UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

RELATÓRIO DO PROJETO DE MC548

Aluno: Murilo Fossa Vicentini **RA**: 082335
Aluno: Tiago Chedraoui Silva **RA**: 082941

Sumário

Lista de Tabelas

Lista de Figuras

1 Integrantes

Aluno: Murilo Fossa Vicentini **RA**: 082335
Aluno: Tiago Chedraoui Silva **RA**: 082941

2 Parte 1

2.1 [nd30]

Variáveis usadas no modelo

- Para cada aresta $(i, j) \in A$, criou-se a variável binária y_{ij} que assume valor $y_{ij} = 1$ se e somente se a aresta (i,j) pertence ao caminho mínimo.

Restrições do modelo

- Todo vértice diferente do inicial e do final deve conter ou nenhuma aresta entrando e saindo ou uma entrando e saindo.
$$\sum_{i \in V}^{m} y_{ik} = \sum_{j \in V}^{m} y_{kj}, \quad \forall k \in V, \forall (i,k) \text{ e } (k, j) \in A$$

- Peso total do caminho não deve exceder K
$$\sum_{i,j \in A} w_{i,j} y_{i,j} \leq K$$

- Deve existir uma aresta que sai de s
$$\sum_{j \in V} y_{s,j} = 1$$

- Deve existir uma aresta que chega em t
$$\sum_{j \in V} y_{j,t} = 1$$

Função objetivo

Objetivo: minimizar o custo do caminho
$$min \sum_{i,j \in A} c_{i,j} y_{i,j} \tag{1}$$

2.2 [mn27]

Variáveis usadas no modelo

- Para cada vértice $u \in V$ e para cada cor $k \in \{1, 2, \ldots, m\}$, criou-se a variável binária x_{uk} que assume valor $x_{uk} = 1$ se e somente se o vértice u foi colorido com a cor k.

- Criou-se uma variável binária y_k para toda cor $k \in \{1, 2, \ldots, m\}$. $y_k = 1$ se e somente se pelo menos um vértice recebeu essa cor.

Restrições do modelo

- Todo vértice deve receber exatamente uma cor
$$\sum_{k=1}^{m} x_{uk} = 1, \quad \forall u \in V$$

- Se um vértice recebe a cor k, esta deve ser usada

$$x_{uk} \leq y_k, \quad \forall u \in V, k \in \{1...m\}$$

- Os Vértices vizinhos não podem ter a mesma cor

$$x_{uk} + x_{vk} \leq 1, \quad \forall (u,v) \in E, k \in \{1...m\}$$

Função objetivo

Objetivo: minimizar o número de cores usadas:

$$min \sum_{k=1}^{m} y_k \tag{2}$$

2.3 [ss2]

Variáveis usadas no modelo

- Criou-se uma variável binária x_{ij} para toda tarefa $i, j \in T$ que recebe valor $x_{ij} = 1$ se e somente se a tarefa i precede j.
- Para cada tarefa $i \in T$ criou-se uma variável binária y_i que recebe valor $y_i = 1$ se e somente se a tarefa i não cumpriu o deadline.

Restrições do modelo

- Todo par de tarefas (i, j) deve ter uma precedência, em que se i precede j, j não pode preceder i.

$$x_{ji} + x_{ij} = 1, \quad \forall i, j \in T$$

- Todo par de tarefas (i, j) deve ter uma transitoriedade de precedência, em que se i precede j, e j precede k, i precede k. Se $x_{ij} = 1$ e $x_{jk} = 1$, então $x_{ik} = 1$.

$$x_{ij} + x_{jk} - 1 \leq x_{i,k}, \quad \forall i, j, k \in T$$

- Para cada par de tarefas (i, j) em S, a tarefa i, obrigatoriamente tem que preceder j.

$$x_{ij} = 1, \quad \forall (i, j) \in S$$

- Se uma j tarefa é precedida por outras n tarefas, o tempo de término da tarefa j deve ser no mínimo o tempo de execução de todas as tarefas predecessoras, mais o seu tempo para ser executada. Se for esse término for menor que o deadline, $y_j = 0$, senão $y_j = 1$

$$\sum_{i \in T, i \neq j} x_{ij} * t_i \leq d_j - t_j + M * y_j \quad \forall j \in T$$

