SPOLM 2007

ISSN 2175-6295 Rio de Janeiro- Brasil, 08 e 09 novembro de 2007.

ALGORITMO EVOLUTIVO PARA O PROBLEMA DE CLUSTERIZAÇÃO EM GRAFOS ORIENTADOS

Gustavo Silva Semaan, Luiz Satoru Ochi
Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brasil
gsemaan@ic.uff.br, satoru@ic.uff.br

Resumo

Esse trabalho apresenta uma nova heurística para o problema de clusterização de dados utilizando conceitos de algoritmos evolutivos. São propostas técnicas de particionamento da população, reconexão de caminhos, e busca local para obtenção de soluções de melhor qualidade. É apresentada uma análise dos resultados obtidos em que é possível observar o aumento da qualidade das soluções obtidas com a utilização dessas técnicas.

Palavras-chave: Problemas de Clusterização Automática, Algoritmos Evolutivos

Abstract

This work presents a new heuristic to clustering problem using an evolutionary algorithm. We present techniques of population partition, path-relinking and local search for extraction of best solutions. It presents experiments where is possible to observe the improvement of quality when we utilize these techniques.

Keywords: *Automatic Clustering Problem, Evolutionary Algorithm*

1. INTRODUÇÃO

Segundo [2], o problema de clusterização consiste em agrupar os elementos de uma base de dados em subconjuntos disjuntos denominados *clusters*, de forma a maximizar a similaridade entre os elementos de um mesmo *cluster* e minimizar a similaridade entre elementos de *clusters* distintos. Dado um conjunto com *n* elementos $X = \{x_1,.., x_n\}$, encontrar partições do conjunto *X* em *k clusters* disjuntos C_i respeitando as seguintes condições:

$$C_i \neq \varnothing \qquad i = 1...k$$
$$C_i \cap C_j = \varnothing \qquad i,\ j = 1,...k$$
$$C_i \cup ... \cup C_k = X$$

Embora grande parte das abordagens que tratam do problema de clusterização utiliza uma quantidade de *clusters* pré-definida, aplicações reais podem não possuir conhecimento prévio de qual é a quantidade ideal de clusters para um problema. Quando a quantidade de *clusters* é definida préviamente, o problema é conhecido como *Problema de k-clusterização* ou simplesmente por Problema de Clusterização (PC) e a quantidade de soluções viáveis é obtida pela Fórmula 1. Caso a quantidade ótima de clusters não é conhecida, ou seja, a quantidade de clusters a ser utilizada faz parte do problema, trata-se de um Problema de C*lusterização* Automática (PCA) e a quantidade de soluções viáveis é obtida pela Fórmula 2. Ambos os problemas são classificados como NP-Completo embora o PCA seja muito difícil de se resolver devido ao maior numero de soluções alternativas.

$$N(x) = \left(\frac{1}{k!}\right) \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} (j)^n$$

Fórmula 1: quantidade de soluções viáveis em um problema de *k-clusterização onde k = número de clusters e n = número de elementos*

$$N(x) = \sum_{k=1}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$$

Fórmula 2: quantidade de soluções viáveis em um problema de *clusterização* automática

Conforme [8,11] a análise de *clusters* é um problema fundamental para ciências experimentais em que deseja-se classificar objetos em grupos e existem aplicações em biologia, medicina, economia, psicologia entre muitas outras áreas. Em muitas aplicações o PC ou o PCA pode ser representado na estrutura de um grafo e o problema passa a ser a de particionar um grafos em conjuntos de subgrafos distintos como mostrado a seguir.

2. PARTICIONAMENTO DE GRAFOS ORIENTADOS

Segundo [4], o problema de clusterização visto no contexto de particionamento de grafos consiste em, dado um grafo, agrupar os vértices deste grafo em subconjuntos ou *clusters* disjuntos, conforme uma medida de similaridade ou aptidão.

