Universidade de Aveiro Departamento de Matemática
2012

Julieta Margarida Alexandre Lopes

O problema da gestão ótima da diversidade de cablagens

Universidade de Aveiro Departamento de Matemática
2011

Julieta Margarida Alexandre Lopes

O problema da gestão ótima da diversidade de cablagens

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica do Doutor Agostinho Miguel Mendes Agra, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro.

Dedico este trabalho aos meus pais, ao meu marido e ao meu filho.

o júri

presidente

Prof. Doutora Maria Paula Lopes dos Reis Carvalho
professora auxiliar da Universidade de Aveiro

Prof. Doutora Maria Adelaide da Cruz Cerveira
professora auxiliar da Universidade de Trás-Os-Montes e Alto Douro

Prof. Doutor Agostinho Miguel Mendes Agra
professor auxiliar da Universidade de Aveiro

agradecimentos Ao meu orientador, Dr. Agostinho Miguel Mendes Agra, pela disponibilidade e grande apoio.

palavras-chave

Problema da gestão ótima da diversidade, problema da p-mediana, heurística *Greedy*.

resumo

Neste trabalho, aborda-se o problema da gestão ótima da diversidade, que pode ser considerado um caso particular do problema da p-mediana, aplicado na indústria automóvel. Em primeiro lugar, descreve-se o problema e estudam-se algumas das suas caraterísticas combinatórias.

Sendo um problema de otimização combinatória classificado como NP-difícil, usualmente, aplicam-se na sua resolução métodos heurísticos. Nesta tese, estuda-se um algoritmo *Greedy* com detalhe, de modo a inferir estratégias que o tornem mais eficiente na resolução do problema. Assim, estudam-se algumas caraterísticas combinatórias desse algoritmo *Greedy*, para um caso particular, e apresentam-se algumas propriedades do mesmo.

De seguida, dado que na realidade o problema da gestão ótima da diversidade de cablagens pode ser decomposto em vários subproblemas, mais fáceis de resolver, apresenta-se a abordagem que consiste na aplicação dum algoritmo *Greedy* em duas etapas, numa primeira etapa o algoritmo é aplicado a cada subproblema e, numa segunda etapa, é aplicado para determinar a melhor forma de combinar as soluções dos vários subproblemas.

Por último, é feito um estudo estatístico, com uma amostra de 5 problemas reais de uma empresa de cablagens, de modo a analisar a existência de caraterísticas comuns na resolução dos mesmos, através dum algoritmo *Greedy*. A partir de alguns resultados observados nesse estudo e pelo facto de, para problemas de grande dimensão, esse algoritmo demorar mais tempo que o desejado a encontrar uma solução, consideram-se duas variantes do mesmo que resultam de restrições na pesquisa que é feita pelo algoritmo original, de forma a torná-lo mais rápido. É apresentado um pequeno estudo computacional para comparar as diferentes variantes do algoritmo.

keywords Optimal diversity management problem, p-median problem, *Greedy* heuristic.

abstract

This study covers the optimal diversity management problem, which can be seen as a particular case of p-median problem, applied to automotive industry. The starting point is the problem description as well as a study of some of its combinatorial characteristics.

As a combinatorial optimization problem classified as NP-hard, heuristic methods are often used on its resolution. In this thesis, a Greedy algorithm is studied in deep detail, in order to find some strategies to improve it for a more efficient problem resolution. Thus, some particular combinatorial characteristics of this Greedy algorithm are studied, for a particular case, and some of its properties are presented.

Thereafter, since optimal diversity management problem can be decomposed into several sub problems – easier to solve – an approach which consists in the application of a Greedy algorithm in two steps is presented: a first step where the algorithm is applied to each sub problem, and a second step where it is applied to determine the best way to combine the several sub problem solutions.

Finally, a statistical study is made with a sample of 5 real problems of a wiring harness company in order to analyze the existence of common characteristics in their resolutions, by applying a Greedy algorithm. Based on some results of the previous study and since for real world applications this algorithm may take longer than desired to find a solution, two variants of this algorithm are considered, those resulting from restrictions in the search made by the original algorithm in order to make it faster. A small computational study is presented in order to compare the several variants of the algorithm.

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

O problema da Gestão Ótima da Diversidade é um caso particular do problema da p-mediana [11,14,15]. No problema da p-mediana é considerado um conjunto de potenciais localizações de equipamentos, um conjunto de clientes, as distâncias entre os clientes e as potenciais localizações e uma constante *p*, que representa o número de equipamentos a localizar, ou seja, as medianas. Este problema pode ser representado num grafo onde os nós representam as localizações dos clientes e dos potenciais equipamentos e os arcos representam a possibilidade de um cliente ser servido por um equipamento. A cada arco é associado um peso que representa a distância entre os clientes e o equipamento. Neste problema, as potenciais localizações dos equipamentos coincidem com as localizações dos clientes. O objetivo é localizar os p equipamentos nos nós do grafo, de modo a minimizar a soma das distâncias de cada cliente ao equipamento que lhe fica mais próximo.

São diversas as aplicações do problema da p-mediana no dia-a-dia, por exemplo, na localização de hospitais, paragens de autocarro e antenas de telecomunicações, entre outras.

Nesta tese, estuda-se o Problema da Gestão Ótima da Diversidade de Cablagens (PGODC), um outro exemplo da aplicação do problema da p-mediana na indústria automóvel. O Problema da Gestão Ótima da Diversidade foi estudado extensivamente, pela primeira vez, em 2000, por Briant [7]. Em 2004, foi aplicado à indústria automóvel por Brian e Naddef [8].

Na indústria automóvel, existe uma enorme diversidade de possibilidades de configurações diferentes na produção da cablagem de cada modelo. A cablagem de um automóvel é constituída por um conjunto de componentes, cada uma composta por um determinado número de fios de cobre, em que cada fio corresponde a uma caraterística do automóvel. Dada essa enorme diversidade de opções para as configurações das cablagens, torna-se impossível para uma empresa produzi-las todas, quer por motivos económicos quer por questões técnicas e administrativas, sendo assim necessário substituir algumas configurações não produzidas por outras. Uma configuração poderá ser substituída por outra mais completa, contendo todos os fios de cobre que aquela teria e mais alguns que não serão utilizados. Esta substituição

acarreta custos adicionais para a produção da cablagem, pois estão a ser gastos fios de cobre não utilizados. Há, então, que garantir que sempre que uma configuração é substituída, esta seja substituída pela configuração mais barata que contém os fios de cobre necessários, de entre as produzidas. Torna-se necessário encontrar um meio de simplificar e otimizar todo o processo de fabrico, tendo em conta as escolhas dos clientes.

Assim, o PGODC consiste em selecionar *p* configurações a produzir de entre um conjunto de *m* configurações possíveis, com o objetivo de minimizar os custos de produção e satisfazer todos os pedidos dos clientes.

Como tal, o PGODC pode ser modelado como um problema da p-mediana num grafo orientado, em que cada nó do grafo se designa por configuração e corresponde a um cliente, as configurações a produzir correspondem aos equipamentos e uma configuração está ligada a outra se for substituída por ela.

Em 2004, segundo Briant e Naddef [8], as empresas consideravam a existência de 7000 configurações diferentes mas produziam apenas entre 6 e 40, dependendo dos fabricantes. Em 2005, Avella *et al* [5] refere que um grafo associado ao PGODC podia ter mais de 80 000 nós e 6000000 arcos, mas em geral, o número máximo de configurações produzidas por uma empresa era 60. Hoje em dia, sabe-se que um grafo pode ter 2 milhões de nós [2].

Briant em [7] e Agra *et al* em [1] provam que o PGODC teoricamente é classificado como NP-difícil, ou seja, não é conhecido nenhum algoritmo polinomial para este problema, e a menos que P=NP, não existe nenhum algoritmo polinomial para o resolver. Assim, é importante uma abordagem heurística para a sua resolução.

Empregam-se métodos heurísticos construtivos, que são algoritmos, em geral simples, que iterativamente constroem uma solução admissível para o problema.

O PGODC é um problema de otimização combinatória. O número de combinações a analisar na sua resolução aumenta exponencialmente com o tamanho do problema (como se verifica no Capítulo 3) e, portanto, o tempo de execução destes algoritmos cresce exponencialmente com a dimensão do problema. Assim, neste trabalho, estudam-se algumas caraterísticas combinatórias do PGODC que facilitam a procura da solução para o mesmo.

São vários os estudos já realizados acerca do PGODC, sendo a principal preocupação de todos eles encontrar uma forma de ultrapassar o facto de os problemas reais, apresentados pelas empresas, possuírem um elevadíssimo número de configurações, impraticável de analisar.

Em 2004, Briant e Naddef [8] usam a relaxação lagrangeana para reduzir o tamanho do problema de forma a ser possível resolvê-lo com otimalidade, através da programação linear inteira, naquele que é considerado o primeiro trabalho importante acerca do PGODC.

Em 2005, Avella *et al* [5] referiu, pela primeira vez, que o grafo associado ao PGODC poderia ser composto por várias componentes desconectadas entre si, e sendo assim, o PGODC (visto como um problema da p-mediana) poderia ser decomposto em vários subproblemas da p-mediana de menor dimensão, e portanto mais simples de resolver. Assim, usou a heurística *Greedy* (que se revelou mais rápida) e a heurística Lagrangeana (que se revelou de melhor qualidade) para cada subproblema, referindo que estas heurísticas permitiam resolver o problema com 1% de otimalidade, num tempo computacional razoável, e obteve a solução de todo o problema através do algoritmo *Branch&Bound*. Foram publicados resultados de estudos computacionais em [6].

Esta decomposição do problema em subproblemas pode ser modelada como um problema saco-mochila [4,13] e este pode ser resolvido eficientemente [4,10].

Em 2009, Agra *et al* [3] aborda novamente a questão da decomposição do problema, usando por duas vezes um algoritmo *Greedy:* em primeiro lugar executa o algoritmo para cada componente do grafo e para todas as possíveis escolhas de medianas, e em segundo lugar escolhe a melhor combinação de medianas (baseando-se nas soluções obtidas pelo algoritmo *Greedy*). Agra e Requejo, provam ainda em [4] que resolver o problema de uma só vez ou resolver o problema decomposto em subproblemas, conduz-nos exatamente à mesma solução e que podem ser obtidas boas soluções.

Assim, tem sido usado um algoritmo *Greedy* para resolver problemas reais de grande dimensão e os estudos realizados normalmente têm referido bons resultados [1,5,7]. Têm sido também estudados várias estratégias *Greedy* e algoritmos genéticos,

que usando informação do algoritmo *Greedy* têm revelado ainda melhores resultados [3].

Nesta tese, utiliza-se um algoritmo *Greedy* [1,2,5,7] dado que, na prática, este é simples de implementar e rápido de executar, produzindo, em geral, boas soluções num tempo computacional bastante razoável. No entanto, para problemas muito complexos que envolvam milhões de nós [2], esse algoritmo *Greedy* torna-se ainda bastante pesado na resolução dos subproblemas. Daí o interesse desta tese em estudá-lo em detalhe, de modo a inferir processos que o tornem mais rápido.

Em suma, os objetivos principais desta tese são estudar algumas caraterísticas combinatórias do PGODC e estudar um algoritmo *Greedy* e as suas caraterísticas combinatórias, procurando encontrar modos de resolver o PGODC de forma mais eficiente.

Assim sendo, no Capítulo 2 começa-se por descrever o PGODC e apresentam-se algumas formulações do mesmo.

Em seguida, no Capítulo 3, estudam-se algumas caraterísticas do PGODC, através da análise do grafo orientado, usando a Análise Combinatória. Procede-se a contagens, nomeadamente, do número de configurações de cada grafo e do número de configurações por nível do grafo, e apresentam-se resultados importantes para a resolução de problemas de grandes dimensões.

O Capítulo 4 introduz um algoritmo *Greedy* e apresenta alguns exemplos da sua utilização na resolução do PGODC, para grafos conexos de pequena dimensão.

O Capítulo 5 é dedicado à análise de algumas características combinatórias do algoritmo *Greedy* apresentado, num caso particular em que o grafo é completo (contém todas as configurações possíveis), todas as configurações têm procura 1 (ou constante positiva) e o custo de cada fio de cobre é unitário (ou constante positivo). Neste caso, apresentam-se alguns resultados que permitem resolver o PGODC de forma mais rápida, aplicando esse algoritmo *Greedy* com algumas restrições.

Como foi referido por Avella *et al* [5], na realidade o PGODC não corresponde a um grafo conexo de pequena dimensão, mas sim a um grafo com várias componentes conexas, em que resolver o problema corresponde a resolver vários subproblemas, cada um deles associado a uma componente conexa. Deste modo, no Capítulo 6, apresenta-se a decomposição do PGODC e um exemplo real de um PGODC, associado

a um grafo com 8 componentes, em que se aplica um algoritmo *Greedy*, tal como em [3], primeiro em cada subgrafo e depois na escolha do número de medianas de cada subgrafo.

No Capítulo 7 são apresentados cinco exemplos reais de PGODC de uma empresa de cablagens e é feito um estudo estatístico para averiguar a existência de padrões nas soluções obtidas. Este estudo torna-se muito importante dado que, para casos em que o grafo não inclui todos os nós, as procuras de cada nó e os custos de cada fio não são constantes, não há resultados teóricos comprovados. Depois de feito o estudo estatístico, são apresentadas duas variantes dum algoritmo *Greedy* que nos permitem resolver os PGODC mais rapidamente, apesar de elevarem um pouco o custo da solução.

Por último, no Capítulo 8, apresentam-se as conclusões sobre o trabalho realizado.

2. Problema da Gestão Ótima da Diversidade de Cablagens

Neste Capítulo, é apresentado o problema da Gestão Ótima da Diversidade de Cablagens (PGODC). Começa-se por descrever o problema, apresentando-se, de seguida, a modelação matemática do mesmo e, posteriormente, a sua formulação em programação linear inteira.

2.1 Introdução – Descrição do problema

A cablagem de um automóvel é constituída por uma imensa quantidade de fios de cobre, cada um com uma função. Por exemplo, um dos fios permite que o automóvel tenha ar condicionado, outro permite que tenha *airbag* para o condutor, etc. Deste modo, a quantidade de fios que cada automóvel possui está associada a um conjunto de funções que o caraterizam.

Esses fios de cobre estão distribuídos por vários módulos, sendo alguns deles incompatíveis quando não é possível que o automóvel tenha determinadas caraterísticas em simultâneo. Por exemplo, o módulo responsável pela localização do volante à esquerda é incompatível com o módulo correspondente ao volante à direita, mas é compatível com o módulo responsável pela existência de caixa automática. Além disso, há módulos de escolha obrigatória e módulos de escolha opcional. Por exemplo, é obrigatório escolher de que lado fica o volante, mas não é obrigatório escolher sensores de estacionamento ou ar condicionado. Ao longo deste trabalho são estudados apenas os módulos opcionais.

Assim sendo, cada cliente poderá escolher um conjunto de caraterísticas que deseja que o seu automóvel possua, dentro de um conjunto de opções possíveis, designadas opções livres, oferecidas pelo fabricante. Cada um desses conjuntos escolhidos pelos clientes resulta numa cablagem diferente para o automóvel, e corresponde a uma configuração diferente.

O PGODC surge devido ao elevado número de possibilidades existentes para a construção de cablagens. Dado que não é possível, por questões técnicas, que um

fabricante possa produzi-las todas, o que acontece é que algumas cablagens possuem mais módulos opcionais do que aqueles de que necessitam, reduzindo-se desta forma a diversidade de cablagens produzidas. Por exemplo, um cliente que escolheu ar condicionado, rádio com leitor de cd's e *airbag* para o condutor, poderá ter no seu automóvel uma cablagem que além destas funções tenha também um fio de cobre correspondente ao *airbag* do passageiro, que não está a ser usado.

Por conseguinte, o PGODC consiste em determinar, face aos pedidos dos clientes (ou à previsão desses pedidos), quais as configurações de cablagem que devem ser produzidas e, para cada configuração pedida, qual a configuração a entregar de modo a minimizar o custo global associado aos fios desnecessários.

De seguida, na secção 2.2, expõe-se o PGODC através de um grafo, em que cada nó representa uma configuração. Trata-se de um problema de otimização combinatória, que consiste em selecionar, de um conjunto discreto e finito de dados (nós do grafo), um subconjunto, que minimiza uma função designada custo, que mede o custo extra resultante de cada substituição efetuada.

O PGODC pode ser modelado como um problema da p-mediana, onde p representa o número de configurações distintas que serão produzidas, e resolver o problema consiste em encontrar essas p configurações de modo a minimizar a função custo já referida acima. Na secção 2.3 apresenta-se a formulação do problema desta forma.

2.2 Modelação matemática do PGODC

Nesta secção modela-se o problema da Gestão Ótima da Diversidade de Cablagens (PGODC).

Sejam n o número de opções possíveis para escolha, com $n \in \mathbb{N}$, e V o conjunto de configurações resultantes das escolhas possíveis. Cada configuração resulta num vetor com n componentes, que tomam os valores 0 ou 1, onde 1 significa que a opção foi escolhida (ou ativada) e 0 o contrário.

Sejam $i, j \in V$ que representam, respetivamente, o conjunto de opções que as escolhas i e j ativam.