  Em que M é um número grande e para calculá-lo somou-se todos os tempos das tarefas mais um.

$$M = \sum_{i \in T} (t_i) + 1$$

Função objetivo

Objetivo: minimizar o número de tarefas que terminem fora do prazo:

$$min \sum_{i=1}^{n} y_i \tag{3}$$

2.4 [ss15]

Variáveis usadas no modelo

- Criou-se uma variável binária $x_{i,j,k}$ para toda tarefa $k \in T$ de todo projeto $i, j \in J$ que recebe valor $x_{i,j,k} = 1$ se e somente se a tarefa k do projeto i precede a tarefa k do projeto j.
- Para cada projeto $j \in J$ e cada tarefa $i \in T$ criou-se uma variável inteira $begin_{j,i}$ que recebe valor o valor de início da tarefa i do projeto j.
- Criou-se uma variável inteira fim que recebe o tempo de término do último projeto.

Restrições do modelo

- Toda tarefa dos pares de projetos (i, j) deve ter uma precedência, em que se i precede j, j não pode preceder i.

$$x_{j,i,k} + x_{i,j,k} = 1, \quad \forall i, j \in J$$

- Toda tarefa $i \in T$ do projeto $j \in J$ tem um tempo mínimo de início que é equivalente ao tempo de início da tarefa que a antecede $i-1$ mais o tempo da execução da tarefa predecessora $t_{j,i-1}$ para o mesmo projeto.

$$begin_{j,i} \geq begin_{j,i-1} + t_{j,i-1}, \quad \forall i \in T, \forall j \in J$$

- Se um projeto j precede um projeto i o tempo de término da tarefa k do projeto i deve ser maior que o tempo de término da tarefa z do projeto j mais o seu tempo de execução.

$$begin_{j,i} + t_{j,i} \leq begin_{k,i} + (1 - x_{j,k,i}) * M, \quad \forall i \in T, \forall j, k \in J : k! = j$$

  Em que M é um número grande e para calculá-lo somou-se todos os tempos das tarefas de todos os projetos.

- A tarefa k do projeto j precede a tarefa z do mesmo projeto, logo o tempo de término da tarefa z deve ser maior que o tempo de término da tarefa k mais o seu tempo de execução.

$$begin_{k,i} + t_{k,i} \leq begin_{j,i} + x_{j,k,i} * M, \quad \forall i \in T, \forall j, k \in J : k! = j$$

  Em que M é um número grande e para calculá-lo somou-se todos os tempos das tarefas de todos os projetos.

$$M = \sum_{j \in J, i \in T} (t_{j,i})$$

- O tempo total da execução dos projetos deve ser igual ao tempo do último terminar.

$$fim \geq begin_{j,m} + t_{j,m}, \quad \forall j \in J$$

  Em que m é o tempo em que a última tarefa do projeto é executada (tarefa no último processador).

Função objetivo

Objetivo: minimizar o tempo de término de todos os projetos:

$$\min fim \tag{4}$$

2.5 [mn22]

Variáveis usadas no modelo

- Para cada máquina $m \in V$ e para cada sala $r \in \{1,2,\ldots,|V|\}$, criou-se a variável binária x_{mr} que assume valor $x_{mr} = 1$ se e somente se a máquina m foi colocada na sala r.
- Para cada peça $p \in U$ e para cada sala $r \in \{1,2,\ldots,|U|\}$, criou-se a variável binária y_{pr} que assume valor $y_{pr} = 1$ se e somente se a peça p foi colocada na sala r.
- Criou-se uma variável binária $rdiff_{mp}$ para cada aresta da máquina $(m,p) \in E$ para a qual $rdiff_{mp} = 1$ se e somente se a máquina e a peça especificada pela aresta estão em salas diferentes.

