Algoritmo Genético Construtivo na otimização de problemas combinatoriais de agrupamentos

João Carlos Furtado
LACESM/CT - UFSM - Universidade Federal de Santa Maria
Campus Universitário - Santa Maria - RS

Luiz Antonio Nogueira Lorena
LAC/INPE- Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas 1758 - São José dos Campos - SP

Resumo

O Algoritmo Genético Construtivo (AGC) introduz novas características aos Algoritmos Genéticos (AG) tradicionais. É bem aceito que "blocos construtivos" (esquemas) formam a base para um bom comportamento dos AGs. Os esquemas entretanto são usualmente avaliados indiretamente, avaliando uma estrutura que o contenha. O AGC considera uma população de tamanho variável que é composta por esquemas e estruturas, que são avaliados através de um processo de adaptação proporcional. Eles são incluídos na população se seus ranks ultrapassam um limiar dado por um parâmetro de evolução, que é atualizado a cada geração, e os esquemas e estruturas que já se encontram na população podem sair pelo mesmo teste. A recombinação procura preservar os bons esquemas, e uma mutação de busca local é aplicada a estruturas para obter a diversificação da população. Para os problemas de agrupamentos os esquemas e estruturas são representados por cadeias binárias, e heurísticas de atribuição complementam a representação de uma maneira única para três problemas de agrupamentos em grafos. Os problemas de agrupamento estudados são, o problema clássico das p-medianas (PPM), o problema capacitado das p-medianas (PPMC) e o problema de agrupamento de corte mínimo (PCM). Resultados muito bons foram obtidos quando o AGC foi aplicado a instancias disponíveis na literatura.

Palavras chave: *Problemas de agrupamentos, problema das p-medianas, problema capacitado das p-medianas, e problema de agrupamento de corte mínimo.*

1. Introdução

Problemas de agrupamento (*clustering*) ocorrem geralmente na classificação de dados para algum propósito, por exemplo, para a simples recuperação ou para efetuar uma análise nos dados. Qualquer algoritmo de agrupamento procura grupos naturais ou inerentes aos dados, usando medidas de "*distancia"* ou "*similaridades"* entre dados individuais [39,41].

Recentemente foi proposta uma *versão construtiva de Algoritmos Evolutivos* (chamada de *Algoritmo Genético Construtivo (AGC)*) e sua aplicação testada em alguns problemas de Otimização Combinatória (veja tese de doutorado de *J. C. Furtado* em http://www.lac.inpe.br/~lorena/teseJC/CGA-tese.ps). Os Algoritmos Genéticos baseiam-se na evolução controlada de uma população de estruturas, e estão muito difundidos nos dias de hoje, apresentando várias aplicações bem sucedidas a

problemas de otimização [1,6,7,13,14,15,17,20,23,26,27,28,29]. Apresenta-se neste trabalho a aplicação do *AGC* a três problemas de agrupamentos em grafos.

Suponha que desejamos encontrar em um grafo $G=(V,E)$ uma partição do conjunto de vértices V em um número pré definido (em geral) de grupos, otimizando alguma medida tomada na combinação dos pesos associados aos vértices e/ou arestas. Os problemas de agrupamento examinados são o *problema clássico das p-medianas (PPM)*, o *problema capacitado das p-medianas (PPMC)* e o *problema agrupamento de corte mínimo (PCM).*

No *PPM* o objetivo é localizar *p* facilidades *(medianas)*, de forma a minimizar a soma das distancias de cada vértice a sua facilidade mais próxima. Hakimi [18,19], foi o primeiro pesquisador a formular o problema, para a localização de uma única facilidade, em seguida, generalizando para múltiplas medianas. Ele propôs um simples procedimento de enumeração para o problema. O problema é reconhecidamente *NP-hard* [12]. Diversas heurísticas têm sido desenvolvidas para o problema das *p*-medianas. Algumas são usadas para obter boas soluções iniciais ou para calcular soluções intermediárias em nós numa árvore de busca. Teitz e Bart [40], Goodchild e Noronha [16] e Denshan e Rushton [8] propuseram *heurísticas de troca*. Rolland, Schilling e Current [35] aplicaram *busca tabu*, Rosing e ReVelle uma *heurística de concentração* [36] e Rosing, ReVelle, Schilling e Current [37] compararam as duas propostas. Algoritmos mais complexos exploram uma *árvore de busca*. Estes apareceram em Efroymson e Ray [9], Jarniven e Rajala e Sinervo [21], Neebe [31], Christofides e Beasley [5], Beasley [4] e Galvão e Raggi [10, 11]. Algumas abordagens bem sucedidas usam informações primal/dual do problema (Senne e Lorena [38] e Beasley [2,4]).

No *PPMC* são consideradas capacidades ao serviço que será oferecido pelas medianas. O serviço total demandado pelos vértices identificados nos grupos das medianas não pode ultrapassar as capacidades de serviço das medianas. Aparentemente este problema não foi tão estudado como o *PPM* . Problemas semelhantes aparecem em Klein e Aronson [24], Mulvey e Beck [30] e Osman e Christofides [32]. Neste último trabalho encontra-se uma bibliografia razoavelmente completa de outros trabalhos relacionados ao problema, e alguns problemas testes.