Considere-se definida em V uma relação de ordem parcial $\lessdot$ tal que $i \lessdot j$ se e somente se a configuração i pode ser substituída por j, ou seja, se e somente se as opções ativadas em i também se encontram ativadas em j, com $i \neq j$.

Seja $G = (V, A)$, o grafo orientado associado à relação de ordem parcial $\lessdot$, onde V é o conjunto de todas as configurações e $A = \{(i,j) \epsilon\, V \times V | i \lessdot j\}$, ou seja A é constituído por todos os pares de configurações (i,j) tais que j inclui todas as opções ativas em i. Os elementos de V correspondem aos nós do grafo. Portanto, a cada configuração faz-se corresponder um nó.

Para cada opção $t \epsilon \{1, \ldots, n\}$ seja f_t o custo unitário dessa opção (ou seja, o custo do fio de cobre associado a essa opção).

Para cada configuração $i, j \in V$, sejam, respetivamente, c_i e c_j (c_i, $c_j > 0$) os custos unitários da produção de i e j e d_i e d_j o número de exemplares de i e j que devem ser produzidos.

Portanto, $c_i = \sum_{t=1}^{n} \delta_{it} f_t$, onde $\delta_{it} = \begin{cases} 1, se\ a\ opção\ t\ está\ na\ configuração\ i \\ 0, caso\ contrário \end{cases}$.

Verifica-se que se $i \lessdot j \Rightarrow c_i < c_j$, pois i tem menos escolhas ativas do que j.

Considere-se um exemplo para n=3, ou seja, são 3 as opções livres disponíveis para as caraterísticas de determinado automóvel: *airbag* para passageiro, *cruise control* e DVD. Os custos associados a cada uma das 3 opções são, respetivamente, 3, 12 e 6:

Opção		Custo do fio
1	*Airbag*	3
2	*Cruise control*	12
3	DVD	6

Tabela 2.1 - Opções livres e respetivos custos.

Neste caso,

V ={(1,1,1); (1,1,0); (1,0,1); (0,1,1), (1,0,0); (0,1,0); (0,0,1); (0,0,0)}.

Por exemplo, a configuração (1,1,0) significa que estão ativas as opções *airbag* para passageiro e *cruise control* e tem um custo unitário de 15 u.m., correspondente à soma dos custos das opções ativas.

O grafo orientado associado à relação de ordem parcial é o seguinte:

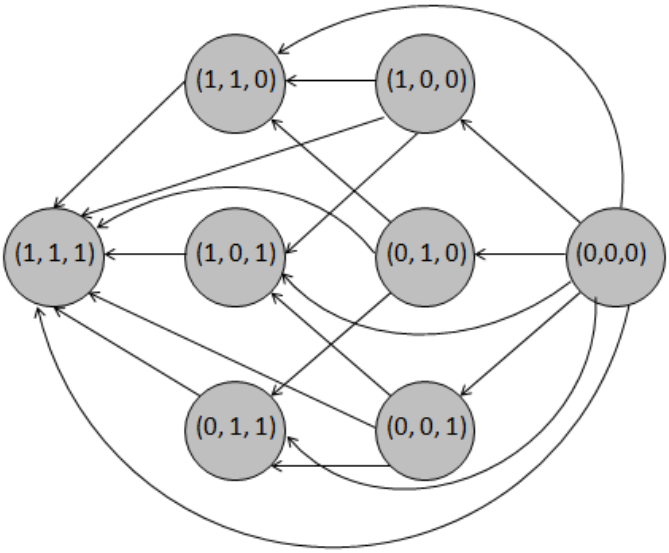

Figura 2.1 – Grafo associado ao problema em que n=3.

Considerem-se, agora, o número de pedidos (configurações a serem produzidas), correspondentes às configurações existentes e o custo de cada configuração:

Configurações	Nº de pedidos	Custo unitário
(1,1,1)	3	21
(1,1,0)	2	15
(1,0,1)	0	9
(0,1,1)	0	18
(1,0,0)	2	3
(0,1,0)	3	12
(0,0,1)	2	6
(0,0,0)	1	0

Tabela 2.2 - Número de pedidos e custos unitários das configurações.

Algumas das configurações pedidas poderão não ser produzidas, o que se pretende é saber quais devem ser produzidas tendo em vista a minimização de custos. Para tal, é necessário analisar qual o custo da substituição da configuração i pela configuração j.

Sejam $j \equiv (1,1,0)$ e $i \equiv (1,0,0)$. Então $i < j$. Em j estão ativas as opções 1 e 2, enquanto que em i está apenas ativa a opção 1.

Dado que $f_1 = 3$, $f_2 = 12$, $f_3 = 6$ e que $\delta_{j1} = \delta_{j2} = 1$, $\delta_{j3} = 0$, $\delta_{i1} = 1$ e $\delta_{i2} = \delta_{i3} = 0$, então c_j = 15 e c_i= 3.

Neste caso, o custo da substituição da configuração i pela configuração j é igual a 12, ou seja, $c_j - c_i$, o que corresponde ao custo das opções ativas em j que não serão utilizadas (neste caso, a opção 2 está ativa em j mas não está ativa em i). Adiante, designa-se este custo por c'_{ij}.

Assumindo que só podem ser produzidas 3 configurações, o problema de otimização consiste em escolher essas 3 configurações de modo a minimizar o custo total de substituição. Uma solução seria, por exemplo, a escolha das 3 configurações: (1,1,1); (1,1,0) e (1,0,0). Assim, as configurações (0,0,1) são substituídas pela configuração (1,1,1) com um custo de 15 u.m. cada, as configurações (0,1,0) são substituídas pela configuração (1,1,0) com um custo de 3 u.m. e a configuração (0,0,0) é substituída pela configuração (1,0,0) com um custo de 3 u.m.

A figura seguinte representa a solução.

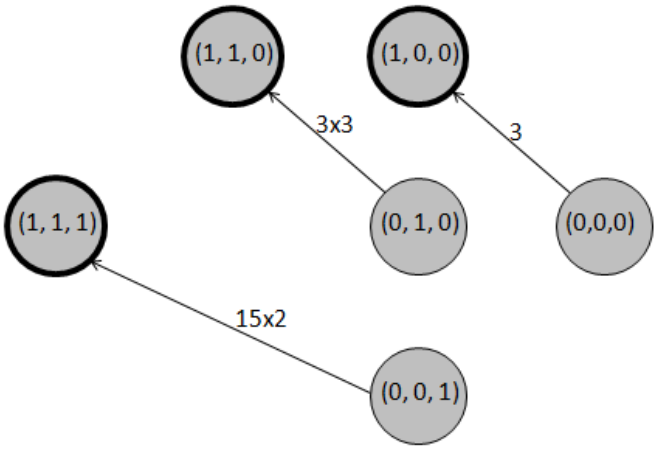

Figura 2.2 – Solução do problema da tabela 2.2.

2.3 Formulações em Programação Linear Inteira

Apresentam-se, nesta secção, várias formulações para o PGODC, em Programação linear inteira. As formulações de fluxos modelam todas as soluções admissíveis para o problema e as formulações de localização modelam um subconjunto de soluções admissíveis, baseando-se numa propriedade de otimalidade.

2.3.1 Formulações de Fluxos para o PGODC

Formulação forte

Sejam p o número máximo de configurações a produzir e $c_{ij}^{'}$ o custo resultante da substituição da configuração i pela j (ou seja, o custo das opções que estão ativas em j e não estão ativas em i), $(i,j) \in A$.

Considerem-se, ainda, as variáveis de decisão:

$$y_i = \begin{cases} 1, & se\ a\ configuração\ i\ é\ produzida \\ 0, & caso\ contrário \end{cases}, i \in V,$$

w_{ij} = número de unidades da configuração j que substituem a configuração $i, (i,j) \in A$, ou seja, o fluxo que passa no arco (i,j).

Pretende-se, então,

$$\min \sum_{(i,j)\in A} c'_{ij}\, w_{ij}$$

sujeito a:

$$\sum_{i\epsilon V} y_i = p, \quad (1)$$

$$\sum_{(ij)\epsilon A} w_{ij} = d_i(1 - y_i), \forall i \epsilon V, \quad (2)$$

$$w_{ij} \leq d_i y_j, \forall (i,j) \in A, \quad (3)$$

$$w_{ij} \epsilon \{0, \dots, d_i\}, \forall (i,j) \in A, \quad (4)$$

$$y_i \epsilon \{0,1\}, \forall i \epsilon V. \quad (5)$$

Onde,

(1) indica que têm de ser produzidas p configurações;

(2) indica que se a configuração i não é produzida então deve ser substituída por outra configuração (se $y_i = 0$ então $\sum_{(i,j)\epsilon A} w_{ij} = d_i$), se a configuração i for produzida não será substituída por nenhuma outra (se $y_i = 1$ então $\sum_{(i,j)\epsilon A} w_{ij} = 0$);

(3) indica se a configuração j é necessária para substituir alguma configuração então ela deve ser produzida, isto é, se $w_{ij} > 0$ então y_j=1;

(4) o número de unidades de i que são substituídas por j variam entre 0 e o máximo de unidades de i pedidas;

(5) a configuração i ou é produzida ou não é produzida.

Formulação fraca

Em vez das restrições (3) que estão escritas para cada par $(i,j) \in A$ individualmente, pode reformular-se o problema garantindo que se j substitui algum i, com $(i,j) \in A$, então j tem de ser produzido.

Assim, considere-se a seguinte formulação:

$$\min \sum_{(i,j)\in A} c'_{ij}\, w_{ij}$$

sujeito a:

$$\begin{aligned}
&\textstyle\sum_{i\epsilon V} y_i = p,\\
&\textstyle\sum_{j:(i,j)\epsilon A} w_{ij} = d_i(1 - y_i), \forall i \epsilon V,\\
&\textstyle\sum_{i:(i,j)\epsilon A} w_{ij} \le \left(\sum_{i:(i,j)\epsilon A} d_i\right) y_j, \forall j \in V, \qquad (6)\\
&w_{ij} \epsilon \{0, \dots, d_i\}, \forall (i,j) \in A,\\
&y_i \epsilon \{0,1\}, \forall i \epsilon V.
\end{aligned}$$

Onde (6) garante que se $w_{ij} > 0$, para algum i tal que $(i,j) \in A$, então y_j=1.

2.3.2 Formulações de Localização para o PGODC

As formulações que vão ser apresentadas nesta subsecção baseiam-se na observação de que a solução ótima, caso seja única, não será obtida através de alguns tipos de soluções.

Considere-se o seguinte exemplo:

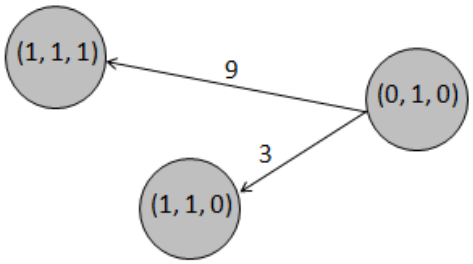

Figura 2.3 – Exemplo de um problema para n=3.

Suponha-se que são necessários dois exemplares da configuração (0,1,0), mas como esta não será produzida, suponha-se, também, que na solução obtida um desses exemplares é substituído pela configuração (1,1,1), com um custo igual a 9 u.m., e o outro é substituído pela configuração (1,1,0), com um custo igual a 3 u.m. O custo total será de 12 u.m.

Mas se os dois exemplares da configuração (0, 1, 0) fossem substituídos apenas pela configuração de menor custo, (1, 1, 0), o custo total seria de 6 u.m.

De facto, será mais vantajoso formular o problema dessa forma.

***Propriedade 1**:* Existe uma solução ótima para o PGODC que verifica o seguinte: $w_{ij} = 0$ ou $w_{ij} = d_i, \forall (i,j) \in A$.

Prova: Suponha-se que existe uma solução (x, y) que não satisfaz o seguinte: $w_{ij} = 0$ ou $w_{ij} = d_i, \forall (i,j) \in A$. Então, o número de unidades de j que substituem i não é igual a zero nem é igual ao número de exemplares de i que devem ser produzidos. Portanto, i não é produzida e tem de ser substituída por j e por k. Comparem-se, então, c'_{ij} e c'_{ik}. Se $c'_{ij} < c'_{ik}$, então a solução com $w_{ij} = d_i$ e $w_{ik} = 0$ é a solução mais barata. Caso contrário, a solução com $w_{ik} = d_i$ e $w_{ij} = 0$ tem um custo igual ou inferior.

Com base na propriedade anterior, podem desenvolver-se novas formulações que modelem as soluções que verificam $w_{ij} = 0$ ou $w_{ij} = d_i$. Neste caso, se a configuração i é substituída pela configuração j então todas as unidades de i são substituídas pelas da configuração j.

Verifica-se ainda que, se $i < j$, o custo adicional resultante da substituição de i por j, é dado por: $c_{ij} = d_i(c_j - c_i)$.

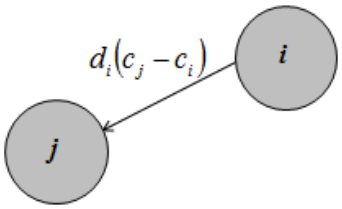

Figura 2.4 – Grafo que relaciona *i* com *j.*

Considere-se, novamente, o exemplo em que $j \equiv (1,1,0)$ e $i \equiv (1,0,0)$ e suponha-se que uma empresa necessita de 100 unidades de j e 200 unidades de i. Os custos unitários das opções 1 e 2 são, respetivamente, $f_1 = 3$ e $f_2 = 12$ e, por conseguinte, c_i=3, $c_j = 15$, e $d_i(c_j - c_i)$= 200(15 – 3) = 2400. De facto, o custo da produção de 200 unidades de i seria de 600 u.m. e o custo de 200 unidades de j será de 3000 u.m.

O custo da produção de 200 unidades de i e 100 de j seria de 2100 u.m e o custo da produção de 300 unidades de j será de 4500 u.m.

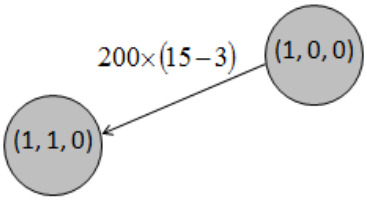

Figura 2.5 – Grafo ilustrativo do custo da substituição de *i* por *j.*

De seguida, formula-se o problema, com base na propriedade 1.

Formulação forte

Considerem-se as variáveis de decisão,

$$y_i = \begin{cases} 1, se\ a\ configuração\ i\ é\ produzida \\ 0, \qquad\qquad\qquad caso\ contrário \end{cases}'$$

$$x_{ij} = \begin{cases} 1, se\ a\ configuração\ i\ é\ substituída\ pela\ configuração\ j \\ 0, \qquad\qquad\qquad\qquad\qquad\qquad caso\ contrário \end{cases} \Leftrightarrow$$

$\Leftrightarrow x_{ij} = 1$ sse $w_{ij} = d_i$.

Formule-se, então, o problema:

Minimizar $c = \sum_{(i,j)\epsilon A} c_{ij}\, x_{ij}$

sujeito a

$$\sum_{i\epsilon V} y_i = p, \tag{7}$$

$$\sum_{(i,j)\epsilon A} x_{ij} + y_i = 1, \forall i \epsilon V, \tag{8}$$

$$x_{ij} \leq y_j, \forall (i,j) \in A, \tag{9}$$

$$x_{ij} \epsilon \{0,1\}, \forall (i,j) \in A, \tag{10}$$

$$y_i \epsilon \{0,1\}, \forall i \in V. \tag{11}$$

Onde:

(7) indica que têm de ser produzidas p configurações;

(8) indica que cada configuração $i \in V$ é produzida, $y_i = 1$, ou é substituída por alguma outra configuração $j \in V, j \neq i, \left(x_{ij} = 1\right)$;

(9) indica que se a configuração j substitui a configuração i, isto é , se $x_{ij} = 1$, então a configuração j tem de ser produzida;

(10) a configuração i é substituída pela configuração j ou não é substituída pela configuração j;

(11) a configuração i ou é produzida ou não é produzida.

Formulação fraca

Pode ainda reescrever-se o problema da seguinte forma:

Minimizar $c = \sum_{(i,j)\epsilon A} c_{ij}\, x_{ij}$

sujeito a

$$\begin{aligned} &\textstyle\sum_{i\epsilon V} y_i = p, \\ &\textstyle\sum_{(i,j)\epsilon A} x_{ij} + y_i = 1, \forall i \epsilon V, \\ &\textstyle\sum_{(i,j)\epsilon A} x_{ij} \leq M_j y_j, \forall j \in V, \\ &x_{ij}\epsilon\{0,1\}, \forall (\, i,j) \in A, \\ &y_i \epsilon \{0,1\}, \forall i \in V. \end{aligned} \qquad (12)$$

Onde (12) garante que se a configuração j substitui alguma configuração i então j tem de ser produzida, sendo M_j o número máximo de configurações que j pode substituir. Por exemplo, $M_j = \sum_{(i,j)\in A} d_i$.