Restrições do modelo

- Toda máquina deve estar em uma única sala

$$\sum_{r \in \{1,2,\ldots,|V|\}} x_{mr} = 1, \quad \forall m \in V$$

- Toda peça deve estar em uma única sala

$$\sum_{r \in \{1,2,\ldots,|V|\}} y_{pr} = 1, \quad \forall p \in U$$

- Número de máquinas por sala não deve exceder limite K

$$\sum_{m \in V} x_{mr} \leq K, \quad \forall r \in \{1,2,\ldots,|V|\}$$

- Se uma máquina estiver em sala diferente de sua peça, $rdiff = 1$

$$rdiff_{mp} \geq x_{mr} - y_{pr}, \quad \forall r \in \{1,2,\ldots,|V|\}, \forall (p,m) \in E$$

Função objetivo

Objetivo: minimizar a soma do custo de transporte de uma peça p para a mesma sala da máquina m:

$$min \sum_{(i,j) \in E} c_{i,j} * rdiff_{i,j} \tag{5}$$

2.6 Resultados

ID exercício	1	2	3
[nd30]	3	13	21
[mn27]	3	7	13[1]
[ss2]	1	6	17
[ss15]	8	165	245[2]
[mn22]	1	4131	3691[3]

Tabela I: Resultados da parte 1 - Modelagem gmpl

[1]Rodou-se durante 30 minutos para somente 13 cores (13 cores havia sido uma solução encontrada anteriormente, diminui-se os número de cores para o número de restrições diminuir e assim o processamento ser mais rápido);

[2]Rodou-se durante 30 minutos para somente 4 salas;

[3]Utilizando a abordagem apresentada na seção do problema, e utilizando-a durante 8 horas em um servidor do laboratório de redes da Unicamp encontrou-se o resultado não ótimo 257. Contudo, se considerarmos somente a ordem dos projetos a solução é encontrada em questão de segundos e o resultado é melhor, 245, porém talvez não seja o ótimo. Como sabe-se que esse resultado é um resultado para a abordagem da seção do problema (seção 2.4) esse resultado foi considerado o melhor que encontramos;

ID exercício	1	2	3
[nd30]	3	13	21
[mn27]	3	7	Não terminou
[ss2]	1	Não terminou	Não terminou
[ss15]	8	Não terminou	Não terminou
[mn22]	1	Não terminou	Não terminou

Tabela II: Resultados da parte 1 - Modelagem planilha

3 Parte 2

Na segunda parte desenvolveu-se uma heurística gulosa, no qual ordenava-se os shards em relação ao ganho do shard com o mínimo de custo entre o satélite vertical e o horizontal. A partir disso pegava-se os shards até que não fossem mais possível. Posteriormente, desenvolveu-se uma busca local a partir da solução gulosa.

Na busca local retirava-se um shard da solução ótima e tentava colocar outros candidatos no lugar. A remoção de um shard da solução ótima é realizada do que possuir menor relação custo/benefício para a melhor, ou seja, se um shard ocupa mais espaço e possibilita um menor ganho.

A lista de candidatos (LRC) foi limitada a 10 candidatos, estes constituídos dos 10 que possuem melhores relação custo/benefício. A escolha dos candidatos foi novamente feito através de uma busca gulosa. Percorre-se a lista lrc e tenta inserir cada um.

Isso era feito para todos os shards da solução ótima, se o resultado fosse melhorado ele era salvo. Caso não fosse, o estado anterior era restaurado, de modo que a mudança na solução que foi ruim, fosse descartado. Feito isso uma vez, isso é repetido até que o tempo limite seja atingido.

A idéia inicial do algoritmo guloso era dar uma solução boa em tempo curto e a busca local seria justamente inserir uma variabilidade no algoritmo.

3.1 Estruturas implementadas

Foram utilizadas algumas estruturas, dentre elas:

Estrutura Shard contendo dois inteiros x e y correspondentes aos satélites que conseguem tirar foto do shard. Dois inteiro hcost e vcost representam o custo de armazenamento dos shards para cada satélite x e y. E uma flag active, se o shard já está ou não na solução.

Estrutura Satélite Contem dois inteiro que contem o tamanho da memória de cada satélite sendo um na horizontal e outro na vertical.

