Um Algoritmo Genético Paralelo Aplicado ao Problema de Cobertura de Conjuntos

Francisco Jhonatas M. da Silva, Antonio C. de Oliveira, Rodrigo de M. S. Veras

Departamento de Computação – DC
Universidade Federal do Piauí (UFPI), Teresina – PI, Brasil

jhonatasintelectus@yahoo.com.br, costa@ufpi.edu.br, rveras@ufpi.br

Abstract. *The set covering problem (SCP) is one of the most important problems in combinatorial optimization. The aim of this paper is to show the application of a Parallel Genetic Algorithm to the SCP. The parallelization of the genetic algorithm was based on the island model with migration unilateral. The preliminary computational results show that the proposed algorithm produces good quality solutions at a reduced computational time.*

Resumo. *O problema de cobertura de conjuntos (PCC) é um dos problemas mais importantes de otimização combinatória. O objetivo desse artigo é mostrar a aplicação de um Algoritmo Genético Paralelo ao PCC. A paralelização do Algoritmo Genético foi baseada no modelo de ilhas com migração unilateral. Os resultados computacionais preliminares mostram que o algoritmo proposto produz soluções de boa qualidade em um reduzido tempo computacional.*

1. Introdução

Existem nas Organizações, na Indústria, na Automação, na Logística e na própria Computação, vários problemas difíceis de otimização combinatória que são do tipo "*NP-Completo"*, ou seja, não é conhecido ainda um algoritmo polinomial para resolvê-los [Garey e Johnsom 1979]. Exemplos desses problemas são: o problema do caixeiro viajante, o problema de roteamento de veículos, o problema de cobertura de conjuntos, etc. O Problema de Cobertura de Conjuntos (PCC) possui importantes aplicações práticas, tais como: alocação de serviços de emergência [Benvenites 1982], recuperação de informações em bancos de dados [Balas 1980a], escalonamento de tripulações [Bartholdi 1981], determinação do número mínimo de policias para patrulhamento [Taylor e Huxley 1989], entre outras.

O PCC é um problema que visa encontrar um subconjunto de colunas com custo mínimo que faça a cobertura de todas as linhas de uma matriz 0-1 (a_{ij}), onde a matriz (a_{ij}) é formada por *m* (linhas) e *n* (colunas). Diz-se que a linha a_i é coberta pela coluna *j* quando $a_{ij} = 1$. Considere a variável binária x_j associada à coluna *j*, $x_j = 1$ se a coluna *j* (com custo $c_j > 0$) está na solução e $x_j = 0$ caso contrário. A modelagem do PCC é apresentada nas Equações 1-3.

Minimizar
$$Z = \sum_{j=1}^{n} c_j x_j \qquad (1)$$

Sujeito a: $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \; i = 1,\ldots, m \qquad (2)$$

$$x_j \in \{0,1\}, \;\; j = 1,\ldots, n \qquad (3)$$

A restrição 2 garante que todas as linhas *m* serão cobertas por pelo menos uma coluna e, a restrição da 3 é uma restrição de integrabilidade 0-1.

Na literatura encontram-se vários trabalhos explorando a utilização de algoritmos heurísticos aplicados ao PCC. Chvàtal (1979) utiliza algoritmos gulosos (*greedy*). Balas e Ho (1980b) utilizaram algoritmos gulosos com novas funções gulosas. Vasko e Wilson (1984) também utilizaram algoritmos gulosos com novas funções gulosas e apresentaram um procedimento de remoção de colunas redundantes da solução viável. Beasley (1990a) propôs um algoritmo baseado em heurísticas lagrangeanas. Beasley e Chu (1996) desenvolveram algoritmos fundamentados em algoritmos genéticos. Constantino *et al* (2003) utilizou algoritmos genéticos com uma nova abordagem da representação das soluções utilizando listas encadeadas e novos operadores genéticos. Solar *et al* (2002) apresentou um algoritmo genético paralelo baseado no modelo mestre-escravo.

Este artigo apresenta um Algoritmo Genético Paralelo (AGP) baseado no modelo de ilhas aplicado ao PCC, estando organizado da seguinte maneira: na próxima seção apresentamos os conceitos de Algoritmos Genéticos e descrevemos o Algoritmo Genético Seqüencial (AGS) desenvolvido; na seção 3 apresentamos os conceitos e fundamentos dos Algoritmos Genéticos Paralelos e mostramos o AGP desenvolvido; na seção 4 são apresentados os resultados computacionais e a análise de desempenho do AGP; na seção 5 são mostrados as conclusões e os trabalhos futuros.

