O problema da cobertura única

Regina Beretta Mazaro **Edna Ayako Hoshino (orientadora)**
Universidade Federal de Mato Grosso do Sul - UFMS
Faculdade de Computação - Facom
79070-900 Campo Grande-MS, Brasil - reginabmre@gmail.com, eah@facom.ufms.br

Resumo

Este artigo introduz uma formulação por programação linear inteira e duas heurísticas GRASP (Greedy Randomized Adaptative Search Procedure) para o problema da cobertura única, que é uma variante do clássico problema da cobertura máxima e que ocorre em aplicações de telefonia celular e redes sem fio. A formulação e as heurísticas propostas são utilizadas para construir um algoritmo exato *branch-and-bound* para o problema. Esta é uma das contribuições do trabalho, uma vez que não se conhece nenhum algoritmo exato proposto na literatura para resolver este problema. Experimentos empíricos foram conduzidos para avaliar o modelo proposto e a qualidade das soluções obtidas pelas heurísticas, assim como para identificar o grau de dificuldade em resolver diferentes classes de instâncias testes do problema.

Palavras Chave. Cobertura única, Cobertura Máxima, Programação Linear Inteira, GRASP, Heurísticas.
Área principal Otimização Combinatória

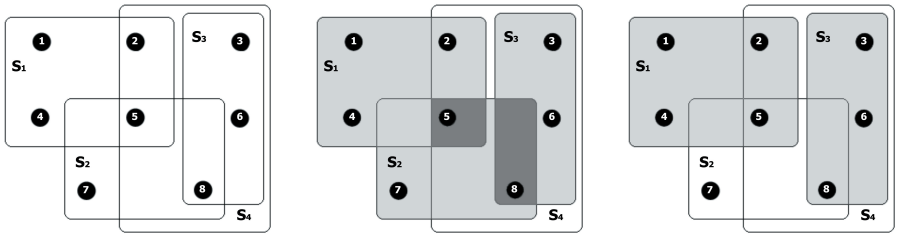
Abstract

This article introduces a formulation by integer linear programming and two GRASP *(Greedy Randomized Adaptative Search Procedure)* heuristics for the unique coverage problem, a variation of the classical set cover problem that occurs in cell phones and wireless networks applications. The formulation and heuristics proposed are used to build an exact *branch-and-bound* algorithm for the problem. This is one of the paper's contributions, since that it isn't known any exact algorithm in literature to solve this problem. Empirical experiments were conducted to evaluate the proposed model and quality of solutions obtained by the heuristics, as to identify the level of difficulty in solving different instance classes for the problem.

Keywords. Unique Coverage, Set Cover, Integer Linear Programming, GRASP, Heuristics.
Main area Combinatorial Optimization.

1 Introdução

O **problema da cobertura única** (*Unique Coverage Problem* - UCP) consiste em, dados um universo U de n elementos e uma coleção S de m subconjuntos de U, encontrar uma subcoleção $S' \subseteq S$ que maximize o número de elementos cobertos uma única vez, isto é, maximizar o número de elementos que pertençam a exatamente um conjunto de S'. Na figura 1(a), tem-se $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$ com $n = 8$, e $S = \{S_1, S_2, S_3, S_4\}$, sendo $m = 4$. Uma solução para o problema é mostrada na Figura 1(b), em que os conjuntos S_1, S_2 e S_3 são selecionados. Essa não é uma solução ótima, pois apesar de todos os pontos serem cobertos, 5 e 8 pertencem a mais de um conjunto escolhido, de forma que apenas 6 pontos foram cobertos de maneira única. Já em 1(c), escolhidos apenas os conjuntos S_1 e S_3, encontrou-se uma solução ótima em que 7 pontos aparecem em apenas um dos conjuntos selecionados, e portanto são cobertos uma única vez. Esse resultado é a maximização procurada no problema.



