UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
ENGENHARIA DE COMPUTAÇÃO

Análise e Projeto de Algoritmos
Problema de Otimização Combinatória

Gabriel Gonçalves - 11318331
Thiago Gonzaga - 11504760

Orientador: Prof. Dr. Gilberto Farias

João Pessoa – 29 de maio de 2018

Resumo

Este artigo propõe uma heurística gulosa chamada "*heurística gulosa de grandes passos" e* investiga sua aplicação para computar uma solução aproximada para o problema de cobertura máxima. Algoritmos gulosos constroem a solução em múltiplos passos, o algoritmo guloso clássico para o problema de cobertura máxima, em cada etapa seleciona um conjunto que contém o maior número de elementos descobertos. A *heurística gulosa de grandes passos*, em cada passo seleciona *p* (1 <= p <= k) conjuntos de tal forma que a união de p conjuntos selecionados contenham o maior número de elementos descobertos, avaliando todas as possíveis combinações *p* de conjuntos dados. Quando p = k o *algoritmo guloso de grandes passos* se comporta como um algoritmo exato que calcula a solução ideal avaliando todas as combinações *k* possíveis dos conjuntos dados. Quando p = 1, ele se comporta como o algoritmo guloso clássico. A *heurística gulosa de grandes passos* pode ser combinada com métodos de pesquisa local para calcular a melhor solução aproximada.

Introdução

O problema da cobertura máxima é selecionar *k* conjuntos {$Sx_1, Sx_2, Sx_3, ..., Sx_k$} da coleção de conjuntos S = {S_1, S_2, ..., S_n} de forma que o número de elementos na união de k conjuntos selecionados | Sx_1 U Sx_2 U… ... U Sx_k | seja o máximo possível. O problema de cobertura máxima é um problema **NP-difícil [1]**. Algoritmos gulosos constroem a solução em várias etapas, tomando uma decisão ótima localmente em cada etapa. O algoritmo guloso clássico para o problema de cobertura máxima, em cada etapa, seleciona um conjunto que contém o maior número de elementos descobertos. O algoritmo proposto chamado *algoritmo guloso de grandes passos*, em cada passo seleciona conjuntos *p* (1 <= p <= k) de tal forma que a união de *p* conjuntos selecionados contenha o maior número de elementos descobertos, avaliando todas as possíveis combinações entre os dados conjuntos. Algoritmos de aproximação para o problema da cobertura máxima e problema de cobertura de conjuntos usam técnicas similares. *Grossman* e *Wool* [1] conduziram uma comparação de desempenho de nove algoritmos de aproximação para o problema de cobertura de conjuntos, e descobriram que o algoritmo guloso randomizado é o melhor algoritmo geral entre os nove algoritmos de aproximação.

Algoritmos de aproximação existentes

O algoritmo guloso clássico para o problema de cobertura máxima é mostrado abaixo. O algoritmo guloso clássico começa com um conjunto vazio, e em cada etapa seleciona um conjunto que contém o maior número de elementos remanescentes descobertos pela solução parcial atual e adiciona o conjunto selecionado para solução parcial.

Algoritmo GulosoCoberturaMaxima (S,k)

```
S : Coleção de conjuntos {S1,S2, … Sn}
k : Número de conjuntos a serem selecionados de S
begin
 C ← ϕ
 W ← S1 U S2 U …..... U Sn
 S' ← S
        while (|C| < k)
        Select T ∈ S' que maximiza |T ∩ W|
        S' ← S' \{T }
        C ← C U {T}
        W ← W \ T
 end while
return C
end
```

O processo de adicionar um conjunto à solução parcial é repetido *k* vezes para selecionar *k* conjuntos. *Hochbaum* e *Pathria*[3] fornecem uma análise do algoritmo guloso clássico para o problema de cobertura máxima. Os algoritmos de aproximação anteriores[4,5,6] usaram a heurística gulosa para o problema de cobertura de conjuntos. O ***exemplo 1*** (abaixo) explica o método guloso com a ajuda de uma coleção de conjuntos pequenos e a mesma coleção de conjuntos é usada no ***exemplo 2*** (abaixo) para explicar o *algoritmo guloso de grandes passos*.

