Abordagem baseada em Continuous GRASP para clusterização de dados

Eduardo Vieira Queiroga
Centro de Informática - Universidade Federal da Paraíba - UFPB
Campus V, Av. dos Escoteiros - Mangabeira, 58055-000, João Pessoa - PB
eduardo.queiroga@ci.ufpb.br

Anand Subramanian
Centro de Tecnologia - Universidade Federal da Paraíba - UFPB
Campus I, Bloco G, Cidade Universitária, 58051-970, João Pessoa, PB
anand@ct.ufpb.br

Lucídio dos Anjos Formiga Cabral
Centro de Informática - Universidade Federal da Paraíba - UFPB
Campus V, Av. dos Escoteiros - Mangabeira, 58055-000, João Pessoa - PB
lucidio@ci.ufpb.br

RESUMO

Clusterização de dados compreende uma vasta gama de problemas em reconhecimento de padrões e aprendizado de máquina não-supervisionado, cujo objetivo é encontrar grupos de objetos similares em uma base de dados de um contexto específico. O problema de k-clusterização particional consiste em encontrar k conjuntos disjuntos e não vazios. Em métodos baseados em centros de *cluster*, minimizar a distância intracluster é um dos critérios mais utilizados. A meta-heurística *Continuous GRASP* (C-GRASP) é uma adaptação do método clássico GRASP que tem obtido sucesso em problemas de otimização global contínua. Este trabalho propõe uma abordagem baseada em C-GRASP para k-clusterização particional que busca o conjunto ótimo de centros de *cluster*. Experimentos computacionais foram realizados em bases de dados de propósito geral. O algoritmo proposto mostrou-se capaz de gerar soluções de alta qualidade quando comparado com a versão original do C-GRASP, o new C-GRASP e outras meta-heurísticas da literatura.

PALAVRAS CHAVE. Continuous GRASP, Clusterização de dados, Otimização contínua.

Tópicos: MH - Metaheuristicas, OA - Outras aplicações em PO

ABSTRACT

Data clustering includes a wide range of problems in pattern recognition and unsupervised machine learning, whose objective is to find similar groups of objects in a database of a specific context. The partitional k-clustering problem consists of finding k disjoint non-empty sets. In center-based clustering methods, minimizing the intra-cluster distances is one of the most commonly used criteria. The *Continuous Greedy Randomized Adaptive Search Procedure* (C-GRASP) metaheuristic is an adaptation of GRASP that has been successful in continuous global optimization problems. This paper proposes a C-GRASP based approach for partitional k-clustering that searches the optimal cluster centers. Computational experients were carried out in general purpose databases.The proposed algorithm produced high quality solutions when compared to the original version of C-GRASP, new C-GRASP and other metaheuristics from the literature.

KEYWORDS. Continuous GRASP. Data clustering. Continuous optimization.

Paper topics: MH - Metaheuristics, OA - Other applications in OR

1. Introdução

Muitos problemas do mundo real que envolvem grandes conjuntos de dados possuem uma demanda eminente por reconhecimento de padrões, utilizando as informações extraídas para apoiar processos de tomada de decisão. Clusterização de dados ou análise de *cluster* é uma tarefa de aprendizagem não supervisionada, onde busca-se encontrar grupos de objetos mais similares dada uma métrica preestabelecida [Jain et al., 1999]. Esse conjunto de técnicas possui aplicações em diversas áreas, tais como: recuperação da informação [Rasmussen, 1992], [Aggarwal e Zhai, 2012], análise de imagens [Celenk, 1990],[Das e Sil, 2010] e bioinformática [Li et al., 2001],[Hruschka et al., 2006].

Clusterização hierárquica e particional são os tipos mais conhecidos e estudados. No primeiro tipo, uma hierarquia de *clusters* é construída sobre os dados e em cada nível existe uma clusterização distinta. No segundo tipo, busca-se uma separação dos dados em *clusters* dijuntos (sem sobreposição) e não-vazios, onde cada elemento precisa pertencer a um único *cluster*. Muitos problemas de clusterização são NP-Difíceis [Gonzalez, 1982], sendo adequado o uso de métodos heurísticos para a resolução sem garantia de otimalidade. O algoritmo *k*-means, por exemplo, é um dos mais utilizados em clusterização particional. Contudo, a fase de inicialização do método pode facilmente induzir soluções em mínimos locais [Bubeck et al., 2012].