O *PCM* considerado neste trabalho considera a formulação dada em Johnson e Mehrotra [22]. É o problema de particionar o conjunto de vértices V em p *agrupamentos*, tal que a soma dos pesos dos vértices nos agrupamentos esteja entre limites inferior e superior, enquanto que a soma dos pesos das arestas é maximizada (ou a soma dos pesos das aresta que estão fora dos agrupamentos é minimizada).

Os problemas são formalizados a seguir.

PPM :

Dado um conjunto de n pontos de demanda (vértices) $V = \{1,\ldots,n\}$ e uma matriz de distancias (pesos) $[\mu_{jl}]$, $\mu_{jl} \geq 0$, $\mu_{jj}=0$ e $\mu_{jl}=\mu_{lj}$ para todo $j,l \in V$, indicando distancias entre pares de vértices. O conjunto V é particionado em

p grupos, $2 \leq p < n$, $C_1, C_2, \ldots, C_p$, e $V_1 = \{\zeta_1, \ldots, \zeta_p\}$, $V_1 \subset V$, é o conjunto de índices das medianas de cada grupo (os pontos de serviço) $\zeta_i \in C_i$, $i = 1,\ldots,p$. O problema *PPM* procura por partições de V que

$$\textit{minimizam} \quad \sum_{i=1}^{p} \left(\sum_{j \in C_i} \mu_{\zeta_i j} \right).$$

PPMC:

Suponha agora que para cada ponto de demanda $j \in V$ existe um serviço positivo demandado a_j. Seja Q_i a capacidade do grupo i. O *PPMC* procura por partições de V que

$$\textit{minimizam} \quad \sum_{i=1}^{p} \left(\sum_{j \in C_i} \mu_{\zeta_i j} \right),$$

$$\text{satisfazendo} \quad \sum_{j \in C_i} a_j \leq Q_i, \; i=1,\ldots,p.$$

PCM:

Seja $E(C_i)$ o conjunto de arestas referente a C_i, e μ_e, $e \in E(C_i)$ os custos não negativos das arestas em C_i, e admita que cada grupo tenha limites de capacidade ($Q_{\min}$ *e* $Q_{\max}$). As medianas não são consideradas para o *PCM* e o objetivo é identificar a partição $C_1, C_2, \ldots, C_p$ que

$$\textit{maximiza} \quad \sum_{i=1}^{p} \left(\sum_{e \in E(C_i)} \mu_e \right),$$

$$\text{satisfazendo} \quad Q_{\min} \leq \sum_{j \in C_i} a_j \leq Q_{\max}, \; i=1,\ldots,p.$$

Neste trabalho, inicialmente apresentamos uma revisão do *AGC*, destacando suas principais características, bem como sua aplicação aos problemas *PPM*, *PPMC* e *PCM*. Finalmente os resultados computacionais e conclusões são apresentados.

2. Algoritmo Genético Construtivo - resumo

Apresentamos nesta seção um resumo do *AGC* e as definições necessárias para sua aplicação aos problemas de agrupamento.

Uma visão geral do *AGC* pode ser dada através do pseudocódigo:

AGC {Algoritmo Genético Construtivo}

```
α := 0 ;
ε := 0.05;                          { intervalo de tempo }
Inicialize P_α ;                    { população inicial }
```

Avalie P_α ; *{ adaptação proporcional }*
Para todo $s_k \in P_\alpha$ **calcule** $\delta(s_k)$ *{ calculo do rank }*
fim_para
Enquanto (não – condições de parada) **faça**
Para todo $s_k \in P_\alpha$ satisfazendo $\alpha < \delta(s_k)$ **faça** *{ teste de evolução }*
$\alpha := \alpha + \varepsilon$;
Selecione P_α de $P_{\alpha-\varepsilon}$; *{ operador de reprodução }*
Recombine P_α ; *{ operadores de recombinação }*
Avalie P_α ; *{ adaptação proporcional }*
fim_para
Para todo novo $s_k \in P_\alpha$ **calcule** $\delta(s_k)$ *{ calculo do rank }*
fim_para
fim_enquanto

Alguns passos no algoritmo *AGC* são sensivelmente diferentes de um Algoritmo Genético clássico. O *AGC* trabalha com uma *população dinâmica*, que cresce com o uso dos *operadores de recombinação*, e pode decrescer guiado pelo *parâmetro de evolução* α (aumentado de ε a cada geração). Quando uma estrutura ou esquema é criada, recebe uma *avaliação proporcional* e seu *rank* $\delta(s_k)$ que será usado no *teste de evolução*. Outra diferença central é a consideração explicita de *esquemas* e o novo processo de *avaliação-fg* (veja os detalhes em http://www.lac.inpe.br/~lorena/teseJC/CGA-tese.ps).

2.1. Representação de estruturas e esquemas

Vamos usar uma *cadeia* com alfabeto binário para a representação dos três problemas de agrupamentos. *Heurísticas de atribuição* fazem a correspondência entre estruturas e esquemas e o problema de agrupamento em questão. Alguns vértices são eleitos como *vértices sementes*, isto é, os vértices iniciais de cada grupo, que irão atrair de alguma maneira os outros vértices que participam da representação.