Em [5] um grafo orientado representa os módulos de um sistema, em que os vértices são as classes e os arcos suas dependências. Para tornar o sistema mais prático operacionalmente, esse foi dividido em subsistemas. Mesmo em pequenos grafos, como o que representa as classes e dependências do sistema, a quantidade de possibilidades de agrupamentos pode ser muito elevada. A Figura 1 apresenta um exemplo de grafo e duas possíveis soluções para o problema de clusterização automática.

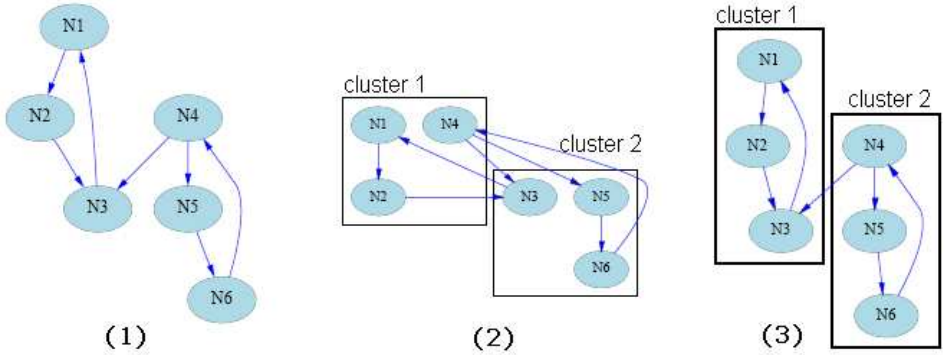

Figura 1: exemplos de particionamento de grafos [5]

Em seus trabalhos, [5] e [4] buscam maximizar a quantidade de relacionamentos entre as classes de um mesmo cluster e minimizar a quantidade de relacionamento entre classes de clusters distintos. Na primeira solução apresentada na Figura 1 (item 2) pode-se observar que existem muitos relacionamentos entre classes de clusters distintos e no cluster 2, por exemplo, existe apenas um relacionamento interno, o que relaciona os vértices N5 e N6. Já na Figura 2 (item 3) apresenta apenas 1 relacionamento entre classes de clusters distintos e vários relacionamentos entre classes de um mesmo cluster. Assim a *clusterização* representada pelo item 3 é considerada melhor que a representada no item 2. Pequenas alterações podem alterar significamente a qualidade de uma solução. Por exemplo, no item 2, a troca entre o vértices N4 do cluster 1 e o vértices N3 do cluster 2 faria com que a solução fosse semelhante a solução apresentada no item 3. A função de aptidão que avalia a qualidade de uma solução utilizada nesse trabalho é apresentada na Sessão 3.2.

3. ALGORITMO EVOLUTIVO HÍBRIDO

Conforme [10] algoritmos evolutivos e algoritmos genéticos são muito utilizados na área de inteligência artificial, inspirados na teoria da evolução natural e genética, conhecidas como computação evolucionária. Estes algoritmos tentam simular alguns aspectos da teoria da seleção natural de Darwin e são utilizados em muitos problemas considerados complexos.

Esse trabalho apresenta um algoritmo evolutivo híbrido (AEH) para o Problema de Particionamento de Grafos (PPG) da literatura problema este proposto inicialmente por [5]. No presente algoritmo, técnicas propostas por [4] que tornaram algoritmo mais eficiente bem como implementações de técnicas de divisão da população utilizada por [9] e reconexão de caminhos são incorporadas ao algoritmo evolutivo.

3.1. REPRESENTAÇÃO DA SOLUÇÃO

Segundo [5] uma boa representação para o problema é extremamente importante para a performance do algoritmo e pode ser crucial para a rápida convergência e qualidade das soluções obtidas pelo mesmo.

A estrutura utilizada para representação da solução aqui adotada foi a *group-number* que, conforme descrito em [3], o índice do vetor representa o vértice do grafo e seu conteúdo representa o *cluster* a que o vértice pertence. A solução apresentada pela Figura 2 indica que os vértices do grafo foram divididos em 3 clusters, onde o cluster 1 possui os vértices 1,3,5 e 7, o cluster 2 possui os vértices 2 e 6 e o cluster 3 possui os vértices 4 e 8.

