Universidade Federal do Rio Grande do Sul
Instituto de Informática
Engenharia de Software N
Profa. Lucinéia Heloisa Thom

RELATÓRIO TRABALHO FINAL

Problema da Mochila com Grafos de Conflito
VNS

Augusto Boranga - 242262
Matheus Pereira - 242247

1 - Problema

O Problema da Mochila como problema de otimização consiste em encontrar a melhor combinação de elementos escolhidos para fazer parte de uma solução. Cada um destes elementos possui um peso e um valor associado e o objetivo é obter a maior soma possível dos valores dos elementos presentes na solução, ao mesmo tempo em que se respeita a restrição de que a soma dos pesos dos elementos presentes na solução não pode ultrapassar o peso máximo dado na instância do problema.

O Problema da Mochila com Grafos de Conflito possui como base as restrições explicadas acima, com adição de restrições que proíbem dois elementos de estarem presentes na solução simultaneamente. Isto é, se houver uma restrição de conflito com os itens A e B, as seguintes situações são possíveis:

-**A** faz parte da solução mas **B** não faz parte.
-**B** faz parte da solução mas **A** não faz parte.
-**A** e **B** estão fora da solução.

Estas restrições de conflito são dados de entrada da instância do problema.

1.1 - Formulação Matemática

A formulação do Problema da Mochila com Grafos de Conflito que criamos consiste na formulação do Problema da Mochila com as restrições adicionais das arestas conflitantes dadas na entrada.

Formulação matemática:

$$\textbf{max} \quad \sum_{i=1}^{n} Pi * Xi$$

$$\textbf{s.a.} \quad \sum_{i=1}^{n} Wi * Xi \leq c$$

$$Xi + Xj \leq 1 \quad \forall \ (i,j) \in E$$

$$Xi \in \{1,0\}$$

onde:
i = 1,2, …, n
j = 1,2, …, n
c = capacidade da mochila
Pi = valor do item i
Wi = peso do item i
Xi = 1 se item i está na solução, 0 caso contrário
W = conjunto de restrições: [(i1, j1), (i2, j2), …]

1.2 - Formulação GLPK

Utilizamos o solver *glpk* para resolver a formulação abaixo, modelada na linguagem de programação *MathProg*.

```
set V;
/* set dos itens */

set E dimen 2;
/* set das restricoes (arestas) */

param c;
/* da capacidade da mochila */

param n;
/* numero de itens */

param w{v in V};
/* peso dos itens */

param p{v in V};
/* valor de cada item */

var x{v in V} >= 0 binary;
/* presença de um item na mochila em uma solução (binária) */


maximize cost: sum{v in V} p[v] * x[v];
/* função de maximização */

s.t. capacity: sum{v in V} w[v] * x[v] <= c;
/* restricoes de capacidade */

s.t. conflict{(i,j) in E}: x[i] + x[j] <= 1;
/* restricoes de conflito */

end;
```

Para executar a avaliação, executamos o comando:

```
glpsol -m formulacao.mod --data kpcg_instances/HB10.dat -o
output/HB10.out --tmlim 3600
```

2 - Meta-heurística

Nossa meta-heurística é a **VNS**: *Variable Neighborhood Search*. Isto é, Busca de Vizinhanças Variáveis. A ideia principal por trás deste conceito é realizar a busca por máximos locais alternando sistematicamente as diferentes vizinhanças geradas utilizando-se de diversos critérios.

Esta característica a diferencia de métodos de busca local como *hill climbing*, por exemplo, onde apenas uma vizinhança (utilizando um critério apenas) é gerada para que a busca seja então realizada.

2.1 - Implementação

2.1.1 - Modelagem

Para este trabalho optamos por *python* como linguagem. Apesar de ser uma linguagem mais lenta, é muito prática e ambos integrantes do grupo possuíam conhecimento.

Implementamos um algoritmo de busca utilizando uma versão básica do VNS. As estruturas de dados que escolhemos usar foram:

- Item: tupla **(número do item, valor, peso)**
- Solução (mochila com itens): array com itens
- Restrições: matriz n x n, onde n é o número de itens. As células a_{ij} e a_{ji} desta matriz terão valor **1** se houver restrição entre os itens i e j e **0** caso contrário.

2.1.2 - Vizinhanças

Nossa solução final gera um total de 3 vizinhanças na execução do algoritmo. Os critérios escolhidos para geração de vizinhanças foram:

Para cada vizinhança Ni, com i = {1, 2, 3}:

N1, N2, N3

retiramos i elementos da mochila e colocamos elementos aleatórios que satisfaçam as restrições tanto de peso da mochila (não ultrapassar o máximo) quanto de conflito (não haver restrição no conjunto R que contenha este item). O número de elementos adicionados na mochila varia. Conforme couber, mais elementos vão sendo adicionados.