Nas últimas décadas, uma grande quantidade de meta-heurísticas tem sido propostas para tratar do problema de clusterização particional, principalmente através de algoritmos bio-inspirados. Por exemplo, foram estudadas abordagens baseadas em *Particle Swarm Optimization* (PSO) [Chuang et al., 2011],[Ahmadi et al., 2012],[Yeh e Lai, 2015], *Ant Colony Optimization* (ACO) [Shelokar et al., 2004],[Zhang e Cao, 2011], *Ant Bee Colony Algorithm* [Zhang et al., 2010],[Karaboga e Ozturk, 2011], *Tabu Search* [Al-Sultan, 1995],[Liu et al., 2008], *Simulated Annealing* (SA) [Selim e Alsultan, 1991] e Algoritmos Genéticos (GA, do inglês *Genetic Algorithms*) [Murthy e Chowdhury, 1996],[Maulik e Bandyopadhyay, 2000],[Liu et al., 2012]. Alguns trabalhos propuseram uma hibridização com o algoritmo *k*-means [Niknam e Amiri, 2010],[Krishnasamy et al., 2014],[Boobord et al., 2015]. Meta-heurísticas menos conhecidas também foram propostas em [Hatamlou, 2013],[Wang et al., 2016].

Em clusterização baseada em centros, métodos de otimização global contínua são mais adequados e geralmente possuem maior capacidade de convergência do que adaptações simples de métodos de otimização discreta. Neste trabalho, é proposto a resolução de clusterização particional minimizando a distância intracluster por uma abordagem beaseada em Continuous GRASP (C-GRASP). A capacidade de resolução do algoritmo proposto é investigada através de experimentos computacionais que envolvem o C-GRASP original [Hirsch et al., 2007], um melhoramento chamado new C-GRASP [Hirsch et al., 2010] e outras meta-heurísticas da literatura. O conjunto de testes é composto por bases de dados de um repositório de aprendizado de máquina conhecido.

O artigo está organizado como segue. Na seção 2, o problema de clusterização é definido formalmente. Na seção 3, o método C-GRASP é apresentado. Na seção 4, o C-GRASP proposto para clusterização é descrito em detalhes. Na seção 5, os experimentos computacionais são apresentados e os resultados obtidos são discutidos. Na seção 6, algumas considerações finais e propostas de trabalhos futuros são feitas.

2. Problema de clusterização

O problema de *k*-clusterização particional com número de *clusters* predeterminado, pode ser definido formalmente como [Hansen e Jaumard, 1997]:

- Para um particionamento $P = \{C_1, C_2, ..., C_k\}$ de D em k *clusters*, temos:

  1. $C_i \neq \emptyset$ para $i = 1, ..., k$
  2. $C_i \cap C_j = \emptyset$, para $i, j = 1, ..., k$ e $i \neq j$;

3. $\bigcup_{i=1}^{k} C_i = D$;

Em alguns métodos de clusterização, um conjunto de k pontos virtuais ou centros de *cluster* $c = \{c_1, c_2, ..., c_k\}$ de mesma dimensionalidade dos objetos são manipulados. Dado um posicionamento de c, cada objeto em D é atribuído ao centro mais próximo, produzindo uma clusterização resultante. Diferentes métricas de avaliação que consideram dissimilaridade podem ser utilizadas para guiar a busca pelos melhores centros. Um dos critérios mais utilizados minimiza o somatório das distâncias, por norma euclidiana, entre cada ponto (objeto da base de dados) d-dimensional p_i e o seu respectivo centro c_j, para $i = 1, ..., m$, $j = 1, ..., k$, como descrito na expressão (2). Se c_j for o centro mais próximo do objeto i, $a_{ij} = 1$, caso contrário, $a_{ij} = 0$.

$$d(p_1, p_2) = \|p_1 - p_2\| = \sqrt{(\sum_{j=1}^{d}(p_{1j} - p_{2j})^2)} \tag{1}$$

$$\min_{c_1, c_2, ..., c_k} \sum_{i=1}^{m}\sum_{j=1}^{k} a_{ij}\|c_j - p_i\| \tag{2}$$

3. Continuous GRASP

O Continuous GRASP (C-GRASP), proposto por [Hirsch et al., 2007], é uma adaptação da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo e Resende, 1995] para resolução de problemas de otimização global contínua. O algoritmo consiste de uma estratégia *multi-start* sobre uma fase de contrução e busca local. A construção utiliza um critério guloso e aleatoriedade para prover boas soluções de partida, mantendo um nível de diversificação apropriado. Em seguida, a fase de busca local é realizada sobre as soluções geradas anteriormente. No C-GRASP, a tarefa de otimização pode ser formalmente definida como encontrar o ponto ótimo x^* para uma funcao $f : \mathbb{R}^n \mapsto \mathbb{R}$ em uma região $S = \{x \in \mathbb{R}^n : l \leq x \leq u\}$, onde $l, u \in \mathbb{R}^n$ tal que $u \geq l$. O algoritmo explora a discretização da região em uma grade com densidade crescente sem realizar cálculos de derivada, que custam caro computacionalmente para algumas funções.