2.3.3 Formulações de Fluxos versus Formulações de Localização

Enquanto que, nas formulações de fluxos $w_{ij}\epsilon\{0, \dots, d_i\}, \forall (\, i,j) \in A$, nas formulações de localização $w_{ij} = 0$ ou $w_{ij} = d_i, \forall (i,j) \in A$. As soluções excluídas pelas formulações de localização nunca darão a solução ótima, se esta for única, uma vez que todas as configurações não produzidas serão substituídas pela configuração mais barata, de entre as opções possíveis. Para além disso, o facto de as formulações de localização modelarem apenas um subconjunto das soluções admissíveis faz com que o problema seja resolvido com maior rapidez.

As formulações fortes são mais "apertadas" do que as fracas porque cortam soluções fracionárias, mas têm mais restrições o que torna a resolução da sua relaxação linear mais lenta.

Para resolver problemas de dimensão reduzida é possível usar um software comercial, como o XPRESS (ver [12]).

3. Características Combinatórias do PGODC

Neste Capítulo estudam-se algumas características combinatórias do PGODC. Começa-se por contabilizar o número de configurações de um grafo (ou o número de possibilidades para a cablagem de um automóvel) e de cada nível desse grafo, a partir do número de opções disponíveis para escolha (ou número de caraterísticas opcionais). Seguidamente, contabiliza-se o número de arcos de um grafo e de cada nível desse grafo, analisa-se quais os arcos que podem ser obtidos por transitividade através de outros e as consequências que daí advêm para o PGODC.

3.1 Número de Configurações

Designe-se por n o número de opções possíveis num determinado automóvel, ou seja, o número de opções disponíveis para escolha, com $n \in \mathbb{N}$.

Assume-se que todas as configurações são possíveis.

Se o cliente escolher as n opções existe apenas uma configuração (C_0^n) possível para a cablagem do seu automóvel: (1,1, …, 1).

Com $n-1$ opções existem C_1^n configurações possíveis para a cablagem: (0, 1, 1, …, 1), (1, 0, 1, 1, …, 1), …, (1, 1, …, 0, 1), (1, 1, …, 1, 0).

E assim sucessivamente, com $n-p$ opções, com $0 \leq p \leq n$ e $n, p \in \mathbb{N}$, existem C_p^n configurações possíveis.

Assim, a soma de todas as configurações possíveis será:

$$C_0^n + C_1^n + C_2^n + \cdots + C_{n-1}^n + C_n^n = 2^n.$$

De facto, trata-se da fórmula binomial de Newton, $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$, para a =1 e b =1 e n inteiro positivo: $(1+1)^n = \sum_{k=0}^{n} \binom{n}{k}$, ver [9].

Portanto, o grafo associado ao problema tem 2^n nós e cada um deles representa uma configuração diferente.

Note-se que, à medida que o número de opções livres aumenta, ou seja, que n aumenta, o número de possíveis configurações aumenta exponencialmente, segundo o modelo 2^n, como se observa na figura seguinte.

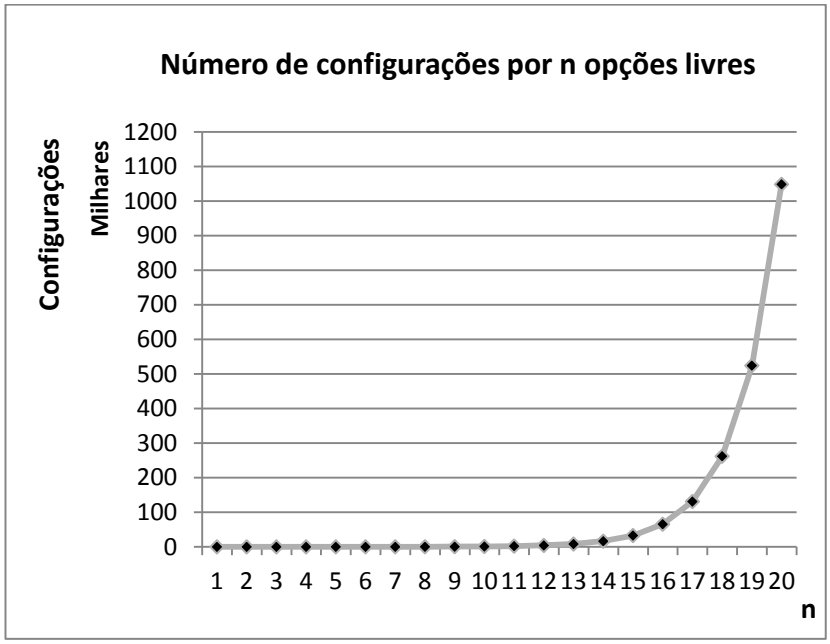

Figura 3.1 – Representação gráfica do número de configurações, em milhares, em função do número de opções.

No entanto, na prática algumas configurações podem não existir.

Relativamente ao número de configurações por nível, o grafo associado ao problema com n opções possíveis tem $n + 1$ níveis e cada nível corresponde a um número de opções ativas: no nível 0 temos C_0^n configurações possíveis, no nível 1 temos C_1^n, … e no nível n temos C_n^n. Portanto, no nível k temos C_k^n nós. No primeiro nível do grafo (nível 0) as opções estão todas ativas, isto é, o cliente escolheu n opções, no segundo nível $n - 1$ opções estão ativas, o que significa que o cliente escolheu $n - 1$ opções, e assim sucessivamente.

Observe-se, na figura seguinte, um exemplo do grafo para $n = 4$:

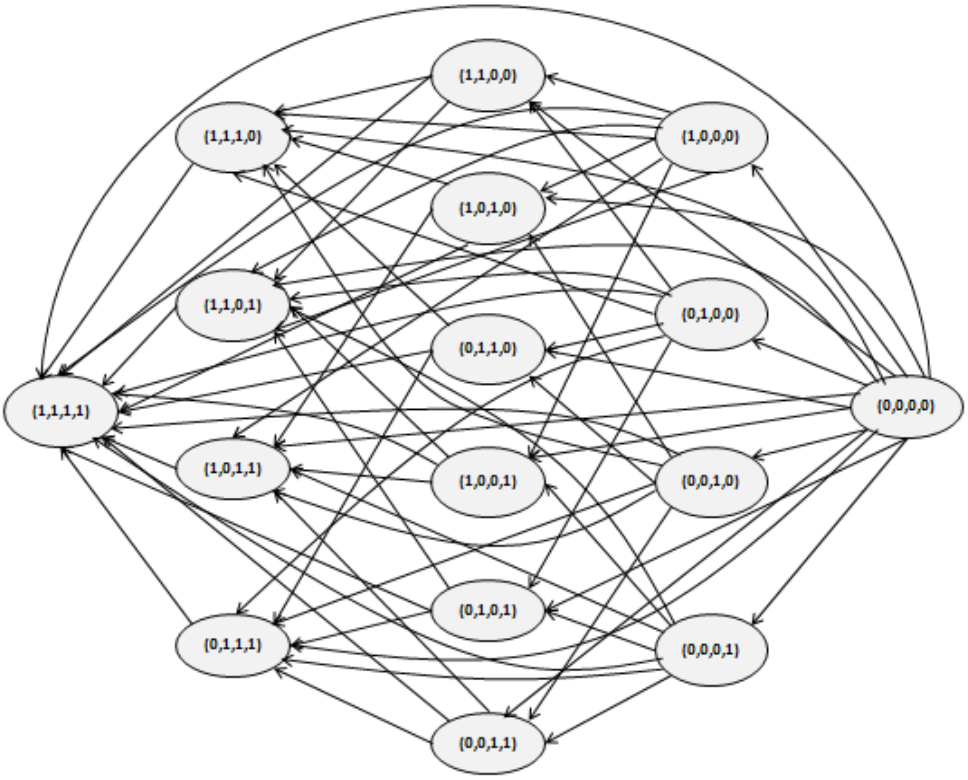

Figura 3.2 - Grafo associado ao problema em que n=4.

Este grafo tem 16 nós, correspondentes a 2^4configurações possíveis, e 5 níveis.

3.2 Número de arcos

Considere-se, ainda, o exemplo apresentado na secção 3.1, representado na figura 3.2. A partir deste exemplo, pode inferir-se qual o número de arcos de cada nível do grafo.

No nível 0 entram 15 arcos pois a configuração do nível 0 (C_0^4) poderá substituir todas as configurações dos níveis seguintes: C_1^4 do nível 1, C_2^4 do nível 2, C_3^4 do nível 3 e C_4^4 do nível 4. No total, pode substituir $C_1^4 + C_2^4 + C_3^4 + C_4^4 = 2^4 - 1$ configurações. Assim, $C_0^4 \times (2^{4-0} - 1) = 15$.

No nível 1 (correspondente a configurações com um zero e $n-1$ uns) entram 28 arcos, pois cada uma das C_1^4 configurações deste nível poderá substituir C_1^3 do nível 2, C_2^3 do nível 3 e C_3^3 do nível 4 (no total, $C_1^3 + C_2^3 + C_3^3 = 2^{4-1} - 1$): $C_1^4 \times (2^{4-1} - 1) = 28$.

Note-se que, a configuração j poderá substituir a configuração i desde que $i < j$. Por exemplo, se $j = (1,1,1,0)$, j pode substituir qualquer configuração que não tenha ativa a opção 4. Portanto, poderá substituir as configurações $(1,1,0,0)$, $(1,0,1,0)$, $(0,1,1,0)$ do nível 2, as configurações $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ do nível 3 e a configuração $(0,0,0,0)$ do nível 4.

No nível 2 entram $C_2^4 \times (2^{4-2} - 1) = 18$ arcos e no nível 3 $C_3^4 \times (2^{4-3} - 1) = 4$ arcos. No nível 4 entram $C_4^4 \times (2^{4-4} - 1) = 0$, ou seja, zero arcos divergentes.

No caso geral, para n opções, em cada nó do nível k do grafo entram $2^{n-k} - 1$ arcos. Portanto, no nível k entram $C_k^n \times (2^{n-k} - 1)$ arcos.

No total, o grafo tem $\sum_{k=0}^{n} C_k^n \times (2^{n-k} - 1)$ arcos.

3.3 Arcos que podem ser obtidos por transitividade

Considerando novamente o exemplo da figura 3.2, note-se que, por exemplo, alguns arcos que entram no nível 1 podem ser obtidos por transitividade a partir de nós do nível 2. Assim, o arco construído do nó $i = (0,0,1,0)$ para o nó $k = (1,1,1,0)$ pode ser obtido por transitividade através do nó $j = (1,0,1,0)$ pois existe um arco de j para k e outro de i para j.

***Propriedade 1**:* Sejam $i, j \; e \; k \; \epsilon \; V$. Se $i < j$ e $j < k$ então $i < k$.

Portanto, alguns dos arcos que entram no nível k podem ser obtidos por transitividade, isto é, se um nó i liga a um nó j e j liga a um nó k, então i também liga a k.

Note-se, ainda, que o número de opções ativas no nível k é igual a $n-k$.

Observe-se, também, que dos arcos que entram no nível 0, todos podem ser obtidos por transitividade, à exceção dos que saem do nível 1, e assim sucessivamente, todos os arcos que entram no nível k podem ser obtidos por transitividade, à exceção dos que saem do nível $k+1$.

Além disso, para cada nó do nível k, entram $2^{n-k}-1$ arcos dos quais $n-k$ arcos saem do nível seguinte, logo não podem ser obtidos por transitividade. Portanto, o número de arcos que entram no nível k e que podem ser obtidos por transitividade é dado por: $C_k^n \times [(2^{n-k}-1)-(n-k)]$.

Assim, o número de arcos estritamente necessários, ou seja, que não podem ser obtidos por transitividade é $C_k^n\ (n-k)$.

Proposição 1:

Sejam $T(n)$ *o* nº de arcos obtidos por transitividade *e* $N(n)$ *o* nº de arcos total. Verifica-se que,

$$\lim_{n\to\infty} \frac{T(n)}{N(n)} = 1,$$

ou seja,

$$\lim_{n\to\infty} \frac{C_k^n \times [(2^{n-k}-1)-(n-k)]}{C_k^n \times (2^{n-k}-1)} = 1$$

Prova:

$$\lim_{n\to\infty} \frac{C_k^n \times [(2^{n-k}-1)-(n-k)]}{C_k^n \times (2^{n-k}-1)} = \lim_{n\to\infty} \frac{C_k^n \times (2^{n-k}-1) - C_k^n(n-k)}{C_k^n \times (2^{n-k}-1)} =$$

$$= \lim_{n\to\infty} \frac{C_k^n \times (2^{n-k}-1)}{C_k^n \times (2^{n-k}-1)} - \lim_{n\to\infty} \frac{C_k^n \times (n-k)}{C_k^n \times (2^{n-k}-1)} =$$

$$= 1 - \lim_{n\to\infty} \frac{(n-k)}{(2^{n-k}-1)} = 1 - 0 = 1$$

De facto, analisem-se os resultados da tabela seguinte, para n configurações:

n	Nº de nós	Nº de arcos no grafo (a)	Nº de arcos no grafo, por transitividade(b)	$\frac{b}{a}$
1	2	1	0	0
2	4	5	1	0,200000000000000
3	8	19	7	0,368421052631579
4	16	65	33	0,507692307692308
5	32	211	131	0,620853080568720
6	64	665	473	0,711278195488722
7	128	2059	1611	0,782418649830015
8	256	6305	5281	0,837589214908803
9	512	19171	16867	0,879818475822857
10	1024	58025	52905	0,911762171477811
11	2048	175099	163835	0,935670677730884
12	4096	527345	502769	0,953396732689226
13	8192	1586131	1532883	0,966429002396397
14	16384	4766585	4651897	0,975939168188546
15	32768	14316139	14070379	0,982833360307552
16	65536	42981185	42456897	0,987801918444082
17	131072	129009091	127894979	0,991364081466166
18	262144	387158345	384799049	0,993906121279654
19	524288	1161737179,00000	1156756443,00000	0,995712682618725
20	1048576	3485735825,00000	3475250065,00000	0,996991808752461
25	33554432	847255055011,000	846835624611,000	0,999504953794587
30	1073741824,00000	205890058352825	205873952225465	0,999921773166277
35	34359738368,0000	5,00315107392613e+16	5,00309094438399e+16	0,999987981665703
40	1099511627776,00	1,21576643595453e+19	1,21576423693127e+19	0,999998191245300
45	35184372088832,0	2,95431267136646e+21	2,95431187971809e+21	0,999999732036362
50	1,12589990684262e+15	7,17897986565953e+23	7,17897958418455e+23	0,999999960791786

Tabela 3. 1 – Razão entre o nº de arcos obtidos por transitividade e o nº de arcos total.

Analise-se, agora, o gráfico que mostra a evolução do quociente $\frac{b}{a}$:

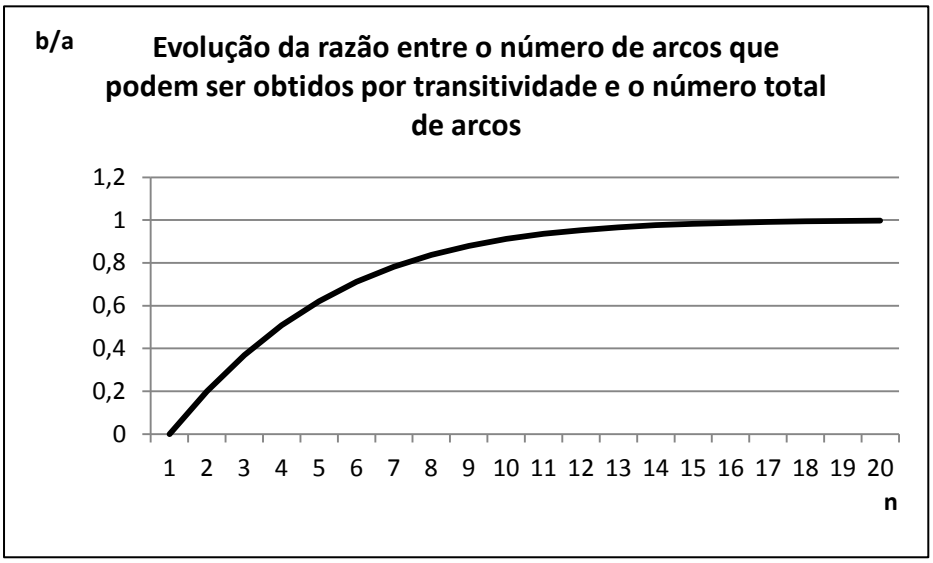

Figura 3.3 – Análise da evolução da razão entre o nº de arcos obtidos por transitividade e o nº de arcos total.

Assim, a proposição 1 assume uma enorme importância na resolução de um problema de grande dimensão, pois indica que não é necessário guardar a informação acerca dos custos relativos à substituição de determinado nó por todos os seus antecessores, mas apenas guardar a informação acerca dos arcos que não se obtêm por transitividade.

4. Algoritmo *Greedy*

Neste Capítulo apresenta-se um algoritmo *Greedy*, bem como alguns exemplos da sua aplicação em problemas simples.

4.1 Introdução

Um algoritmo *Greedy* é uma heurística que, iterativamente, constrói uma solução para determinado problema. Partindo de um conjunto inicial de vários nós que poderão fazer parte da solução e do número de nós que se pretende que a solução contenha, o algoritmo procura em cada iteração o melhor nó, de entre os disponíveis, para incluir na solução parcial. Ou seja, em cada iteração o algoritmo escolhe o nó que conduz a uma solução mais vantajosa, sendo essa escolha feita através de uma função que mede o benefício da introdução de cada um dos nós disponíveis na solução parcial. No entanto, o algoritmo não tem em conta se essa vantagem encontrada numa determinada iteração se manterá após as iterações e escolhas seguintes.

