APLICAÇÃO DE ALGORITMOS GENÉTICOS AO PROBLEMA DE COBERTURA DE CONJUNTO

Ademir Aparecido Constantino
Universidade Estadual de Maringá - Departamento de Informática
Av. Colombo, 5790, CEP 87020-290, Maringá-PR
e-mail: ademir@din.uem.br

Paulo Alexandre dos Reis
Universidade Estadual de Maringá - Departamento de Informática
e-mail: pareis@din.uem.br

Candido Ferreira Xavier de Mendonça Neto
Universidade Estadual de Maringá - Departamento de Informática
e-mail: xavier@din.uem.br

Mauricio Fernandes Figueiredo
Universidade Estadual de Maringá - Departamento de Informática
e-mail: mauricio@din.uem.br

Resumo
O Problema de Cobertura de Conjuntos (PCC) tem um importante papel dentro da Pesquisa Operacional e surge em muitos problemas práticos. O objetivo deste artigo é apresentar um algoritmo heurístico baseado em Algoritmos Genéticos para a resolução do PCC. O algoritmo é inicializado com uma população gerada por um algoritmo guloso randomizado. Um novo operador de cruzamento e um novo operador de mutação adaptativo foram incorporados ao algoritmo para intensificar a busca. O algoritmo foi testado com uma classe de problemas de custos não unitários da OR-Library, sem aplicar técnicas de redução do problema, o qual apresentou resultados melhores que os encontrados na literatura.
Palavras chaves: cobertura de conjunto, heurística, Algoritmos Genéticos

Abstract
The Set Covering Problem (SPC) plays an important role in Operational Research since it can be found as part of several real world problems. In this work we report the use of a genetic algorithm to solve SCP. The algorithm starts with a population chosen by a randomized greedy algorithm. A new crossover operator and a new adaptive mutation operator were incorporated into the algorithm to intensify the search. Our algorithm was tested for a class of non-unicost SPC obtained from OR-Library without applying reduction techniques. The algorithms found solutions better than the ones found in the literature.
Keywords: set covering, heuristic, Genetics Algorithms

1 O Problema de Cobertura de Conjuntos (PCC)

O Problema de Cobertura de Conjuntos (PCC) - ou *Set Covering Problem* - é um problema de otimização definido como segue. Denote-se $I = \{1,2, ..., m\}$ e $J=\{1, 2, ..., n\}$ conjuntos de índices. Seja $\Gamma=\{ P_j \subseteq I,\ j \in J\}$, tal que, $I = \bigcup_{j \in J} P_j$. Seja c_j é um custo associado ao conjunto P_j. Define-se de *cobertura* um subconjunto $J^* \subseteq J$ tal que $I = \bigcup_{j \in J^*} P_j$ com custo $C^* = \sum_{j \in J^*} c_j$. O PCC consiste em encontrar

uma cobertura de custo mínimo. Tal problema pode ser formulado como um problema de programação matemática da seguinte forma:

$$\text{Minimizar} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{Sujeito a} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1, 2, ..., m \tag{2}$$

$$x_j \in \{0,1\} \qquad j = 1, 2, ..., n \tag{3}$$

Onde A é uma matriz-(0,1) $m \times n$, tal que, $a_{ij}=1$ se e somente se $i \in P_j$ e $a_{ij}=0$ em caso contrário. A variável x_j é a variável de decisão do problema que indica se a coluna j foi selecionada ($x_j = 1$) ou não ($x_j = 0$). Note que I e J representam os conjuntos de índices de linhas e colunas, respectivamente, da matriz A. Além disso, P_j representa o vetor da coluna j da matriz A.