Para qualquer dos três problemas de agrupamento (*PPM, PPMC e PCM*), `(1,0,0,0,0,0,0,0,1,1)`, é uma *estrutura* para um problema de 10 vértices onde procura-se a formação de 3 grupos, enquanto que `(1,#,0,0,0,#,0,0,1,1)` é o que chamamos de *esquema* (os vértices 2 e 6, representados por #, não estão sendo considerados). Seja $V_1 = \{\zeta_1,...,\zeta_p\}$, o conjunto de índices dos *vértices sementes* (representados por 1 nas cadeias) de cada grupo que será formado. Para a estrutura e esquema acima, $V_1 = \{1,9,10\}$. Seja V_2 o conjunto de índices dos vértices *não-sementes* (representados por 0 nas cadeias). A correspondência com cada um dos problemas é feita aplicando as seguintes *heurísticas de atribuição*:

HA-PPM

1) *Inicialize os grupos com a atribuição dos vértices sementes*
$C_1 = \{\zeta_1\}, C_2 = \{\zeta_2\}, ..., C_p = \{\zeta_p\},$

2) *Atribua os vértices não-sementes a seu vértice semente mais próximo, formando a partição* $C_1,C_2,...,C_p$

HA-PPMC

1) *Inicialize os grupos com a atribuição dos vértices sementes* $C_1 = \{\zeta_1\}, C_2 = \{\zeta_2\}, ..., C_p = \{\zeta_p\}$,
2) *Enquanto a capacidade de cada grupo permitir, atribua os vértices não-sementes a seu vértice semente mais próximo, formando a partição* $C_1,C_2,...,C_p$

HA-PCM

1) *Inicialize os grupos com a atribuição dos vértices sementes* $C_1 = \{\zeta_1\}, C_2 = \{\zeta_2\}, ..., C_p = \{\zeta_p\}$,
2) *Enquanto a capacidade de cada grupo permitir, atribua os vértices não-sementes ao grupo que traga maior retorno da soma dos pesos das arestas formadas pelo conjunto que compõe os vértices do grupo e o vértice não-semente considerado, ponderado pelo número de elementos do grupo.*

No caso dos problemas *PPM* e *PPMC* os vértices sementes são as medianas que atraem os outros vértices mais próximos para seus grupos. No caso do problema *PCM*, os vértices sementes não são medianas, mas somente vértices iniciais que atraem os outros vértices para os grupos.

2.2. População inicial

A *população inicial* P_0 é formada somente por *esquemas*, que são gerados aleatoriamente. Em cada esquema existem exatamente *p sementes* distribuídas em posições aleatórias e 20% dos demais vértices são também distribuídos em posições aleatórias (*não-sementes*). Os vértices restantes não são considerados. Conforme o problema considerado, as heurísticas *HA-PPM*, *HA-PPMC* e *HA-PCM* são então aplicadas a cada esquema, obtendo seus grupos correspondentes.

2.3. Avaliação de esquemas e estruturas

Como as estruturas da população inicial são esquemas, não representam soluções para o problema (alguns vértices não estão sendo considerados), uma avaliação direta com a função objetivo de cada problema não faz sentido. Assim, uma das principais características deste novo método de otimização esta baseada em uma avaliação dupla, chamada de *avaliação-fg*, descrita a seguir.

Considerando P_α, a população no tempo α, são definidas as funções $f: P_\alpha \rightarrow \Re_+$ e $g: P_\alpha \rightarrow \Re_+$, admitindo-se, sem perda de generalidade, que $g(s_k) \geq f(s_k)$. É também definido um limite superior para o valor da função g, i.e., $g_{\max} \geq \max_{s_k \in P_\alpha} g(s_k)$. A *figura 1* mostra graficamente a definição das funções f, g e g_{max}.

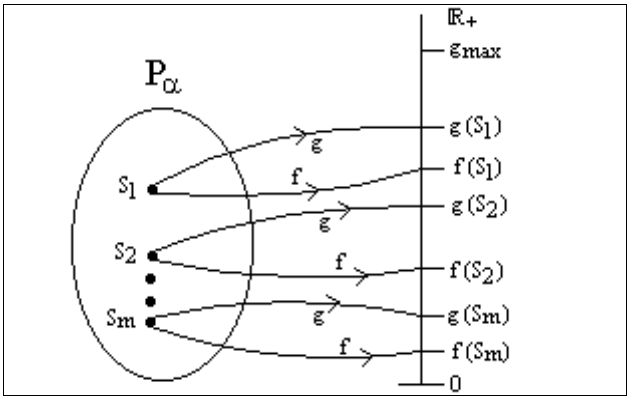


Figura 1: Representação gráfica das funções de avaliação.

A população é formada de *m* estruturas. Para cada estrutura existem duas funções de avaliação que sugerem o processo de adaptação da estrutura ao meio. O valor dessas funções produzem limites que serão usados em um processo semelhante ao usado no algoritmo A* [33] e permitem que a população consiga evoluir, usando como informação o desvio em relação ao valor máximo g_{max}. O desvio percentual esperado em relação a g_{max} será dado por *d*.

