Aplicação de Algoritmos Heurísticos ao Problema de Cobertura de Conjunto

Prof. Ademir A. Constantino
Departamento de Informática
Universidade Estadual de Maringá
www.din.uem.br/~ademir

- Objetivo:
  * Apresentar alguns algoritmos heurísticos para um problema de Cobertura de Conjunto
- Tópicos:
  * Formalização do problema;
  * Algoritmo de Construção Guloso (*Greedy*);
  * Algoritmo de Busca Local – Vizinhança Aleatória
  * Algoritmo GRASP
  * Algoritmo *Simulated Annealing*.

Modelo *Set Covering*

- O problema de cobertura de conjunto (*Set Covering*) é um problema de programação linear inteira.

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

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

Define-se ainda:

Matriz $m \times n$ $A = [a_{ij}]$;

$M = \{1, 2, \ldots, n\}$ o conjunto das linhas;

$N = \{1, 2, \ldots, n\}$ o conjunto das colunas

onde

c_j =custo da coluna j;

$$a_{ij} = \begin{cases} 1 \text{ se linha } i \text{ é coberta pela coluna } j; \\ 0 \text{ caso contrário} \end{cases}$$

$$x_j = \begin{cases} 1 \text{ se a coluna } j \text{ está na solução;} \\ 0 \text{ caso contrário.} \end{cases}$$

Problema de Cobertura de Conjunto

- Exemplo de *Set Covering* na forma matricial :

$$\text{Minimizar } (c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7)$$

$$\text{Sujeito à} \quad \begin{vmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{vmatrix} \geq \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix}$$

Uma solução viável

Aplicações do *Set Covering*

- Existem muitos problemas práticos que são resolvidos como um problema *Set Covering*, por exemplo:
  * escalonamento de tarefas;
  * balanceamento de carga em linha de produção;
  * investimento de capital;
  * localização de facilidades;
  * roteirização de veículos;
  * recuperação de informação.

Proposta de um Algoritmo Guloso (*Greedy*) para o *Set Covering*

Notações utilizadas pelo algoritmo:

- $M=\{1, 2, ..., m\}$ //conjunto das linhas
- $P_j= \{i \in M \ / \ a_{ij}=1\}$ //conjunto de linhas cobertas pela coluna j;
- k_j = número de linhas (ainda não cobertas) que podem ser cobertas com a coluna j;
- S = conjunto contendo uma solução (subconjunto de colunas).
- R = conjunto das linhas ainda não cobertas.

Passo 0: Faça $R=M$, $S=\emptyset$, $t=1$ e vá para o passo 1;

Passo 1: Se $R= \emptyset$ vá para o passo2. Caso contrário, faça $k_j = |P_j \cap R|$ e escolha a coluna $j_{(t)}$ tal que $f(C_{j(t)}, k_{j(t)}) = \min \{f(c_j, k_j) \ / \ k_j>0, j= 1,...,m\}$ Considere $R = R \backslash P_{j(t)}$, $S= S \cup \{j(t)\}$, $t= t+1$ e vá para o passo1.

Passo 2: Ordene os elementos de S em ordem decrescente do valor C_j. Considere os elementos $i \in S$ em ordem, e se $S\backslash\{i\}$ é uma solução (cobertura) viável, então faça $S=S\backslash\{i\}$. Quando todos os elementos $i \in S$ forem considerados então S será a cobertura primária.

Propostas de Funções *Greedy* para o *Set Covering*.

- Funções $f(c_j, k_j)$ propostas por Vasco and Wilson (1984) :
  1. c_j
  2. c_j / k_j
  3. $c_j / \log_2 k_j$
  4. $c_j / k_j \log_2 k_j$
  5. $c_j / k_j \ln k_j$
  6. $c_j / (k_j)^2$
  7. $(c_j)^{1/2} / (k_j)^2$
- Algoritmo *Greedy* proposto Vasco and Wilson (1984):
  - * Modificar o Passo 1 do algoritmo *Greedy* anterior;
  - * uma função *Greedy* (de 1 a 7) é selecionada aleatoriamente cada vez que uma coluna for selecionada para entrar na solução.