No caso do PGODC, escolhido o número de configurações a produzir (designado por número de medianas), o algoritmo constrói iterativamente uma solução que indica quais as configurações que devem ser produzidas, de forma a poupar o máximo possível nos custos de produção. É introduzida no algoritmo a função poupança, que se pretende maximizar. Em cada iteração, a função mede a poupança que advém da introdução de cada uma das configurações na solução e o algoritmo devolve a configuração mais vantajosa nessa iteração, ou seja, aquela que conduz à poupança máxima. No entanto, como já se referiu, não tem em conta após as iterações seguintes, se as escolhas feitas anteriormente ainda são as mais vantajosas.

De seguida, descreve-se um algoritmo *Greedy* e apresentam-se exemplos concretos da sua aplicação a problemas simples.

4.2 Descrição do Algoritmo

4.2.1 Exemplo

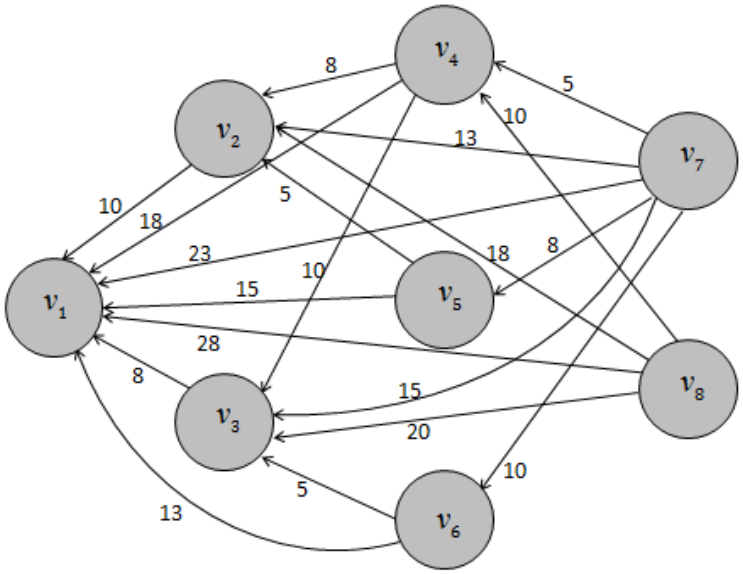

Figura 4.1- Grafo exemplificativo de um problema.

Suponha-se que vão ser produzidas as configurações v_1, v_3 e v_5. Designa-se por X, o conjunto dessas configurações: $X = \{v_1, v_3, v_5\}$.

Uma vez definidas as configurações a produzir, a solução final pode ser obtida facilmente substituindo cada nó v não produzido ($v \epsilon V/X$), pelo nó (em X) produzido que o substitui com menor custo.

Por exemplo, v_7 será substituído por v_5, pois $min\{c_{v_1\, v_7},\ c_{v_3\, v_7},\ c_{v_5\, v_7}\} = min\{23, 15, 8\} = 8 = c_{v_5\, v_7}$. Neste caso, diz-se que v_5 é o sucessor de v_7 na solução X e denota-se por $v_5 = s(v_7)$.

4.2.2 Pseudo-código do algoritmo *Greedy*

Sejam $V = \{v_1, v_2, \dots, v_n\}$ o conjunto de configurações possíveis, onde v_1 corresponde à configuração que contém todas as opções, p o número de configurações distintas a produzir, ou seja, o número de medianas da solução e c a matriz de custos (associados à substituição de uma configuração por outra).

O algoritmo pode descrever-se como se expõe de seguida.

Procedimento *Greedy* $\left(V, p, c\ (.,.)\right)\{$

$X \longleftarrow \{v_1\}$\\ conjunto de configurações selecionadas

enquanto $|X| < p\{$

calcular poupança $(v, \mathrm{X}), \forall v \in V \backslash X$ \\ poupança obtida por incluir v em X

$v^* \longleftarrow \mathrm{argmax}\{\text{poupança}(v, \mathrm{X}): v \in V \backslash X\}$

$X \longleftarrow XU\{v^*\}$

$\}$

retornar X

$\}$

Onde:

$\text{poupança}\ (v, \mathrm{X}) = \min_{\substack{\mathrm{i} \in \mathrm{X} \\ \mathrm{i} > v}} c_{iv} + \sum_{\substack{j \in V \backslash X \\ j < v}} max\left\{0, \min_{\substack{\mathrm{i} \in \mathrm{x} \\ (\mathrm{i},j) \epsilon A}} c_{ij} - c_{v_j}\right\}$, sendo $\min_{\substack{\mathrm{i} \in \mathrm{X} \\ \mathrm{i} > v}} c_{iv}$ a poupança relativa a v (o que significa que v deixa de ser substituído) e $\sum_{\substack{j \in V \backslash X \\ j < v}} max\left\{0, \min_{\substack{\mathrm{i} \in \mathrm{x} \\ \mathrm{i} > j}} c_{ij} - c_{v_j}\right\}$ a poupança relativa às configurações que v passa a substituir, com custo inferior ao da atual solução.

Pode ainda escrever-se:

$\text{poupança}\ (v, \mathrm{X}) = c_{s(v),v} + \sum_{\substack{j \in V \backslash X \\ j < v}} max\left\{0, c_{s(j),j} - c_{v_j}\right\}$,

sendo $s(v) = argmin_{\substack{i \epsilon X \\ i > v}} c_{iv}$, ou seja, $s(v)$ é o sucessor de v na atual solução X.

4.3 Exemplo da aplicação do algoritmo a um grafo

Recorde-se o exemplo da secção 2.2, em que n=3, sendo os custos associados a cada uma das 3 opções 3, 12 e 6, respetivamente.

O conjunto V, já descrito no exemplo, tem 2^3 elementos, ou seja, há 8 configurações possíveis.

Denotam-se por 1, 2, …, 8 cada uma das configurações, respetivamente. No grafo correspondente a configuração i corresponde ao nó v_i, com $1 \leq i \leq 8$, ou seja, $v_1 = (1,1,1)$; $v_2 = (1,1,0)$; $v_3 = (1,0,1)$; $v_4 = (0,1,1)$; $v_5 = (1,0,0)$; $v_6 = (0,1,0)$; $v_7 = (0,0,1)$; $v_8 = (0,0,0)$.

O número de pedidos, correspondentes às configurações existentes, e o custo de cada configuração encontram-se na tabela 2.2 da secção 2.2.

Assim, grafo associado ao problema, já com os custos associados às possíveis substituições, é o seguinte:

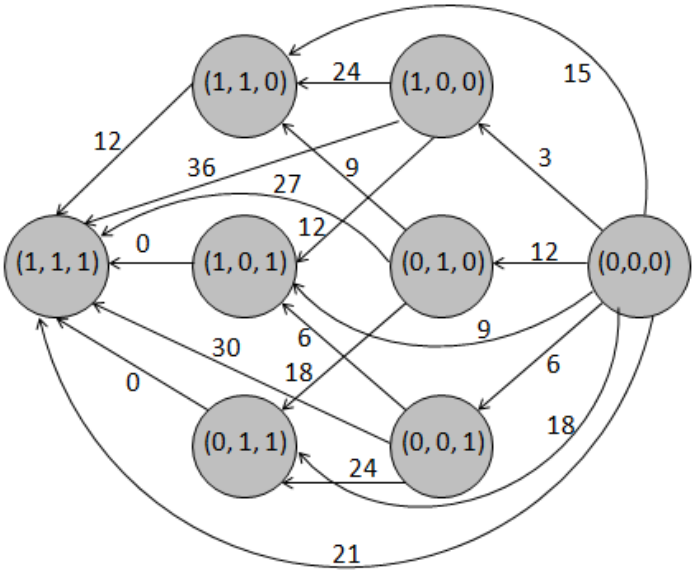

Figura 4.2 - Grafo associado ao problema descrito acima.

Consideram-se duas situações: na primeira incluem-se no algoritmo *Greedy* apenas as configurações pedidas e na segunda incluem-se no algoritmo todas as configurações. Em ambas as situações considera-se p=3.

Situação 1

Analise-se, então, o algoritmo *Greedy* para este exemplo:

Procedimento *Greedy* $(V, 3, c(i,j))$ onde $c(i,j) = \begin{bmatrix} 0 & 12 & 0 & 0 & 36 & 27 & 30 & 21 \\ 0 & 0 & 0 & 0 & 24 & 9 & 0 & 15 \\ 0 & 0 & 0 & 0 & 12 & 0 & 6 & 9 \\ 0 & 0 & 0 & 0 & 0 & 18 & 24 & 18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$X \leftarrow \{v_1\}$

Note-se que, o custo inicial corresponde ao custo resultante da substituição de cada uma das restantes configurações pedidas por v_1 (a única que é produzida) e é igual a: $c_{v_1v_2} + c_{v_1v_5} + c_{v_1v_6} + c_{v_1v_7} + c_{v_1v_8}$=12 + 36 + 27 + 30 + 21= 126 u.m.

Ou seja, desperdiçam-se, 126 u.m.

Enquanto $|X| < 3$

Calcular poupança $(v, \mathrm{X}), \forall v \in V \backslash X$

Portanto, pretende-se agora escolher as duas outras configurações a serem produzidas. Calcula-se, então, a poupança relativa à produção de cada uma das restantes configurações:

→ poupança $(v_2, \mathrm{X}) = c_{s(v_2),v_2} + \sum_{\substack{j \in V \backslash X \\ j < v_2}} max\{0, c_{s(j),j} - c_{v_2,j}\} =$

$= c_{v_1,v_2} + max\{0, c_{v_1,v_5} - c_{v_2,v_5}\} + max\{0, c_{v_1,v_6} - c_{v_2,v_6}\} + max\{0, c_{v_1,v_8} - c_{v_2,v_8}\} =$

$= 12 + max\{0, (36 - 24)\} + max\{0, (27 - 9)\} + max\{0, (21 - 15)\} = 48$

→ poupança $(v_5, \mathrm{X}) = c_{s(v_5),v_5} + \sum_{\substack{j \in V \backslash X \\ j < v_5}} max\{0, c_{s(j),j} - c_{v_5,j}\} =$

$= c_{v_1,v_5} + max\{0, c_{v_1,v_8} - c_{v_5,v_8}\} =$

$= 36 + max\{0, (21 - 3)\} = 54$

→ poupança $(v_6, \mathrm{X}) = c_{s(v_6),v_6} + \sum_{\substack{j \in V \backslash X \\ j < v_6}} max\{0, c_{s(j),j} - c_{v_6,j}\} =$

$= c_{v_1,v_6} + max\{0, c_{v_1,v_8} - c_{v_6,v_8}\} =$

$$= 27 + max\{0, (21 - 12)\} = 36$$

→ poupança $(v_7, \mathrm{X}) = c_{s(v_7),v_7} + \sum_{\substack{j \in V \setminus X \\ j < v_7}} max\{0, c_{s(j),j} - c_{v_7,j}\} =$

$$= c_{v_1,v_7} + max\{0, c_{v_1,v_8} - c_{v_5,v_8}\} =$$

$$= 30 + max\{0, (21 - 6)\} = 45$$

→ poupança $(v_8, \mathrm{X}) = c_{s(v_8),v_8} + \sum_{\substack{j \in V \setminus X \\ j < v_8}} max\{0, c_{s(j),j} - c_{v_8,j}\} =$

$$= c_{v_1,v_8} = 21$$

Assim, na primeira iteração do algoritmo obteve-se:

Nó v_i	v_2	v_5	v_6	v_7	v_8
poupança (v_i, X)	48	**54**	36	45	21

Tabela 4.1 - Tabela resumo da primeira iteração do algoritmo *Greedy,* do problema acima descrito.

De acordo com os dados da tabela, escolhe-se, v_5, a configuração que conduz a uma maior poupança:

$v^* \leftarrow \mathrm{argmax}\{\mathrm{poupança}(v, \mathrm{X}): v \in V \setminus X\} = v_5$

$\mathrm{X} \leftarrow XU\{v_5\} = \{v_1, v_5\}$

Então, para além da configuração correspondente a v_1, será também produzida a configuração correspondente a v_5. Como X ainda só tem 2 elementos, o algoritmo prossegue, sendo agora menor o desperdício resultante das substituições. O custo é igual a 72 u.m.

Na segunda iteração, obtém-se:

Nó v_i	v_2	v_6	v_7	v_8
poupança (v_i, X)	**30**	27	**30**	3

Tabela 4.2 - Tabela resumo da segunda iteração do algoritmo *Greedy,* do problema acima descrito.

Assim, escolhe-se, por exemplo, v_2.

$v^* \leftarrow \operatorname{argmax}\{\text{poupança}(v, \mathrm{X}): v \in V \backslash X\} = v_2$,

$\mathrm{X} \leftarrow XU\{v_2\} = \{v_1, v_2, v_5\}$.

Serão, então, produzidas as configurações associadas aos nós v_1, v_2, v_5. Portanto, o conjunto X já possui 3 elementos e o algoritmo termina aqui.

O desperdício, correspondente ao custo da substituição de v_6 por v_2, de v_7 por v_1, de v_8 por v_5 é igual a 42 u.m.

Como tal, tem-se:

Nós	Substituição	Configurações	Nº de pedidos	Custos de produção
v_1	Não	(1,1,1)	3	63
v_2	Não	(1,1,0)	2	30
v_5	Não	(1,0,0)	2	6
v_6	v_2	(0,1,0)	3	45
v_7	v_1	(0,0,1)	2	42
v_8	v_5	(0,0,0)	1	3
				Total=189

Tabela 4.3 - Tabela com a solução, através do algoritmo *Greedy,* do problema acima descrito.

Situação 2

Na primeira iteração do algoritmo, obtém-se:

Nó v_i	v_2	v_3	v_4	v_5	v_6	v_7	v_8
poupança (v_i, X)	48	**60**	18	54	36	45	21

Tabela 4.4 - Tabela resumo da primeira iteração do algoritmo *Greedy,* do problema acima descrito.

Escolhe-se v_3 e o algoritmo prossegue.

Na segunda iteração, obtém-se:

Nó v_i	v_2	v_4	v_5	v_6	v_7	v_8
poupança (v_i, X)	**30**	9	18	27	9	9

Tabela 4.5 - Tabela resumo da segunda iteração do algoritmo *Greedy,* do problema acima descrito.

Assim, serão produzidas as configurações associadas aos nós v_1, v_2 e v_3.
Como tal, tem-se:

Nós	Substituição	Configurações	Nº de pedidos	Custos de produção
v_1	Não	(1,1,1)	3	63
v_2	Não	(1,1,0)	2	30
v_3	Não	(1,0,1)	0	0
v_5	v_3	(1,0,0)	2	18
v_6	v_2	(0,1,0)	3	45
v_7	v_3	(0,0,1)	2	18
v_8	v_3	(0,0,0)	1	9
				Total=183

Tabela 4.6 - Tabela com a solução, através do algoritmo *Greedy,* do problema acima descrito.

Neste caso, é vantajoso incluir no algoritmo todas as configurações, mesmo as que não são pedidas, pois a sua produção reduz os custos globais.

4.4 Estudo da otimalidade do algoritmo *Greedy*

De seguida, estuda-se o algoritmo *Greedy* relativamente à otimalidade da solução que o algoritmo constrói. A seguinte proposição pode ser facilmente verificada.

Proposição 1:

O algoritmo *Greedy* que considera todas as configurações dá sempre a solução ótima para $p \leq 2$.

O algoritmo *Greedy* nem sempre dá a solução ótima. Veja-se o exemplo seguinte.

Considere-se $n = 3$.

As configurações a ser produzidas são: (1,1,1); (1,1,0), (1,0,0) e (0,1,0). Cada opção tem um custo de 5, 5, e 10, respetivamente para a 1ª, 2ª e 3ª opções.

Os pedidos e custos fixos de cada opção estão de acordo com a seguinte tabela:

Configurações	Pedidos	Custos
(1,1,1)	1	20
(1,1,0)	1	10
(1,0,0)	3	5
(0,1,0)	3	5

Tabela 4.7 – Configurações, número de pedidos e custos unitários.

O gráfico associado ao problema é o seguinte:

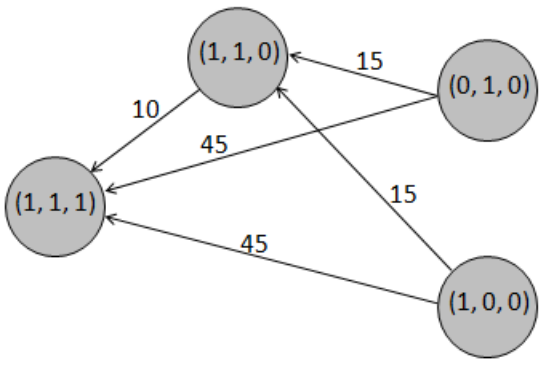

Figura 4.3 – Grafo associado ao problema descrito na tabela 4.7.

Aplique-se, então, o algoritmo *Greedy*:

Sejam,

$$v_1 = (1,1,1);\ v_2 = (1,1,0);\ v_3 = (0,1,0); v_4 = (1,0,0).$$

X⟵$\{v_1\}$

Note-se que o desperdício resultante da substituição das restantes configurações por v_1 é igual a 100 u.m.