1	2	3	4	5	6	7	8
1	2	1	3	1	2	1	3

Figura 2: representação de uma solução

3.2. FUNÇÃO DE APTIDÃO

Conforme [5] uma solução é considerada de boa qualidade quando possui baixa quantidade de relacionamentos *inter-clusters* e uma grande quantidade de relacionamentos *intra-cluster*. A intra-conectividade é uma medida que mensura a densidade de conexões entre vértices de mesmo um *cluster*. A Fórmula 3 é utilizada para calcular a intra-conectividade de um cluster *i*, em que μ_i é o total de arcos que possuem em uma das extremidades algum vértice do cluster *i* e N_i é a quantidade de vértices do cluster. Já a inter-conectividade mensura a conectividade entre vértices de *clusters* distintos. A Fórmula 4 é utilizada para calcular a inter-conectividade entre dois clusters, em que N_i é o total de vértices do cluster *i*, N_j é o total de vértices do cluster *j* e ε_{ij} o total de arcos que possuem as extremidades em i e j.

A função de aptidão utilizada consiste em uma conjugação entre a intra-conectividade e a inter-conectividade, conforme apresenta a Fórmula 5. Esta medida foi denominada *Qualidade de Modularização (MQ: Modularization Quality*) e retorna valores no intervalo [-1,1]. O objetivo do AEH é maximizar a qualidade de modularização, sendo para isto necessário maximizar a intra-conectividade e/ou minimizar a inter-conectividade.

$$A_i = \frac{\mu_i}{N_i^2}$$

Fórmula 3: intra-conectividade

$$E_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \frac{\varepsilon_{ij}}{2N_i N_j} & \text{if } i \neq j \end{cases}$$

Fórmula 4: inter-conectividade

$$MQ = \begin{cases} \frac{\sum_{i=1}^{k} A_i}{k} - \frac{\sum_{i,j=1}^{k} E_{i,j}}{\frac{k(k-1)}{2}} & \forall k > 1 \\ A_i & k = 1 \end{cases}$$

Fórmula 5: Qualidade de modularização

3.3. SELEÇÃO

Esse trabalho utiliza o critério de elitismo para manter a melhor solução da população atual na próxima geração e a técnica de seleção por torneio. Na técnica de seleção por torneio uma quantidade *t* de soluções é aleatoriamente selecionada e dessas soluções somente a melhor será mantida na próxima geração. Esse processo será repetido até que uma nova população seja completada. A Figura 3 apresenta um exemplo de solução por torneio considerando *t* = 3. As soluções 5, 6 e 7 foram selecionadas aleatoriamente e a solução 5, que possui melhor aptidão, será selecionada para a próxima geração.

1	2	3	1	1	1	3	2	3
2	1	3	1	1	1	3	1	3
3	1	2	1	1	1	2	1	3
4	2	1	1	3	1	3	2	3
5	1	3	3	1	1	3	1	3
6	1	2	1	2	3	2	3	3
7	2	3	1	3	1	2	1	3

Figura 3: seleção por torneio (*t* = 3)

3.4. CRUZAMENTO

O operador de cruzamento é utilizado para, perante uma probabilidade determinada como parâmetro do algoritmo, combinar pares de indivíduos e obter novos indivíduos potencialmente com melhor aptidão. Esse trabalho utiliza o cruzamento de 1 ponto que consiste em, dado um ponto aleatório p para $1 \leq p \leq n$ em que n é a quantidade de vértices do grafo, fazer a troca entre todos os demais genes (vértices) da solução, conforme ilustra a Figura 4. Esta operação não gera soluções inválidas, uma vez que no problema de particionamento de grafos orientados um vértice pode pertencer a qualquer cluster e os clusters podem conter quaisquer vértices.