Exemplo 1. Seja S = {{a, b, c, d, e, f}, {a, b, c, g, h}, {d, e, f, i, j}, {g, h, i}, {k, l}} seja a coleção dada de conjuntos e K = 3.

Suponha rótulos para os dados conjuntos $\mathbf{S_1}$ = {a, b, c, d, e, f}, $\mathbf{S_2}$ = {a, b, c, g, h}, $\mathbf{S_3}$ = {d, e, f, i, j}, $\mathbf{S_4}$ = {g, h, i}, $\mathbf{S_5}$ = {k, l}. Cobertura inicialmente parcial C = {}.

No **primeiro pass**o do algoritmo, entre os cinco conjuntos, $\mathbf{S_1}$ tem seis elementos descobertos {a, b, c, d, e, f} e é melhor que a cobertura dos conjuntos $\mathbf{S_2}$, $\mathbf{S_3}$, $\mathbf{S_4}$ e $\mathbf{S_5}$. Então o primeiro passo seleciona $\mathbf{S_1}$ e agora, a cobertura parcial C = {{a, b, c, d, e, f}}.

No **segundo passo**, $\mathbf{S_4}$ tem três elementos descobertos {g, h, i}, $\mathbf{S_2}$ tem dois elementos descobertos {g, h}, $\mathbf{S_3}$ tem dois elementos descobertos {i, j} e $\mathbf{S_5}$ tem dois elementos descobertos {k, l}. Então, o segundo passo seleciona $\mathbf{S_4}$, e agora a cobertura parcial C = {{a, b, c, d, e, f} {g, h, i}}

No **terceiro passo**, $\mathbf{S_5}$ tem dois elementos descobertos {k, l}, $\mathbf{S_2}$ não tem elementos descobertos e $\mathbf{S_3}$ tem um elemento {j}. Então, o terceiro passo seleciona $\mathbf{S_5}$, e a cobertura parcial C = {{a, b, c, d, e, f} {g, h, i}, {k, l}}.

Agora |C| = 3 e C cobre 11 elementos. *Algoritmos gulosos aleatórios e probabilísticos aproximados*[7,8,9] produzem soluções melhores do que o algoritmo guloso clássico para o problema de cobertura de conjuntos. O algoritmo guloso randomizado Grossman e Wool[2] é o mesmo que o algoritmo guloso clássico, exceto que os laços são quebrados aleatoriamente e o algoritmo básico é repetido **N** vezes, retornando a melhor solução entre as soluções N. O estudo computacional de Grossman e Wool[2] mostrou que o algoritmo guloso randomizado é o melhor algoritmo de aproximação entre os nove algoritmos para o problema de cobertura de conjuntos.

Aickelin[10], Beasley e Chu[11] usaram algoritmos genéticos para problemas de cobertura de conjuntos. Gomes[12] comparou quatro algoritmos: *Round*, *Dual-LP*, *Primal-Dual* e *Guloso* e concluiu que o algoritmo guloso tem um bom desempenho entre os quatro algoritmos para problemas de cobertura de conjuntos.

O procedimento de busca adaptativa aleatória gulosa (*GRASP*)[13] é uma metaheurística iterativa que pode ser aplicada a muitos problemas de otimização combinatória. O *GRASP* em cada iteração constrói uma solução viável usando o método adaptativo guloso randomizado e aplica a pesquisa local para encontrar a solução localmente ideal na vizinhança da solução construída.

DePuy[14] propôs uma metaheurística chamada *Meta-RaPS* para resolver problemas combinatórios e DePuyet[15] investigou diferenças entre *Meta-RaPS* e *GRASP.* O *Meta-RaPS* foi aplicado para problemas de cobertura de conjuntos e compara-se com os cinco melhores algoritmos usados por Grossman e Wool[2].

Resende[17] aplicou o *GRASP* para o problema de cobertura máxima e mostrou que o *GRASP* encontra soluções quase ideais para a maioria dos problemas testados.

Heurística gulosa de grandes passos

A heurística gulosa de grandes passos começa com uma coleção de conjuntos vazios, em cada passo que ela seleciona p (1 <= p <= k) se estabelece que a união dos conjuntos selecionados contém o maior número de elementos descobertos, avaliando todas as possíveis *p* combinações de conjuntos restantes e adiciona os *p* conjuntos selecionados à cobertura de conjunto parcial.