Na primeira iteração, obtém-se:

Nó v_i	v_2	v_3	v_4
poupança (v_i, X)	70	45	45

Tabela 4.8 - Primeira iteração do algoritmo *Greedy,* para o problema da figura 4.3.

Considera-se, então, $v^* \longleftarrow v_2$. Assim, X⟵ $\{v_1, v_2\}$.

Como p= 3, o algoritmo prossegue e o desperdício é agora igual 30 u.m.

Na segunda iteração, obtém-se:

Nó v_i	v_3	v_4
poupança (v_i, X)	15	15

Tabela 4.9 - Segunda iteração do algoritmo *Greedy,* para o problema da figura 4.3.

Pode agora considerar-se $\mathrm{X} = \{v_1, v_{2,} v_{3,}\}$ ou $\mathrm{X} = \{v_1, v_{2,} v_{4,}\}$.

Portanto, se forem apenas produzidas 3 configurações, o algoritmo indica as seguintes:

(1,1,1); (1,1,0), (1,0,0) ou (1,1,1); (1,1,0) e (0,1,0). Em ambos os casos, há um desperdício de 15 u.m.

Assim, há duas soluções possíveis, representadas nas tabelas seguintes:

Configurações	Pedidos	Configurações produzidas	Custos de produção
(1,1,1)	1	1	20
(1,1,0)	1	1+3	40
(1,0,0)	3	3	15
(0,1,0)	3	0	0

Tabela 4.10 – Solução 1 do problema, obtida pelo algoritmo *Greedy.*

Configurações	Pedidos	Configurações produzidas	Custos de produção
(1,1,1)	1	1	20
(1,1,0)	1	1+3	40
(1,0,0)	3	0	0
(0,1,0)	3	3	15

Tabela 4.11 – Solução 2 do problema, obtida pelo algoritmo *Greedy.*

Nas duas opções, o custo total é de 75 u.m.

Contudo, a melhor opção é produzir (1,1,1); (1,0,0) e (0,1,0), tal como indica a tabela seguinte:

Configurações	Pedidos	Configurações produzidas	Custos de produção
(1,1,1)	1	2	40
(1,1,0)	1	0	0
(1,0,0)	3	3	15
(0,1,0)	3	3	15

Tabela 4.12 - Solução não obtida pelo algoritmo *Greedy.*

Nesta opção, o custo total é de 70 u.m, existindo um desperdício menor.

5. Propriedades combinatórias do algoritmo *Greedy* num caso particular

Recorrendo à Análise Combinatória, é possível determinar em que nível estão os nós escolhidos para a solução de um problema, dada pelo algoritmo *Greedy*, no caso em que o grafo tem uma estrutura particular: inclui todos os nós, a procura de cada nó é igual a 1 (ou constante positiva) e o custo por cada fio é unitário (ou constante positivo).

Constatou-se no Capítulo 3 que, por exemplo, cada nó do nível 1 da figura 3.2 pode substituir C_1^3 nós do nível 2, C_2^3 do nível 3 e C_3^3 do nível 4, ou seja, cada nó do nível 1 tem C_1^3 antecessores no nível 2, C_2^3 antecessores no nível 3 e C_3^3 antecessores no nível 4. No total, cada nó do nível 1 tem $2^{4-1} - 1$ antecessores.

De facto, note-se que, no nível k, com $k\epsilon\{1, \dots, n-1\}$, cada nó tem C_1^{n-k} antecessores (ou nós cobertos) no nível k+1, C_2^{n-k} antecessores no nível k+2, C_3^{n-k} antecessores no nível k+3, e assim sucessivamente, sendo que no nível $n-1$ (penúltimo nível) cada nó tem C_{n-k}^{n-k} antecessores. O nó do nível zero tem $2^{n-k} - 1$ antecessores e o nó do nível n não tem antecessores. No total, cada nó do nível k tem $2^{n-k} - 1$ antecessores.

No algoritmo *Greedy*, o primeiro nó é escolhido na inicialização do algoritmo e pertence ao nível zero, é aquele que tem 1 em todas as opções, (1,1,1, …, 1), e portanto, aquele que poderá substituir qualquer um dos restantes. O segundo nó será escolhido de forma que a poupança seja máxima, e assim sucessivamente.

Ao longo deste capítulo, considera-se que a procura de cada nó e o custo de cada ligação são unitários. No entanto, os resultados que aqui se apresentam são válidos para o caso em que as procuras e os custos são constantes (positivas).

Propriedade 1: Para a escolha do segundo nó, pelo algoritmo *Greedy*, a poupança no nível k, $p(k)$, é dada pelo produto de k com o número de arcos que entram em cada nó desse nível mais o próprio nó:

$p(k) = k \times (2^{n-k} - 1) + k = k \times 2^{n-k}$,

onde k é a poupança que se obtém por substituir o nó com todas as opções pelo nó no nível k, relativamente a cada antecessor deste último nó e a ele próprio.

Proposição 1: O segundo nó a ser escolhido através do algoritmo *Greedy* pertence ao nível 1.

Prova: Pretende-se obter a poupança máxima: $max\{p(k)\} = max\{k \times 2^{n-k}\}$.

Seja k =1, $p(1) = 2^{n-1}$.

No geral, para o nível k, $p(k) = k \times 2^{n-k}$.

Aplicando o Lema 3 (para $k=0$ e $t=k$), em apêndice, para $k \geq 2$ temos que $p(k) \leq p(1)$.

Portanto, a função poupança é decrescente em função do nível, logo a poupança máxima é obtida no nível 1.

Proposição 2: O terceiro nó a ser escolhido, através do algoritmo *Greedy,* pertence ao segundo nível e não é antecessor do nó já escolhido no primeiro nível.

Prova:

Na escolha do terceiro nó, podem ocorrer duas situações:

1) O nó escolhido pertence ao nível 1;
2) O nó escolhido pertence a um nível k, superior a 1.

Designe-se por i o nó a instalar e calcule-se, então, a poupança que se obtém em cada uma das situações.

1) Seja A a poupança que se obtém escolhendo um nó do nível 1.

A poupança no nível 1 é igual ao número de antecessores do nó i que não são antecessores do nó já instalado em 1. Ou seja, é igual a todos os antecessores de i com 1 na posição onde o nó já instalado tem o zero, isto é, metade dos seus antecessores (incluindo o próprio i): $= \frac{2^{n-1} - 1 + 1}{2} = 2^{n-2}$.

Assim, $A = 2^{n-2}, n \epsilon \mathbb{N}$.

2) Nesta situação há, ainda, que distinguir dois casos.

   a) i é antecessor do vértice instalado no nível 1.

Seja $B(k)$ a poupança que se obtém pela escolha de um nó do nível k, antecessor do vértice já instalado no nível 1, com $k\epsilon\{2, \dots, n\}$.

Neste caso, a poupança é igual ao número de antecessores de i, incluindo o próprio i, multiplicado por $(k-1)$ que é a poupança relativa ao vértice no nível 1: $(k-1) \times (2^{n-k} - 1 + 1) = (k-1) \times (2^{n-k}) = (2k-2)(2^{n-k-1})$.

Assim, $B(k) = (2k-2)(2^{n-k-1}),\ n\epsilon\mathbb{N},\ k\epsilon\{2, \dots, n\}$.

b) i não é antecessor do vértice instalado no nível 1.

Seja $C(k)$ a poupança que se obtém pela escolha de um nó do nível k, que não seja antecessor do vértice já instalado no nível 1, com $k\epsilon\{2, \dots, n\}$.

Neste caso, a poupança é igual ao número de antecessores de i, incluindo o próprio, que não são antecessores do vértice instalado no nível 1, multiplicado por k (poupança relativa ao vértice instalado no nível 0), e ao número de antecessores de i que são também antecessores do vértice instalado no nível 1, multiplicado por $(k-1)$.

Note-se que, os antecessores de i com 1 na posição onde o vértice instalado no nível 1 tem zero não são antecessores do vértice do nível 1, ou seja, metade dos antecessores de i não são antecessores do vértice do nível 1. Além disso, os antecessores de i com 0 na posição onde o vértice do nível 1 tem 0, são antecessores deste, ou seja, metade dos antecessores de i são antecessores do vértice do nível 1. Assim, tem-se:

$(k-0)(2^{n-k-1}) + (k-1)(2^{n-k-1}) = (2k-1)(2^{n-k-1})$.

Logo, $C(k) = (2k-1)(2^{n-k-1}),\ n\epsilon\mathbb{N},\ k\epsilon\{2, \dots, n\}$.

Basta, agora, calcular em qual das situações a poupança é máxima.

Analisem-se, em primeiro lugar, $C(k)$ e $B(k)$, para para $k \geq 2$.

Para $n \geq 4$, $2k-1 > 2k-2$ e $2^{n-k-1} > 0$, logo $C(k) \geq B(k)$, para $k \geq 2$.

Analise-se, agora, a relação entre $C(k)$ e A.

Para $k = 2$, $C(k) = 3(2^{n-3}) \geq 2(2^{n-3}) = 2^{n-2} = A$. Logo $C(k) \geq A$, para $k = 2$.

Pelo Lema 2, em apêndice, conclui-se que $C(k) \geq C(k+1)$ e portanto a poupança máxima é obtida em $C(2)$, ou seja, na situação 2b) para k =2.

Proposição 3: Após p iterações do algoritmo *Greedy,* seja k o maior nível onde foi escolhido um nó para fazer parte da solução. Na iteração $p + 1$ é apenas necessário considerar os primeiros $k + 1$ níveis.

Prova:

Compare-se a poupança por instalar no nível $k + 1$ com a poupança de instalar no nível $k + t$, para $t \geq 2$.

Seja j o nó a instalar no nível $k + t$, para $t \geq 2$, e seja i um nó antecessor de j, pertencente ao nível $k + 1$.

Pretende-se mostrar que $p(i) - p(j) \geq 0$.

$$p(i) - p(j) \geq$$

$$1 \times (nº\ de\ antecessores\ de\ i + 1) - t \times (nº\ de\ antecessores\ de\ j + 1).$$

Na escolha de i poupa-se pelo menos 1 por cada antecessor de i e por i, pois todos estes nós estão a ser cobertos por nós do nível 0 ao nível k.

Relativamente aos antecessores de j e a j, poupa-se, pelo menos, t unidades.

$$p(i) - p(j) \geq 1 \times \left(2^{n-(k+1)}\right) - t \times \left(2^{n-(k+t)}\right) =$$

$$= 2^{n-(k+1)} - t \times 2^{n-(k+t)} \geq 0, \quad \text{para} \quad t \geq 2, \quad \text{pelo Lema 3, em apêndice.}$$

6. Decomposição do Problema da Gestão Ótima da Diversidade de Cablagens

Neste Capítulo apresenta-se a decomposição do PGODC, assim como um exemplo de um problema real resolvido através do algoritmo *Greedy*, em duas fases.

6.1 Exemplo

Na realidade, o PGODC é usualmente composto por um grafo com várias componentes conexas e pode ser decomposto em vários subproblemas, correspondentes a cada uma das componentes, dado que as configurações a produzir são escolhidas analisando os vários subgrafos. Assim, a solução do problema será obtida através das soluções dos subproblemas, que serão muito mais simples de resolver dado o menor número de configurações envolvidas em cada um deles.

Vejamos um exemplo para um grafo com 4 componentes, em que cada componente tem como opções livres *airbag*, *cruise control* e DVD.

Componentes / Configurações	Subgrafo 1 Volante à esquerda e caixa manual (1,1)	Subgrafo 2 Volante à esquerda e caixa automática (1,0)	Subgrafo 3 Volante à direita e caixa manual (0,1)	Subgrafo 4 Volante à direita e caixa automática (0,0)
(1, 1,1)	(1,1,1,1,1)	(1,0,1,1,1)	(0,1,1,1,1)	(0,0,1,1,1)
(1,1,0)	(1,1,1,1,0)	(1,0,1,1,0)	(0,1,1,1,0)	(0,0,1,1,0)
(1,0,1)	(1,1,1,0,1)	(1,0,1,0,1)	(0,1,1,0,1)	(0,0,1,0,1)
(0,1,1)	(1,1,0,1,1)	(1,0,0,1,1)	(0,1,0,1,1)	(0,0,0,1,1)
(1,0,0)	(1,1,1,0,0)	(1,0,1,0,0)	(0,1,1,0,0)	(0,0,1,0,0)
(0,1,0)	(1,1,0,1,0)	(1,0,0,1,0)	(0,1,0,1,0)	(0,0,0,1,0)
(0,0,1)	(1,1,0,0,1)	(1,0,0,0,1)	(0,1,0,0,1)	(0,0,0,0,1)
(0,0,0)	(1,1,0,0,0)	(1,0,0,0,0)	(0,1,0,0,0)	(0,0,0,0,0)

Tabela 6.1 – Exemplo de um problema associado a um grafo com 4 componentes.

Neste exemplo, serão produzidas apenas 11 configurações, de entre as várias possibilidades. Quatro têm que ser obrigatoriamente produzidas: aquelas que têm todas as opções ativas, para cada componente. Assim, resta escolher 7.

Note-se que, uma configuração de uma componente não pode substituir uma configuração de outra componente. Por exemplo, a configuração (1,1,1) da componente 1 não pode substituir a configuração (1,1,1) da componente 2. Na realidade, a configuração (1,1,1) da componente 1 designa-se (1,1,1,1,1) e a da componente 2 designa-se (1,0,1,1,1).

Sabendo que poderão ser produzidas 11 configurações, ou seja, se $p = 11$, uma solução poderá ser composta por 2 configurações do subgrafo1, 4 do subgrafo 2, 2 do subgrafo 3 e 3 do subgrafo 4. A escolha das configurações a produzir terá sempre como objetivo minimizar os custos de produção.

6.2 (Re)Formulação do PGODC

O PGODC pode ser reformulado da seguinte forma.

Sejam m o número de componentes, $K = \{1, 2, \dots, m\}$ o conjunto dos índices das componentes conexas e $G_k = (V_k, A_k)$ o subgrafo correspondente à componente k, com $k \in K$. Consideremos, ainda, um número inteiro positivo p_k, com $k \in K$, que representa o número de medianas escolhidas na componente k (para cada componente podem ser produzidas de 1 até $p - m + 1$ configurações, sendo p o número total de medianas).

O problema consiste em :

Minimizar $c = \sum_{(i,j)\epsilon A} c_{ij}\, x_{ij}$

sujeito a

$$\sum_{i\epsilon V} y_i = p_k, k\epsilon K, \quad (7a)$$

$$\sum_{k\epsilon K} p_k = p, \quad (7b)$$

$$\sum_{(i,j)\epsilon A} x_{ij} + y_i = 1, \forall i\epsilon V_k, k\epsilon K, \quad (8)$$

$$x_{ij} \leq y_j, \forall (i,j) \in A_k, k \epsilon K, \quad (9)$$

$$x_{ij} \epsilon \{0,1\}, \forall (i,j) \in A_k, k \epsilon K, \quad (10)$$

$$y_i \epsilon \{0,1\}, \forall i \in V_k, k \epsilon K, \quad (11)$$

$$p_k \in \{1, \dots, p-m+1\}, k \epsilon K. \quad (12)$$

Onde:

(7a) indica o número de configurações, $p_k, k \epsilon K$, que serão escolhidas em cada componente;

(7b) indica que a soma das configurações escolhidas em cada uma das componentes tem que ser igual ao número total de configurações a produzir, p;

(12) indica que o número de configurações escolhidas em cada componente varia de 1 até $p-m+1$.

As restrições (8), (9), (10) e (11) são semelhantes às que foram já descritas na subsecção 2.3.2.

O PGODC pode resolver-se em duas fases, numa primeira fase, pelo facto de o grafo não ser conexo, decompõe-se o problema em m subproblemas, um para cada componente, fazendo variar o número de medianas de 1 até $p-m+1$, e de seguida, resolve-se o PGODC em cada componente. Daqui resulta uma tabela de custos γ_{lk}, em que γ_{lk} é o custo ótimo de escolher l configurações no subgrafo k.

Numa segunda fase, escolhe-se a melhor forma de combinar as p configurações entre todas as componentes. Para isso, pode resolver-se o problema da decomposição, formalizado da seguinte forma:

Minimizar $\sum_{l=1}^{m-p+1} \sum_{k=1}^{m} \gamma_{lk} z_{lk}$

sujeito a

$$\sum_{l=1}^{m-p+1} \sum_{k=1}^{m} l \, z_{lk} = p \quad (13)$$

$$\sum_{l=1}^{m-p+1} z_{lk} = 1, \forall k \in K \quad (14)$$

$$z_{lk} \epsilon \{0,1\} \quad (15)$$

Onde:

(13) indica que, no total, têm que ser produzidas p configurações;

(14) indica que para cada configuração l, o número de configurações escolhido varia entre 1 e $p - m + 1$;

(15) z_{lk} é igual a 1 (se são escolhidas l configurações no subgrafo k) ou zero (caso contrário).

Designa-se o problema anterior por problema da decomposição ótima (PDO).

O PDO pode ser modelado como um caso particular de um problema saco-mochila de escolha múltipla ([4], [13]) e pode ser resolvido de forma eficiente, segundo Agra e Requejo em [4] e Cardoso e Cerdeira em [10].