Para cada estrutura s_k, existe um desvio percentual entre os valores $g(s_k)$ e $f(s_k)$ em relação a $g(s_k)$, que será dada por:

$$d_k = \frac{g(s_k) - f(s_k)}{g(s_k)}, i = 1,..., m.$$

Para um mesmo desvio absoluto $g(s_k) - f(s_k)$ o desvio percentual diminui com o aumento de $g(s_k)$. Uma estrutura deve ser eliminada da população se, para algum valor de α ($\alpha \geq 0$), $d\, g_{max} \leq d_k\, g(s_k) + \alpha\, d\, (g_{max} - g(s_k))$. O objetivo a ser alcançado (esperado) é representado por $d.g_{max}$, enquanto a estrutura s_k possui um desvio $d_k\, g(s_k)$. O desvio $d\, (g_{max} - g(s_k))$ representa o desvio esperado para atingir g_{max} a partir de $g(s_k)$. Para $0 \leq \alpha \leq 1$ a heurística seria admissível.

O parâmetro $\alpha \geq 0$ poderá ser isolado como $\alpha \geq \frac{dg_{\max} - d_i g(s_k)}{d(g_{\max} - g(s_k))} = \delta(s_k)$ para indicar quando a estrutura deixaria de ser admissível na população. Isso corresponde a retirar a estrutura s_k da população. O parâmetro α é considerado um elemento de controle da evolução da população inicial. Considera-se variar α a partir de zero, em pequenos intervalos de evolução, onde novas gerações são criadas. Cada estrutura s_k, quando criada recebe um valor $\delta(s_k)$ que indica o seu tempo de permanência na população. Quanto menor o seu valor d_k, melhor será sua adaptação, e poderá permanecer mais tempo na população, e com maior probabilidade de participar dos operadores recombinação e mutação. Quanto maior a diferença $g_{max} - g(s_k)$, menor seu tempo de permanência na população.

Nossa condição de parada será quando a população for completamente eliminada ou para um determinado número de gerações. No processo a melhor estrutura será preservada.

2.4. Funções de avaliação para os problemas PPM, PPMC e PCM

Particularizando as funções f, g e o parâmetro g_{max} para cada problema:

PPM e PPMC

Após a aplicação das respectivas heurísticas *HA-PPM* ou *HA-PPMC* aos esquemas e/ou estruturas da população P_α, a função g é definida como $g(s_k) = \sum_{i=1}^{p} \sum_{j \in C_i(s_k)} \mu_{\zeta_i j}$; e a função f definida como $f(s_k) = \sum_{i=1}^{p} \lambda_i . \left[\left| C_i(s_k) \right| - 1 \right]$,

onde $\lambda_i = \min_{j \in C_i(s_k)} \left\{ \mu_{\zeta_i j} \right\}$ é a menor distancia que foi atribuída ao grupo i, e $\left| C_i(s_k) \right|$ é a cardinalidade do conjunto $C_i(s_k)$.

Claramente $f(s_k) \leq g(s_k)$, para todo $s_k \in P_\alpha$. Idealmente a diferença $g(s_k) - f(s_k)$ deve suficientemente pequena. Se s_k é uma estrutura então a função g produz o valor de uma solução ao problema. Este valor isolado não tem significado quando s_k é um esquema, e o uso da função f em conjunto com a função g apresenta uma avaliação proporcional do esquema (ou da estrutura) que esta sendo considerado.

PCM

Após a aplicação da heurística *HA-PCM* os p $(= \left| V_1(s_k) \right|)$ grupos $C_i(s_k)$ são identificados, correspondentes ao conjunto semente $V_1(s_k) = \left\{ \zeta_1, \zeta_2, ..., \zeta_p \right\}$. $E\left[C_i(s_k) \right]$ é o conjunto de arestas no subgrafo induzido pelo grupo $C_i(s_k)$. O *PCM* foi formulado como um problema de maximização, e neste caso, quando s_k é uma estrutura, a função f dará o valor da função objetivo. Ela é definida como $f(s_k) = \sum_{i=1}^{p} \sum_{e \in E\left[C_i(s_k) \right]} \mu_e$, e a função g é definida como $g(s_k) = \sum_{i=1}^{p} \lambda_i \left| E\left[C_i(s_k) \right] \right|$,

onde $\lambda_i = \max_{e \in E\left[C_i(s_k) \right]} \left\{ \mu_e \right\}$ é o maior peso entre as arestas de $C_i(s_k)$.

Para os três problemas, o limite superior comum $g_{\max}$ é obtido no início do algoritmo AGC , gerando uma estrutura completa de forma aleatória e fazendo $g_{\max}$ receber a avaliação g para esta estrutura. Para assegurar de que $g_{\max}$ será sempre um limite superior, depois da recombinação cada estrutura nova s_{nova} será rejeitada se $g_{\max} \leq g(s_{nova})$.

2.5. Seleção e recombinação

Após a geração de cada estrutura e posterior avaliação, é atribuída uma ordem na população baseada em dois critérios:

- esquemas que consideram uma fração maior dos dados do problema;
- estruturas ou esquemas onde a diferença relativa entre os valores das funções *f* e *g* forem menores.

Assim, a população é ordenada pela chave: $\Delta(s_k) = \frac{1+d_k}{V_1+V_2}$.

Um operador irá selecionar duas estruturas e/ou esquemas que devem recombinar-se de forma a gerar uma ou várias novas estruturas e/ou esquemas. Na escolha das duas estruturas e/ou esquemas, s_{base} e s_{guia} , a estrutura s_{base} é selecionada entre as *n* primeiras (melhores) estruturas na ordenação e s_{guia} é selecionada na população inteira;

Uma vez selecionadas as duas estruturas, se s_{base} é um esquema, o operador recombinação é usado para a construção de uma nova geração na população. Denominaremos s_{base} de esquema base e de s_{guia} esquema ou estrutura guia.