O processo de adicionar *p* subconjuntos é repetido em *k*/*p* vezes. O último passo do algoritmo seleciona menos de *p* conjuntos quando *k* não é um múltiplo de *p*. O *algoritmo guloso de grandes passos* é mostrado abaixo.

Quando **p = k**, o algoritmo guloso de grandes passos se comporta como um algoritmo exato que calcula a solução ótima, avaliando todas as combinações *k* possíveis de conjuntos dados. Quando **p = 1**, ele se comporta como o algoritmo guloso clássico. Quando o tamanho do passo é *p*, o *algoritmo guloso de grandes passos* é executado no tempo **O** $((k/p) * |S|^p)$. O **exemplo 2** (abaixo) explica o **algoritmo guloso de grandes passos** com ajuda da coleção de conjuntos usada no **exemplo 1** (acima).

Algoritmo GulosoDeGrandesPassos (S, k, p)

```
S: Coleção de conjuntos {S1, S2, … , Sn}
k : Número de conjuntos a serem selecionados
p : Passo atual do algoritmo
begin
C ← φ
W ← S1 U S2 U … U Sn
        while (|C| < k)
      if ( (k - |C|) < p) then
          q ← k - |C|
      else
        q ← p
      end if

Select T = {T1, T2, …, Tq}, T ⊆ S\C que maximiza |W ∩ (T1 U T2 U … U Tq)|
W ← W \ (T1 U T2 U … U Tq)
C ← C U {T1, T2,..., Tq}
 end while
   return C
end
```

Exemplo 2. Seja S = {{a, b, c, d, e, f}, {a, b, c, g, h}, {d, e, f, i, j}, {g, h, i}, {k, l}} a coleção de conjuntos, K = 3 e o tamanho do passo do algoritmo é p = 2. E determinam-se os conjuntos S_1 = {a, b, c, d, e, f} , S_2 = {a, b, c, g, h}, S_3 = {d, e, f, i, j}, S_4 = {g, h, i} e S_5 = {k, l}.

Como p = 2, cada etapa do algoritmo escolhe dois conjuntos de modo que a união deles contenha o maior número de elementos descobertos.

Cobertura parcial inicial C = {}.

No **primeiro passo** do algoritmo, os candidatos são (S_1, S_2), (S_1, S_3), (S_1, S_4) (S_1, S_5), (S_2, S_3), (S_2, S_4), (S_3, S_4), (S_3, S_5) e (S_4, S_5); entre os dez candidatos (S_2, S_3) é melhor que todos os outros candidatos, pois S_2 U S_3 tem dez elementos descobertos e é maior que o dos outros candidatos. Assim, o **primeiro passo** seleciona (S_2, S_3) e agora a cobertura parcial C = {{a, b, c, g, h} {d, e, f, i, j}}.

No **segundo passo**, ele seleciona apenas um conjunto ao invés de dois, pois K = 3 e dois conjuntos (S_2, S_3) já estão selecionados pelo primeiro passo. Os candidatos são S_1, S_4 e S_5. S_5 tem dois elementos descobertos {k, l}, e S_1 e S_2 não têm nenhum elemento descoberto.

Algoritmo MelhorDosGrandesPassos-1-2-3-4(S, k)

```
S : Uma coleção de conjuntos {S1,S2, … Sn}
k : Número de conjuntos a serem selecionados de S
begin
 Melhor ← ϕ
                for (p = 1 to 4)
          C ← GulosoDeGrandesPassos (S,k,p)
                if ( |U Melhor| < |U C|) then
          Melhor ← C
           end if
         end for
       return Melhor
end
```

Então o **segundo passo** seleciona S_5 e agora, finalmente, solução C = {{a, b, c, g, h} {d, e, f, i, j} {k, l}} e C cobre 12 elementos. Isto é melhor que a cobertura dos conjuntos selecionados pelo algoritmo guloso clássico no **Exemplo 1** (acima).

O melhor do *algoritmo dos grandes passos* 1,2,3,4 (BBS-1,2,3,4) mostrado acima calcula quatro soluções aproximadas usando o *algoritmo guloso dos grandes passos* com tamanhos de passo p = 1, 2, 3, 4 e retorna a melhor solução entre as quatro computadas.