(a) (b) (c)

Figura 1: 1(a) Representação de U e S. 1(b) Uma possível solução. 1(c) A solução ótima.

O problema foi originalmente introduzido por Demaine et al. (2006) e motivado por aplicações em redes sem fio. Numa versão simplificada da aplicação, é dado um conjunto de locais candidatos à instalação de transmissores da rede, cada um sendo capaz de atender um certo conjunto de clientes. O objetivo consiste em maximizar o número de clientes cobertos pela rede. Porém, diferentemente do que ocorre no problema da cobertuma máxima (*Set Covering Problem* - SCP), um cliente só é considerado coberto se estiver dentro do alcance de uma única base. Isso porque o sinal de um cliente coberto por mais de uma base acaba sofrendo interferência e tendo sua qualidade reduzida, o que deve ser evitado. Assim, fica evidente que o UCP é uma variante do já bastante conhecido SCP. Na verdade, a versão abordada por Demaine et al. (2006) contava também com um orçamento total para a construção das bases, com cada uma possuindo um determinado custo de instalação. Essa é uma forma mais generalizada do UCP, conhecida como *Budgeted Unique Coverage Problem* - BUCP, e não será tratada neste artigo. O problema da cobertura única apresenta ainda outras aplicações relacionadas, como no *radio broadcast* e no *Envy-Free Pricing*.

Além disso, o UCP pode ser associado a dois dos 21 problemas NP-completos de Karp (1972): o do corte máximo e o da cobertura máxima. O primeiro, considerando-se cada vértice como um conjunto e cada aresta como um elemento, nada mais é do que um caso particular do problema da cobertura máxima em que todo elemento aparece em exatamente dois conjuntos. Já o segundo é uma generalização do problema em questão, uma vez que não possui a restrição de que um elemento seja unicamente coberto. Assim, é possível que o trabalho focado no problema da cobertura única possa trazer contribuições também ao estudo desses problemas clássicos da Computação.

Quando o problema da cobertura única foi introduzido por Demaine et al. (2006), provou-se a existência de uma aproximação semi-logarítmica para o problema. Mais tarde, Moser et al. (2007) analisam a complexidade da versão parametrizada do UCP, em que cada ponto pode ser coberto até k vezes. Eles sugerem um algoritmo com $O(4^{k^2})$, e deixam em aberto a questão da existência ou não de um bom algoritmo de *branching* para o problema. Misra et al. (2009) apresentam um algoritmo mais eficiente para essa versão parametrizada do problema, com $O(2^{O(kloglogk)}.|F|.k+|F|^2)$, em que F é a coleção de conjuntos disponíveis, fazendo uso de métodos amplamente conhecidos de *Color-Coding*. Porém, não se conhece a existência de nenhum trabalho que trate o UCP original de forma exata.

O objetivo deste trabalho é apresentar um algoritmo exato para o problema da cobertura única. Para tanto, são propostas uma formulação de programação linear inteira (PLI) para o problema e algumas heurísticas *Greedy Randomized Adaptative Search Procedure* (GRASP). Inicialmente, fez-se uso de um resolvedor comercial de PLI para implementar um algoritmo exato *branch-and-bound* (BB) usando o modelo proposto. As heurísticas foram comparadas e usadas dentro de cada nó da árvore de enumeração do algoritmo BB proposto, para obter limitantes inferiores mais justos e melhorar o desempenho do algoritmo. Experimentos empíricos foram realizados em um conjunto expressivo de instâncias testes. A análise dos resultados permitiu obter algumas conclusões acerca do tema, como a qualidade das soluções obtidas pelas heurísticas e especialmente a respeito das instâncias difíceis.

Ao longo deste artigo, a Seção 2 traz uma descrição mais detalhada do problema, assim como o modelo de PLI proposto. Os algoritmos das heurísticas são detalhados na Seção 3, e os resultados obtidos podem ser conferidos na Seção 4, juntamente com uma análise dos mesmos. Finalizando, as conclusões são apresentadas na Seção 5.