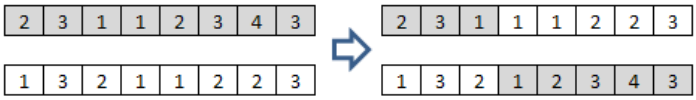

Figura 4: cruzamento de 1 ponto

3.5. MUTAÇÃO

Cada gene das soluções, perante uma probabilidade determinada como parâmetro do algoritmo, pode ser alterado. Esta técnica é utilizada para explorar novas regiões, aumentando o espaço de busca e abrindo novas possibilidades de soluções. Como os genes no AEH aqui proposto representam os *clusters* aos quais seus vértice pertencem, a aplicação da mutação altera o vértice de *cluster*. A Figura 5 ilustra a aplicação do operador de mutação alterando o vértice 6 do cluster 2 para o cluster 3. Assim como no operador de cruzamento não há necessidade de ajustes, uma vez que não são geradas soluções inválidas.

Figura 5: operador de mutação

3.6. BUSCA LOCAL

A busca local proposta para o AEH tem como objetivo efetuar uma busca numa vizinhança da melhor solução de cada geração. Para isto ela analisa perturbações na solução semente buscando a melhoria desta, ou seja, encontrar um ótimo local associado a vizinhança analisada. Nesse contexto, neste procedimento, cada vértice da solução semente é deslocado para o cluster ao qual ele compartilha a maior quantidade de arcos. Somente se ocorrer a melhoria da aptidão dessa solução a nova solução será mantida.

3.7. RECONEXÃO DE CAMINHOS

A reconexão de caminhos (*path relinking*) foi proposta por [6,7]. Segundo [1,6,7] a reconexão de caminhos é uma técnica que integra intensificação e diversificação na busca por soluções melhores e consiste em gerar soluções intermediárias entre uma solução base e uma solução alvo, com o objetivo de encontrar uma solução intermediária melhor. Nesse trabalho a solução base é a melhor solução encon-

trada até o momento pelo algoritmo enquanto a solução alvo é uma solução entre as soluções elite e é suficientemente distinta da solução base.

A busca por soluções intermediárias melhores pode ocorrer no sentido solução base para solução alvo, solução alvo para solução base ou mesmo em ambos sentidos, com o objetivo de encontrar uma solução intermediária melhor que os extremos. A Figura 6 ilustra a aplicação da reconexão de caminhos.

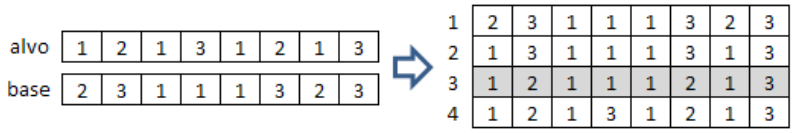

Figura 6: reconexão de caminhos

Nesse trabalho, a solução base é copiada para a primeira posição da lista de soluções intermediárias. A partir dela soluções intermediárias serão reconstruídas até que uma delas seja semelhante a solução alvo. Esta reconstrução será realizada mantendo os clusters da solução alvo nas soluções intermediárias, *cluster* a *cluster*.

A solução intermediária 2, por exemplo, é formada pela solução intermediária 1 e os vértices que pertencem ao cluster 1 da solução alvo são atualizados para permanecerem no cluster 1. Já a solução intermediária 3 é formada pela solução intermediária 2 e os vértices que pertencem ao cluster 2 da solução alvo são atualizados para permanecerem no cluster 2. Esta operação é repetida para todos os clusters da solução alvo.

Na Figura 6, a solução intermediária 3 possui a aptidão melhor que a solução base. Isto indica que foi encontrada uma solução intermediária melhor que a melhor solução encontrada até o momento e esta solução substituirá a solução base na população.

3.8. PARTICIONAMENTO DE POPULAÇÃO

Com intuito de acelerar a convergência da população para soluções melhores foi utilizado o particionamento da população conforme proposto em [9]. As soluções são ordenadas descendentemente conforme suas aptidões. A população, então, é dividida nas classes A, B e C conforme a qualidade das soluções e o tamanho das classes são parâmetros no algoritmo.