É conhecido que o problema de decisão do PCC é da classe NP-completo (Garey e Johnson, 1979). Várias aplicações de PCC podem ser encontradas na literatura, tais como: escalonamento de condutores de empresa de transporte (Desrochers e Soumis, 1989), locação de unidades emergenciais (Toregas *et al*, 1971), balanceamento em linha de montagem (Salveson, 1955), entre outras. Além da complexidade teórica, nessas aplicações, é muito comum que o problema tenha centenas de linhas e milhares de colunas. Portanto, por essas características é perfeitamente recomendado o uso de algoritmos heurísticos para a resolução do PCC. Na literatura podem ser encontrados vários trabalhos explorando algoritmos heurísticos para a solução do problema de cobertura de conjunto. Chvàtal (1979) foi um dos primeiros trabalhos utilizando algoritmos gulosos (*greedy*). Balas e Ho (1980) também propuseram um algoritmo guloso baseado em cinco novas funções gulosas. Posteriormente, surgiu o trabalho de Vasco e Wilson (1984) que explorou as heurísticas de Chvàtal e Balas e Ho, adicionando novas funções gulosas e um procedimento de remoção de colunas redundantes da solução viável. Feo e Resende (1989) propuseram uma variação não-determinística da heurística de Chvàtal conjugando com busca local. Beasley (1990a) e Haddadi (1997) desenvolveram algoritmos baseados em heurísticas lagrangeanas. Jacobs e Brusco (1993) investigaram a aplicação de *simulated anneling*. Mais recentemente, Bealey e Chu (1996) e Al-Sultan *et al* (1996) desenvolveram algoritmos baseados em Algoritmos Genéticos, dos quais o primeiro foi o que apresentou os melhores resultados entre todos trabalhos encontrados na literatura. Neste trabalho, foi desenvolvido um algoritmo heurístico baseado em Algoritmos Genéticos que será apresentado em detalhes na seção seguinte.

2 O algoritmo

A seguir serão apresentadas as implementações dos operadores genéticos, assim como representação computacional da população e dos cromossomos.

2.1 Função de aptidão

Em algoritmos genéticos é comum considerar a medida de aptidão de um indivíduo como sendo um valor positivo. Todos as instâncias do problema utilizadas neste trabalho possuem somente custos positivos. Assim, a função de aptidão deste trabalho corresponde exatamente à função objetivo de cada PCC.

2.2 Esquema de Codificação

Uma forma natural e bastante utilizada na literatura é a codificação da estrutura (solução) na forma de um vetor binário, cujo tamanho é igual ao número de colunas da matriz de coeficientes do PCC (Lopes, 1995; Al-Sultan *et al* 1996) onde a posição j assume valor 1 se a coluna j fazer parte da solução, e 0 em caso contrário. Essa codificação é ineficiente considerando que o número de colunas de uma solução raramente ultrapassa a metade do número total de colunas da matriz de coeficientes.

Dessa forma, optou-se por utilizar uma codificação na forma de conjunto *cobertura* $S \subseteq J$ contendo os índices das colunas pertencentes à solução, cuja codificação computacional foi através de lista dinâmica.

2.3 Estrutura de dados

Na literatura e nas aplicações práticas é bastante comum encontrar problemas de cobertura de conjunto de baixa densidade (matriz esparsa). Dessa forma, optou-se por uma codificação compacta e mais eficiente através de um vetor $M=[m_j]$, onde $m_j=P_j$, $j \in J$. Neste caso, a implementação computacional de m_j também foi através de lista dinâmica.

Quanto à representação da população, também, utilizou-se de uma implementação em lista dinâmica contendo todas as soluções (estruturas) em ordem decrescente de seu custo (aptidão). Aqui a população será denotada por um conjunto $Pop=\{S_l \subseteq J, l=1,2, ..., Npop\}$, onde $Npop=|Pop|$.