2 Descrição do Problema

Como já exposto na Seção 1, o problema da cobertura única consiste em determinar quais conjuntos selecionar de uma dada coleção S de modo a maximizar o número de elementos pertencentes a exatamente um dos conjuntos selecionados.

Considere $|U| = n$, $|S| = m$ e constantes a_{ij} para representar os dados da instância, em que $a_{ij} = 1$ se e somente se o elemento i é coberto pelo conjunto j. O modelo de PLI para o UCP considera variáveis x, y e z, tais que $x_j = 1$ se e somente se o conjunto j é escolhido enquanto z_i representa quantas vezes o elemento i é coberto pelos conjuntos escolhidos. Já a variável $y_i = 1$ se e somente se o elemento i é coberto de forma única.

Com as estas definições, tem-se a seguinte formulação de PLI para o UCP:

$$
\begin{aligned}
&\text{(F)} \quad && \max \sum_{i=1}^{n} y_i && && (1)\\
&\text{sujeito a} && \sum_{j=1}^{m} a_{ij} x_j = z_i && , \forall i, 1 \leq i \leq n && (2)\\
& && y_i \leq z_i && , \forall i, 1 \leq i \leq n && (3)\\
& && (M-1)y_i + z_i \leq M && , \forall i, 1 \leq i \leq n && (4)\\
& && y \in \{0,1\}^n, z \in \mathbb{Z}^n, x \in \{0,1\}^m. && && (5)
\end{aligned}
$$

No modelo, M é uma constante inteira arbitrariamente grande. De fato, basta que $M \geq |U|$, já que esse é um limitante superior natural de z. Ainda sobre as variáveis, com exceção de z que é inteira, todas as demais são variáveis binárias.

A função objetivo (1), maximiza o número de pontos cobertos de maneira única. Na restrição (2), z_i indica quantas vezes um ponto i é coberto pelos conjuntos selecionados. As restrições (3) e (4) são necessárias para que y_i assuma os valores desejados. A restrição (3), ao estabelecer que z_i é o limitante superior de y_i, resolve os casos em que $z_i = 0$, e portanto y_i deve ser igual a 0 também. As demais situações são tratadas pela restrição (4): para atender a desigualdade proposta, y_i só pode ser igual a 1 se z_i também o for. Caso contrário, y_i deve ser obrigatoriamente 0. Assim, garante-se que y_i irá representar quais pontos foram cobertos apenas uma vez pelos conjuntos selecionados, como desejado. Por fim, as restrições (5) garantem a integralidade das variáveis.

3 Detalhes dos Algoritmos

Durante o processo de testes e aperfeiçoamento das heurísticas, foram avaliadas cinco versões funcionais, cujos algoritmos serão descritos nesta seção.

As heurísticas propostas são baseadas nos princípios descritos por Resende e Ribeiro (2003), e portanto podem ser consideradas heurísticas GRASP. Elas apresentam diferenças apenas quanto à construção da solução inicial e à escolha do conjunto para substituição na busca local. Para evitar confusão, cada versão das heurísticas será nomeada como segue:

1. *Gulosa*: algoritmo guloso adaptativo que adiciona à solução sendo construída o conjunto de maior peso ainda não selecionado. A função peso está descrita adiante e é atualizada após cada conjunto ser incluído na solução.

2. *GRAP*: utiliza um parâmetro α para controlar a aleatoriedade na construção da solução, mas não realiza a fase de busca local.

3. *GRASP_FirstFit*: possui todos os conceitos de uma heurística GRASP. Na busca local, é selecionado o primeiro conjunto de peso maior que a solução atual.

4. *GRASP_BestFit*: outro GRASP completo, mas que seleciona o conjunto de maior peso dentre a vizinhança da solução atual, a cada iteração da busca local.