2. Algoritmos Genéticos

Os fundamentos teóricos dos Algoritmos Genéticos (AG) foram apresentados por John Holland no final da década de 60 [Holland 1975]. Os AGs são uma poderosa ferramenta para a resolução de muitos problemas difíceis de otimização combinatória.

Algoritmos Genéticos são técnicas estocásticas de busca, inspiradas no processo de evolução natural. Diferente dos algoritmos de busca local - busca tabu e *simulated annealing* - que são baseados na manipulação de uma solução viável, um algoritmo genético inicia com um conjunto de soluções inicial chamada população. Cada indivíduo na população é chamado de cromossomo, representando uma solução do problema. Um cromossomo é usualmente, uma string de símbolos, mas não necessariamente uma string é formada por bits binários. Os cromossomos evoluem através de sucessivas iterações, chamada gerações. Durante cada geração, os cromossomos são avaliados usando uma função de avaliação, também chamada *fitness*. Para criar a próxima geração novos cromossomos chamados prole (*offspring*) são formados por: a) combinando dois cromossomos da geração presente usando um operador de cruzamento(crossover) e/ou b) modificando um cromossomo usando um operador de mutação. Uma nova geração é formada por: a) selecionando de acordo com os valores da função de avaliação alguns dos pais e/ou proles, e b) rejeitando outros de modo a manter o tamanho da população constante. Cromossomos com valores altos para as funções de avaliação possuem probabilidades maiores de serem selecionados (problema de maximização). Depois de diversas gerações, o algoritmo converge para o melhor cromossomo, o qual se espera que represente uma solução ótima ou sub-ótima

para o problema.

2.1. Algoritmo Genético Aplicado ao PCC

As características do AG implementado é mostrado na Tabela 1.

Tabela 1. Características do AG

População inicial	Guloso randomizado
Codificação do cromossomo	*String* binária de tamanho *n* (número de colunas)
Seleção dos pais	Torneio estocástico
Crossover	*Crossover* de um ponto
Mutação	Mutação variável
Método de substituição da população	Substituição total
Tamanho da população	300
Critério de parada	2.500 gerações

2.1.1 Operador de Reparação

Após o processo de cruzamento e mutação o indivíduo (solução) gerado pode violar as restrições do problema, por exemplo, pode haver na solução linhas não-cobertas. É necessário que se faça uso de um operador de reparação. Este operador verifica se a solução contém todas as linhas cobertas, se isso não for o caso, o operador acrescenta na solução novas colunas (com o menor custo) que vão cobrir todas as linhas da matriz. Este operador além de garantir a viabilidade da solução, ele também retira da solução colunas redundantes de maior custo.

3. Algoritmos Genéticos Paralelos (AGP)

Com o avanço da tecnologia hoje se tem a disposição computadores mais velozes, que podem auxiliar na resolução de diversos problemas difíceis que exigem um grande esforço computacional. Com a computação paralela podem-se utilizar os múltiplos processadores para realizar uma tarefa, dividindo-as em subtarefas e executando-as simultaneamente, com isso, pode-se ganhar melhoria no tempo de processamento e na qualidade de solução.

Os principais de modelos de paralelização dos AGs são: o modelo mestre-escravo, *finely grained* (modelo de vizinhança) e modelo de ilhas [Tanese 1989] [Alba e Troya 1999]. No modelo mestre-escravo, a população é mantida no processo mestre e este fica a cargo de realizar a seleção dos pais, o cruzamento e a mutação, ele também é encarregado de enviar aos processos escravos os indivíduos gerados para serem feitas as suas avaliações. No modelo de vizinhança (*finely grained*), cada indivíduo é colocado em uma célula (processo) e cada um deles só pode interagir (cruzamento) com células vizinhas. No modelo de ilhas dividimos a população em subpopulações que são agrupadas em ilhas (processadores). Em cada ilha é executado um AGS (todos os passos de AG: seleção, cruzamento, etc.). Neste modelo se faz necessário o uso de um novo operador: o operador de migração. Este operador define a taxa de migração e a quantidade de indivíduos que vão ser transferidos para as outras ilhas. Com esse modelo podem-se explorar diferentes regiões do espaço de busca.