No Algoritmo 1, é apresentado o pseudocódigo do C-GRASP em sua versão original. O algoritmo possui os seguintes parâmetros de entrada: n é a dimensão do problema; l e u são, respectivamente, os vetores *lower bound* e *upper bound* que determinam as restrições do espaço de busca; $f(.)$ é a função objetivo; MaxIters é o número de iterações sobre a fase de construção e busca local para uma nova solução de partida; MaxNumIterNoImprov é o número máximo de iterações sem melhora tolerado sem redução do tamanho do passo h; NumTimesToRun determina o número de execuções do procedimento para uma nova solução de partida; MaxDirToTry o número de direções que a busca local deve analisar; e o α determina o equilíbrio entre aleatoriedade e escolha gulosa na fase de construção. Para cada iteração do laço na linha 3, o método é realizado para um ponto inicializado aleatoriamente com distribuição uniforme (linha 4) e tamanho do passo h inicialmente igual a 1, sendo este último parâmetro responsável pela densidade da grade a ser explorada. No laço da linha 5, a fase de construção *ConstrucãoGulosaAleatória* é executada sobre x seguido pelo procedimento *BuscaLocal* que recebe o x resultante da fase anterior. Em seguida a melhor solução encontrada x^* é atualizada quando houver melhora (linhas 8 e 9). O tamanho do passo h é reduzido pela metade (aumentando a densidade da grade), sempre que houver MaxNumIterNoImprov iterações sem melhora (linhas 12 e 13).

A etapa de construção, descrita no Algoritmo 2, é inicializada com todas as coordenadas de x não fixadas (que podem ser alteradas) e armazenadas no conjunto C (linha 2). Em seguida, uma busca linear (linha 7) é realizada para cada coordenada não fixada. A Lista Restrita de Candidatos (RCL, do inglês *Restricted Candidate List*) consiste em uma lista de soluções candidatas que

respeitam um limiar definido pelo parâmetro α e as soluções min e max obtidas da etapa anterior. O procedimento se repete até que todas as coordenadas sejam fixadas ($C = \emptyset$).

Algoritmo 1: C-GRASP Original [Hirsch et al., 2007]

```
1  procedimento C-GRASP (n, l, u, f(.), MaxIters, MaxNumIterNoImprov, NumTimesToRun, MaxDirToTry, α)
2      x* ← nil; f* ← ∞;
3      para cada j = 1, ..., NumTimesToRun fazer
4          x ← UnifRand(l, u); h ← 1; NumIterNoImprov ← 0;
5          para cada Iter = 1, ..., MaxIters fazer
6              x ← ConstruçãoGulosaAleatória(x, f(.), n, h, l, u, α);
7              x ← BuscaLocal(x, f(.), n, h, l, u, MaxDirToTry);
8              se f(x) < f* então
9                  x* ← x; f* ← f(x); MaxNumIterNoImprov ← 0;
10             senão
11                 NumIterNoImprov ← NumIterNoImprov + 1;
12             se NumIterNoImprov ≥ MaxNumIterNoImprov então
13                 h ← h/2; NumIterNoImprov ← 0;
14     retorne x*;
```

Algoritmo 2: ConstruçãoGulosaAleatória

```
1  procedimento ConstruçãoGulosaAleatória (x, f(.), n, h, l, u, α)
2      C ← {1, 2, ..., n};
3      enquanto C ≠ ∅ faça
4          min ← +∞; max ← −∞;
5          para cada i = 1, ..., n fazer
6              se i ∈ C então
7                  z_i ← BuscaLinear(x, h, i, n, f(Δ), l, u);
8                  g_i ← f(z_i);
9                  se min > g_i então min ← g_i ;
10                 se max < g_i então max ← g_i ;
11         RCL ← ∅;
12         para cada i = 1, ..., n fazer
13             se i ∈ C and g_i ≤ (1 − α) * min + α * max então
14                 RCL ← RCL ∪{i};
15         j ← RandomlySelectElement(RCL);
16         x ← z_j; C ← C \ {j};
17     retorne x*;
```

Na fase de busca local, descrita no algoritmo 3, busca-se melhorar a solução por explorar o conjunto de direções $\Gamma = \{d \in \mathbb{R}^n : d_i = -1 \vee d_i = 0 \vee d_i = 1,\ i = 1, ..., n\} \setminus \{d \in \mathbb{R}^n : d_i = 0, i = 1, ..., n\}$ que consiste em todas as possibilidades de vetores com $\{-1, 0, 1\}$ em cada coordenada, exceto o vetor $\{0\}^n$. A cardinalidade do conjunto de direções é dada por $3^n - 1$, podendo-se realizar um mapeamento bijetivo $T : \{1, ..., 3^n - 1\} \mapsto \Gamma$.

4. Abordagem baseada em C-GRASP para clusterização de dados

Para k-clusterização particional, a dimensão do problema no C-GRASP é dada por $n = k \times d$, tal que k é o número de *clusters* e d é o número de dimensões (atributos) de cada ponto (objeto) do conjunto de dados. Um ponto x na região representa uma solução que armazena as coordenadas dos centros de *cluster*, como ilustrado na Figura 1 para um problema bidimensional de 3-clusterização, onde c_i^j consiste na j-ésima coordenada do i-ésimo centro de *cluster*.