Exemplo

Considere-se um exemplo de um problema real, com 8 componentes onde serão produzidas 11 configurações. Numa primeira fase, resolvem-se os 8 subproblemas para os várias valores possíveis para o número de configurações (este varia de 1 até 11– 8 +1). As soluções ótimas para os subproblemas encontram-se na tabela abaixo.

Componentes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1 configuração	63106.8	7011.2	15776.5	1751.9	510588	56731.3	127646	14182.4
2 configurações	35117.4	3901.4	8779.3	974.4	284130	31569.4	71032	7891.9
3 configurações	27604.4	3066.6	6901.1	765.8	223344	24815.3	55835.4	6203.3
4 configurações	23772.9	2640.9	5943.1	659.4	192342	21370.7	48085.1	5342.2

Tabela 6.2 – Exemplo de um PDO associado a um grafo com 8 componentes.

Das 11 configurações a produzir resta escolher 3, pois sabe-se que será produzida uma configuração de cada componente, nomeadamente aquela que contém todas as opções ativas. Essas configurações serão escolhidas calculando a poupança máxima, ou seja, escolhe-se uma configuração da componente que permitir uma maior poupança. Assim, neste caso, são escolhidas mais 2 configurações da componente 5 e uma da componente 7.

6.3 Algoritmo *Greedy* para o PDO

Para resolver o PDO, pode usar-se o algoritmo *Greedy* em duas fases distintas. Numa primeira fase, aplica-se o algoritmo *Greedy* a cada um dos m subproblemas (correspondentes às m componentes do grafo), fazendo variar o número de medianas de 1 até $p - m + 1$, sendo p o número de medianas pretendido. Daqui resulta uma tabela de valores, com as soluções dos vários subproblemas para os vários valores de p.

Numa segunda fase, determina-se a melhor forma de combinar as p medianas entre as m componentes. Assim, usa-se o algoritmo *Greedy* para o PDO, para a escolha das p configurações, tal como se descreve de seguida.

Procedimento *Greedy* $\left(m, p\ , \gamma\ (.,.) \right)\{$

$l \longleftarrow \{l_1, l_2, \ldots, l_m\}$ \\ indica o número de medianas que são instaladas em cada componente

$l_k \longleftarrow 1$ \\ l_k dá a posição (número de medianas) na componente k

$conta = m$

$$custo = \sum_{k=1}^{m} \gamma_{1k}$$

enquanto $conta < p$ {

fazer:

$p_k = \gamma_{1k} - \gamma_{2k};$ \\ poupança obtida por escolher uma configuração da componente k

$k^* \leftarrow argmax\{p_k : k \in K\};$

$l_{k^*} \leftarrow l_{k^*} + 1;$

$custo \leftarrow custo - p_{k^*};$

$conta = conta + 1;$

se $l_{k^*} < m - p + 1$ então $p_{k^*} = \gamma_{l_{k^*},k^*} - \gamma_{l_{k^*+1},k^*}$

}

retornar l

}

Foi provado em Agra e Requejo, 2009 [4], que este procedimento de resolução do PDO, fornece exatamente a mesma solução que executar o algoritmo *Greedy* sobre todo o grafo do PGODC.

Exemplo da aplicação do algoritmo *Greedy* a um grafo com 8 componentes.

Considere-se novamente o exemplo da secção 6.2, mas agora resolvido através do algoritmo *Greedy*.

Para cada componente, calcula-se o custo com 1, 2, até $p - m + 1$ medianas, ou seja, com 1, 2, 3 e 4 configurações. Os resultados obtidos, para cada um dos subproblemas, usando o algoritmo *Greedy*, encontram-se na tabela seguinte:

Componentes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1 configuração	63106,7	7011,2	15776,5	1751,9	510588,3	56731,3	127646,4	14182,4
2 configurações	35117,4	3901,4	8779,3	974,5	284129,8	31569,4	71031,9	7891,9
3 configurações	27604,4	3066,6	6901,1	765,8	223343,5	24815,3	55835,4	6203,3
4 configurações	23848	2649,2	5961,8	661,4	192950,3	21438,3	48237,1	5359,1

Tabela 6.3 Exemplo de um PDO associado a um grafo com 8 componentes, resolvido usando o algoritmo *Greedy*.

Depois de calcular os custos das várias configurações em cada componente, o algoritmo calcula o valor da poupança da escolha de uma nova configuração em cada componente, da seguinte forma:

Procedimento *Greedy* $\left(8{,}11\,,\gamma\,(.,.)\right)$, onde γ é uma matriz composta pelos valores da tabela 6.3.

$l \leftarrow \{l_1, l_2, \dots, l_8\}$

$l_k \leftarrow 1$

$conta = 8$

$$custo = \sum_{k=1}^{m} \gamma_{1k} = 796794{,}7$$

enquanto $conta < 11$

fazer:

$p_1 = \gamma_{11} - \gamma_{21} = 63106{,}7 - 35117{,}4 = 27\,989{,}3;$

$p_2 = \gamma_{12} - \gamma_{22} = 7011{,}2 - 3901{,}4 = 3\,109{,}8;$

$p_3 = \gamma_{13} - \gamma_{23} = 15776{,}5 - 8779{,}3 = 6\,997{,}2;$

$p_4 = \gamma_{14} - \gamma_{24} = 1751{,}9 - 974{,}5 = 777{,}5;$

$p_5 = \gamma_{15} - \gamma_{25} = 510588{,}3 - 284129{,}8 = 226\,458{,}5$

$p_6 = \gamma_{16} - \gamma_{26} = 56731{,}3 - 31569{,}4 = 25\,161{,}9;$

$p_7 = \gamma_{17} - \gamma_{27} = 127646{,}4 - 71031{,}9 = 56\,614{,}5;$

$p_8 = \gamma_{18} - \gamma_{28} = 14182{,}4 - 7891{,}9 = 6\,290{,}5.$

$k^* \leftarrow argmax\{27989{,}3; 3109{,}8; \dots; 6290{,}5\} = 5;$

Assim, a primeira mediana, das 3 que faltam instalar, pertence à componente 5. O algoritmo continua:

$l_{k^*} \leftarrow l_{k^*} + 1 = 2;$

$custo \leftarrow 796794{,}7 - 226458{,}5 = 570336{,}2;$

$conta = 8 + 1;$

se $l_{k^*} < 11 - 8 + 1$ então $p_5 = \gamma_{25} - \gamma_{35} = 284129{,}8 - 223343{,}5 = 60\,786{,}3$

Nesta iteração, o algoritmo teve apenas que calcular a poupança para a componente 5, pois foi a única que se alterou devido à instalação de uma nova mediana. Comparando novamente as poupanças, conclui-se que a maior é, mais uma vez, registada na componente 5. O custo passa a ser igual a 570336,2 – 60786,3 = 509549,9.

O algoritmo tem agora que calcular a nova poupança para a componente 5 e compará-la com as restantes. Da mesma forma, conclui-se que a terceira mediana é instalada na componente 7, sendo agora o custo igual a 452935,4.

7. Estudo estatístico do algoritmo *Greedy*

No Capítulo 5 apresentaram-se alguns resultados relativos ao algoritmo *Greedy*, válidos para o caso de o grafo incluir todos os nós e a procura de cada nó e o custo de cada ligação serem constantes positivas. No caso geral, de procuras e custos distintos, não existem resultados teóricos comprovados. Como tal, realizou-se um estudo estatístico com base numa amostra composta por 5 exemplos reais de cablagem automóvel, de determinados modelos de automóveis de uma empresa de cablagens, com vista a estudar algumas caraterísticas do algoritmo *Greedy,* para casos gerais.

As caraterísticas gerais das cablagens encontram-se na tabela seguinte:

Exemplos	**Nº de componentes conexas (m)**	**Descrição geral do tamanho das componentes**			**Nº total de nós**
		Nº de componentes	**Nº de nós por componente**	**Nº de níveis por componente**	
A	8	8	384	11	3072
B	46	15	384	11	10848
		23	192	10	
		8	96	9	
C	14	3	2048	13	15360
		7	1024	12	
		4	512	11	
D	16	1	6144	18	22080
		1	3072	17	
		1	2048	16	
		2	1536	15	
		5	1024	12	
		4	512	11	
		1	384	14	
		1	192	13	

E	60	4	2304	17	51840
		2	1920	18	
		4	1536	17	
		8		16	
		2	1152	14	
		4	768	15	
		2		16	
		4	576	12	
		4		14	
		2	480	15	
		12	384	13	
		4		11	
		2	288	11	
		6	192	12	

Tabela 7.1 – Resumo das caraterísticas de 5 problemas reais, de uma empresa de cablagem.

Neste estudo, usou-se um computador pessoal com um processador Intel Core 2 Duo, de 2.20 GHz, com 4GB de memória RAM. Os programas utilizados foram feitos em C++.

7.1 Estudo de algumas características do algoritmo *Greedy*

Neste Capítulo, estudam-se algumas caraterísticas relativas às medianas das soluções obtidas com o algoritmo Greedy, nos vários exemplos, nomeadamente, a existência de regularidades na localização das medianas e a existência de correlação entre o número de medianas por componente e o custo de cada componente.

7.1.1 Localização das medianas de cada solução

Partindo da análise de cada um dos exemplos referidos, começa-se por estudar a localização dos nós selecionados em cada solução obtida pelo algoritmo *Greedy*. Apresentam-se, de seguida, os resultados obtidos com os cinco exemplos, para o cálculo de 4 conjuntos de medianas, p=50, p=100, p=150 e p=200.

Apresenta-se, para cada exemplo, uma tabela que indica o número de nós selecionados por nível e um gráfico com a percentagem de nós selecionados relativamente ao número de nós de cada nível. No gráfico não se considera o nível

zero pois são sempre selecionados 100% dos nós desse nível, tornando-se assim o gráfico mais fácil de ler.

Exemplo A

Nível	Nº de nós total	Nº de nós selecionados			
		p=50	p=100	p=150	p=200
0	8	8	8	8	8
1	64	2	6	8	9
2	224	15	26	33	40
3	464	10	25	35	47
4	648	11	22	34	49
5	656	3	10	22	31
6	504	1	3	8	13
7	304	0	0	2	3
8	144	0	0	0	0
9	48	0	0	0	0
10	8	0	0	0	0

Tabela 7.2 – Localização das medianas por componente, para o exemplo A.

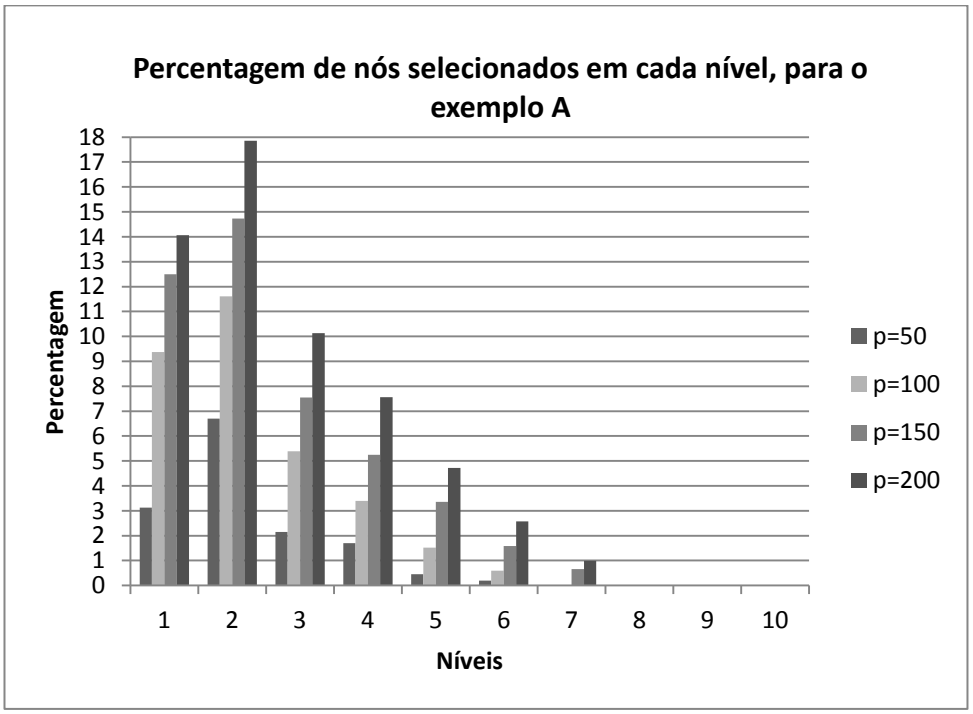

Figura 7.1 – Percentagem de medianas por nível, para o exemplo A.

Exemplo B

Nível	Nº de nós total	Nº de nós selecionados			
		p=50	p=100	p=150	p=200
0	46	46	46	46	46
1	328	0	0	0	1
2	1063	1	13	23	31
3	2026	3	23	34	45
4	2546	0	1	5	11
5	2246	0	5	12	21
6	1447	0	10	24	33
7	727	0	2	5	7
8	307	0	0	0	2
9	97	0	0	1	3
10	15	0	0	0	0

Tabela 7.3 – Localização das medianas por componente, para o exemplo B.

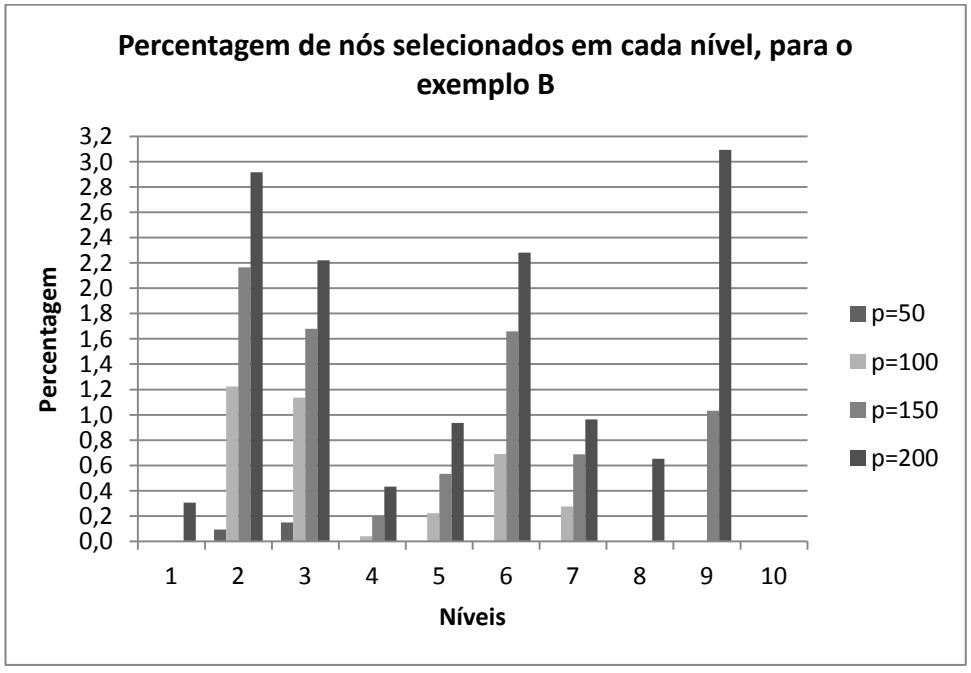

Figura 7.2 – Percentagem de medianas por nível, para o exemplo B.

Note-se que, nestes dois exemplos, mais de 50% dos nós selecionados encontram-se nos primeiros 4 níveis e que a partir do nível 6 a percentagem de nós selecionados é muito baixa ou nula.

Exemplo C

Nível	Nº de nós total	Nº de nós selecionados			
		p=50	p=100	p=150	p=200
0	14	14	14	14	14
1	139	0	0	0	0
2	610	2	4	5	8
3	1574	0	2	3	6
4	2687	6	12	14	19
5	3258	4	8	14	17
6	2968	10	21	31	35
7	2128	2	8	20	29
8	1224	7	17	25	37
9	547	4	9	15	22
10	174	1	4	8	10
11	34	0	1	1	3
12	3	0	0	0	0

Tabela 7.4 – Localização das medianas por componente, para o exemplo C.

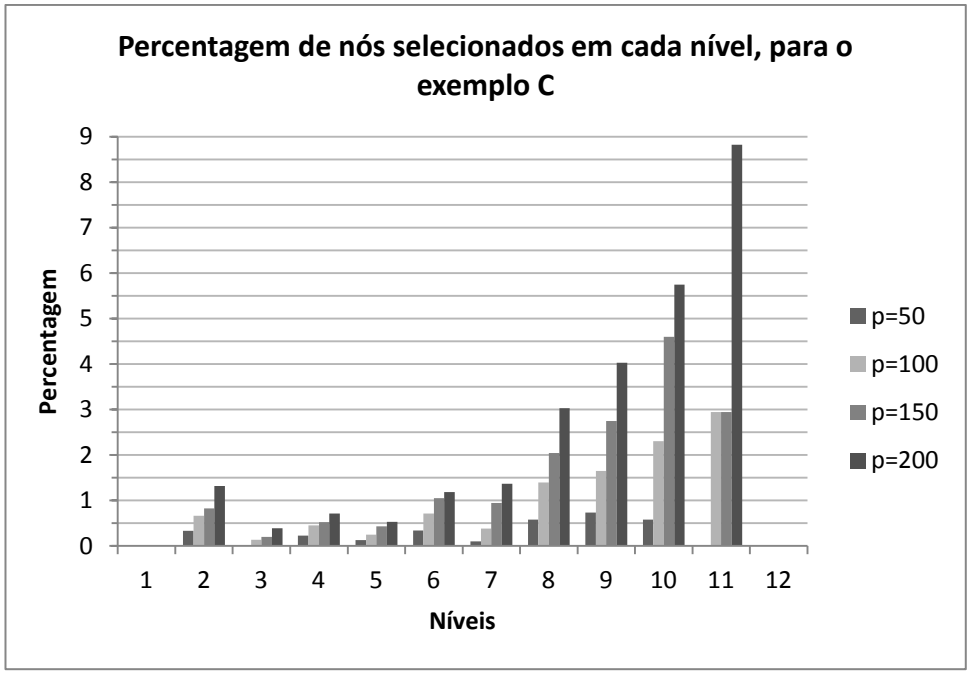

Figura 7.3 – Percentagem de medianas por nível, para o exemplo C.