Para cada nova geração as soluções contidas em A são mantidas, as soluções contidas em C são excluídas e novas soluções aleatórias são geradas e as soluções pertencentes a classe B são submetidas aos operadores de cruzamento e mutação. Quando este módulo é utilizado em conjunto ao módulo de reconexão de caminhos, apresentado na sessão 3.7, a solução alvo é a melhor solução da população e a solução base é a melhor solução da população classe B.

4. RESULTADOS OBTIDOS

Para a realização da análise experimental do algoritmo AEH aqui proposto foram utilizados os grafos apresentados na Tabela 1. O trabalho [5] utilizam apenas instâncias pequenas, de até 20 vértices. Já o trabalho [4] utilizou instâncias de até 500 vértices.

Grafo	Quantidade de Vértices	Quantidade de Arcos
1	100	514
2	250	2345
3	500	4290

Tabela 1: grafos utilizados nos experimentos.

Cada grafo apresentado na Tabela 1 foi submetido a cada configuração apresentada na Tabela 3 cinco vezes. A Tabela 2 apresenta a relação de siglas utilizadas nas Tabelas 3 e 4. O critério de parada utilizado nesse trabalho foi a quantidade de gerações. Nas configurações em que o particionamento da população foi utilizado o tamanho das classes A, B e C foram de 10, 70 e 20 soluções, respectivamente.

Sigla	Descrição
BL	Busca Local
RC	Reconexão de caminhos
PP	Particionamento da população
PC	Probabilidade de cruzamento
PM	Probabilidade de mutação
P	Tamanho da população
ST	Seleção por torneio de ST soluções
G	Quantidade de gerações
MS	Melhor solução (aptidão)
T	Tempo de execução da melhor solução (segundos)
C	Quantidade de clusters da melhor solução
MMS	Média das melhores soluções nas execuções

Tabela 2: legenda de técnicas e parâmetros citados nas Tabelas 3 e 4.

	Configurações para experimentos							
Cod	**BL**	**RC**	**PP**	**PC**	**PM**	**P**	**ST**	**G**
1				0,8	0,05	100	2	100
2			X	0,8	0,05	100	2	100
3		X		0,8	0,05	100	2	100
4		X	X	0,8	0,05	100	2	100
5	X			0,8	0,05	100	2	100
6	X		X	0,8	0,05	100	2	100
7	X	X		0,8	0,05	100	2	100
8	X	X	X	0,8	0,05	100	2	100

Tabela 3: parâmetros utilizados na execução do AE.

A Tabela 4 apresenta uma relação com as melhores soluções obtidas para cada grafo em cada configuração. A relação contém o valor da aptidão da solução, a quantidade de clusters formados e o tempo gasto na execução do algoritmo em segundos.

Adicionalmente com o objetivo de avaliar as técnicas de busca local, reconexão de caminhos e particionamento da população foram considerados as melhores aptidões e tempo de execução de cada melhor solução com e sem a aplicação de cada uma destes módulos. Por exemplo, as configurações 1 e 5, 2 e 6, 3 e 7 e as configurações 4 e 8 se diferenciam apenas pela aplicação de busca local. Assim o tempo de execução e a aptidão serão comparados entre essas configurações.

		Grafos											
		1				**2**				**3**			
		MS	**MMS**	**C**	**T**	**MS**	**MMS**	**C**	**T**	**MS**	**MMS**	**C**	**T**
Configuração	**1**	0.759	0.754	73	1	0.689	0.673	171	3	0.659	0.650	171	16
	2	0.766	0.761	74	1	0.695	0.684	169	4	0.680	0.671	174	16
	3	0.793	0.777	66	1	0.692	0.684	164	12	0.675	0.663	156	21
	4	0.800	0.790	72	1	0.715	0.704	165	12	0.682	0.675	168	20
	5	0.869	0.853	55	1	0.915	0.902	106	6	0.965	0.955	106	22
	6	0.867	0.865	54	1	0.902	0.897	126	6	0.957	0.952	126	25
	7	0.872	0.860	58	2	0.922	0.906	97	10	0.966	0.955	121	42
	8	***0.877***	0.868	56	2	***0.923***	0.905	*85*	*9*	***0.966***	0.958	85	48

Tabela 4: relação dos melhores resultados obtidos para cada grafo.