A heurística do *algoritmo guloso de grandes passos* não usa pesquisa local e ela pode ser usada na primeira fase das iterações do *GRASP* para construir soluções mais viáveis.

Resultados experimentais

A heurística gulosa de grandes passos foi comparada ao *algoritmo guloso randomizado*, o melhor algoritmo entre os nove algoritmos testados por Grossman e Wool[2].

A Tabela abaixo fornece a comparação do algoritmo *BS-2* (tamanho de passo p = 2) com o *algoritmo guloso clássico* e a comparação do algoritmo *BS-4* (tamanho de passo p = 4) com o *algoritmo guloso clássico*. [*BS = Big Step = Grande Passo].*

|X|	Tamanho da Coleção	Tamanho médio dos subconjuntos	k	Número de problemas	Guloso	Vs BS-2	Guloso	Vs BS-4
1000	100	70	10	100	9	31	9	49
1000	100	80	10	100	18	30	16	47
1000	150	60	15	100	18	36	17	59
1000	150	25	5	100	2	11	5	20
1000	150	30	5	100	3	3	3	4
1000	150	40	5	100	3	11	3	18
1000	150	50	5	100	3	12	1	26
1000	150	25	5	100	1	1	3	10
1000	150	30	5	100	0	9	0	18
1000	150	40	5	100	5	5	4	15
1000	150	50	5	100	3	19	1	36
1000	150	25	10	100	6	10	9	29
1000	150	30	10	100	4	19	1	29
1000	150	40	10	100	5	19	8	33
1000	150	50	10	100	10	27	3	50
1000	150	25	15	100	10	21	8	33
1000	150	30	15	100	10	26	14	45
1000	150	40	15	100	12	31	12	53

1000	150	60	15	100	16	27	10	57
1000	150	35	20	100	9	29	6	53
1000	150	40	20	100	17	21	22	39
1000	150	60	20	100	15	35	13	51
1000	150	70	5	100	2	14	1	25
1000	150	80	5	100	3	12	4	31
1000	150	60	5	100	6	12	6	26
1000	150	70	10	100	11	29	13	38
1000	150	80	10	100	15	30	10	46
1000	150	90	10	100	10	26	10	43
1000	150	90	5	100	8	23	9	40

Tabela: Guloso vs Guloso de grandes passos em instâncias de problemas randômicos

Na tabela acima,
- "**|X|**" é o número de elementos no conjunto universal;
- "**Tamanho da Coleção**" é o número de conjuntos na coleção definida *S* da instância do problema;
- "**k**" é o número de conjuntos a serem selecionados da coleção de conjuntos;
- "**Número de problemas**" é o número de problemas usados para comparação de desempenho;
- "**Vs BS-2**" ao lado de "**Guloso Vs BS-2**" é o número de ocorrências de problemas para qual a *heurística gulosa de grandes passos* com p = 2 está computando soluções aproximadas melhores do que o algoritmo guloso clássico;
- "**Guloso**" ao lado de "**Guloso Vs BS-2**" representa o número de instâncias de problema que o algoritmo guloso clássico está computando melhores soluções do que a *heurística gulosa de grandes passos* com p = 2.
- As duas colunas em "**Greedy Vs BS-4**" têm um significado semelhante às colunas em "**Greedy Vs BS-2**".

|X|	**Tamanho da Coleção**	**Tamanho médio dos subconjuntos**	**k**	**Número de problemas**	**Guloso-R**	**Vs BS-3**	**Guloso-R**	**Vs BS-4**
1000	100	70	10	100	16	32	19	39
1000	100	80	10	100	34	27	30	28