Estrutura CSShard Contém shards inseridos na solução, contém x e y do shard, a direção do satélite responsável pela foto, o indíce idx do shard na estrutura com todos os shards.

Estrutura Solution Contém valor da solução ótima encontrada, o número de shards inseridos na solução, e um vetor de CSshards.

3.2 Análise de complexidade do algoritmo

Denotaremos a $|Sat| = m$ e $|Shards| = n$.

1. função main $O(n^2 + nm)$
   (a) função get_args: $O(1)$
   (b) função read_instances: $O(n+m)$
   (c) função quicksort: $O(n^2)$
   (d) função Greedy_solver: $O(n)$

(e) Local_search: $O(n^2 + nm)$

i. copy_sol: $O(n)$
ii. save_state: $O(n+m)$
iii. recover_statel: $O(n+m)$
iv. find_lrc: $O(n)$
v. choose_lrc: $O(n)$
vi. save_sol: $O(n)$

Observações:

- Local_search executa $|n|$ as funções de *i* até *vi* listadas acima;
- A função main tem sua complexidade delimitada pela função Local_search; e exata é executada até que o tempo seja atingido.

3.3 Resultados

Apresentamos dois resultados, o primeiro para a solução gulosa (ver tabela III) e o segundo aplicando a busca local (ver tabela IV)sobre a solução gulosa encontrada.

Posteriormente, utilizando duas instâncias de entrada (big-0 e small-0) projetou-se um gráfico (ver figuras 1 e 2) com a melhora da solução em relação ao número de iterações para a heurística.

Instância	Melhor solução encontrada	Parâmetro Tempo
small-0	8874	5s
small-1	56435	5s
small-2	29430	5s
small-3	14848	5s
small-4	31403	5s
medium-0	3354366	5s
medium-1	1334090	5s
medium-2	3590494	5s
medium-3	1927658	5s
medium-4	3696487	5s
medium-5	2350322	5s
medium-6	743412	5s
medium-7	1157211	5s
big-0	10148418	5s
big-1	19194050	5s

Tabela III: Resultados da parte 2 - Somente parte gulosa

4 Ambientes computacionais

1. Máquina 1

Memória 2.0 GB

Processador Intel® Core™2 Duo CPU T5250 @ 1.50GHz × 2

Gráfico Intel® 965GM

Tipo SO 32 bits

SO Fedora 15 (Lovelock)

Instância	Melhor solução encontrada	Parâmetro Tempo
small-0	9937	2 min
small-1	58763	2 min
small-2	33013	2 min
small-3	17449	2 min
small-4	33029	2 min
medium-0	3484321	2 min
medium-1	1394863	2 min
medium-2	3756501	2 min
medium-3	2001891	2 min
medium-4	3858310	2 min
medium-5	2454255	2 min
medium-6	779016	2 min
medium-7	1203455	2 min
big-0	10538294	2 min
big-1	19714711	2 min