Neste exemplo, os dados encontram-se mais dispersos, mas mais de 50% dos nós selecionados encontram-se em níveis iguais ou inferiores ao nível 6.

É importante referir que no exemplo C, as procuras estimadas são, na maioria, nulas ou quase nulas nos níveis superiores, o que explica o facto de, neste caso, serem escolhidas mais medianas nos níveis inferiores do que nos superiores.

Exemplo D

Nível	Nº de nós total	Nº de nós selecionados			
		p=50	p=100	p=150	p=200
0	16	16	16	16	16
1	146	5	8	11	11
2	611	2	5	5	5
3	1541	8	11	13	16
4	2625	3	7	11	13
5	3293	2	7	16	21
6	3359	1	10	18	27
7	3071	5	11	19	25
8	2607	3	6	10	17
9	1992	2	7	14	19
10	1332	2	5	5	14
11	786	0	3	6	8
12	414	1	3	4	4
13	188	0	1	2	2
14	70	0	0	0	2
15	22	0	0	0	0
16	6	0	0	0	0
17	1	0	0	0	0

Tabela 7.5 – Localização das medianas por componente, para o exemplo D.

Observe-se que, mais de 50% dos nós selecionados encontram-se nos primeiros 6 níveis (ou menos) e que, a partir do nível 11, a percentagem de nós selecionados é muito baixa ou nula.

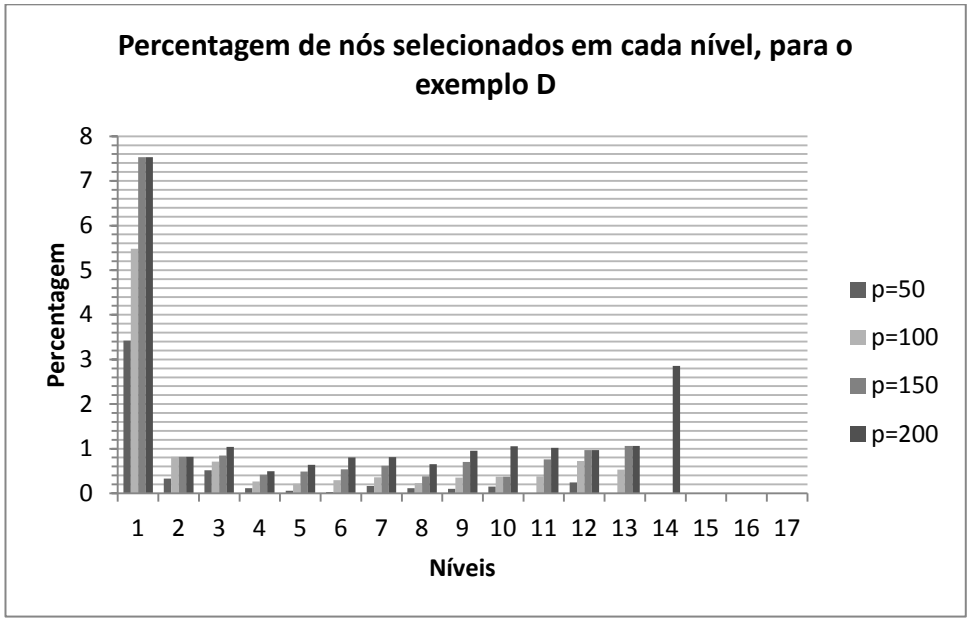

Figura 7.4 – Percentagem de medianas por nível, para o exemplo D.

Exemplo E

Nível	Nº de nós total	Nº de nós selecionados		
		p=100	p=150	p=200
0	60	60	60	60
1	456	0	1	1
2	1654	5	10	13
3	3768	2	5	7
4	6084	6	13	23
5	7614	9	14	22
6	7986	3	10	16
7	7350	4	7	12
8	5972	7	15	20
9	4340	1	4	8
10	2932	2	5	7
11	1830	1	5	8
12	1002	0	1	3
13	484	0	0	0
14	212	0	0	0
15	76	0	0	0
16	18	0	0	0
17	2	0	0	0

Tabela 7.6 – Localização das medianas por componente, para o exemplo E.

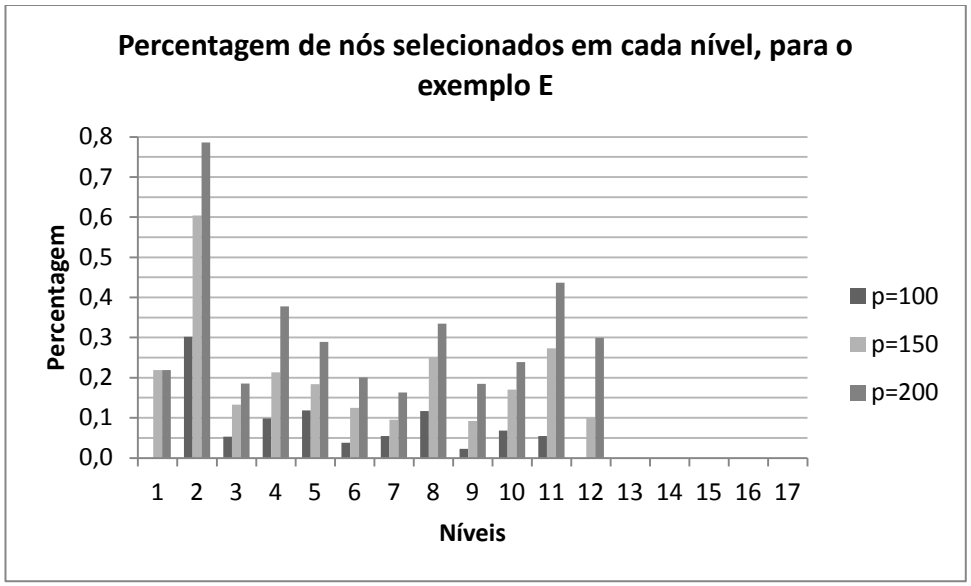

Figura 7.5 – Percentagem de medianas por nível, para o exemplo E.

Neste exemplo, à semelhança dos anteriores, mais de 50% dos nós selecionados encontram-se nos primeiros 4 níveis (ou menos) e, a partir do nível 12, a percentagem de nós selecionados é muito baixa ou nula.

Deste estudo, conclui-se que a maioria das medianas está situada nos primeiros níveis e que nos últimos níveis são selecionadas poucas medianas ou até mesmo, em alguns casos, nenhumas.

7.1.2 Número de medianas em cada componente versus custo da componente

Pretende-se, agora, averiguar se o número de medianas escolhidas em cada componente está relacionado com o custo da componente (considerando que o custo da componente corresponde ao custo da substituição de todos os nós pelo nó do nível zero).

Analise-se, como motivação, o seguinte gráfico para o exemplo A, correspondente à escolha de 100 medianas.

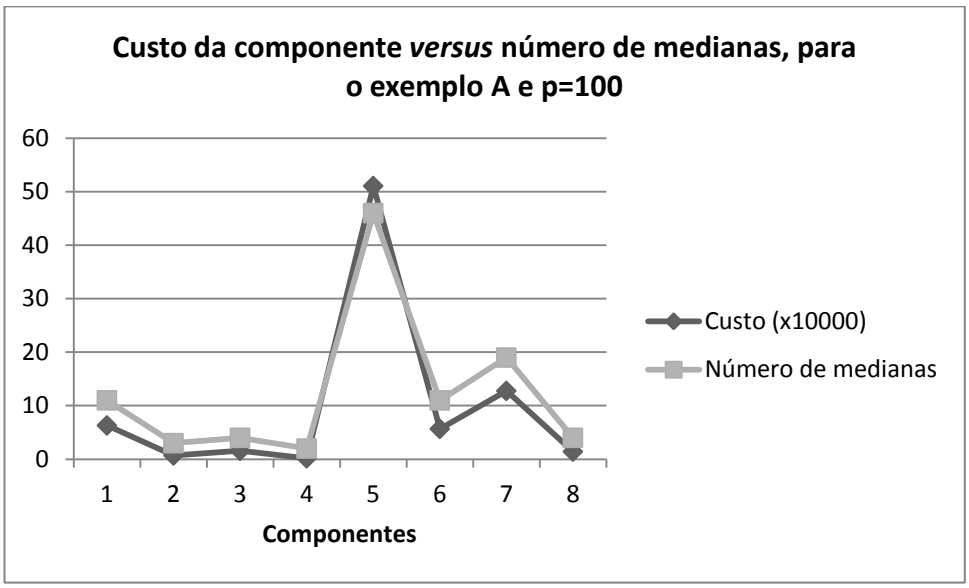

Figura 7.6 – Comparação entre o nº de medianas e o custo de cada componente, para o exemplo A.

Parece evidente a relação entre o número de medianas de cada componente e o custo dessa componente.

Apresenta-se, de seguida, um estudo feito a partir de uma amostra global contendo os cinco exemplos apresentados, utilizando os valores de p=50, 100, 150 e 200. Para cada um destes valores constroem-se diagramas de dispersão, relacionando o peso de cada componente (quociente entre o custo de cada componente e o custo global) com a proporção de medianas nessa componente.

Observe-se o diagrama para p=50.

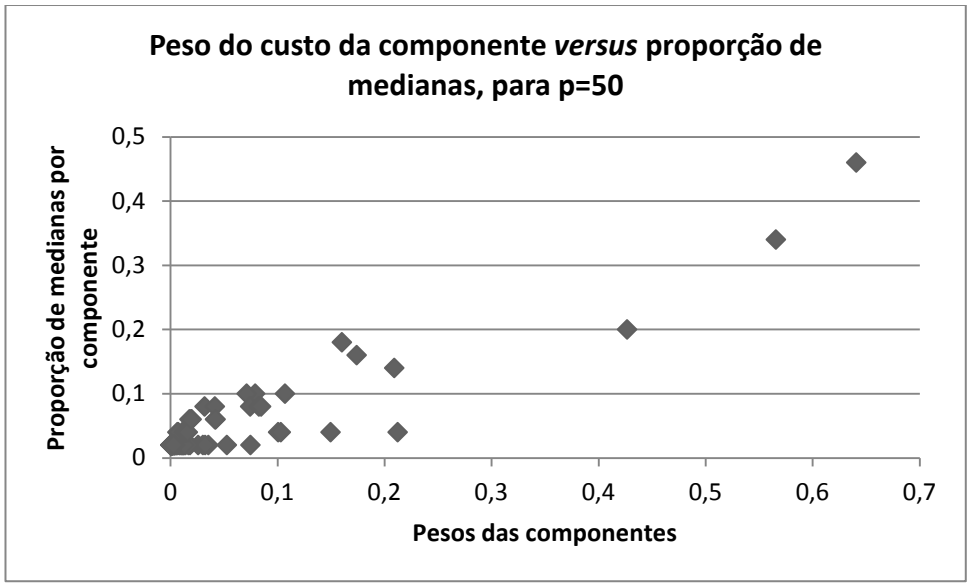

Figura 7.7 – Diagrama de dispersão, para p=50.

O diagrama de dispersão sugere uma correlação forte e positiva entre o peso da componente em termos de custo, e a proporção de medianas de cada componente. O coeficiente de correlação linear é igual a 0,932.

De facto, calculando a correlação entre o número de medianas de cada componente e o custo dessa componente, para os diferentes exemplos, obtêm-se os seguintes resultados:

Exemplo	A	B	C	D
Coeficiente de correlação	0,995	0,882	0,985	0,959

Tabela 7.7 – Coeficientes de correlação, para p=50.

Conclui-se que há uma correlação forte e positiva em todos os casos.

Analise-se, de seguida, o diagrama de dispersão para p=100.

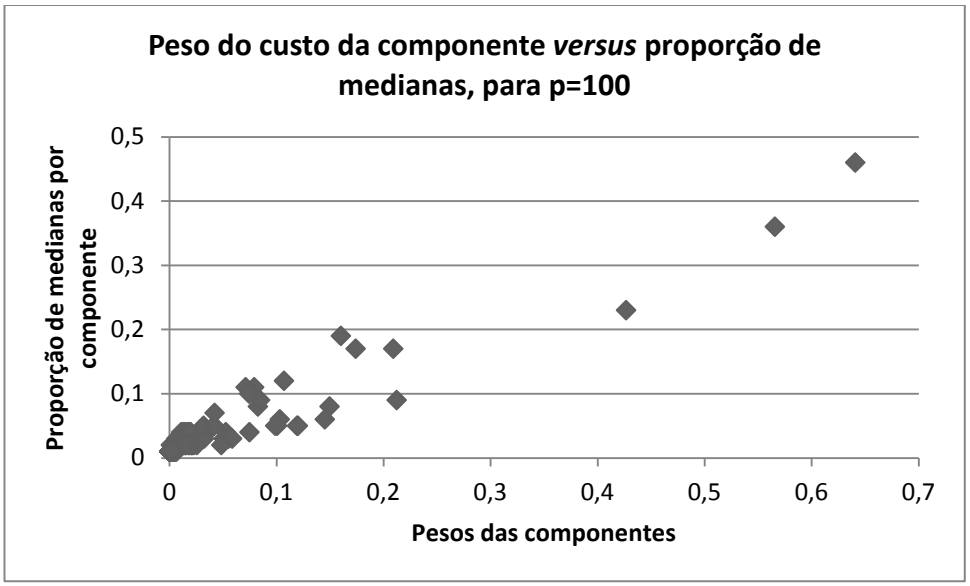

Figura 7.8 – Diagrama de dispersão, para p=100.

Verifica-se que há uma correlação linear positiva forte entre as duas variáveis em estudo. De facto, o coeficiente de correlação linear é igual 0,956.

Analise-se, ainda, a correlação existente entre a proporção de medianas e o custo das componentes, para cada exemplo:

Número de medianas	A	B	C	D	E
Coeficiente de Correlação	0,988	0,966	0,981	0,974	0,967

Tabela 7.8 – Coeficientes de correlação, para p=100.

Verifica-se que existe uma correlação forte e positiva em todos os casos.

De seguida, apresenta-se o diagrama de dispersão correspondente a p=150.

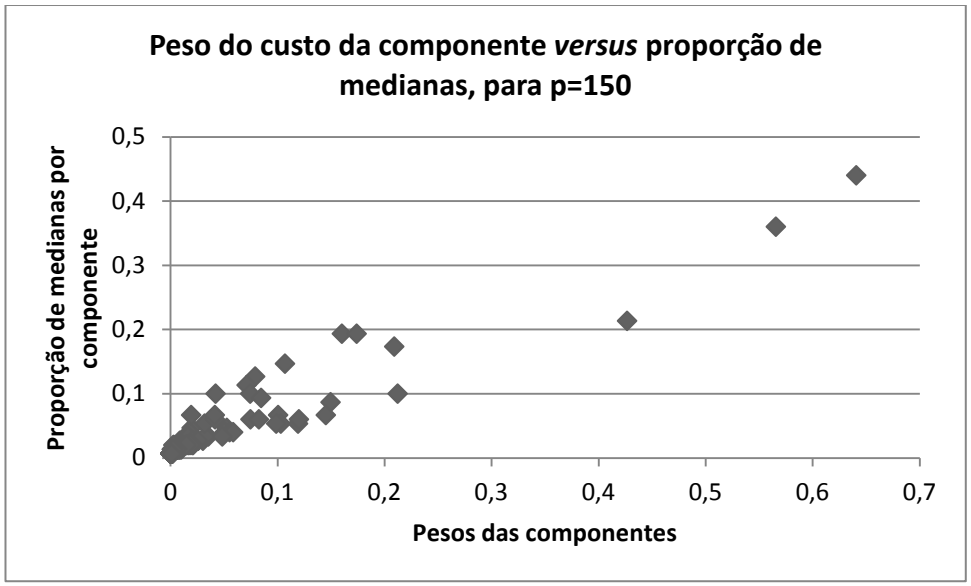

Figura 7.9 – Diagrama de dispersão, para p=150.

Mais uma vez, o diagrama de dispersão sugere a existência de uma forte correlação entre as variáveis, sendo o coeficiente de correlação linear igual a 0,947.

A tabela seguinte indica os coeficientes de correlação para cada um dos exemplos.

Número de medianas	A	B	C	D	E
Coeficiente de Correlação	0,979	0,970	0,965	0,922	0,967

Tabela 7.9 – Coeficientes de correlação, para p=150.