2.4 População Inicial.

A população inicial na maioria dos casos de aplicação de algoritmos genéticos é gerada de maneira totalmente aleatoria. Neste trabalho, foi utilizado um algoritmo de construção aleatorizado, baseado em um algoritmo guloso (*greedy*), para gerar cada estrutura da população. Dessa forma, a população gerada será constituída somente de soluções viáveis. Experiências com esse processo demonstraram melhores resultados ao invés de gerar a cada indivíduo aleatoriamente. O algoritmo para gerar a população inicial é formalmente descrito a seguir:

α_i = o conjunto de colunas que cobrem a linha $i, i \in I$;

β_j = o conjunto de linhas cobertas pelo coluna j, $j \in J$;

Função GerarIndividuo;

$S \leftarrow \phi$; // S é o conjunto de colunas na solução
$U \leftarrow I$; // U é o conjunto de linhas não cobertas
$w_i \leftarrow 0, \forall i \in I$; // w_i é o número de colunas em S que cobrem a linha i, $i \in S$
Enquanto ($U \neq \phi$) **faça**
    Selecione aleatoriamente uma linha $i \in U$;
    Selecione uma coluna $j \in \alpha_i$ que minimize $c_j / \left| U \cap \beta_j \right|$;
    $S \leftarrow S \cup \{j\}$;
    $w_i \leftarrow w_i + 1, \forall i \in \beta_j$;
    $U \leftarrow U - \beta_j$;
Retorne S.

Ao final da construção, na solução encontrada pode haver colunas redundantes, cuja remoção não afeta a viabilidade da solução. O procedimento de eliminação de redundâncias é dado a seguir.

Função EliminarRedundancia(S);

$T \leftarrow S$;
Enquanto ($T \neq \phi$) **faça**
    Selecione aleatoriamente uma coluna j, $j \in T$;
    $T \leftarrow T - \{j\}$;
    Se ($w_i \geq 2, \forall i \in \beta_j$) **então**

$S \leftarrow S - \{j\}$;
$w_i \leftarrow w_i - 1, \forall i \in \beta_j$;
Retorne S.

Ao final desse processo teremos um indivíduo (solução) que deve ser avaliado e inserido na população inicial. A construção da população inicial consiste na repetição desses passos até terem sido gerados *NPop* indivíduos.

Função GerarPopulacaoInicial;

$Pop \leftarrow \phi$; // população do algoritmo genético
$t \leftarrow 0$; // número de indivíduos na população
Repetir
$S \leftarrow$ GerarIndividuo;
$S \leftarrow$ EliminarRedundancia(S);
$Pop \leftarrow Pop \cup \{S\}$ mantendo a ordenação dos indivíduos;
$t \leftarrow t + 1$;
até ($t > NPop$).

2.5 Seleção

O princípio de seleção dos indivíduos para participarem do cruzamento é baseado no método de classificação (*ranking*). Dada a população com *Npop* indivíduos, a probabilidade de seleção de um indivíduo para participar do cruzamento é proporcional à sua posição relativa na população, ou seja, a probabilidade de seleção de um cromossomo $S_l \in Pop$ é dada por:

$$p(S_l) = \frac{l}{\sum_{k=1}^{Npop} k} = \frac{2l}{Npop(Npop+1)} \quad (4)$$

Dessa forma, o indivíduo menos apto (o primeiro da população) terá a menor probabilidade de ser selecionado. Observe que o valor da equação 4 é muito simples e rápido de ser calculado e não necessita ser atualizado a cada vez que a população for modificada (supondo um tamanho fixo da população) como é feito no método da roleta.

2.6 Cruzamento

Esse operador consiste em selecionar dois indivíduos, unindo as estruturas de ambos, criando uma terceira estrutura (solução). Então, o algoritmo de eliminação de redundância é aplicado. Assim, gera-se um indivíduo filho.

Função Cruzamento;

$X \leftarrow$ Selecionar; // Selecionar o primeiro indivíduo pai
$Y \leftarrow$ Selecionar; // Selecionar o segundo indivíduo pai
$Z \leftarrow X \cup Y$;
$Z \leftarrow$ EliminarRedundancia(Z);
Retorne Z. // novo indivíduo

2.7 Mutação Variável

O objetivo é implementar um operador de mutação com taxa variável para evitar estagnação da população de soluções, ou seja, que todas as soluções sejam iguais e, com isso, intensificar a busca pela melhor solução.