3.1 AGP Desenvolvido

Dentre os vários modelos de paralelização, escolheu-se o modelo de ilhas com migração

unilateral. Este modelo foi o escolhido porque é o que tem apresentado melhores resultados no que diz respeito ao tempo computacional e a qualidade da solução. Foram utilizadas neste modelo quatro ilhas (processos). O período de migração foi a cada 250 gerações e a quantidade de indivíduos migrados foi de 4% do tamanho da população de cada ilha.

Utilizou-se o MPI (*Message Passing Interface*) para a implementação paralela. De acordo com Pacheco (1999), o MPI é um padrão para comunicação de dados em computação paralela. O MPI possui funções que permitem a comunicação (envio e recebimento de mensagens) entre todos os processos envolvidos na paralelização.

4. Resultados Computacionais

Os algoritmos (AGS e AGP) foram codificados na linguagem de programação C e executados em um computador com Processador Core2 Quad 2.66GHz e 2GB de Memória RAM. Os algoritmos implementados (AGS e AGP) foram testados com arquivos obtidos da OR-Library [Beasley 1990b] e foram executados 10 vezes para cada problema. Os detalhes dos problemas testes na Tabela 2.

Tabela 2. Detalhes dos problemas testes

Conjunto de problemas	Linhas (*m*)	Colunas (*n*)	Número de problemas
4	200	1000	10
5	200	2000	10
6	200	1000	5
A	300	3000	5
B	300	3000	5
C	400	4000	5
D	400	4000	5
E	500	5000	5
F	500	5000	5
G	1000	10000	5
H	1000	10000	5

É apresentado na Tabela 3, para cada um dos algoritmos desenvolvidos (AGS e AGP), a melhor solução encontrada nos 10 testes realizados, o desvio σ médio e a média do tempo de execução. O desvio σ médio é calculado da seguinte forma:

$$\sigma = \sum_{i=1}^{10} (S_{Ti} - S_0)/(10S_0) * 100\% , \quad (4)$$

onde S_{Ti} é o valor da melhor solução encontrada no teste i, S_0 é o valor da solução ótima ou melhor solução conhecida. Analisando a Tabela 3 é possível observar que o AGP conseguiu produzir em 2 casos soluções melhores que o AGS (sendo que em 1 desses casos, o AGP conseguiu encontrar soluções melhores que as conhecidas) e nos outros casos o AGP igualou-se ao AGS. É possível observar também na Tabela 3 que em todos os problemas o AGP conseguiu reduzir significativamente, ou no mínimo ficarem igual, os desvios σ médios. O desvio σ médio negativo no problema G.4 indica que foi encontrada solução melhor que a conhecida (as quais foram utilizadas como parâmetros para o cálculo do desvio pela equação 4). Observa-se também que em relação ao tempo

de execução, o AGP foi executado em aproximadamente 25% do tempo de execução do AGS. Na tabela 3 encontram-se os resultados para 10 problemas dentre os 56 problemas testados.

Tabela 3. Resultados Computacionais

		AGS			AGP		
Problema	Solução Ótima	Melhor Solução	σ	Tempo de Execução	Melhor Solução	σ	Tempo de Execução
4.3	516	516	0,39	22,72	516	0,00	5,81
5.7	293	294	1,40	44,10	293	0,14	10,09
6.1	138	138	0,80	23,40	138	0,00	5,94
A.3	232	233	1,03	58,61	233	0,69	14,96
B.1	69	69	0,29	59,09	69	0,00	14,98
C.4	219	219	1,19	76,35	219	0,32	19,42
D.5	61	61	0,00	77,67	61	0,00	19,50
E.4	28	28	1,43	99,68	28	0,36	25,09
F.1	14	14	0,00	95,42	14	0,00	26,21
G.4	172	171	-0,41	187,32	170	-0,58	48,08
H.5	55	55	0,73	221,15	55	0,18	49,59

O *speed-up* avalia o ganho de processamento paralelo sobre o sequencial [Alba e Luque 2005]. O cálculo do *speed-up* é realizado pela Equação 5:

$$speed\text{-}up = T_s / T_p \ , \tag{5}$$

onde T_s é o tempo de execução do algoritmo seqüencial e T_p é o tempo de execução do algoritmo paralelo. Analisando a Tabela 3 podemos concluir que o *speed-up* do AGP é igual a 4,03, o que garante uma boa avaliação do AGP, pois o ideal é que o *speed-up* seja o mais próximo possível do valor da quantidade de processadores envolvidos no programa paralelo (nesse caso 4 processadores).

