MDIO

Gabriel Poça 56974 Pedro Nunes 54726
Sofia Vieira 54782

7 de Janeiro de 2012

Resumo

1 Introdução

Este relatório é composto por cinco secções, incluindo esta, sendo que cada uma das seguintes corresponde aos quatro primeiros exercicios em enunciado respectivamente. Por questões de apresentação e simplicidade determinada informação, componente de alguns exercicios, encontra-se em outros documentos, como tal este relatório é acompanhado de um documento denominado *ex2.lp*, onde consta código para o lp_solve correspondente ao exercicio 2, e outro documento denominado *resumo.pdf* onde consta um documento que contém o resumo para o exercicio 1.

2 Exercicio 1

Como consta na introdução o resumo encontra-se em outro documento denominado *resumo.pdf*. O artigo de base ao resumo tem por titulo "Ants can solve the team orienteering problem"e é da autoria de *Liangjun Ke*, *Claudia Archetti* e *Claudia Archetti*[1].

3 Exercicio 2

No exercicio 2 deve-se, considerando uma sub-instância do problema original e uma frota de duas viaturas, establecer um modelo de Programação Linear que permita encontrar a solução óptima. Consideremos as seguintes variáveis:

n Numero de vértices a visitar, sendo que o vértice 0 representa o vértice inicial e o vértice n o final.

m Numero de viaturas.

[1]http://www.sciencedirect.com/science/article/pii/S360835207002162

L_i Prémio em cada vértice para $i = 0, \ldots, n$.

$Tmax$ Distância máxima que cada viatura pode percorrer.

t_{ij} Distância entre o vértice i e j.

x_{ijk} Variável binária que representa a passagem, ou não, da viatura k entre o ponto i e j.

O modelo de Programação Linear é:

- $\sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=0}^{m} Lj * x_{ijk}$

s.a:

1. $\sum_{i=0}^{n-1} \sum_{k=1}^{m} x_{ijk} <= 1 \qquad j = 1, \ldots, n$

2. $\sum_{i=0}^{n} x_{ipk} = \sum_{j=0}^{n} x_{pjk} \qquad p = 0, \ldots, n \qquad k = 1, \ldots, m$

3. $\sum_{i=0}^{n} \sum_{j=0}^{n} t_{ij} * x_{ijk} <= Tmax \qquad k = 1, \ldots, m$

4. $\sum_{i=1}^{n-1} x_{0ik} = 1 \qquad k = 1, \ldots, m$

5. $\sum_{i=1}^{n-1} x_{ink} = 1 \qquad k = 1, \ldots, m$

A restrição 1 garante que apenas uma das viaturas pode visitar um ponto. A restrição 2 garante que existe, para cada viatura, um percurso de saida de cada vertice visitado para outro vertice. A restrição em 3 garante que é respeitado um limite máximo de distância percorrida para cada viatura. As restrições em 4 e 5 garantem que uma viatura deve ser 'sair' do vertice inicial e terminar no vértice final.

Este relatório é acompanhado de um documento denominado *ex2.lp* onde se encontra o código para o lp_solve que prova a solução proposta.

4 Exercicio 3

O pseudo código na Figura 1 consiste num algoritmo que permite encontrar os percursos com maior numero de vértices, sem repetidos, entre dois pontos.

Algoritmo 1 Pseudo código para os caminhos mais curtos.

```
nV ← Numero de viaturas.
graph ← Array com todos os vétices a visitar.
dMax ← Valor da distância máxima que cada viatura pode percorrer.
paths := Conjunto que detém os percursos determinados.
for i = 0; i < nV; i ← i + 1 do
  path := Novo percurso composto por sequência de vértices vazia.
  path ← path+ Vértice inicial.
  while Procura não terminada do
    if Distância no percurso com distância ao vertice final < dMax then
      path ← path+ Vértice mais próximo do ultimo no percurso.
    else
      Remover o ultimo vértice inserido.
      path ← path+ Vértice final.
    end if
  end while
  paths ← paths + path
end for
```

Aplicado ao problema em enunciado optemos os seguintes percursos:

1. 1 -> 28 -> 18 -> 6 -> 7 -> 3 -> 32
2. 1 -> 19 -> 20 -> 27 -> 31 -> 22 -> 32
3. 1 -> 13 -> 9 -> 8 -> 10 -> 11 -> 12 -> 21 -> 32
4. 1 -> 17 -> 29 -> 32

Deste modo obtemos um **Lucro Total** de **150**.

5 Exercicio 4

De forma a conseguir melhores resultados que no algoritmo na Figura 1 podemos acrescentar outra etapa de processamento. Deste modo depois de determinados os percursos no algoritmo anterior tenta-se substituir os vértices de maior prémio não visitados por visitados, um a um, para todos os percursos, sem nunca ultrapassar o limite de distância máxima para cada percurso e deste modo aumentar o lucro. O pseudo código encontra-se na Figura 2

Algoritmo 2 Pseudo código para o exercício 3.

```
graph = Array com todos os vértices.
dMax = Valor da distância máxima que cada viatura pode percorrer.
paths ← Caminhos mais curtos a partir do algoritmo anterior.
while Existir vertices não visitados por analisar do
  bestV ← Vértice de maior lucro não visitado.
  for path : paths (para todos os percursos) do
    for v : path (para todos os vértices do percurso) do
      if vb melhor que v (no sentido em que apresenta maior lucro) then
        Trocar o v por vb.
        if Solução inválida then
          Repor v.
        else
          Terminar procura e passar ao vértice seguinte.
        end if
      end if
    end for
  end for
end while
```

Podemos considerar que em caso de empate o melhor vértice é escolhido de forma aleatória, tal como o primeiro percurso onde se procura a substituição. Este seria um dos pontos a melhorar. É também de notar que não se especifica um caso de paragem, para tal pode propor-se duas soluções:

- O algoritmo itera sobre todos os vértices "livres"terminando quando deixa de se registar alterações nos percursos.
- O algoritmo itera sobre todos os vértices livres marcando os visitados e termina após análise de todos.

A decição acaba por apenas influenciar o numero de vezes que o ciclo é executado.

Sendo este algortimo moroso realizamos apenas uma primeira iteração: o vértice selecionado foi o 15 e o percurso o 4 sendo a troca realizada com o vértice 29. Deste modo o **Lucro Total** aumenta de 150 para **155**. Mais iterações acarrentam melhores resultados.