Algoritmo de busca na vizinhança de Jacobs and Brusco (1995)

- Parâmetros:
  - * S = conjunto com as colunas que estão na solução;
  - * S' = conjunto com as colunas que não estão na solução;
  - * w_i= o número de colunas que cobrem a linha i;
  - * U= Conjunto das linhas descobertas;
  - * $Z(S)$ = Custo da solução;
  - * $N(S)$= o número de colunas no conjunto S
  - * $Q(S)$= $\max\{c_j : \forall\, j \in S\}$
  - * $\rho_1 \in [0,1]$ = percentual de $N(S)$
  - * $\rho_2 \in [0,1]$ = percentual de $Q(S)$

Algoritmo de Busca na Vizinhança de Jacobs and Brusco (1995)

0. Faça d=0, $D=\lceil \rho_1 N(S) \rceil$, $E=\lceil \rho_2 Q(S) \rceil$ e $w_i = \Sigma_{k \in S} a_{ik}$, i=1,...,m
1. Selecione aleatoriamente uma coluna k, $k \in S$.
2. Troque k de S para S'. Faça $w_i = w_i - a_{ik}$ para todo $i \in M$. Faça d=d+1. Se d=D, vá para 3, caso contrário retorne para 1.
3. Defina U como o conjunto das linhas com w_i= 0. Se $U = \emptyset$, então vá para 6, caso contrário vá para 4.
4. Defina S'_E como o conjunto das colunas com $c_j \leq E$ / $j \in S'$. Calcule o seguinte:
    - α_{ij} = 1 se w_i= 0 e a_{ij}= 1 para todo $i \in M$, $j \in S'_E$; 0 caso contrário.
    - $v_j = \Sigma_{i \in M} \alpha_{ij}$ para todo $j \in S'_E$.
    - $\beta_j = c_j/v_j$ para todo $j \in S'_E$.
    - $\beta_{min} = \min \{\beta_j / j \in S'_E \}$
    - K = conjunto das colunas com $\beta_j = \beta_{min}$ / $j \in S'_E$.
5. Selecione aleatoriamente uma coluna $k \in K$ e troque esta coluna de S' para S. Faça $w_i = w_i + a_{ik}$ para todo $i \in I$. Retorne para 3.
6. Examine cada coluna k, $k \in S$, na ordem inversa. Se $w_i - a_{ik} \geq 1$ para todo $i \in M$, então troque k de S para S' e faça $w_i = w_i - a_{ik}$ para todo $i \in M$.

GRASP
de Feo and Resend (1995)

- Característica do problema estudado
  - Custo unitário $c_j = 1$ para j=1,...,n

- <u>Fase de Construção</u>

- Escolha gulosa:
  - Selecionar o conjunto P_j que cobre o maior número de linhas ainda não cobertas, isto é , maior valor para k_j.
- Escolha Semi-gulosa (*semi-greedy*):
  - Selecionar aleatoariamente um conjunto P_j de uma lista RCL contendo somente os conjuntos que cobrem ao menos um percentual α do maior valor para k_j.

GRASP
de Feo and Resend (1995)

- Exemplo:

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
					•			1
		•	•			•		2
•	•		•	•		•		3
•			•	•	•		•	4
				•	•			5
2	1	1	**3**	**3**	**3**	**2**	1	

Linhas ainda não cobertas

∀ α = 40%, então RCL={P_1,P_4, P_5, P_6, P_7} para solução inicial

- Suponha que P_5 tenha sido selecionado aleatoriamente. Então, excluir as linhas 3, 4 e 5 que foram cobertas.

GRASP
de Feo and Resend (1995)

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
					•			1
		•	•			•		2
								3
								4
								5
0	0	1	1	0	1	1	0	

- RCL: $=\{P_3, P_4, P_6, P_7\}$
- Próxima escolha aleatória seja P_3 . Então, restará apenas P_6.
- Solução $S = \{P_3, P_5, P_6\}$ de tamanho 3.

GRASP
de Feo and Resend (1995)

- Por outro lado, se P6 tivesse sido escolhido inicialmente em lugar de P5, teríamos outra situação:

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
								1
		•	•			•		2
•	•		•	•		•		3
								4
								5
1	1	1	**2**	1	0	**2**	0	

- Escolhendo P_4, teríamos uma cobertura menor $S = \{P_4, P_6\}$ de tamanho 2.