O particionamento da população favoreceu o acréscimo da aptidão da melhor solução em todas as configurações, exceto quando foi aplicada em conjunto à busca local. Neste caso, para todos os grafos a melhor solução foi pior que a solução encontrada sem a aplicação do particionamento da população e, além disso, o tempo de execução se manteve ou aumentou.

A reconexão de caminhos melhorou a solução em todos os grafos para todas as configurações que foi utilizada, porém com acréscimo no tempo de execução. Já a busca local foi responsável pela grande melhoria na qualidade das soluções em todas as configurações, obtendo destaque perante as demais configurações mesmo com aumento no tempo de execução.

Os resultados indicam que os melhores resultados em todos os gráficos foram obtidos com a configuração 8, que utiliza buscal local, reconexão de caminhos e particionamento da população. A utilização das três técnicas, entretanto, causa um aumento no tempo de execução. Ainda com base na Tabela 4 é possível observar que a configuração 5, que utiliza apenas a busca local, obteve resultados significativos em relação a aptidão das soluções e principalmente em relação ao tempo de execução, o que pode tornar esta configuração a mais adequada.

REFERÊNCIAS

[1] BASTOS, L. O.; OCHI L.S.; MACAMBIRA, E. M. (2005) GRASP with Path Relinking for the SONET Ring Problems. 5th International Conference on Hybrid Intelligent Systems (HIS2005). Proc. of the 5th HIS2005 in cooperation with IEEE Computational Intelligence Society, p. 6-9.

[2] BERKHIN, P. (2002). Survey of Clustering Data Mining Techniques. Accrue Software.

[3] COLE, R. M. (1998) Clustering with Genetic Algorithms. Master's thesis, Department of Computer Science, University of Western Australia.

[4] DIAS, C. R.; OCHI, L. S. (2003).Efficient Evolutionary Algorithms for the Clustering Problem in Directed Graphs, Proceedings of the 2003 IEEE Congress on Evolutionary Computation. v.1, pp.983–988.

[5] DOVAL, D., MANCORIDIS, S. and MITCHELL, B. S. (1999) Automatic Clustering of Software Systems using a Genetic Algorithm. Proc. of the Int. Conf. on Software Tools and Engineering Practice, pp. 73-81.

[6] GLOVER, F. (1996) Tabu search and adaptive memory programing: advances, applications and challenges. Interfaces in Computer Science and Operations Research, pp. 1–75.

[7] GLOVER, F.; LAGUNA, M.; MART, R.(2000). Fundamentals of scatter search and path-relinking. Control Cybernetics, pp. 653–684.

[8] HARTUV, E.; SHAMIR, R. (1999). A Clustering Algorithm based on Graph Connectivity. Technical Report, Tel Aviv University, Dept. of Computer Science.

[9] NORONHA, T.F.; RESENDE, M.G.C.; RIBEIRO, C.C. (2007) A random-keys genetic algorithm for routing and wavelength assignment. Seventh Metaheuristics International Conference, Montréal, Canadá.

[10] SANTOS, H. G., OCHI, L. S., MARINHO, E. H., DRUMMOND, L. M. A. (2006) Combining an Evolutionary Algorithm with Data Mining to solve a Vehicle Routing Problem. NEUROCOMPUTING Journal - ELSEVIER, volume 70 (1-3), pp. 70-77

[11] TRINDADE, A.R.; OCHI, L.S. (2006) Um algoritmo evolutivo híbrido para a formação de células de manufatura em sistemas de produção. Pesquisa Operacional vol. 26(2), pp. 255-294.