5. *GRASP_Disjunto*: considera na construção inicial apenas conjuntos com elementos disjuntos dos já escolhidos até então. Quando nenhum conjunto atenda mais a tal requisito, utiliza o peso geral como critério de seleção, assim como os demais.

O peso total de um conjunto s_i é dado por $peso_i = |s_i| - |s_i \cap C| - |s_i \cap E|$, sendo C o conjunto de pontos cobertos pelos conjuntos já escolhidos anteriormente e E o conjunto de pontos cobertos unicamente até o momento.

O Algoritmo 1 refere-se tanto a heurística *GRASP_FirstFit* como a *GRASP_BestFit*, que toma como entrada o número de elementos, n, o de conjuntos, m, uma matriz de incidência A indicando quais elementos pertencem a cada um dos conjuntos e um parâmetro real α, que será usado no cálculo do *threshold*. Considere vetores de inteiros *escolhidos* ($escolhidos(i) = 1$ se e somente se o conjunto i é selecionado), *cobertos* ($cobertos(j) =$ quantas vezes o ponto j é coberto pelos conjuntos selecionados) e *pesos* de cada conjunto.

Na função *Atualiza_Pesos*, o vetor *pesos* é atualizado considerando $peso_i = \sum_{j=1}^{n} c_j$, para i de 1 a m, com

$$c_j = \begin{cases} 1, & \text{se } cobertos(j) = 0, \text{ pois } j \text{ passa a ser coberto unicamente;} \\ -1, & \text{se } cobertos(j) = 1, \text{ pois } j \text{ deixa de ser coberto unicamente;} \\ 0, & \text{nos demais casos, em que } j \text{ já é coberto mais de uma vez.} \end{cases}$$

Algoritmo 1: GRASP-FirstFit(n, m, A, α)

```
1  início
2      Zera_Vetores(escolhidos, cobertos)
3      maximo ← 0
4      minimo ← n
5      unicos ← 0
6      Atualiza_Pesos()
7      enquanto Encontra_Max_Min() ≠ 0 faça
8          tamanho ← Constroi_RCL(α)
9          Seleciona_Conjunto(tamanho)
10         Atualiza_Pesos()
11     Busca_Local()
12     para i ← 1 ≤ n faça
13         se cobertos[i] = 1 então
14             unicos ← unicos + 1
15     retorna unicos
```

A função *Encontra_Max_Min* atualiza *maximo* e *minimo* para os novos valores de *pesos*, retornando quantos conjuntos com pesos positivos estão disponíveis. Em cada iteração do laço, a lista RCL de conjuntos candidatos, que são aqueles com peso maior ou igual ao *threshold* ($threshold = minimo + (maximo - minimo) * \alpha$) é construída e em seguida, a função *Seleciona_Conjunto* escolhe aleatoriamente um dos conjuntos candidatos para adicionar na solução.

Dois cenários de melhoria são considerados na busca local: a eliminação de um conjunto selecionado que tenha se tornado prejudicial ao longo da construção da solução, ou seja, seu peso tornou-se negativo; e a substituição de um conjunto por outro de peso peso. Na segunda situação, pode-se empregar tanto a estratégia *first-fit*, em que é utilizado na troca o **primeiro** conjunto a ser encontrado de peso maior ao que está sendo avaliado, quanto a *best-fit*, que seleciona o **melhor** conjunto dentre os disponíveis para a substituição. É essa a única diferença entre as heurísticas H_3 e H_4.

GRASP-Disjuntos é muito parecido com o da já apresentada, exceto que a função *Atualiza_Pesos* calcula os pesos considerando $peso_i = \sum_{j=1}^{n} c_j$, para i de 1 a m, onde $c_j = 1$ se e somente se $cobertos(j) = 0$ e $c_j = -n$ caso contrário. Esta restrição é eliminada apenas quando, ao final da fase de construção, existirem conjuntos que podem ser adicionados para melhorar a solução atual.