O C-GRASP original [Hirsch et al., 2007] se mostrou apto à resolução de problemas de otimização de funções matemáticas com poucas dimensões para os parâmetros utilizados nos experimentos. A densidade da grade h possui grande influência na capacidade de resolução do método, sendo necessário uma definição proporcional à dimensionalidade do espaço de busca, que

Algoritmo 3: BuscaLocal

```
1  procedimento BuscaLocal (n, l, u, f(.), MaxDirToTry)
2      Improved ← true; D ← ∅;
3      x* ← x; f* ← f(x);
4      NumDirToTry ← min{3^n − 1, MaxDirToTry};
5      enquanto Improved faça
6          Improved ← false;
7          enquanto |D| ≤ NumDirToTry ∧ ¬Improved faça
8              Generate r ← ⌈UnifRand(1, 3^n − 1)⌉ ∉ D;
9              D ← D ∪ {r};
10             d ← T(r); x ← x* + h * d;
11             se l ≤ x ≤ u então
12                 se f(x) ≤ f* então
13                     x* ← x; f* ← f(x);
14                     D ← ∅;
15                     Improved ← true;
16     retorne x*;
```

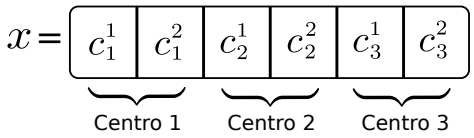


Figura 1. Representação da solução para três *clusters* no espaco euclidiano bidimensional.

possui grande variabilidade em tarefas de clusterização. Para tratar desse problema, é proposto o cálculo do h baseado na média do conjunto de dados, na tentativa de prover adaptabilidade a conjuntos diversos. Além disso, uma modificação estrutural foi aplicada para uma exploração mais intensa sobre a fase de construção, utilizando a fase de busca local para refinamentos de menor escala.

No algoritmo 4, é apresentado o pseudocódigo do método proposto, chamado *C-GRASP-Clustering*. Os parâmetros de entrada $f(.)$, MaxDirToTry e α são os mesmos do C-GRASP original. Os outros parâmetros são: o conjunto de dados D, a dimensionalidade dos dados d, o número de *clusters* k, o tempo máximo de execução $TimeLimit$, e os parâmetros Δ_1 e Δ_2 que configuram um mecanismo de filtro de iterações. Nas linhas 3-5, são obtidos os valores máximos e mínimos de cada coordenada dos elementos de D, para utilizar na definição dos limites que cada centro de *cluster* deve obedecer. Para manter a mesma dimensionalidade de x, os vetores l e u recebem um cópia dos limites para cada centro de *cluster*, como pode ser visualizado nas linhas 6-11. Os valores h_s e h_e, baseados nos melhoramentos propostos em [Hirsch et al., 2010], definem o intervalo em que o parâmetro h pode operar. Eles são calculados como o maior e menor valor médio $\bar{e}_i$ para a i-ésima coordenada de todos os elementos em D (linhas 12), com o intuito de atenuar as diferenças de escala entre os atributos. O procedimento é executado até que o tempo limite seja atingido (linha 14). Para cada iteração uma solução aleatória x é amostrada, e inicia-se uma etapa iterativa sobre a fase de construção. Enquanto h, inicialmente igual a h_s, não for menor do que h_e (linha 17), o procedimento *ConstruçãoGulosaAleatória* é executado (linha 18). A cada iteração dessa etapa, h é reduzido pela metade. Após a primeira melhora interna (linhas 21-22), um mecanismo de filtro é ativado para desistência de iterações consideradas não promissoras (linhas 23-24) através dos parâmetros Δ_1 e Δ_2. Quando o *gap* entre a solução obtida anteriormente f_{prev} e a solução corrente $f(x)$, dado por $gap(f_{prev}, f(x)) = \dfrac{f_{prev} - f(x)}{f_{prev}} \times 100$, for menor do que Δ_1, verifica-se o *gap* para a melhor solução conhecida, dado por $gap(f(x), f^*) = \dfrac{f(x) - f^*}{f(x)} \times 100$. Se esse percentual

for superior a Δ_2, a iteração é abortada e um novo ponto de partida construído. Em seguida, a busca local é realizada para refinamentos de pequena escala, utilizando um tamanho de passo fixo (linhas 29-33).

