Relatório do trabalho prático – MC548 2011s1

Instituto de Computação – Unicamp – Prof. Cid Souza

27/06/2011

**1 - Integrantes do grupo:**

Alberto Arruda de Oliveira RA 093311

Luiz Claudio C. de Carvalho RA 800578

**2 - parte 1 do trabalho:**

Exercício [gt54]:

Variáveis utilizadas: (definição das variáveis usadas no modelo)

Restrições: (descrição de cada (família de) restrições do modelo)

Função Objetivo: (fórmula)

Exercício [ss2]:

Variáveis utilizadas: (definição das variáveis usadas no modelo)

Restrições: (descrição de cada (família de) restrições do modelo)

Função Objetivo: (fórmula)

Exercício [gt10]:

Variáveis utilizadas: (definição das variáveis usadas no modelo)

Restrições: (descrição de cada (família de) restrições do modelo)

Função Objetivo: (fórmula)

Exercício [mn27]:

Variáveis utilizadas: (definição das variáveis usadas no modelo)

Restrições: (descrição de cada (família de) restrições do modelo)

Função Objetivo: (fórmula)

Exercício [nd32]:

Variáveis utilizadas: (definição das variáveis usadas no modelo)

Restrições: (descrição de cada (família de) restrições do modelo)

Função Objetivo: (fórmula)

Resultados:

|  |  |  |
| --- | --- | --- |
| **Exercício** | **Instância** | **Função Objetivo** |
| [gt54] | 1 |  |
| [gt54] | 2 |  |
| [gt54] | 3 |  |
| [ss2] | 1 |  |
| [ss2] | 2 |  |
| [ss2] | 3 |  |
| [gt10] | 1 |  |
| [gt10] | 2 |  |
| [gt10] | 3 |  |
| [mn27] | 1 |  |
| [mn27] | 2 |  |
| [mn27] | 3 |  |
| [nd32] | 1 |  |
| [nd32] | 2 |  |
| [nd32] | 3 |  |

**3 - parte 2 do trabalho:**

Estruturas de dados utilizadas:

Foi construída uma estrutura para cada um dos elementos básicos do problema, a saber, os **satélites** e as **shards**:

shard

int posH; //Posição Horizontal

int posV; //Posição Vertical

float rShard; //Ganho de Armazenagem

float cH; //Custo de armazenagem pelo satélite em

// rota horizontal

float cV; //Custo de armazenagem pelo satélite em

// rota vertical

int lidaPor;

bool lida;

satelite

int ns; //Numero do satélite

float memTotal; //Memória total do satélite

float memRestante; //Memória restante do satélite

Leitura de dados:

Foi criada uma função específica para este fim. A mesma armazena em vetores das estruturas acima os satélites e as shards fornecidas na entrada. Retorna valor total de rShard, ou seja, da recompensa máxima que se pode extrair da instância. Os satélites são numerados de 1 a 2n, sendo que os satélites de rota vertical recebem os números (satélite.ns) de *n+1* a *2n*. Isso permite que estes sejam armazenados em um único vetor, facilitando sua ordenação. Para facilitar as comparações, a coordenada vertical de cada shard recebeu o valor *y+n*, ou seja, ao valor da entrada foi acrescida a quantidade de satélites.

Após a leitura, estes vetores foram **ordenados** da seguinte forma:

* Satélites: ordem crescente de capacidade de memória;
* Shards: ordem decrescente da relação benefício/custo, tendo sido utilizado o menor dos dois custos de leitura (horizontal ou vertical);

Heurística básica:

O problema foi tratado como uma sequência de problemas da mochila, sendo alocadas a cada satélite as shards viáveis e ainda não lidas que maximizam a recompensa capturada por cada um. Para a solução de cada subproblema de preenchimento de um satélite usamos a heurística ingênua de se alocar os melhores R/C, ou seja, as shards são alocadas na sequência em que foram ordenadas após sua leitura. Para minimizar a chance de que um satélite de menor capacidade não fosse utilizado, estes são preenchidos primeiro, ou seja, também de acordo com sua ordenação inicial. A complexidade desta heurística básica é O(n.m), sendo n e m o número de satélites e shards, respectivamente.

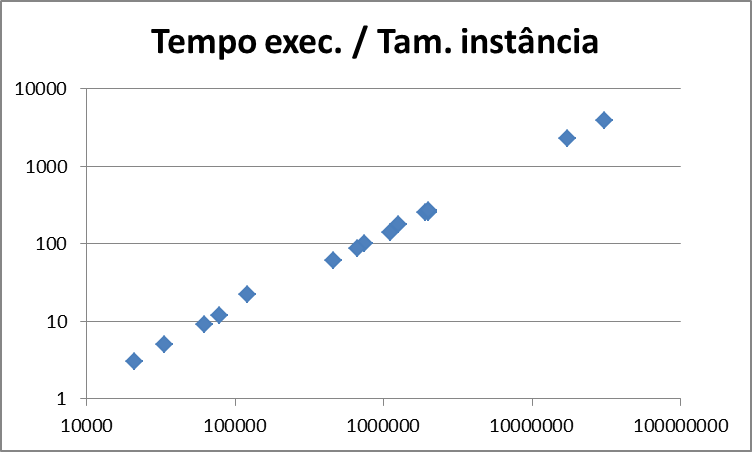
Os resultados da aplicação desta heurística básica são tabulados abaixo.