Em um dos testes foi incluída a heurística *GRASP-FirstFit* na implementação da formulação PLI proposta, com ambas estratégias trabalhando em conjunto para solucionar o problema. A ideia era que a heurística servisse como limitante inferior, mas em muitos casos foi ela mesmo quem encontrou a melhor solução.

4 Resultados Computacionais

Testes computacionais foram realizados para avaliar a qualidade das soluções obtidas pelas heurísticas e o desempenho tanto do algoritmo *branch-and-bound* usando o modelo (F) quanto das heurísticas apresentadas nas seções anteriores.

A linguagem C++ foi utilizada na implementação das heurísticas enquanto a linguagem C e a biblioteca de PLI da FICO, o *Xpress* FICO (2007)[1], foram usados no algoritmo exato.

Uma vez que não foram encontradas instâncias testes disponíveis na literatura para o *UCP*, utilizou-se um conjunto de instâncias do problema da cobertura máxima. As 70 instâncias foram obtidas na *OR-Library* de Beasley (2010), uma coleção de dados de teste para diversos problemas de Pesquisa Operacional. Cada instância foi submetida a 9 testes, divididos em dois grupos: os com uso do resolvedor PLI (de E_1 a E_4), e os submetidos apenas às heurísticas (de E_5 a E_9). A estratégia E_1 equivale ao algoritmo BB sem uso de nenhuma heurística. Já E_2 usa a heurística *GRAP* enquanto E_3 e E_4 usam *GRASP-FirstFIt* com parâmetro $\alpha = 1$. Todas as execuções foram limitadas a 1800 segundos, exceto na E_4 que foi limitado a 3600 segundos. As estratégias E_5, E_6, E_7, E_8 e E_9 equivalem, respectivamente, às heurísticas 1 a 5 descritas na Seção 3.

Os testes foram realizados numa máquina com processador Intel(R) Core(TM)2 Duo CPU E7400 de 2.80GHz e 4 GB de memória.

As instâncias foram agrupadas em 12 grupos, de acordo com propriedades como número de pontos e de conjuntos. Acredita-se que isso não traga nenhum prejuízo à exposição dos dados obtidos, pois o comportamento das instâncias de um mesmo grupo foi bastante semelhante em todos os casos observados. Na Tabela 1, é traçado um perfil das instâncias de teste, mostrando quantas instâncias fazem parte do grupo(q), quantos pontos (n) e quantos conjuntos (m) elas possuem. Além disso, derivou-se mais duas informações: a cardinalidade média de cada conjunto C, denotado por $|C|$ e em quantos conjuntos um mesmo ponto aparece em média, s. Os valores finais foram arredondados, pois não faz sentido referir-se a números reais nesse contexto.

Grupos	q	n	m	$\|C\|$	s
scp4	10	200	1000	4	20
scp5	10	200	2000	4	40
scp6	5	200	1000	10	50
scpa	5	300	3000	6	60
scpb	5	300	3000	15	150
scpc	5	400	4000	8	80

Grupos	q	n	m	$\|C\|$	s
scpd	5	400	4000	20	200
scpe	5	50	500	10	100
scpnre	5	500	5000	50	499
scpnrf	5	500	5000	100	999
scpnrg	5	1000	10000	20	199
scpnrh	5	1000	10000	50	499

Tabela 1: **Perfil das instâncias.**

Vale notar que o número de conjuntos é quase sempre 10 vezes superior ao do número de pontos, com exceção dos grupos scp4 e scp6. Como consequência, fica evidente que cada ponto será coberto por mais do que um conjunto, fato comprovado pela cobertura média de um ponto: esse valor varia de 20 a 999. Já a cardinalidade média dos conjuntos não é tão elevada, variando de 4 a 100 ao longo dos 12 grupos utilizados.