Algoritmo 4: Abordagem baseada em C-GRASP para clusterização de dados

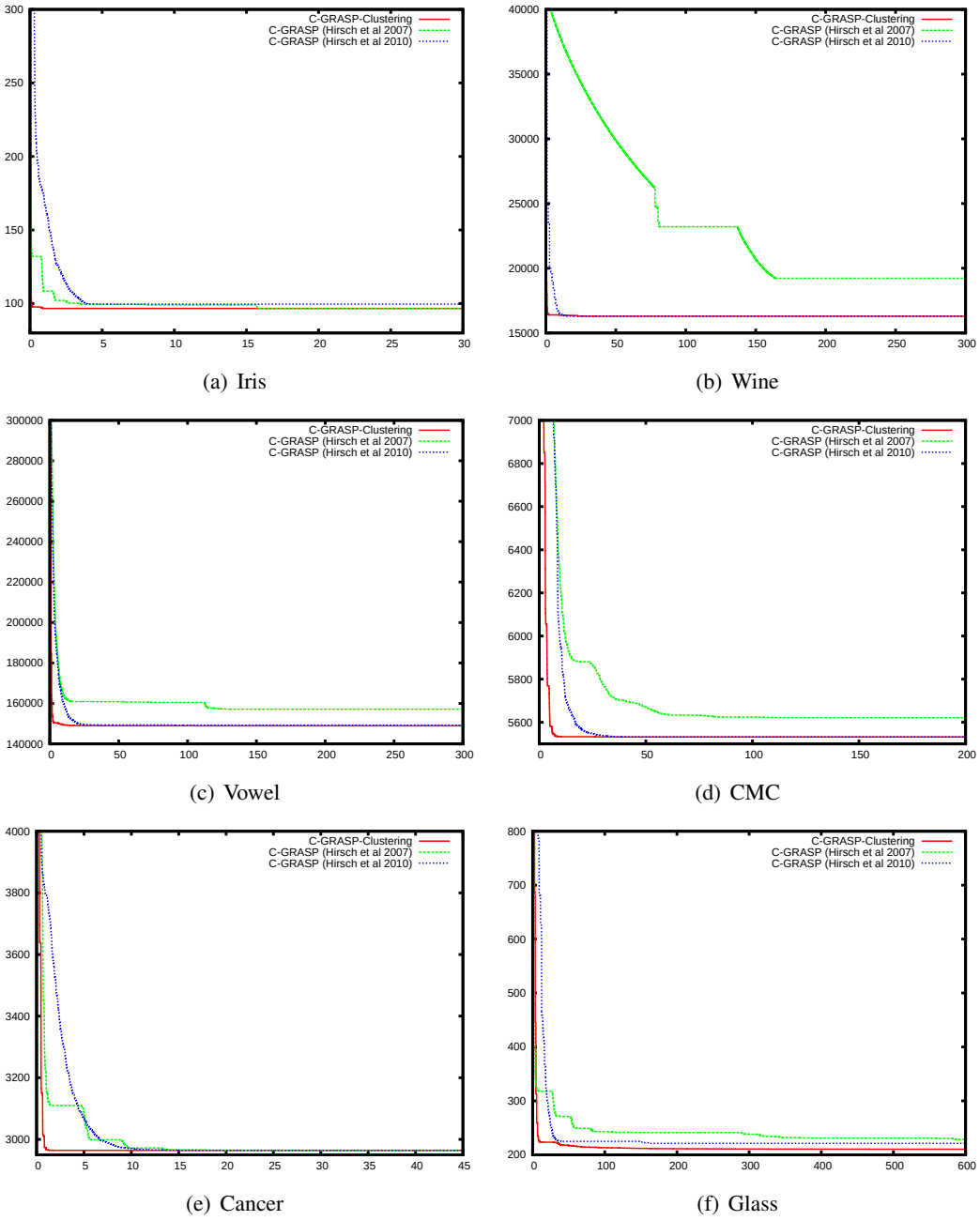
```
1  procedimento C-GRASP-Clustering(D, d, k, f(.), TimeLimit, MaxDirToTry, α, Δ1, Δ2)
2      n ← k × d; // dimensão do problema
3      para cada i = 1, ..., d fazer
4          min_i ← min({e_i : e ∈ D}); // armazena valores extremos de
5          max_i ← max({e_i : e ∈ D}); // D para cada coordenada
6      i ← 0;
7      enquanto i < n faça
8          para cada j = 1, ..., d fazer
9              l_{i+j} ← min_j; // armazena vetores l e u
10             u_{i+j} ← max_j; // que definem as restrições em caixa
11         i ← i + d; // pula para o id do próximo centro de cluster
12     h_s ← max({ē_i, i = 1, ..., d}); h_e ← min({ē_i, i = 1, ..., d})/10^4;
13     x* ← nil; f* ← ∞;
14     enquanto TimeLimit não atingido faça
15         x ← UnifRand(l, u); h ← h_s;
16         f_prev ← ∞; imp_g ← false; imp_l ← false;
17         enquanto h ≥ h_e faça
18             x ← ConstruçãoGulosaAleatória(x, f(.), n, h, l, u, α);
19             se f(x) < f* então
20                 x* ← x; f* ← f(x); imp_g ← true
21             se ¬imp_l ∧ f_prev ≠ ∞ ∧ f_prev − f(x) ≥ 0 então
22                 imp_l ← true
23             se imp_l ∧ gap(f_prev, f(x)) < Δ1 ∧ gap(f(x), f*) > Δ2 então
24                 Go to linha 14
25             f_prev ← f(x);
26             h ← h/2;
27         se ¬imp_g então
28             Go to linha 14
29         h ← 1; // densidade da grade fixada para a busca local
30         enquanto h ≥ h_e/10 faça
31             x ← BuscaLocal(x*, f(.), n, h, l, u, MaxDirToTry);
32             se f(x) < f* então x* ← x; f* ← f(x) ;
33             h ← h/2;
34     retorne x*;
```