Tabela.3.1 – resultados da heurística básica

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instância** | **Satélites** | **Shards** | **R. Máximo** | **Capturado** | **%** | **Tempo exec.\*** |
| Small0 | 80 | 264 | 13.666 | 8.450 | 62 | 3 |
| Small1 | 80 | 1.520 | 76.022 | 53571 | 70 | 22 |
| Small2 | 80 | 774 | 39.212 | 27.166 | 69 | 9 |
| Small3 | 80 | 420 | 21.307 | 14.106 | 66 | 5 |
| Small4 | 80 | 975 | 49.391 | 30763 | 62 | 12 |
| Med0 | 200 | 9.508 | 4.718.409 | 3.185.460 | 68 | 253 |
| Med1 | 200 | 3.719 | 1.845.683 | 1.290.119 | 70 | 100 |
| Med2 | 200 | 10.000 | 5.002.484 | 3.423.897 | 68 | 255 |
| Med3 | 200 | 5.528 | 2.804.919 | 1.833.207 | 65 | 141 |
| Med4 | 200 | 10.000 | 5.021.152 | 3.524.435 | 70 | 267 |
| Med5 | 200 | 6.341 | 3.194.956 | 2.257.615 | 71 | 175 |
| Med6 | 200 | 2.297 | 1.139.797 | 715.828 | 63 | 61 |
| Med7 | 200 | 3.330 | 1.690.257 | 1.115.662 | 66 | 87 |
| Big0 | 600 | 29.069 | 14.510.868 | 9.807.963 | 68 | 2262 |
| Big1 | 600 | 50.913 | 25.453.577 | 18.316.737 | 72 | 3870 |

\* variação em clock() durante a rotina de alocação;

Fig. 3.1 tempos de execução – heurística básica\*



\* escala di-logaritmica. Tempo de execução em ticks versus dimensão da entrada

(shards x satélites).

O exame da alocação produzida por esta heurística mostrou que os satélites de maior capacidade, preenchidos por último, frequentemente ficavam bastante ociosos. Esta constatação inspirou o aprimoramento introduzido, descrito a seguir.

(\*\*\* Caso ainda haja memória disponível em algum dos satélites e existam shards que não foram lidas\*\*\*), os satélites são novamente percorridos, desta vez no sentido inverso da ordenação (dos maiores para os menores). Para cada um deles, são percorridas novamente todas as shards, testando-se as seguintes condições:

* Pode ser lida pelo satélite atual;
* Não está alocada ou foi alocada anteriormente a um satélite de menor capacidade;

Preenchidas estas condições, a shard é alocada ao satélite atual e a memória consumida por ela é devolvida ao satélite onde se encontrava anteriormente alocada. (\*\*\* Se este procedimento já não reduz a zero a quantidade de shards não lidas\*\*\*), as duas rotinas (*heurística básica* e *deslocamento para os maiores*) são executadas sequencialmente, enquanto houver melhorias ou até que todas as shards tenham sido lidas.

No que diz respeito à sequência em que as shards são percorridas em cada laço do deslocamento para os satélites maiores, foram feitas três tentativas: o percurso na ordem original (recompensa/custo decrescente), a ordem inversa e a escolha aleatória de se uma shard apropriada para deslocamento é efetivamente deslocada ou não. Neste último caso, foi considerado um número máximo de iterações sem melhoria para a interrupção da rotina. Os resultados de cada variante são sintetizados na tabela abaixo (apenas para as instâncias em que a heurística básica não foi 100% eficiente). A memória total restante nos satélites tabulada é apenas uma indicação e não uma comprovação de que foi atingida uma solução ótima para a instância. (\*\*\* Os testes efetuados com substituição da heurística básica por uma invocação ao solver glpk para a alocação de shards aos satélites produziram os limitantes superiores indicados na última coluna\*\*\*).

O código entregue no trabalho corresponde ao da variação \*\*\*N\*\*\*

Tabela 3.2 – Aprimoramentos comparados

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Heurística** | **Básica** | | | **Variação 1** | | | **Variação 2** | | | **Variação 3** | | | **LS** |
| **Instância** | **NC** | **MR** | **TE** | **NC** | **MR** | **TE** | **NC** | **MR** | **TE** | **NC** | **MR** | **TE** | **NC** |
| Small2 | 31 |  | 0 |  |  |  |  |  |  |  |  |  |  |
| Med0 | 1.114 |  | 37 |  |  |  |  |  |  |  |  |  |  |
| Med1 | 360 |  | 14 |  |  |  |  |  |  |  |  |  |  |
| Med2 | 130 |  | 16 |  |  |  |  |  |  |  |  |  |  |
| Med4 | 96 |  | 13 |  |  |  |  |  |  |  |  |  |  |
| Med5 | 184 |  | 13 |  |  |  |  |  |  |  |  |  |  |
| Med6 | 6 |  | 13 |  |  |  |  |  |  |  |  |  |  |
| Med7 | 236 |  | 23 |  |  |  |  |  |  |  |  |  |  |
| Big0 | 27 |  | 724 |  |  |  |  |  |  |  |  |  |  |
| Big1 | 5.743 |  | 1.566 |  |  |  |  |  |  |  |  |  |  |

NC – Recompensa não capturada

MR – Memória não utilizada dos satélites

TE – tempo de execução

LS – limitante superior obtido com solver glpk

A complexidade de pior caso das variantes 1 e 2 é potencialmente exponencial, dependendo do número de iterações que forem executadas até que se chegue à solução ótima ou que não se obtenham melhorias na solução. A da variação 3 depende do tamanho da entrada e do número máximo de iterações que forem permitidas. Na prática foi observado que o número máximo de iterações realizadas foi de (\*\*\*N\*\*\*). O gráfico abaixo os tempos de execução em função do tamanho da entrada de cada uma das instâncias.