O objetivo na mutação é dar a oportunidade de inserir características não presentes nas estruturas pertencentes à população e, também, evitar a convergência prematura da população de soluções. O procedimento de mutação consiste em selecionar um número, proporcional ao tamanho da solução, de colunas de J e inseri-las na solução. Posteriormente, o algoritmo de eliminação de redundância é aplicado. Tal procedimento é resumido abaixo.

Função Mutação(S);

Selecione aleatoriamente λ, $\lambda \in [0,1]$;
Para $k \leftarrow 1$ to $\lambda * |S|$ **faça**
    Selecione uma coluna j, $j \in J$;
    $S \leftarrow S \cup \{j\}$;
$S \leftarrow$ EliminarRedundancia(S);
Retorne S.

Similarmente ao que ocorre no processo natural, a mutação não ocorre em todos os indivíduos, mas existe uma probabilidade de ocorrer chamada de *taxa de mutação* (*TxM*). Tradicionalmente, é utilizada nos algoritmos genéticos uma taxa de mutação constante, mas aqui foi proposto o uso de uma taxa de mutação variável que aumente de acordo com a convergência da população. Assim, utilizou-se uma taxa de mutação variável calculada da seguinte forma:

$$TxM(t) = \frac{TxMinM}{1 - e^{\left(-(c_1(t) - c_{NPop}(t))/c_1(t)\right)}} \tag{5}$$

Onde:

$TxMinM$ = a taxa mínima de mutação (parâmetro);
$c_1(t)$ = o custo do indivíduo menos apto da população na iteração (ou geração) t;
$c_{Npop}(t)$ = o custo do indivíduo mais apto da população na iteração t.

Como se pode perceber, a taxa de mutação cresce conforme o custo do indivíduo menos apto na população se aproxima do custo do melhor indivíduo fazendo com que as mutações sejam mais freqüentes, possibilitando uma intensificação da busca.

2.8 Atualização da População

De acordo com os operadores de cruzamento e mutação apresentados, a população sempre será constituída de soluções viáveis. Cada novo indivíduo gerado é inserido ou não na população de acordo com o seu custo, se este for melhor que o custo do indivíduo menos apto, então, ele entra na população saindo o menos apto, caso contrário, ele é descartado. Dessa forma, o algoritmo genético desenvolvido é baseado na abordagem *steady-state*, pois cada novo indivíduo gerado pode imediatamente substituir um outro indivíduo da população atual. Ao contrário da abordagem geracional na qual a população inteira pode ser substituída por uma nova população.

2.9 O algoritmo genético desenvolvido

Após ter sido feita uma descrição dos operadores principais do algoritmo genético, esta seção apresenta o algoritmo que resume todo o processo. Além dos parâmetros *NPop*, *TxMinM*, o algoritmo também utiliza um outro parâmetro denominado de *NMax*, que estabelece um critério de parada baseado no número máximo de iterações sem que a população tenha sofrido alterações. Nos

experimentos realizados foram utilizados os seguintes valores: *NPop*= 500, *TxMinM*= 5%, $NMax = 1000$.

Função AlgoritmoGenetico;

GerarPopulacaoInicial;
$t \leftarrow 0$;
Repetir
  $S \leftarrow$ Cruzamento;
  Escolha aleatoriamente um $\rho, \rho \in [0,1]$;
  Se ($\rho <$ TxM(t)) **então** $S \leftarrow$ Mutacao(S);
  Faça R receber a solução com o maior custo em *Pop*;
  Se (Avaliar(S)<Avaliar(R)) **então**
    $Pop \leftarrow Pop - \{R\}$;
    $Pop \leftarrow Pop \cup \{S\}$; // mantendo a ordenação da população
  Se (foram feitas modificações na população)
    então $t \leftarrow 0$;
    senão $t \leftarrow t + 1$;
Até ($t > NMax$).