Em todos os testes, o limitante dual utilizado foi o número de pontos da instância. Para facilitar o entendimento da qualidade das soluções obtidas por cada estratégia, a Tabela 2 apresenta o GAP (em %) entre a solução encontrada e o limitante dual correspondente, calculado da seguinte forma: $gap = \dfrac{\text{limitante dual - melhor solução}}{\text{melhor solução}}$.

Instâncias	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
scp4	0,0	0,0	0,0	0,0	14,9	8,8	7,7	7,9	4,5
scp5	0,0	0,0	0,0	0,0	10,6	4,6	4,6	4,6	1,4
scp6	28,9	24,5	24,2	24,2	30,0	25,2	24,1	24,7	21,5
scpa	4,4	4,8	4,5	4,0	18,2	11,4	11,4	11,5	7,4
scpb	40,2	32,0	31,6	31,4	37,1	32,4	31,5	31,4	30,2
scpc	19,7	17,4	17,3	17,2	23,5	18,0	17,9	17,8	13,9

[1]Disponível para uso no laboratório da FACOM/UFMS por meio de licença acadêmica

scpd	51,3	39,7	40,3	40,2	47,1	40,2	39,9	39,2	40,0
scpe	13,6	14,2	13,6	13,6	29,5	16,3	14,2	15,7	12,1
scpnre	79,1	65,0	65,0	65,0	70,2	62,8	62,8	62,2	64,7
scpnrf	86,6	79,2	79,2	79,2	80,5	74,2	72,4	73,5	73,0
scpnrg	53,8	43,6	43,4	42,1	45,3	41,0	40,9	41,2	44,1
scpnrh	79,3	72,1	67,4	67,2	70,9	63,9	63,9	63,9	68,2
média	38,1	32,7	32,2	32,0	39,8	33,2	32,6	32,8	31,7

Tabela 2: **Gap dos resultados, em %.**

Nota-se que os algoritmos exatos (E_1 a E_4) conseguiram resolver de forma ótima as 20 instâncias dos grupos scp4 e scp5, pois seus gaps são nulos. Tal desempenho não foi obtido em nenhuma outra situação, o que pode indicar tanto um maior grau de dificuldade das instâncias ou apenas que o limitante atual não é muito bom. Infelizmente, como essas instâncias estão sendo aplicadas para o UCP pela primeira vez nesse trabalho, não existem resultados anteriores para comparação.

Pode-se observar também uma melhora na qualidade da solução conforme a abordagem torna-se mais sofisticada. Entre os testes com o resolvedor PLI, os piores resultados foram de E_1 e os melhores foram de E_4. Porém, a diferença entre E_2, E_3 e E_4 é bastante discreta, ficando abaixo de 1% de melhoria, de forma que ter dobrado o tempo limite de execução em E_4 não refletiu tão expressivamente no resultado final, assim como a heurística GRASP completa também não trouxe muito ganho em relação a abordagem mais simples da GRAP.

Entre as heurísticas, o comportamento foi parecido: E_5 obteve os piores resultados, e E_9, os melhores. Novamente a diferença entre E_6, E_7, E_8 e E_9 foi tímida, ficando na casa dos 1,5% apenas. É interessante observar que, pelo menos no que diz respeito aos resultados, a estratégia $FirstFit(E_7)$ foi sutilmente superior à $BestFit(E_8)$, o que apoia a decisão de incluir essa versão nos testes com a formulação PLI.

Em relação à PLI X heurísticas, o melhor desempenho varia de acordo com o grupo de instâncias. Isso provavelmente seja reflexo das características das instâncias: as com menos conjuntos e pontos foram resolvidas melhor por PLI, e mais extensas pelas heurísticas. Analisando os resultados de maneira global, tem-se que E_9 obteve o melhor desempenho.

Um parâmetro com influência direta no desempenho e eficiência das heurísticas é o critério de aleatoriedade α. Pensando nisso, variou-se o seu valor nos testes em 0.1 no intervalo de 0 a 1. Ao todo, foram testados 11 valores para α, realizando 200 execuções da heurística para cada um. Na Tabela 3, tem-se quantas vezes cada α encontrou a melhor solução para uma instância.