Na recombinação, cada *label* do esquema base é comparado ao correspondente (na mesma posição) no esquema ou estrutura guia. Diversas recombinações podem ocorrer e existirá um tratamento para cada caso *(examinados nesta ordem)*:

Caso 1: Quando base e guia apresentam *labels* iguais, o esquema ou estrutura resultante não é alterado.

```
base:       ( 1,0,#,0,1,#,1,#)
guia:       ( #,0,1,1,1,#,0,0)
resultante: ( _,0,_,_,1,#,_,_)
```

Caso 2: Quando a base apresenta um *label* diferente de # e a guia apresenta um *label* igual a #, o *label* da base é preservado.

```
base:       ( 1,0,#,0,1,0,1,#)
guia:       ( #,0,1,1,1,#,0,0)
resultante: ( 1,_,_,_,_,0,_,_)
```

Caso 3: Quando a base apresenta um *label* igual a # e a guia um *label* igual a 0, o *label* resultante será 0.

```
base:       ( 1,0,#,0,1,0,1,#)
guia:       ( #,0,1,1,1,#,0,0)
resultante: ( _,_,_,_,_,_,_,0)
```

Caso 4: Quando a base apresenta um *label* diferente de 1 e a guia um *label* igual a 1. Neste caso, uma semente esta sendo introduzida na estrutura resultante. Uma vez que o número de sementes em cada estrutura deve ser igual a *p*, uma semente da base deve ser retirada e substituída por um vértice que não seja semente, ou seja, um *label* igual a 0. Como podemos retirar da base qualquer uma das *p* sementes existentes, dois critérios podem ser usados:

- escolher aleatoriamente uma das sementes e realizar sua substituição, o que acarretará em um nova estrutura;
- realizar a substituição das *p* sementes, o que acarretará em *p* novas estruturas.

Usando o segundo critério, temos como exemplo:

```
base:          ( 1,0,#,0,1,#,1,#)
guia:          ( #,0,1,1,1,#,0,0)
resultantes:   ( 0,0,1,0,1,#,1,#)
               ( 1,0,1,0,0,#,1,#)
               ( 1,0,1,0,1,#,0,#)
```

Caso 5: Quando a base apresenta um *label* igual a 1 e a estrutura guia apresenta um *label* igual a 0. Neste caso, o *label* resultante será igual a 0, no entanto, a estrutura terá p -1 sementes. Desta forma, é necessário que algum vértice que não pertencia ao conjunto de sementes passe a ser uma semente. Novamente existem duas possibilidades:

- escolher aleatoriamente um vértice que não é semente e torná-lo semente;
- substituir todos os vértices que não são sementes por sementes, dando origem a tantas novas estruturas quanto o número de vértices que não são sementes na estrutura base.

Usando o segundo critério, temos:

```
base:          ( 1,0,#,0,1,#,1,#)
guia:          ( #,0,1,1,1,#,0,0)
resultantes:   ( 1,1,#,0,1,#,0,#)
               ( 1,0,#,1,1,#,0,#)
```

Além do operador recombinação, usamos o operador *mutação*. Este operador é aplicado no caso em que s_{base} é uma estrutura completa. Na mutação, cada semente da estrutura permuta de posição com os vértices que não pertencem ao conjunto das sementes, gerando novas estruturas ($|V|$-$|V_1|$ estruturas). No algoritmo implementado, iniciamos com o operador recombinação, não havendo mutação. A medida que novas estruturas são geradas, o operador mutação torna-se mais freqüente, chegando ao final do processo como único operador.

3. Resultados computacionais

O *AGC* foi testado inicialmente em uma amostra dos dados para o *PPM* que se encontram na *OR-library* [3]. Os tamanhos das instancias são 100, 200, 300, 400, e 500 vértices. O número de medianas varia de 5 a 67. Os resultados computacionais estão apresentados na *tabela 1*. A solução do *AGC* é a avaliação de $g(s_k)$ para a melhor estrutura gerada. O *gap* = [(solução *AGC* - Solução ótima)*100]/(solução *AGC*).

Todos os *gaps* foram menores que 0.73% e nulos para nove instancias. O algoritmo foi codificado em *C*, e os testes avaliados em um *Pentium 166 Mhz*. Pode ser notado na *tabela 1* que um número grande de medianas aumenta o tempo computacional. Isso ocorre pelo uso mais intensivo da busca local de mutação. O controle de evolução (intervalo de tempo) usado foi: $\varepsilon = 0.05$ para $0 \leq \alpha \leq 1$ e $\varepsilon = 0.025$ para $\alpha > 1$; e o desvio proporcional admitido $d = 0.1$.