3 Resultados Computacionais

O algoritmo apresentado neste artigo foi codificado em Pascal e testado em um Pentium III 900 MHz. Os testes foram conduzidos sobre com uma coleção de dados (65 problemas) de vários tamanhos e densidades obtidos da OR-Library (Beasley, 1990b). Os resultados dos testes estão resumidos na Tabela 1. Para cada problema foram executados 10 testes. Para cada teste a tabela informa o custo da função objetivo do melhor indivíduo da população (célula com '-' significa que o custo da solução foi igual ao da coluna "Melhor Solução"). As colunas AG_B e AG_P informam os desvios σ médios do algoritmo de Beasley e Chu (1996) e do proposto por este trabalho, respectivamente. O desvio σ médio é calculado da seguinte forma: $\sigma = \sum_{i:=1}^{10} (S_i - S_o)/(10S_o)$, onde S_i é o valor da solução do *i*-ésimo teste e S_o é o valor da melhor solução conhecida segundo Beasley e Chu(1996). Salientando que, para os problemas de 4 a 6 e A a D os valores da solução ótima são conhecidos, enquanto que para os problemas de E a H o valor ótimo ainda é desconhecido.

Tabela 1. Resultados Computacionais.

Problema	A melhor solução de cada teste										Melhor	AG_B	AG_P	Tempo de
	1	2	3	4	5	6	7	8	9	10	Solução[a]	σ	σ	execução[b]
4.1	-	-	-	-	-	-	-	-	-	-	429	0,16	0,00	18
4.2	-	-	-	-	-	-	-	-	-	-	512	0,00	0,00	16
4.3	-	-	-	-	-	-	-	-	-	-	516	0,00	0,00	21
4.4	495	495	495	495	495	495	-	495	495	495	494	0,00	0,18	22
4.5	-	-	-	-	-	-	-	-	-	-	512	0,00	0,00	16
4.6	-	-	-	-	-	-	-	-	-	-	560	0,00	0,00	21
4.7	-	-	-	-	-	-	-	-	-	-	430	0,05	0,00	15
4.8	-	-	-	-	-	-	-	-	-	-	492	0,02	0,00	17
4.9	-	-	-	-	-	-	-	-	-	-	641	0,33	0,00	17
4.10	-	-	-	-	-	-	-	-	-	-	514	0,00	0,00	16
5.1	-	-	-	-	-	-	-	-	-	-	253	0,00	0,00	19
5.2	-	-	-	-	-	-	-	-	-	-	302	0,50	0,00	20
5.3	-	-	-	-	-	-	-	-	-	-	226	0,88	0,00	20
5.4	-	-	-	-	-	-	-	-	-	243	242	0,04	0,04	25
5.5	-	-	-	-	-	-	-	-	-	-	211	0,00	0,00	21