1000	150	60	15	100	49	15	37	32
1000	150	25	5	100	11	6	11	10
1000	150	30	5	100	4	2	10	3
1000	150	40	5	100	8	5	8	9
1000	150	50	5	100	6	16	4	16
1000	150	25	5	100	6	1	9	5
1000	150	30	5	100	9	1	4	5
1000	150	40	5	100	7	6	8	8
1000	150	50	5	100	3	11	1	26
1000	150	25	10	100	26	2	26	6
1000	150	30	10	100	23	7	17	7
1000	150	40	10	100	26	15	35	12
1000	150	50	10	100	27	15	22	23
1000	150	25	15	100	49	5	41	9
1000	150	30	15	100	38	7	36	5
1000	150	40	15	100	42	14	36	17
1000	150	60	15	100	45	14	36	23
1000	150	35	20	100	52	6	39	11
1000	150	40	20	100	55	8	49	10
1000	150	60	20	100	62	10	51	14
1000	150	70	5	100	9	11	6	17
1000	150	80	5	100	8	19	12	25
1000	150	60	5	100	9	14	7	17
1000	150	70	10	100	23	19	22	17
1000	150	80	10	100	21	25	20	28
1000	150	90	10	100	32	20	26	27

1000	150	90	5	100	17	26	15	28

Tabela 2: Guloso randomizado vs Guloso de grandes passos

Entre o **BS-2** (grandes passos com p = 2) e o algoritmo guloso clássico, o **BS-2** computou melhores soluções aproximadas do que o **algoritmo guloso clássico** para 21% dos problemas, e o **algoritmo guloso clássico** apresentou melhor desempenho que o **BS-2** para 8 % dos problemas.

Entre o **BS-4** (grande passo com p = 4) e o **algoritmo guloso clássico**, o **BS-4** computou melhores soluções aproximadas do que o **algoritmo guloso clássico** para 36% dos problemas, e o **algoritmo guloso clássico** apresentou melhor desempenho que o **BS-4** para 8 % dos problemas.

A **Tabela 2** (abaixo) fornece a comparação de desempenho do **algoritmo guloso randomizado** com N = 20 e o algoritmo guloso de grandes passos **BS-3** (com tamanho de passo p = 3) e o grande algoritmo **BS-4** (com tamanho de passo p = 4) em 3000 problemas gerados aleatoriamente instâncias.

Entre o **BS-3** (grande passo com p = 3) e o **algoritmo guloso randomizado**, o **BS-3** calculou melhores soluções aproximadas do que o **algoritmo guloso randomizado** para 13% dos problemas, e o **algoritmo guloso randomizado** foi melhoro que o **BS-3** para 25% dos problemas.

E entre o **BS-4** (grande passo com p = 4) e o **algoritmo guloso randomizado**, o **BS-4** calculou melhores soluções aproximadas do que o **algoritmo guloso randomizado** para 17% dos problemas, e o **algoritmo guloso randomizado** foi melhor que o **BS-4** para 22% dos os problemas.

|X|	**Tamanho da Coleção**	**Tamanho médio dos subconjuntos**	**k**	**Número de problemas**	**Guloso-R**	**Vs Melhor de BS-1,2,3,4**
1000	100	70	10	100	7	47
1000	100	80	10	100	7	38
1000	150	60	15	100	18	35
1000	150	25	5	100	2	10
1000	150	30	5	100	2	4
1000	150	40	5	100	2	10
1000	150	50	5	100	0	22

1000	150	25	5	100	4	6
1000	150	30	5	100	2	5
1000	150	40	5	100	3	11
1000	150	50	5	100	0	28
1000	150	25	10	100	10	7
1000	150	30	10	100	10	10
1000	150	40	10	100	12	22
1000	150	50	10	100	10	28
1000	150	25	15	100	32	12
1000	150	30	15	100	23	11
1000	150	40	15	100	18	23
1000	150	60	15	100	25	26
1000	150	35	20	100	30	12
1000	150	40	20	100	29	16
1000	150	60	20	100	36	21
1000	150	70	5	100	3	21
1000	150	80	5	100	2	31
1000	150	60	5	100	2	22
1000	150	70	10	100	9	27
1000	150	80	10	100	3	36
1000	150	90	10	100	11	39
1000	150	90	5	100	3	38

Tabela 3: Guloso randomizado vs Melhor de grandes passos 1,2,3,4

A **Tabela 3** fornece comparação de desempenho do algoritmo **MelhorDosGrandesPassos** (BBS-1,2,3,4) e do **algoritmo guloso randomizado** em 3000 instâncias de problemas gerados aleatoriamente. O algoritmo **BBS-1,2,3,4** computou melhores soluções aproximadas para 22% dos problemas e o **algoritmo guloso randomizado** computou melhores soluções aproximadas para 11% dos problemas.