Problema	*Vertices (n)*	*Medianas (p)*	*Solução ótima*	*Solução AGC*	*gap (%)*	*Tempo (segundos)*
pmed1	100	5	5819	5819	0	28
pmed2	100	10	4093	4093	0	37
pmed3	100	10	4250	4250	0	34
pmed4	100	20	3034	3034	0	230
pmed5	100	33	1355	1360	0.36	375
pmed6	200	5	7824	7824	0	172
pmed7	200	10	5631	5631	0	238
pmed8	200	20	4445	4454	0.20	1055
pmed9	200	40	2734	2754	0.73	3331
pmed10	200	67	1255	1257	0.15	4325
pmed11	300	5	7696	7696	0	369
pmed12	300	10	6634	6637	0.04	677
pmed13	400	5	8162	8162	0	555
Pmed14	500	5	9138	9138	0	1875

Tabela 1: Resultados computacionais dos testes realizados.

Para o problema *pmed1* traçamos os seguintes gráficos (*figuras 1 a 4*):

Figura 1: Tamanho da população por geração

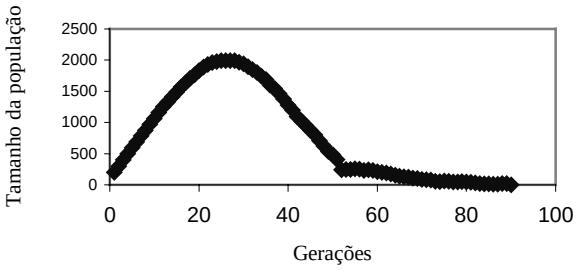


Figura 2: Número máximo de vértices por geração

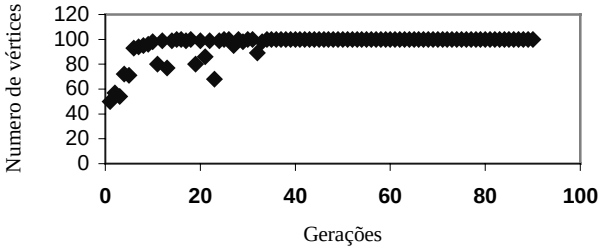


Figura 3: Melhor dk por geração

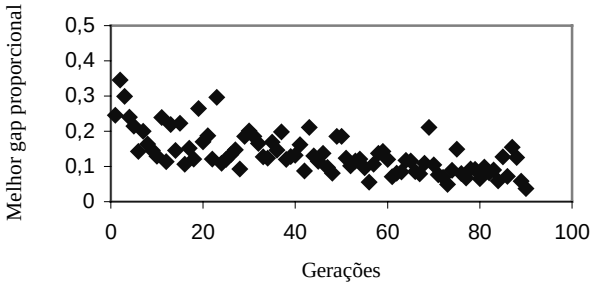
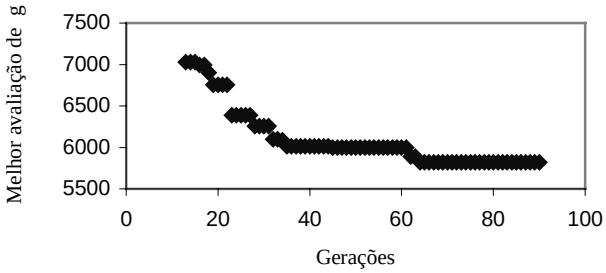


Figura 4: Melhor valor de g por geração



Onde:

Figura 1: Mostra como varia o tamanho da população;
Figura 2: Mostra o crescimento do número de vértices da estrutura com máximo valor $|V_1|+|V_2|$ por geração;
Figura 3: Mostra os valores do desvio d_k em função do número de gerações; e
Figura 4: Mostra a evolução da função objetivo $g(s_k)$ em relação as gerações.

O *AGC* foi também aplicado a um conjunto de instancias do *PPMC*, propostas no trabalho de Osman e Christofides [32]. Foram considerados dois conjuntos de 10 instancias, com (50x5) e (100x10) vértices e medianas, respectivamente. A *tabela 2* apresenta os *gaps* para a melhor solução de 7 heurísticas, e o tempo médio para 5 replicações do *AGC*. As colunas representam a identificação do problema, resultados de heurísticas usadas em [32] (*H.OC, H1+F1, H1+B1, HSS.OC, HSS.C e TS1+FBA*), os gaps obtidos pelo *AGC* e os seus tempos.

A heurística *H.OC* é uma heurística construtiva simples, enquanto que *H1+F1* e *H1+B1* iniciam com o resultado de *H.OC* e realizam permutações usando as estratégias "*first improve*" e "*best improve*". Os algoritmos *HSS.OC* e *HSS.C* são implementações de *simulated annealing* que usam estratégias distintas de resfriamento da temperatura. O algoritmo *TS1+FBA* implementa uma versão da *busca tabu* [32].

Observando os resultados na *tabela 2*, pode-se concluir que o *AGC* apresenta resultados tão bons quanto *TS1+FBA* e ligeiramente pior que *HSS.OC* e *HSS.C*, isto é, os melhores resultados são comparáveis.

Prosseguimos os testes computacionais com instancias do *PCM* propostas no trabalho de Johnson e Mehrotra [22]. Dois conjuntos de instancias foram testados, o primeiro com Q=450 e o segundo com Q=512. O número de vértices, grupos e arestas variam, e as soluções ótimas dos problemas foram obtidas em [22] usando um método exato. A *tabela 3* apresenta os resultados da aplicação do *AGC* nas instancias. O *gap* neste caso é calculado por *gap* = [(solução ótima – solução *AGC*)*100]/(solução ótima). O resultados do *AGC* podem ser considerados bons, com tempos computacionais baixos.