Tabela 1. Continuação

Problema	A melhor solução de cada teste										Melhor	AG_B	AG_P	Tempo de
	1	2	3	4	5	6	7	8	9	10	Solução[a]	σ	σ	execução[b]
5.6	-	-	-	-	-	-	-	-	-	-	213	0,00	0,00	19
5.7	-	-	-	-	-	-	-	-	-	-	293	0,00	0,00	21
5.8	-	-	-	-	-	-	-	-	-	-	288	0,28	0,00	26
5.9	-	-	-	-	-	-	-	-	-	-	279	0,00	0,00	18
5.10	-	-	-	-	-	-	-	-	-	-	265	0,00	0,00	19
6.1	-	-	-	-	-	-	-	-	-	-	138	0,00	0,00	9
6.2	-	-	-	-	-	-	-	-	-	-	146	0,14	0,00	10
6.3	-	-	-	-	-	-	-	-	-	-	145	0,00	0,00	9
6.4	-	-	-	-	-	-	-	-	-	-	131	0,00	0,00	11
6.5	-	-	-	-	-	-	-	-	-	-	161	0,19	0,00	10
A.1	255	254	254	254	254	254	254	254	254	254	253	0,12	0,43	36
A.2	-	-	-	-	-	-	-	-	-	-	252	0,00	0,00	39
A.3	-	233	233	233	-	233	233	233	233	233	232	0,22	0,34	41
A.4	235	-	-	-	-	-	-	-	-	-	234	0,00	0,04	35
A.5	237	237	237	237	237	237	237	237	237	237	236	0,00	0,42	28
B.1	-	-	-	-	-	-	-	-	-	-	69	0,00	0,00	30
B.2	-	-	-	-	-	-	-	-	-	-	76	0,00	0,00	31
B.3	-	-	-	-	-	-	-	-	-	-	80	0,00	0,00	27
B.4	-	-	-	-	-	-	-	-	-	-	79	0,00	0,00	25
B.5	-	-	-	-	-	-	-	-	-	-	72	0,00	0,00	27
C.1	-	-	-	-	-	229	-	228	228	228	227	1,00	0,22	71
C.2	221	220	222	-	223	220	222	-	223	220	219	0,46	0,87	64
C.3	-	-	-	-	-	-	-	245	-	-	243	1,40	0,08	56
C.4	-	-	-	-	-	-	-	-	-	-	219	0,05	0,00	62
C.5	-	-	-	-	-	-	-	-	-	-	215	0,05	0,00	41
D.1	-	-	-	-	-	-	-	-	-	-	60	0,00	0,00	48
D.2	-	-	-	-	-	-	-	-	-	-	66	0,00	0,00	51
D.3	-	-	-	-	-	-	-	-	73	73	72	0,28	0,28	43
D.4	-	-	-	-	-	-	-	-	-	-	62	0,00	0,00	48
D.5	-	-	-	-	-	-	-	-	-	-	61	0,00	0,00	47
E.1	-	-	-	-	-	-	-	-	-	-	29	0,00	0,00	4
E.2	-	-	-	-	-	-	-	-	-	-	30	0,00	0,00	4
E.3	-	-	-	-	-	-	-	-	-	-	27	2,00	0,00	7
E.4	-	-	-	-	-	-	-	-	-	-	28	2,60	0,00	9
E.5	-	-	-	-	-	-	-	-	-	-	28	0,00	0,00	7
F.1	-	-	-	-	-	-	-	-	-	-	14	0,00	0,00	236
F.2	-	-	-	-	-	-	-	-	-	-	15	0,00	0,00	202
F.3	-	-	-	-	-	-	-	-	15	-	14	0,00	0,71	203
F.4	-	-	-	-	-	-	-	-	-	-	14	0,00	0,00	201
F.5	13	13	13	13	13	13	-	-	-	13	14	-2,14	-5,00	219
G.1	178	176	178	178	-	177	178	178	176	177	179	-0,73	-0,84	152
G.2	155	156	155	156	155	155	156	155	156	156	158	-1,08	-1,58	167
G.3	168	170	168	168	170	170	167	170	-	168	169	-0,65	-0,12	179
G.4	171	170	170	171	170	170	-	-	169	171	172	-0,99	-0,81	162
G.5	-	-	-	169	-	-	169	-	169	-	168	0,83	0,18	165
H.1	-	-	65	-	-	-	-	65	-	65	64	0,00	0,47	244
H.2	-	-	-	-	-	-	-	-	-	-	64	0,00	0,00	219
H.3	61	59	-	61	-	59	-	-	-	59	60	-1,50	-0,17	260
H.4	58	58	58	-	-	-	-	58	-	-	59	-0,17	-0,68	275
H.5	-	-	-	-	-	-	-	-	-	-	55	0,18	0,00	252
											Soma	4,52	-4,92	4.453

[a] Solução ótima ou a melhor solução conhecida obtida de Beasley e Chu (1996) para fins de comparação.
[b] Tempo médio de execução em segundos.