Conclusão

Esta pesquisa propôs uma nova heurística gulosa chamada *heurística gulosa de grande passo*. O *algoritmo guloso de grande passo* foi comparado com o *algoritmo guloso clássico* e o *algoritmo guloso randomizado* [2]. Experimentos em muitos casos de problema de cobertura máxima mostraram que o algoritmo guloso de grande passo com p = 2, p = 3 e p = 4 computa soluções aproximadas melhores que *o algoritmo guloso clássico* em muitos casos. À medida que o tamanho do passo *p* é aumentado, o *algoritmo guloso de grande passo* computou melhores soluções aproximadas do que o *algoritmo guloso clássico*.

O *algoritmo guloso randomizado* com 20 repetições calculou melhor solução aproximada do que o *algoritmo guloso de grande passo* com tamanho de passo p = 3 e com tamanho de passo p = 4 na média. O algoritmo ***MelhorDosGrandesPassos*** 1,2,3,4 computou melhor solução aproximada do que o *algoritmo guloso randomizado* com 20 repetições na média. O algoritmo ***MelhorDosGrandesPassos*** proposto neste trabalho pode ser combinado com métodos de busca local para encontrar uma solução aproximada.

Referências

[1] Karp, R.M., "Reducibility Among Combinatorial Problems", Complexity of Computer Computations, Plenum Press (1972).

[2] Tal Grossman, Avishai Wool, "Computational experience with approximation algorithms for the set covering problem", European journal of operational research 101, 81-92 (1997).

[3] Dorit S. Hochbaum, Anu Pathria "Analysis of the Greedy Approach in Problems of Maximum kCoverage", Naval Research Logistics, Vol. 45, 615-627 (1998).

[4] Chvatal, V. "A Greedy Heuristic for the Set-Covering Problem", Mathematics of Operations Research, 4(3), 223-235 (1979).

[5] Johnson, D.S., "Approximation algorithms for combinatorial problems", Journal of Computer System Science 9,256-278 (1974).

[6] Lovasz L, "On the ratio of optimal integral and fractional cover", Discrete Mathematics, 13,383-390 (1975).

[7] Haouari, M., Chaouachi, J.S., "A probabilistic greedy search algorithm for combinatorial optimization with application to the set covering problem", Journal of the Operational Research Society 53, 792-799 (2002).

[8] Vasko, F.J., Wilson, G.R., "An efficient heuristic for large set covering problems" Naval Research Logistics Quarterly 31, 163-171 (1984).

[9] Feo, T., Resende, M.G.C., "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters 8, 67-71 (1989).

[10] Aickelin, U., "An indirect genetic algorithm for set covering problems", Journal of the Operational Research Society 53, 1118-1126 (2002).

[11] Beasley, J.E., Chu, P.C., "A genetic algorithm for the set covering problem", European Journal of Operational Research 94, 392-404 (1996).

[12]] Fernando C. Gomes, Claudio N. Meneses, Panos M. Pardalos,Gerardo Valdisio R. Viana, "Experimental Analysis of Approximation Algorithms for the Vertex Cover and Set Covering Problems", Computers & Operations Research, 33, 3520-3534 (2006) .

[13] T. A. Feo and M. G. C. Resende., "Greedy randomized adaptive search procedures. Journal of Global Optimization", 6, 109-133, (1995).

[14] DePuy, G.W., Whitehouse, G.E., Moraga, R.J., "Using the meta-raps approach to solve combinatorial problems", In Proceedings of the 2002 Industrial Engineering Research Conference, vol. 19, p. 21 (2002).

[15] DePuy, G.W., Moraga, R.J., Whitehouse, G., 'MetaRaPS: A simple and effective approach for solving the traveling salesman problem", Transportation Research Part E: Logistics and Transportation Review 41 (2), 115- 130 (2005).

[16] G. Lan, G. W. DePuy, and G. E.Whitehouse, "An effective and simple heuristic for the set covering problem," European Journal of Operational Research, vol. 176, no. 3, pp. 1387-1403 (2007).

[17] Mauricio G.C. Resende, "Computing Approximate Solutions of the Maximum Covering Problem with GRASP", Journal of Heuristics, Vol 4, Issue 2, 161-177 (1998).