5. Experimentos Computacionais

Nesta seção, o procedimento de calibração dos parâmetros do método proposto, bem como os resultados da comparação do mesmo com outras abordagens são apresentados. Os algoritmos C-GRASP-Clustering, C-GRASP original [Hirsch et al., 2007] e new C-GRASP [Hirsch et al., 2010] foram implementados e executadas na plataforma Matlab 7.14 (R2012a) em um computador com processador *Intel Core i5-3210M* (64 bits) de 2 núcleos com 2.50GHz e 8GB de memória RAM no sistema operacional Linux Mint 17 (64 bits).

Um conjunto de seis bases de dados de propósito geral do repositório da *University Californe Irvine* (UCI) [Merz e Blake] foi utilizado nos testes. A tabela 3 apresenta os principais atribu-

tos desse conjunto. O tempo de execução dos métodos implementados foi de 30 segundos para base *Iris*, 200 segundos para as bases *Cancer* e *CMC*, 300 segundos para as bases *Wine*, *Vowel* e 600 segundos para a base *Glass*. Esses valores foram baseados no nível de dificuldade das instâncias, que possui certa proporcionalidade com a quantidade de itens, a dimensão dos dados e o número de *clusters*.



(a) Iris (b) Wine

(c) Vowel (d) CMC

(e) Cancer (f) Glass

Figura 2. Gráficos de convergência do C-GRASP-Clustering, C-GRASP e new C-GRASP para o conjunto de bases de dados considerado.

5.1. Calibração de parâmetros

Para encontrar bons valores para os parâmetros do C-GRASP-Clustering, um projeto experimental fatorial [Winer et al., 1971] foi elaborado e executado. Neste experimento, 2^k configurações são avaliadas para k parâmetros em níveis mínimo e máximo. Para os parâmetros α, MaxDirToTry

(γ), Δ_1, Δ_2, existem $2^4 = 16$ possibilidades distintas. Cada configuração foi executada 10 vezes nas bases de dados *glass* e *vowel*, que são consideradas as mais difíceis do conjunto. A média da função objetivo para cada configuração é apresentada nas tabelas 1 e 2. Para a base *vowel*, nenhuma configuração conseguiu uma vantagem significativa. Para a base *glass*, quando $\alpha = 0.6$, $\gamma = 30$, $\Delta_1 = 0.5\%$, $\Delta_2 = 1\%$, tem-se resultados de melhor qualidade. Essa configuração representa a versão do C-GRASP-Clustering considerada nos testes realizados na seção 5.2 .

Glass		$\Delta_1 = 0.05\%$		$\Delta_1 = 0.1\%$	
		$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$	$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$
$\alpha = 0.4$	$\gamma = 30$	210.2460	210.2776	210.4307	210.2994
	$\gamma = 50$	210.3081	210.6239	210.3835	210.3622
$\alpha = 0.6$	$\gamma = 30$	210.2770	**210.1465**	210.2070	210.2081
	$\gamma = 50$	210.3022	210.3432	210.2841	210.3559

Tabela 1. Resultados da calibração de parâmetros para a base *glass*.

Vowel		$\Delta_1 = 0.05\%$		$\Delta_1 = 0.1\%$	
		$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$	$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$
$\alpha = 0.4$	$\gamma = 30$	148967.2412	148967.2412	148967.2413	148967.2412
	$\gamma = 50$	148967.2411	148967.2411	148967.2411	148967.2411
$\alpha = 0.6$	$\gamma = 30$	148967.2412	148967.2411	148967.2413	148967.2411
	$\gamma = 50$	148967.2411	148967.2411	148967.2411	148967.2411

Tabela 2. Resultados da calibração de parâmetros para a base *vowel*.

5.2. Comparação entre abodagens C-GRASP e outras meta-heurísticas

A performance do C-GRASP-Clustering foi comparada com o C-GRASP original [Hirsch et al., 2007], o new C-GRASP [Hirsch et al., 2010], o algoritmo *k*-means e duas meta-heurísticas eficientes propostas recentemente: o *Black Hole* (BH) [Hatamlou, 2013] e o *Flower Pollination Algorithm with Bee Pollinator* (BPFPA) [Wang et al., 2016]. O BH e BPFPA obtiveram resultados de melhor qualidade e robustez do que outras meta-heurísticas conhecidas, tais como *Particle Swarm Optimization*, *Differential Evolution*, *Ant Bee Colony* e *Gravitational Search Algorithm*. Para viabilizar a comparação com o new C-GRASP, os parâmetros h_s, h_e foram calculados como proposto no C-GRASP-Clustering, e o parâmetro p_{lo} recebeu o valor 0.01, pois o método original foi executado em problemas de dimensionalidades muito distintas. Os demais parâmetros do new C-GRASP, bem como os do C-GRASP original, são os mesmos utilizados em [Hirsch et al., 2007] e [Hirsch et al., 2010].