Tabela IV: Resultados da parte 2 - Parte gulosa e busca local

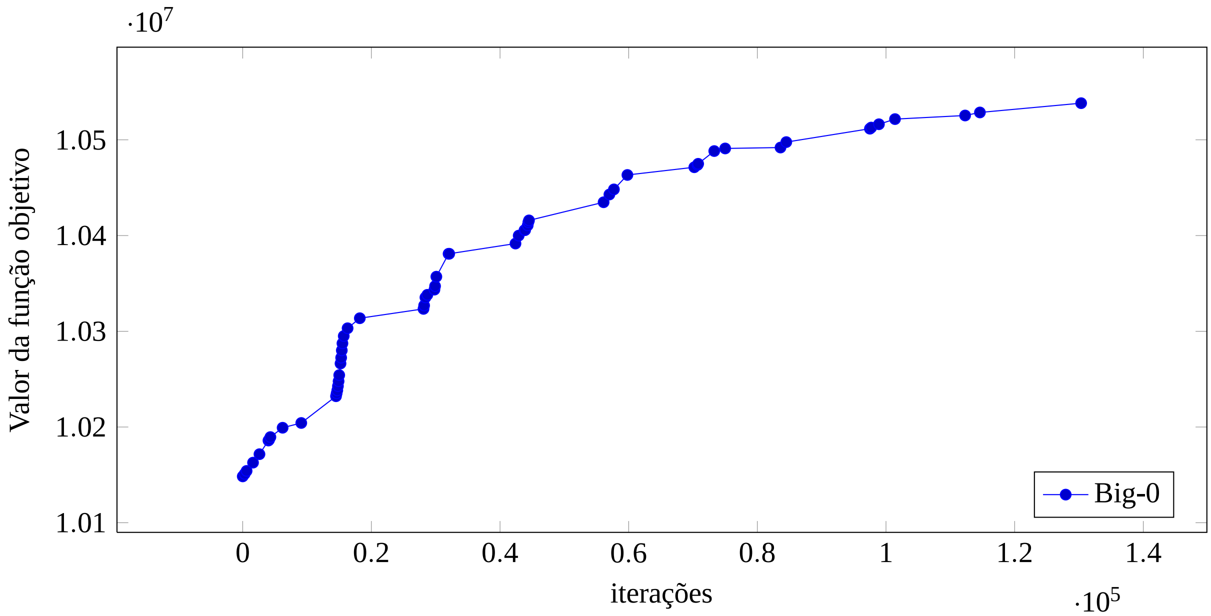

Figura 1: Melhora com busca local - Ganho fotografando shards - Instância big-0

Compilador gcc (GCC) 4.6.0 20110530 (Red Hat 4.6.0-9)

Linguagem de programação C

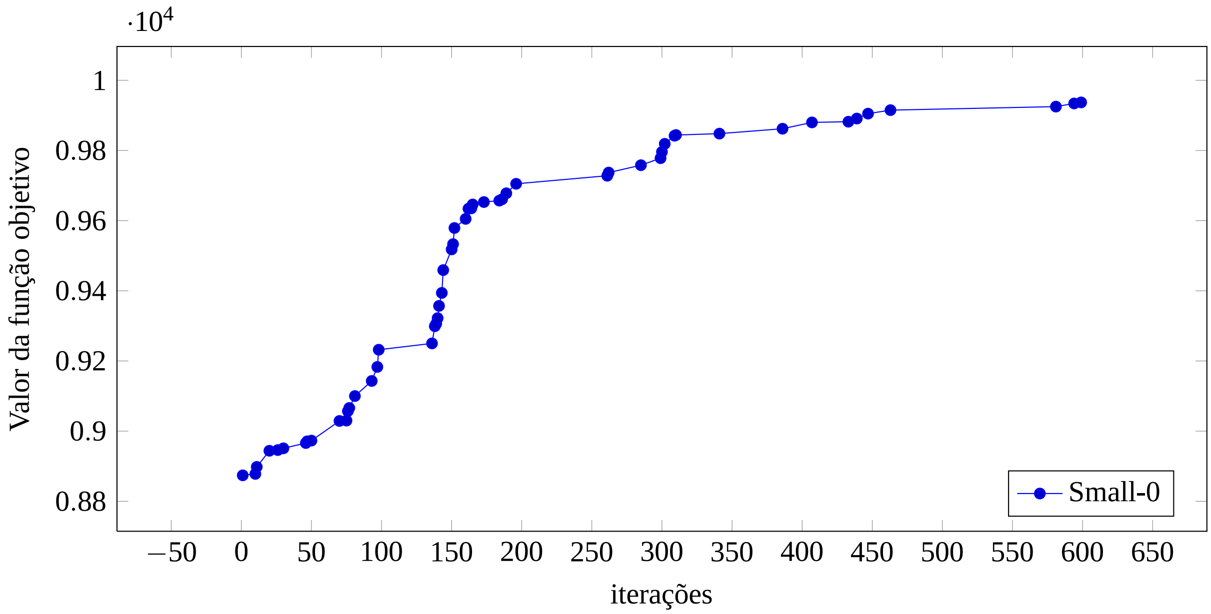

Figura 2: Melhora com busca local - Ganho fotografando shards - Instância small-0