GRASP
de Feo and Resend (1995)

- <u>Fase de Busca Local</u>
- Define-se a vizinhança (*k*,*p*)-*troca* da seguinte forma: para todas as *k*-ênuplas em uma cobertura *S* que seja possível trocá-las por uma *p*-ênupla (p<k) que não está em *S*.
- Exemplo: considere o exemplo abaixo com cobertura $S = \{P_3, P_5, P_6\}$.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
					•			1
		•	•			•		2
•	•		•	•		•		3
•			•	•	•		•	4
				•	•			5
		↑		↑	↑			

GRASP
de Feo and Resend (1995)

- Aplicando (2,1)-*troca* que substitui a
  * 2-ênupla $\{P_3, P_5\}$
  * pela 1-ênupla $\{P_4\}$,
  * resultaria na cobertura ótima $S = \{P_4, P_6\}$.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
					•			1
		•	•			•		2
•	•		•	•		•		3
•			•	•	•		•	4
				•	•			5
			↑		↑			

GRASP de Feo and Resend (1989)

- Característica do problema estudado
  - * Custo unitário $c_j = 1$ para j=1,...,n
  - * N= Número de iterações;

Procedure *GRASP*(n, P_1,...,P_n, α, N, S)
x:= n;
Para i:=1,...,N faça
    Para j=1,...,n faça $\eta_j = P_j$;
    Faça S:= $\emptyset$;
    Enquanto $\eta_j \neq \emptyset$, $\forall$ j=1,...,n faça //*constrói a solução*
        Γ:=max{$|\eta_j|$, $\forall$ j=1,...,n };
        RCL ={ j:$|\eta_j| \geq \alpha\, \Gamma$, $\forall$ j=1,...,n};
        Selecionar aleatoriamente K de RCL ;
        S:= $S \cup k$;
        Para j=1,...,n faça $\eta_j = \eta_j - P_k$; //*retira de todas as colunas as linhas cobertas por* P_k
    Aplicar (1,0)-*Troca* em S; //*busca local*
    Se $|S| < x$ então x:= $|S|$; S^*:=S;

Um Algoritmo Simulated Annealing (Jacobs and Brusco, 1995)

- Entrada: T_0, T_f, N_{it}, α (entre 0 e 1)
- $T \leftarrow T_0$; $S_0 \leftarrow$ gera solução inicial; $S \leftarrow S_0$; $S^* \leftarrow S_0$
- enquanto $T > T_f$ faça (temperatura alta)
- para cont $\leftarrow$ 1 até M_{it} faça (iterações para equilíbrio)
- S' $\leftarrow$ seleciona uma solução vizinha de S
- Δcusto $\leftarrow$ custo(S') -custo(S)
- se Δcusto < 0 ou U[0,1] < exp(-Δcusto/T)
- então $S \leftarrow S'$
- se ($S < S^*$) então $S^* \leftarrow S$
- fim do para
- $T \leftarrow \alpha T$
- fim-enquanto

Parâmetros considerados por Jacobs and Brusco (1995)

Teste	ρ_1	ρ_2	T_0	α	N_{it}	Tempo Máximo
1	0.1	1.1	1.3	0.9	100	240 seg.
2	0.1	2.0	1.3	0.9	100	240 seg.
3	0.4	1.1	1.3	0.9	100	240 seg.
4	0.4	2.0	1.3	0.9	100	240 seg.

Referências

- Feo, T. A. and Resende, M. G. C. A Probabilistic Heuristic for a Computationally Difficult Set Covering. *Operations Research Letters*, 8, pp. 67-71, 1989.

- Feo, T.A. and Resende, M.G.C. Greedy Randomized Adaptive Search Procedure. *J. of Global Optimization*, vol. 6, pp. 109-133, 1995.

- Jacobs, L. W. and Brusco, M. J. A Local-Search Heuristic for Large Set-Covering Problems. *Naval Research Logistic*. Vol. 42, pp. 1129-1140, 1995.

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