Para cada instância, 50 execuções independentes foram realizadas, com exceção do BPFPA, onde foram realizadas 20 execuções para apenas 4 das 6 bases de dados. Na tabela 4, são reportados o cálculo da função objetivo (equação 2) mínimo (melhor), médio e máximo (pior), além do desvio padrão. Com exceção da melhor solução na base *Vowel*, encontrada pelo C-GRASP original, o método proposto obteve os melhores resultados em todas as bases para as métricas consideradas, incluindo empates com o new C-GRASP e BPFPA. Nas bases *Wine*,*Vowel* e *Glass*, o C-GRASP-Clustering obteve solução média superior aos demais algoritmos. O baixo desvio padrão do C-GRASP-Clustering caracteriza a robustez do método em comparação com as outras abordagens. Esses resultados indicam o potencial de técnicas baseadas em C-GRASP para resolução de problemas de clusterização.

Na Figura 2, é ilustrado o processo de convergência dos métodos C-GRASP-Clustering, C-GRASP e new C-GRASP. Essa perspectiva demonstra vantagem do método proposto sobre os outros dois algoritmos, pois a curva do mesmo se mantém abaixo das demais durante todo a execução para todos os testes.

Tabela 3. Informações sobre as bases de dados utilizadas para testes.

Base de dados	Número de atributos	Número de *clusters*	Número de objetos
Iris	4	3	150
Wine	13	3	178
Vowel	3	6	871
CMC	9	3	1473
Cancer	9	2	683
Glass	9	6	214

Tabela 4. Distância intracluster obtida pela abordagem proposta e outras meta-heurísticas da literatura.

Base de dados	Critério	*k*-means	BH	BPFPA	C-GRASP	new C-GRASP	C-GRASP-Clustering
Iris	Melhor	97.3259	96.6558	**96.6555**	**96.6555**	**96.6555**	**96.6555**
	Média	105.7290	96.6558	**96.6555**	96.6556	**96.6555**	**96.6555**
	Pior	128.4042	96.6631	**96.6555**	96.6556	**96.6555**	**96.6555**
	Desvio padrão	12.3875	0.0017	0	3.2040e-05	0	0
Wine	Melhor	16555.6794	16293.4199	16292.1847	16292.1848	**16292.1846**	**16292.1846**
	Média	16963.0449	16294.3176	16292.9230	17084.2035	16292.3198	**16292.1847**
	Pior	23755.0494	16300.2261	16294.1727	18598.7428	16292.6672	**16292.1847**
	Desvio padrão	1180.6942	1.6512	0.7663	863.7451	0.2189	6.5632e-10
Vowel	Melhor	149394.8039	148985.6137	—	**148967.2408**	148967.2421	148967.2410
	Média	153660.8071	149848.1814	—	150763.3570	148980.4264	**148967.2412**
	Pior	168474.2659	153058.9866	—	168340.2868	149062.0805	**148967.2415**
	Desvio padrão	4123.0420	1306.9537	—	4076.1838	30.8673	0.0001
CMC	Melhor	5542.1821	5532.8832	5693.7239	5532.1909	**5532.1847**	**5532.1847**
	Média	5543.4234	5533.6312	5693.7246	5621.3465	**5532.1847**	**5532.1847**
	Pior	5545.3333	5534.7773	5693.7283	7018.0475	**5532.1847**	**5532.1847**
	Desvio padrão	1.5238	0.59940	0.0016	356.4527	0	0
Cancer	Melhor	2986.9613	2964.3888	**2964.3869**	2964.3871	**2964.3869**	**2964.3869**
	Média	3032.2478	2964.3954	**2964.3869**	2964.3872	**2964.3869**	**2964.3869**
	Pior	5216.0894	2964.4501	**2964.3869**	2964.3874	**2964.3869**	**2964.3869**
	Desvio padrão	315.1456	0.0092	0	7.1618e-05	0	0
Glass	Melhor	215.6775	210.5155	—	210.4291	**210.0010**	**210.0010**
	Média	227.9778	211.4986	—	228.0480	211.1786	**210.3177**
	Pior	260.8384	213.9569	—	255.4452	214.8058	**210.4455**
	Desvio padrão	14.1388	1.1823	—	11.8937	1.3954	0.1676

6. Considerações finais

O Continuous GRASP é uma meta-heurística recente que ainda não foi explorada em uma diversidade de problemas conhecidos, especificamente em tarefas de reconhecimento de padrões por clusterização de dados. Esse trabalho propôs uma abordagem C-GRASP para *k*-clusterização particional de bases de dados de propósito geral. Os experimentos realizados deram evidências de que o algoritmo proposto possui melhor performance do que o C-GRASP original, o *new* C-GRASP, e outras meta-heurísticas da literatura. Como trabalhos futuros, novas estratégias para geração de direções na busca local, mecanismos de filtros e hibridizações com outras meta-heurísticas podem ser exploradas. A performance do método também pode ser avaliada em dados artificiais ou reais de maiores dimensões, utilizando testes de significância estatística para fortalecer os resultados.

