operações e Sistemas – Ano 3, nº 4, Out-Dez/2008, p. 155-167

ANEXO A – DEFINIÇÕES DAS FUNÇÕES

Esse anexo lista as definições de todas as funções utilizadas nos experimentos desse artigo.

(BR) Função Branin

Definição: $BR(x) = (x_2 - \frac{5}{4\pi^2} x_1^2 + \frac{1}{\pi} 5x_1 - 6)^2 + 10\,(1 - \frac{1}{8\pi})\cos(x) + 10$

Domínio: $[-5,15]^2$

Mínimo Global (um de vários): $x^* = (\pi, 2{,}275)$; $BR(x^*) = 0{,}397887$

(GP) Função Goldstein e Price

Definição: $GP(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$
$[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$

Domínio: $[-2,2]^2$

Mínimo Global: $x^* = (0,1)$; $GP(x^*) = 3$

($H_{n,m}$) Função Hartmann

Definição: $H_{n,m}(x) = -\sum_{i=1}^{m} a_i e^{-t}$

Domínio: $[0,1]^n$

Mínimo Global (n=3, m=4): $x^* = (0{,}114614, 0{,}555469, 0{,}852547)$;
$H_{3,4}(x^*) = -3{,}86278$

Parâmetros:

$$t = \sum_{j=1}^{n} A_{ij}^{(t)} (x_j - P_{ij}^{(n)})^2$$

$$A^{(3)} = \begin{bmatrix} 3{,}0 & 10{,}0 & 30{,}0 \\ 0{,}1 & 10{,}0 & 35{,}0 \\ 3{,}0 & 10{,}0 & 35{,}0 \\ 0{,}1 & 10{,}0 & 35{,}0 \end{bmatrix}$$

$$P^{(3)} = 10^{-4} \begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8838 \end{bmatrix}$$

(R_n) Função Rosenbrock

Definição: $R_n(x) = \sum_{j=1}^{n-1} [100(x_j - x_{j+1})^2 + (x_j - 1)^2]$

Domínio: $[-10,10]^2$

Mínimo Global: $x^* = (1,...,1)$; $R_n(x^*) = 0$

($S_{4,m}$) Função Shekel

Definição: $S_{4,m}(x) = -\sum_{i=1}^{m} [(x - a_i)^T (x - a_i) + c_i]^{-1}$

Domínio: $[0,10]^4$

Mínimo Global: $x^* = (4,4,4,4)$; $S_{4,5}(x^*) = -0,53628349$
$S_{4,7}(x^*) = -0,40281868$, $S_{4,10}(x^*) = -0,53628349$

Parâmetros:

$$a = \begin{bmatrix} 4,0 & 4,0 & 4,0 & 4,0 \\ 1,0 & 1,0 & 1,0 & 1,0 \\ 8,0 & 8,0 & 8,0 & 8,0 \\ 6,0 & 6,0 & 6,0 & 6,0 \\ 7,0 & 3,0 & 7,0 & 3,0 \\ 2,0 & 9,0 & 2,0 & 9,0 \\ 5,0 & 5,0 & 3,0 & 3,0 \\ 8,0 & 1,0 & 8,0 & 1,0 \\ 6,0 & 2,0 & 6,0 & 2,0 \\ 7,0 & 2,6 & 7,0 & 2,6 \end{bmatrix}$$

$c = [0,1, 0,2, 0,2, 0,4, 0,4, 0,6, 0,3, 0,7, 0,5, 0,5]$

(SH) Função Shubert

Definição: $SH(x) = \sum_{i=1}^{5} [i\cos[(i + 1)x_1 + i] \sum_{i=1}^{5} [i\cos[(i + 1)x_2 + i]$

Domínio: $[-10,10]^2$

Mínimo Global (um de vários): $x^* = (5,48242188, 4,84742188)$; $SH(x^*) = -186,7309$

(Zn) Função Zakharov

Definição: $Z_n(x) = \sum_{i=1}^{n} [x_i^2 + (\sum_{i=1}^{n} 0,5ix_i)^2 + (\sum_{i=1}^{n} 0,5ix_i)^4]$

Domínio: $[-5,10]^n$

Mínimo Global: $x^* = (0, 0, \mathbf{K}, 0)$; $Z_n(x^*) = 0$