Para a vizinhança N3, decidimos descartar todos os itens presentes na mochila e gerar uma solução com itens aleatórios. Esta heurística é uma espécie de último recurso na busca por uma solução melhor.

2.1.1 - Critério de parada

Utilizamos como critério de parada para a execução de nosso algoritmo o número de iterações do método de busca. Escolhemos um total de 10 iterações do algoritmo de busca como limite, pois com muitas iterações a execução se tornava muito lenta e o resultado não melhorava o suficiente para compensar o maior tempo de execução.

3 - Resultados

A seguir temos uma comparação entre os resultados obtidos com o solver *glpk* (com a formulação mostrada acima, em **1.2**) e a nossa implementação utilizando a metaheurística VNS em cada uma das i nstâncias do problema fornecidas.

3.1 - Execução

Abaixo estão descritos: resultado, tempo e porcentagem de acerto com relação à solução ótima para cada método por cada instância.

	Solução ótima	**glpk Resultado**	**glpk Tempo (s)**	**glpk Acerto (%)**	**VNS Resultado**	**VNS Tempo (s)**	**VNS Acerto (%)**
C10	7776	7776	0,1	100	7776	9,7	100
C1	1040	1040	0,0	100	1040	13,4	100
C3	600	600	0,0	100	590	8,8	98
R10	2007	2007	2,8	100	1996	18,6	99
R1	559	559	0,0	100	559	19,4	100
R3	1763	1763	8,0	100	1733	289,4	98
test	7310	5445	3600 (limite)	74	5066	1092,3	69
HB10	49836	17376	3600 (limite)	34	22533	88,2	45
HB11	99702	32407	3600 (limite)	32	38373	501,7	38
HB12	199413	0 (não encontrou)	18000 (limite)	0	71519	5064,3	35

3.2 - Média glpk

A formulação que fizemos no final obteve uma média de **74%** de acertos. Pode-se observar que o motivo que fez com que esta média não fosse 100% ou algo muito próximo disso foi a execução nas instâncias muito grandes.

Imaginamos que esta queda na taxa de acerto conforme as instâncias crescem esteja relacionada com o tempo limite que escolhemos para a execução do solver, que foi de 1 hora para cada instância (com exceção do HB12, que como não obteve nenhum resultado, fomos incrementando o tempo limite até chegarmos em 5 horas).

Não acreditamos que esta queda de resultado esteja relacionada com a formulação, visto que buscamos formular o problema de maneira mais simples e direta possível, nos baseando na formulação apresentada no artigo fornecido no enunciado do trabalho. (http://jgaa.info/accepted/2009/PferschySchauer2009.13.2.pdf).

3.3 - Média do algoritmo com meta-heurística

O algoritmo que implementamos obteve uma média de **78,2%** de acertos. Consideramos este resultado bem satisfatório, visto que foi superior ao encontrado com o solver *glpk*.

Neste caso também pudemos observar uma queda na porcentagem de acerto conforme a instância crescia. Isso se deve ao fato de nossa função de busca por máximos locais ter ficado lenta. Esta função precisa gerar os vizinhos de uma dada solução, portanto, quando há muitos vizinhos a serem gerados (como é o caso das instâncias "grandes", como HB10, HB11 e HB12), ela vira um gargalo e demora a retornar.

4 - Conclusão

Ao término deste trabalho concluímos que o uso de uma meta-heurística pode ajudar na execução de um algoritmo de busca. Além disso, vimos que pode ser muito difícil encontrar a solução ótima de um sistema conforme o tamanho de seus dados de entrada aumenta.

Outro fator que percebemos que influencia bastante foi o critério para a geração de vizinhança. Como a busca em vizinhos é a base dos algoritmos de busca, a forma como eles são gerados vai impactar na qualidade do resultado final.

Tivemos certa dificuldade para encontrar uma maneira eficiente de gerar vizinhos, pois às vezes o algoritmo convergia muito rápido para um máximo local e não melhorava além disso. Mas em geral, obtivemos bons resultados com o algoritmo que implementamos.

5 - Referências

- http://www.ime.unicamp.br/~chico/mt852/slidesvns.pdf
- http://www.unipac.br/site/bb/tcc/tcc-33fc0129a58e28afaa0283f0a0af2f1d.pdf
- http://jgaa.info/accepted/2009/PferschySchauer2009.13.2.pdf
- http://www.cs.uleth.ca/~benkoczi/OR/read/vns-tutorial.pdf