4. Conclusões

Os resultados mostram que o *AGC* é eficiente na otimização dos problemas de agrupamentos considerados. Observamos que em muitas instâncias o algoritmo foi capaz de obter a solução ótima num pequeno intervalo de tempo, ficando no geral a menos de 1% da solução ótima.

Verificamos que existe maior dificuldade em obter boas soluções quando o número de sementes é maior. Acreditamos que isto ocorre devido a representação usada para as estruturas. O *AGC* é um método flexível, i.e., outras restrições poderiam ser facilmente tratadas nos problemas, e existe muita liberdade também na definição das heurísticas de atribuição.

A representação usada e os decodificadores de atribuição (heurísticas) podem ser usados em outros problemas de agrupamentos, mas é importante salientar que os conceitos básicos do *AGC* são independentes da representação e decodificadores. Alguns resultados anteriores usam representações diferentes na aplicação do *AGC* aos problemas de Cortes bi-dimensional [25] e ao problema da k-coloração [34].

Finalmente, acreditamos que uma análise mais profunda dos parâmetros empregados pode trazer ainda melhores resultados.

Agradecimentos: O trabalho do segundo autor recebeu apoio parcial do CNPq através dos projetos de números 350034/91-5, 520844/96-3, 680082/95-6 e da FAPESP nos projetos números 95/9522-0 e 96/04585-6.

Problema	H.OC	H1+FI	H1+BI	HSS.OC	HSS.C	TS1+FBA	AGC	Tempo (s)
1	10,23	9,39	14,72	0	2,94	2,94	0	2
2	10,27	2,97	5,13	0	0	0	0	2
3	29,42	7,98	8,65	0	0	0	0	12
4	36,86	0	0,15	0	0	0	0	2
5	21,08	12,34	1,95	0	0	0	0	8
6	13,36	8,09	8,86	0	0	0	0	2
7	22,99	8,25	4,70	0	2,28	0	0	3
8	15,24	1,70	2,07	0	0	0,12	0,73	11
9	5,17	2,79	2,65	0	0	0	0	2
10	22,67	1,80	7,47	0	0	0	1,44	14
11	75,04	1,39	1,29	0	0	0,29	0,79	614
12	62,21	3,93	0,82	0	0	0,20	0,31	296
13	80,01	11,50	2,63	0	0	0	0	327
14	66,49	1,62	7,33	0,30	0	0,30	0,50	303
15	38,91	0,54	4,21	0	0	0,36	0	315
16	86,58	11,42	4,08	0	0	0,31	0,10	271
17	61,02	6,76	5,60	0,48	0,29	0,58	0	332
18	28,95	4,41	8,91	0,19	0,19	0,19	0,19	380
19	58,48	7,17	9,11	0	0,09	0,29	0,09	410
20	86,26	3,08	2,48	0	1,39	0	3,38	503

Tabela 2: PPMC: gaps do AGC e de heurísticas de Osman & Christofides .

Problema		Q	Vértices	p	Arestas	Solução ótima	AGC	Gap (%)	Tempo (s)
1	Cb450.30.6.47	450	30	6	47	1099	1099	0	20
2	Cb450.48.8.98	450	48	8	98	2928	2928	0	31
3	Cb450.47.8.99	450	47	8	99	1837	1826	0,59	86
4	Cb450.47.9.101	450	47	9	101	3574	3451	3,44	210
5	Cb450.61.9.187	450	61	9	187	22245	21459	3,53	118
6	Cb512.30.5.47	512	30	5	47	1174	1142	2,72	63
7	Cb512.45.7.98	512	45	7	98	3238	3209	0,89	49
8	Cb512.47.7.99	512	47	7	99	1993	1940	2,65	79
9	Cb512.47.8.101	512	47	8	101	3969	3713	6,40	322
10	Cb512.61.8.187	512	61	8	187	23564	22762	3,40	359

Tabela 3: Problema PCM: resultados do AGC.

5. Referências

[1] Bäck, Th.; Schwefel, H.P. *An Overview of evolutionary algorithms for parameter optimization*. Evolutionary Computation, v. 1, p. 1-23, 1993.

[2] Beasley, J.E. *A note on solving large p-median problems*. European Journal of Operational Research. v. 21, p. 270-273, 1985.

[3] Beasley, J.E. *OR-Library: distribution test problems by electronic mail*. Journal of Operational Research Society. v. 41, p. 1069-1072, 1990.

[4] Beasley, J.E. *Lagrangean heuristics for Location problems*. European Journal of Operational Research. v. 65, p. 383-399, 1993.

[5] Christofides, N.; Beasley, J.E. *A tree search algorithm for the p-median problems*. European Journal of Operational Research. v. 10, p. 196-204, 1982.

[6] Davis, L.D., Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1991.

[7] De Jong, K. *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. thesis, University of Michigan, Ann Arbor, MI, 1975.

[8] Densham, P.J., Rushton, G. *A more efficient heuristic for solving large P-median problems.* Papers in Regional Science v.71, p. 307-329, 1992.

[9] Efroymson, M. A. and Ray, T. L. *A branch-and-bound algorithm for plant location* Operations Research v.14, p. 361-368, 1966.