Referências

Aggarwal, C. C. e Zhai, C. (2012). A survey of text clustering algorithms. In *Mining Text Data*, p. 77–128. Springer.

Ahmadi, A., Karray, F., e Kamel, M. S. (2012). Model order selection for multiple cooperative swarms clustering using stability analysis. *Information Sciences*, 182(1):169–183.

Al-Sultan, K. S. (1995). A tabu search approach to the clustering problem. *Pattern Recognition*, 28 (9):1443–1451.

Boobord, F., Othman, Z., e Abu Bakar, A. (2015). A wk-means approach for clustering. *International Arab Journal of Information Technology (IAJIT)*, 12(5).

Bubeck, S., Meilă, M., e von Luxburg, U. (2012). How the initialization affects the stability of the k-means algorithm. *ESAIM: Probability and Statistics*, 16:436–452.

Celenk, M. (1990). A color clustering technique for image segmentation. *Computer Vision, Graphics, and Image Processing*, 52(2):145–170.

Chuang, L.-Y., Hsiao, C.-J., e Yang, C.-H. (2011). Chaotic particle swarm optimization for data clustering. *Expert systems with Applications*, 38(12):14555–14563.

Das, S. e Sil, S. (2010). Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm. *Information Sciences*, 180(8):1237–1256.

Feo, T. e Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133. ISSN 0925-5001. URL `http://dx.doi.org/10.1007/BF01096763`.

Gonzalez, T. F. (1982). On the computational complexity of clustering and related problems. In *System modeling and optimization*, p. 174–182. Springer.

Hansen, P. e Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical programming*, 79(1-3):191–215.

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information sciences*, 222:175–184.

Hirsch, M. J., Meneses, C., Pardalos, P. M., e Resende, M. G. (2007). Global optimization by continuous grasp. *Optimization Letters*, 1(2):201–212.

Hirsch, M. J., Pardalos, P. M., e Resende, M. G. (2010). Speeding up continuous grasp. *European Journal of Operational Research*, 205(3):507–521.

Hruschka, E. R., Campello, R. J., e De Castro, L. N. (2006). Evolving clusters in gene-expression data. *Information Sciences*, 176(13):1898–1927.

Jain, A. K., Murty, M. N., e Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Karaboga, D. e Ozturk, C. (2011). A novel clustering approach: Artificial bee colony (abc) algorithm. *Applied soft computing*, 11(1):652–657.

Krishnasamy, G., Kulkarni, A. J., e Paramesran, R. (2014). A hybrid approach for data clustering based on modified cohort intelligence and k-means. *Expert Systems with Applications*, 41(13): 6009–6016.

Li, W., Jaroszewski, L., e Godzik, A. (2001). Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283.

Liu, R., Jiao, L., Zhang, X., e Li, Y. (2012). Gene transposon based clone selection algorithm for automatic clustering. *Information Sciences*, 204:1–22.

Liu, Y., Yi, Z., Wu, H., Ye, M., e Chen, K. (2008). A tabu search approach for the minimum sum-of-squares clustering problem. *Information Sciences*, 178(12):2680–2704.

Maulik, U. e Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465.

Merz, C. e Blake, C. Uci repository of machine learning databases. URL <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Murthy, C. A. e Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832.

Niknam, T. e Amiri, B. (2010). An efficient hybrid approach based on pso, aco and k-means for cluster analysis. *Applied Soft Computing*, 10(1):183–197.

Rasmussen, E. (1992). Information retrieval. chapter Clustering Algorithms, p. 419–442. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-463837-9. URL http://dl.acm.org/citation.cfm?id=129687.129703.

Selim, S. Z. e Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern recognition*, 24(10):1003–1008.

Shelokar, P., Jayaraman, V. K., e Kulkarni, B. D. (2004). An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195.

Wang, R., Zhou, Y., Qiao, S., e Huang, K. (2016). Flower pollination algorithm with bee pollinator for cluster analysis. *Information Processing Letters*, 116(1):1–14.

Winer, B. J., Brown, D. R., e Michels, K. M. (1971). *Statistical principles in experimental design*, volume 2. McGraw-Hill New York.

Yeh, W.-C. e Lai, C.-M. (2015). Accelerated simplified swarm optimization with exploitation search scheme for data clustering. *PloS one*, 10(9):e0137246.

Zhang, C., Ouyang, D., e Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications*, 37(7):4761–4767.

Zhang, L. e Cao, Q. (2011). A novel ant-based clustering algorithm using the kernel method. *Information Sciences*, 181(20):4658–4672.