Convém salientar que nestes problemas não foram aplicadas técnicas de redução do problema como sugere Garfinkel e Nemhauser (1972). Tomou-se essa decisão a fim ter o mesmo padrão de comparação com o trabalho de Beasley e Chu (1996).

4 Análise e Conclusão

A soma dos desvios médios de AG_B e AG_P são 4,52 e –4,92, respectivamente. Isso significa que, em média, o algoritmo proposto neste trabalho foi melhor que o algoritmo proposto por Beasley e Chu (1996). Ademais, o tempo total do experimento foi de 4.453 segundos em um computador Pentium III 900MHz, enquanto que o AG_B foi executado em Silicon Graphics (R4000, 100MHz) em 95.572 segundos de tempo total. A comparação de performance entre diferentes arquiteturas de computador não é uma tarefa simples (Dongarra, 1992). Porém, a Silicon Graphics[1] mostra que um computador R4000 100MHz tem performance aproximadamente equivalente ao Pentium 66MHz usando SPEC *Benchmark*. Através de experimentos com o AG_P em computadores com processadores Pentium 66MHz, 166MHz, 233MHz e Pentium III 900MHz, pôde-se concluir que este último é aproximadamente 15 vezes mais rápido que o primeiro. Dessa forma, se considerar essa medida como *Benchmark*, tem-se que 4.453×15= 66.799 segundos é ainda 30% inferior ao tempo consumido por AG_B. Portanto, o algoritmo AG_P mostrou ser ligeiramente superior ao algoritmo AG_B em dois aspectos: na qualidade das soluções e no tempo total de execução.

Referência Bibliográfica.

Al-Sultan, K.S., Hussain, M.F. e Nizami, J.S.. A genetic algorithm for set covering problem. *Journal of the Operational Research Society*, 47, pp. 702-709, 1996.

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

Beasley, J.E. A lagrangian heuristic for set-covering problems. *Naval Research Logistic*, 37, pp. 151-164, 1990a.

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

Beasley, J.E.. e Chu, P.C. A genetic algorithm for the set covering problem. European Journal of Operational Research,, 94, pp. 392-404, 1996.

Desrochers, M., e Soumis, F.,. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23, pp. 1-13, 1989.

Dongarra, J.J.. Performance of various computers using standard linear equations software. *Computer Architecture News*, 20, pp. 22-44, 1992.

Feo, T. A. e Resende, M. G. C.. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8, pp. 67-71, 1989.

Garey, M.R. e Johnson, D.S.. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

Garfinkel, R.S. e Nemhauser, G.L.. *Integer Programming*. New York. John Wiley, 1972.

Ghvàtal, V.. A greedy heuristic for the set covering problem. *Management Science*, 21, pp. 591-599, 1979.

Haddadi, S. Simple lagrangian heuristic for the set covering problem. *European Journal of Operational Research*, 97, pp. 200-204, 1997.

[1] Silicon Graphics Indy. Historical Article, 1993. http://futuretech.mirror.vuurwerk.net/pcw9-93indy.html, Acessado em 30 de maio de 2003.

Jacobs, L. W. e Brusco, J.J.. A simulated annealing-based heuristic for the set-covering problem. *Working paper*, Operations Management and Information System Department, Northern Illinois University, 1993.

Lopes, L. S.. Uma heurística baseada em algoritmos genéticos aplicada ao problema de cobertura de conjunto. *Dissertação de Mestrado em Computação Aplicada – INPE*, 1995.

Salveson, M. E.. The assembly line balancing problem. *Journal of Industrial Engineering*, 6, pp. 18-25, 1955.

Toregas, C., Swain, R., Revelle, C., and Bergman, L.. The location of emergency service facilities. *Operations Research*, 19, pp. 1363-1373, 1971.

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

Agradecimentos:

Ao suporte parcial do CNPq através dos processos n. 304058/2002-5 PP e n. 470470/2001-1.