[10] Galvão, R.D.; Raggi, L.A. *A method for solving to optimality uncapacitated location problems.* Annals of Operations Research. v. 18, p. 225-244, 1989.

[11] Galvão, R.D.; Ferreira Filho, V.J.M.; Rivas, M.P.A. *Some computational aspects of p-median type location problems*. VIII Congresso CLAIO/XXVIII SBPO, Rio de Janeiro, p. 1266-1271, 1996.

[12] Garey, M. R.; Johnson, D. S., Computers and Intractability: a Guide to the Theory of NP-Completeness. San Francisco: Freeman, 1979.

[13] Goldberg, D.E., Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA, p. 11-172, 1989.

[14] Goldberg, D.E.; Korb, B.; Deb, K. *Messy genetic algorithms: Motivation, analysis, and first results.* Complex Systems v. 3: p. 493-530, 1989.

[15] Goldberg, D.E.; Deb, K.; Kargupta, H.; Harik, G. *Rapid, accurate optimization of difficult problems using fast messy genetic algorithms.* IlliGAL Report No. 93004, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois, Urbana, 1993.

[16] Goodchild, M. F.; Noronha, V. *Location-allocation for small computers.* Monograph No. 8, Department of Geography, University of Iowa, Iowa city, 1983.

[17] Grefenstette, J.J. *Proceedings of the second int'l conference on genetic algorithms and their applications*. Hillsdale, NJ, Lawrence Erlbaum, 1987.

[18] Hakimi, S.L. *Optimum distribution of switching centers and the absolute centers and the medians of a graph*. Operations Research. v. 12, p. 450-459, 1964.

[19] Hakimi, S.L. *Optimum distribution of switching centers in a communication network and some related graph theoretic problems*. Operations Research. v.13, p. 462-475, 1965.

[20] Holland, J.H., Adaptation in natural and artificial systems. MIT Press, p. 11-147, 1975.

[21] Jarvinen, P.; Rajala, J. ; Sinervo, H. *A branch and bound algorithm for seeking the p-median.* Operations Research, v. 20, p. 173-178, 1972.

[22] Johnson, E.L.; Mehrotra, A. *Min-Cut Clustering*. Working paper. School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, 1992.

[23] Kargupta, H. *Search, Polynomial Complexity, and The Fast Messy Genetic Algorithm*, Ph.D. thesis, IlliGAL Report No. 95008, Illinois Genetic Algorithms Laboratory,

Department of General Engineering, University of Illinois, Urbana, 1995.

[24] Klein, K. ; Aronson, J.E. *Optimal clustering: a model and method.* Naval Research Logistics, v. 38, p. 447-461, 1991.

[25] Lorena, L.A.N.; Lopes, F.B. *A dynamic list heuristic for 2D-cutting*. In J. Dolezal and J. Fidler eds., System Modeling and Optimization, Chapman-hall, p. 481-488, 1996a.

[26] Lorena, L.A.N.; Lopes, L.S. *Computational experiments with genetic algorithms applied to set covering problems.* Pesquisa Operacional, 16, p. 41-53, 1996b.

[27] Lorena, L.A.N.; Lopes, L.S. *Genetic algorithms applied to computationally difficult set covering problems.* Journal of the Operational Research Society, 48, p. 440-445, 1997.

[28] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, 1996.

[29] Mitchell, M., An Introduction to Genetic Algorithms. MIT Press, Cambridge, England, 1996.

[30] Mulvey, J. M.; Beck, M.P. *Solving capacitated clustering problems.* European Journal of Operational Research, v. 18, p. 339-348, 1984.

[31] Neebe, A.W. *A branch and bound algorithm for the p-median transportation problem.* Journal of the Operational Research Society. v. 29, p. 989-995, 1978.

[32] Osman, I. H.; Christofides, N. *Capacitated clustering problems by hybrid simulated annealing and tabu search*. International Transactions in Operational Research, v. 1, n. 3, p. 317-336, 1994.

[33] Pearl, J., Heuristics – Intelligent search strategies for computer problem solving, Addison-Wesley Publishing Company, MA, 1985.

[34] Ribeiro Filho, G. *Uma heurística construtiva para coloração de grafos.* Master thesis, INPE, 1997.

[35] Rolland, E., Schilling, D. A., Current, J. R., *An efficient Tabu search procedure for the p-median problem.* European Journal of Operational Research 96, 329-342,1997.

[36] Rosing, K. E.., ReVelle, C. S., *Heuristic concentration; Two stage solution construction*. European Journal of Operational Research 97, 75-86,1997.

[37] Rosing, K. E., ReVelle, C. S., Schilling D. A. and Current, J. R. *Heuristic concentration and Tabu search: A head to head comparison.* European Journal of Operational Research, 93-99. 1998.

[38] Senne, E.L.F.; Lorena, L.A.N. *A lagrangean/surrogate approach to facility location problems, EURO-TIMS Congress – Barcelona, 1997.*

[39] Spath, H., *Cluster Analysis Algorithms for data reduction and classification of objects. Ellis Horwood Publishers, New York, 1980.*

[40] Teitz, M.B.; Bart, P. *Heuristic methods for estimating the vertex median of a weighted graph*. Operations Research. v. 16, p. 955-961, 1968.

[41] Zupan, J., Clustering of Large Data Sets. John Wiley & Sons, New York, 1982