5. Conclusões e Trabalhos Futuros

Foi desenvolvido um Algoritmo Genético Paralelo, baseado no modelo de ilhas com migração unilateral, aplicado ao PCC. O algoritmo conseguir encontrar, na maioria dos problemas, a solução ótima ou a melhor solução conhecida. No problema G.4 o AGP encontrou uma solução melhor do que a melhor solução conhecida. Quanto à avaliação do desempenho do AGP, observando o valor do *speed-up*, conclui-se que o algoritmo desenvolvido teve um ótimo desempenho computacional, ficando próximo do ideal.

Como possíveis trabalhos futuros, com a intenção de melhorar as soluções dos problemas em que o AGP não conseguiu chegar à solução ótima e reduzir ainda mais o tempo computacional e, assim, poder comparar com os trabalhos de Beasley e Chu (1996) e Constantino *et* al (2003) que são os trabalhos que apresentam os melhores resultados utilizando algoritmos genéticos sequencial, podem-se apontar: adaptação do modelo de ilha do AGP para realizar as migrações no sentido bidirecional ou multidirecional; utilização de comunicação assíncrona para a troca de mensagens entre os processos; adaptação do AGP para outros tipos de paralelismo, como o modelo mestre-escravo ou modelos híbridos.

Referências

Alba, E. e Troya, M. J. M. (1999). A survey of parallel distributed genetic algorithms. *Complexity*, 4(4), p.31-52.

Alba, E. e Luque, G. (2005). Measuring the Performance of Parallel Metaheuristics. In *Parallel Metaheuristics: A new Class of Algorithms*, John Wiley & Sns, Inc., Hoboken, New Jersey, p.43-62.

Balas, E. (1980a). Cutting Planes from Conditional Bounds: A New Approach to Set Covering. *Mathematical Programming*, v. 12, p. 19-36.

Balas, E. e Ho, A. (1980b). Set covering algorithm using cutting planes, heuristics, and subgradient optimization: a computational study. *Mathematical Programming*, 12, p. 37-60.

Bartholdi, J. (1981). A Guaranteed-Accuracy Round-off Algorithm for Cyclic Scheduling and Set Covering, *Operations Research*, 29, p.501-510.

Beasley, J. E. (1990a). A lagrangian heuristic for set-covering problem. *Naval Research Logistic*, 37, p.151-164.

Beasley, J. E. (1990b). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41, p.1990-1072.

Beasley, J. E. e Chu, P. C. (1996). A genetic algorithm for the ser covering problem. *European Journal of Operational Research*, 94, p.392-404.

Benveniste, R. (1982). A Note on the Set Covering Problem. *JORS* 33, p. 261-265.

Chvàtal, V. (1979). A greed heuristic for the set covering problem. *Management Science*, 21, p.591-599.

Constantino, A. A., Reis, P. A., *et al*. (2003). Aplicação de Algoritmos Genéticos ao Problema de Cobertura de Conjunto. *Simpósio Brasileiro de Pesquisa Operacional*.

Garey, M.R.; Johnson. D.S. (1979). *Computers and intractability:* a guide to the theory of NP-completeness. San Francisco, Freeman.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA.

Pacheco, S. P. (1999) A User's Guide to MPI. Disponível em: ftp://math.usfca.edu/pub/MPI/mpi.guide.ps. Acesso em 23 de janeiro de 2013.

Solar, M., Parada, V., Urrutia, R. (2002). A parallel genetic algorithm to solve the set covering problem. *Computers and Operations Research*, v.29, p.1221-1235.

Tanese, R. (1989). Distributed Genetic Algorithms. *Proceedings of the third International Conference on Genetic Algorithms*. Schaffer J. D. Editors, Morgan Kaufmann Publishers, p.434-439.

Taylor, P. e Huxley, S. (1989). A Break from Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support Tool. *Interfaces*, v. 45, p.4-24.

Vasko, F. J. e Wilson, G. R. (1984). An Efficient Heuristic for Large Set Covering Problems. *Naval Research Logistic Quarterly*, 31, p.163-171.