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
E_6	0	0	0	0	0	2	1	3	4	22	**49**
E_7	0	0	2	1	4	3	2	6	8	20	**39**
E_8	0	0	0	0	3	2	4	9	7	23	**41**
E_9	15	**23**	18	15	10	5	6	9	7	5	3

Tabela 3: **Comparativo dos diferentes α.**

Os valores em negrito mostram claramente que para as heurísticas E_6, E_7 e E_8 o melhor α é o 1, que indica um comportamento mais guloso da heurística. Já para a heurística E_9, a *GRASP-Disjunto*, esse cenário se inverte.

Pelo perfil das instâncias apresentado na Tabela 1, percebe-se que a quantidade de pontos e conjuntos não pode ser tomada como medida direta dessa dificuldade. Um exemplo disso está em scp4, resolvido de forma ótima com o uso da formulação PLI, e scp6, cuja solução ótima não foi encontrada mesmo tendo os mesmos valores que o grupo anterior.

Talvez a cardinalidade dos conjuntos e a cobertura média de cada ponto tenha mais a dizer em relação a essa dificuldade. Observando-se scp4 e scp5, que foram resolvidos de forma ótima, tem-se que ambos apresentam os menores valores nessas medidas. A cardinalidade média de seus conjuntos é 4, enquanto a cobertura média de cada ponto é 20 e 40 respectivamente. Um acréscimo de duas unidades na primeira e 20 na segunda (6, 60) já faz de scpa um grupo *difícil*, cuja solução ótima não foi encontrada. Ainda assim, scpa é o grupo que apresenta os menores gaps dentre os não resolvidos de forma exata, reforçando a ideia de que a cardinalidade e a cobertura média de um ponto devem ser fatores determinantes na dificuldade de uma instância do UCP.

5 Conclusões

Foi iniciado com esse trabalho um estudo em busca de um algoritmo exato para o problema da cobertura única, até agora tratado apenas de maneira aproximada. Sugeriu-se uma formulação de programação linear inteira para o problema, além de diferentes heurísticas GRASP para obter limitantes inferiores para serem usados dentro do algoritmo exato proposto.

Embora ainda preliminares, os resultados obtidos já serviram para nortear trabalhos fututos. Pretende-se avaliar outras formulações PLI para o problema, de forma a compará-las com a atual, uma vez que os limitantes duais fornecidos pela formulação (F) são fracos. Técnicas de geração de colunas e o estudo poliédrido do problema podem ser promissores para obter formulações mais apertadas e outros algoritmos exatos, como *branch-and-price* e *branch-and-cut*, para o problema.

Referências

Beasley, J. E., 2010. Or-library. http://people.brunel.ac.uk/˜mastjjb/jeb/orlib/scpinfo.html.

Demaine, E. D., Feige, U., Hajiaghayi, M. T., Salavatipour, M. R., 2006. Combination can be hard: approximability of the unique coverage problem. In: Proceedings of SODA'2006. pp. 162–171.

FICO, 2007. Xpress-Optimizer Reference Manual.

Karp, R. M., 1972. Reducibility among combinatorial problems. In: Complexity of Computer Computations. pp. 85–103.

Misra, N., Raman, V., Saurabh, S., Sikdar, S., 2009. The budgeted unique coverage problem and color-coding. In: Proceedings of the Fourth International Computer Science Symposium in Russia on Computer Science(CSR '09). pp. 310–321.

Moser, H., Raman, V., Sikdar, S., 2007. The parameterized complexity of the unique coverage problem. In: Proceedings of ISAAC'07. pp. 621–631.

Resende, M. G., Ribeiro, C. C., 2003. Handbook of Metaheuristics. Kluwer Academic Publishers, Ch. Greedy Randomized Adaptive Search Procedures, pp. 219–249.