Por último, averigua-se o que acontece para p=200.

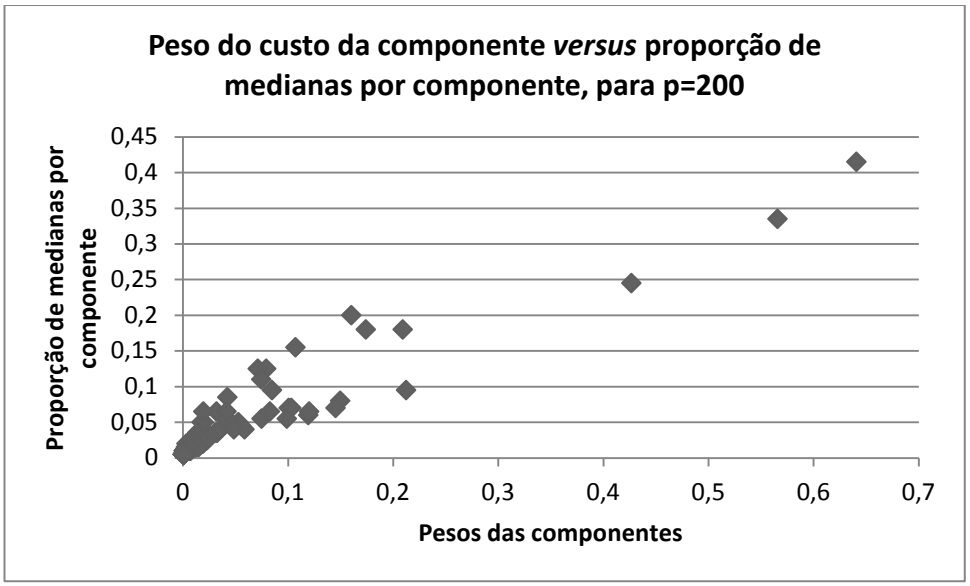

Figura 7.10 – Diagrama de dispersão, para p=200.

Neste caso, continua a verificar-se uma correlação forte e positiva entre as duas variáveis, com coeficiente de correlação linear igual a 0,946.

Para cada exemplo, os coeficientes de correlação linear são os que constam da tabela seguinte.

Número de medianas	A	B	C	D	E
Coeficiente de Correlação	0,967	0,959	0,950	0,955	0,964

Tabela 7.10 – Coeficientes de correlação, para p=200.

É de notar que, com base no custo de cada componente, pode prever-se, com alguma exatidão, o número de medianas dessa componente. Observa-se no gráfico seguinte um modelo de regressão linear, para p=100, para a amostra composta pelos cinco exemplos A, B, C, D e E.

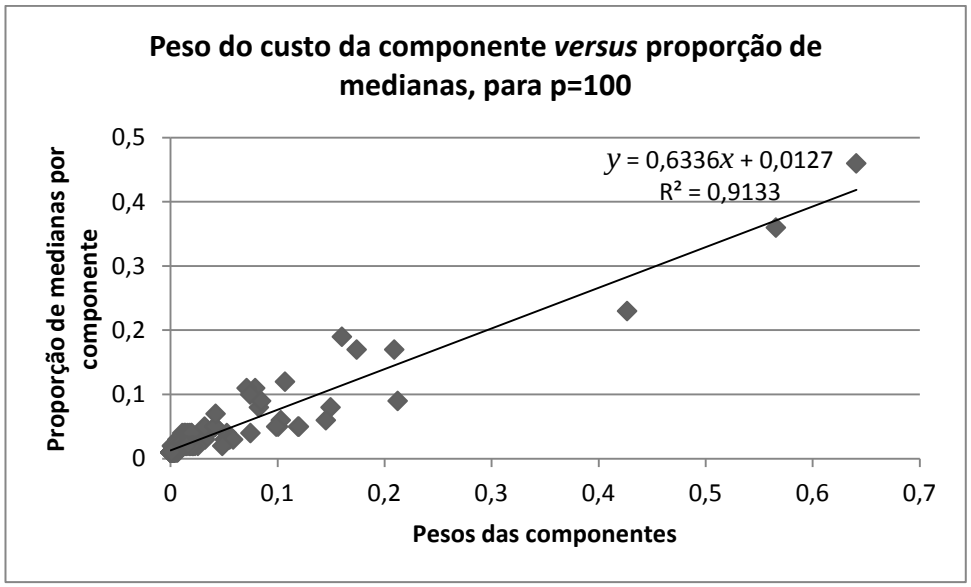

Figura 7.11 – Diagrama de dispersão e reta de regressão, para p=100.

Através da equação da reta de regressão é possível, a partir do peso de uma componente, prever um valor aproximado para a proporção de medianas que fazem parte dessa componente, e consequentemente o número de medianas dessa componente. Neste caso, com um valor para o coeficiente de determinação igual a 0,91, o que é bastante bom.

Com base neste modelo linear, calculou-se, para o exemplo E, o número de medianas estimado e comparou-se com o número de medianas calculado pelo algoritmo *Greedy*:

Componente	Nº de medianas previsto pelo modelo	Nº de medianas instaladas pelo *Greedy*	Componente	Nº de medianas previsto pelo modelo	Nº de medianas instaladas pelo *Greedy*
1	0,97	**1**	31	0,91	**1**
2	3,20	**3**	32	0,96	**1**
3	0,92	**1**	33	0,92	**1**
4	1,31	2	34	1,24	2
5	1,01	**1**	35	0,91	**1**
6	3,41	**3**	36	0,97	**1**
7	0,93	**1**	37	5,38	**5**
8	1,35	2	38	6,34	5
9	0,95	**1**	39	3,10	2
10	1,84	**2**	40	1,70	**2**
11	0,91	**1**	41	1,87	**2**
12	1,07	**1**	42	1,29	2
13	1,13	**1**	43	0,94	**1**
14	6,30	5	44	1,61	**2**
15	1,17	**1**	45	0,94	**1**
16	7,48	6	46	1,77	**2**
17	1,01	**1**	47	0,92	**1**
18	3,56	3	48	1,26	2
19	0,95	**1**	49	0,91	**1**
20	1,95	**2**	50	1,03	**1**
21	0,95	**1**	51	0,91	**1**
22	1,86	**2**	52	1,06	**1**
23	0,95	**1**	53	0,91	**1**
24	2,07	**2**	54	0,97	**1**
25	0,93	**1**	55	1,45	2
26	1,38	2	56	1,58	**2**
27	0,92	**1**	57	1,18	**1**
28	1,09	**1**	58	1,00	**1**
29	0,91	**1**	59	1,03	**1**
30	1,21	**1**	60	0,96	**1**

Tabela 7.11 – Comparação entre o número de medianas instaladas pelo algoritmo *Greedy* e o número de medianas previsto pelo modelo de regressão linear.

Verifica-se que o número de medianas estimadas com base no modelo e o número de medianas instaladas pelo algoritmo *Greedy* são bastante semelhantes.

Analisando outros exemplos, pode concluir-se que o modelo é, em geral, bom para prever o número de medianas que serão escolhidas, principalmente porque dá sempre a perceção de quais as componentes que terão um maior número de medianas e devolve proporções da mesma ordem de grandeza das calculadas pelo algoritmo *Greedy*.

7.2 Comparação entre o algoritmo *Greedy* e algoritmos *Greedy* restritos

Na subsecção 7.1.1, usou-se o algoritmo *Greedy* para encontrar soluções para os 5 problemas dos exemplos apresentados e verificou-se que, na maioria dos casos, a maior parte das medianas são instaladas em nós dos primeiros níveis. Como tal, usam-se agora, para além do algoritmo *Greedy* usual, duas novas versões restritas deste algoritmo. Enquanto que o algoritmo *Greedy* usual percorre em cada iteração todos os nós do grafo em busca da melhor poupança, os algoritmos restritos percorrem apenas uma parte do grafo, em cada iteração. Designa-se por algoritmo "*Greedy* k+1" o algoritmo *Greedy* que, após a instalação de uma mediana no nível k, restringe a procura de uma nova mediana até ao nível k+1 (nível seguinte) e por "*Greedy* Metade" o algoritmo *Greedy* que restringe a procura de uma nova mediana até metade dos níveis do grafo.

Em cada exemplo, estes três algoritmos são executados várias vezes, dependendo do número que medianas a calcular.

Pretende-se comparar os vários tempos que os vários algoritmos demoram a encontrar a solução e os custos dessas soluções.

Nas tabelas seguintes encontram-se os resultados obtidos no estudo realizado para os cinco exemplos, no cálculo de 30, 50, 100, 150, 200 e 250 medianas.

Exemplo A

Nº de medianas	**Tempo de execução** (em segundos)			**Custos/Soluções** (em euros)		
p	*Greedy*	*Greedy* k+1	*Greedy* metade	*Greedy*	*Greedy* k+1	*Greedy* metade
30	0	0	0	208575	215019	208575
50	0	0	0	144699	148407	144823
100	1	0	0	82365	82769	83269
150	1	1	1	56023	56297	58635
200	1	1	1	41066	41200	45301
250	1	1	1	31540	31602	36975

Tabela 7.12 – Resultados, para o exemplo A.

Exemplo B

Nº de medianas	**Tempo de execução** (em segundos)			**Custos/Soluções** (em euros)		
P	*Greedy*	*Greedy* k+1	*Greedy* metade	*Greedy*	*Greedy* k+1	*Greedy* metade
50	1	0	0	10527997	12751265	10527997
100	1	1	1	3508393	4570833	3565318
150	1	1	1	2253028	2818309	2394801
200	1	1	1	1657671	1994248	1868282
250	1	1	1	1304088	1517588	1576957

Tabela 7.13 – Resultados, para o exemplo B.

Exemplo C

Nº de medianas	**Tempo de execução** (em segundos)			**Custos/Soluções** (em euros)		
p	*Greedy*	*Greedy* k+1	*Greedy* metade	*Greedy*	*Greedy* k+1	*Greedy* metade
30	6	1	5	4881	6712	5286
50	9	4	8	3239	4078	3883
100	19	12	18	1874	2169	2858
150	28	21	28	1314	1462	2504
200	37	29	37	1014	1103	2309
250	45	38	47	814	876	2188

Tabela 7.14 – Resultados, para o exemplo C.

Exemplo D

Nº de medianas	**Tempo de execução** (em segundos)			**Custos/Soluções** (em euros)		
p	*Greedy*	*Greedy* k+1	*Greedy* metade	*Greedy*	*Greedy* k+1	*Greedy* metade
30	33	8	33	3866568	4939649	3866568
50	64	37	62	2278937	3067109	2278937
100	152	100	143	1248531	1494683	1254386
150	203	170	203	863039	966847	870992
200	303	263	289	662364	722336	672780
250	396	343	394	536785	569449	549362

Tabela 7.15 – Resultados, para o exemplo D.

Exemplo E

Nº de medianas	**Tempo de execução** (em segundos)			**Custos/Soluções** (em euros)		
p	*Greedy*	*Greedy* k+1	*Greedy* metade	*Greedy*	*Greedy* k+1	*Greedy* metade
100	22	3	11	2395735	4222575	2440383
150	41	8	19	1432132	2542986	1567778
200	66	14	25	1043177	1781787	1224145
250	83	21	31	811943	1347742	1035051

Tabela 7.16 – Resultados, para o exemplo E.

Para o exemplo E, apresentam-se ainda dois gráficos comparativos para os três algoritmos, um para os tempos de execução e outro para os custos das soluções.

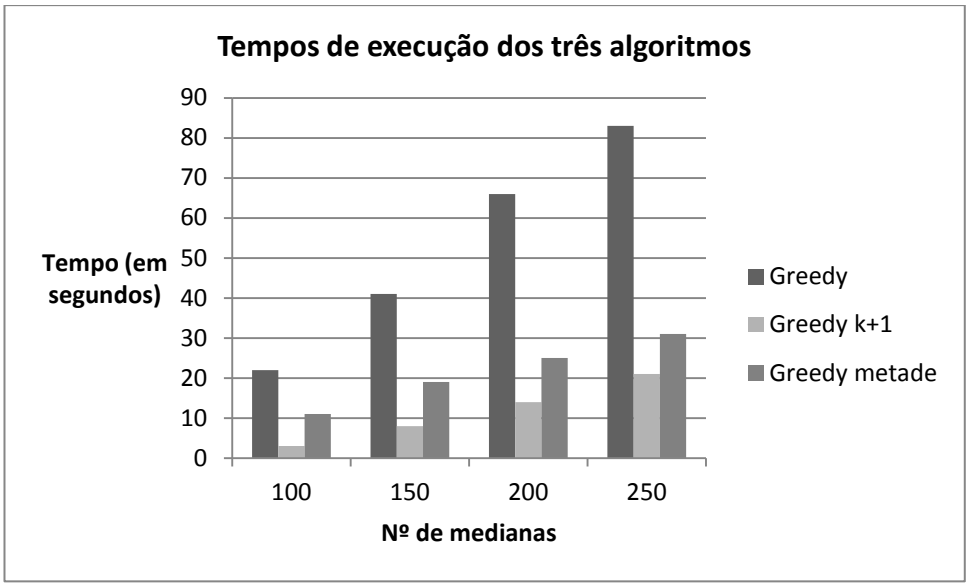

Figura 7.12 – Comparação dos tempos de execução dos 3 algoritmos, para o exemplo E.

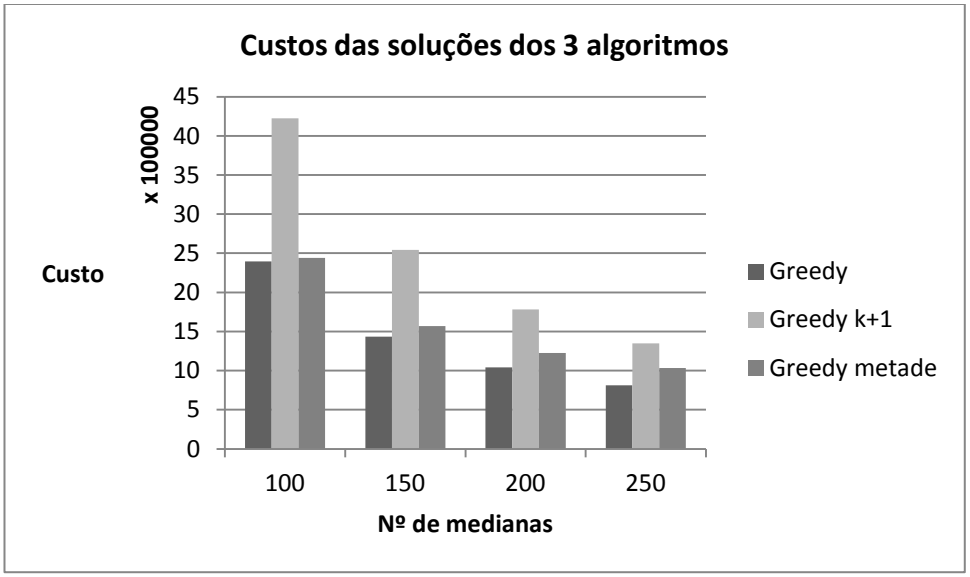

Figura 7.13 – Comparação dos custos das soluções dos 3 algoritmos, para o exemplo E.

Da análise destes dados pode concluir-se, principalmente para os exemplos de maior dimensão (C, D e E), que na maioria dos casos pode ser bastante vantajoso restringir a busca do algoritmo *Greedy* a k+1 níveis pois diminui bastante o tempo de execução do algoritmo, apesar de aumentar o custo da solução.

No exemplo A, o de menor tamanho, não se verifica uma melhoria em termos de tempo (à exceção de um caso) dos algoritmos restritos em relação ao algoritmo

Greedy, mas também não há alterações significativas nos custos da utilização dos algoritmos restritos em relação ao algoritmo *Greedy*.

No exemplo B, também há poucas evidências de vantagem em termos de tempo nos algoritmos restritos e em alguns casos, principalmente na utilização do algoritmo restrito a k+1 níveis, já se verifica um maior aumento ao nível dos custos.

No exemplo C, nota-se já uma melhoria significativa ao nível do tempo de execução do algoritmo restrito a k+1 níveis, em relação ao algoritmo *Greedy*, melhoria essa que compensa o pequeno aumento dos custos. No algoritmo *Greedy* restrito a metade dos níveis a melhoria dos tempos não é significativa e os custos são, em alguns casos, elevados.

No exemplo D, há evidências de vantagem no uso do algoritmo restrito a k+1 níveis, ao nível dos tempos de execução, embora nalguns casos os custos aumentem um pouco. Em relação ao algoritmo restrito a metade dos níveis, verifica-se uma pequena vantagem, pois os tempos e os custos são semelhantes aos do algoritmo *Greedy*.

No exemplo E, a vantagem da utilização dos algoritmos restritos é bastante significativa no que respeita aos tempos de execução, sendo maior no algoritmo restrito a k+1 níveis. Em termos de custos, há um ligeiro aumento na utilização do algoritmo restrito a metade dos níveis e um aumento razoável, por vezes um pouco elevado, na utilização do algoritmo restrito a k+1 níveis.

Desta análise, conclui-se que para os exemplos de pequena dimensão deve usar-se o algoritmo *Greedy*. Contudo, tal como se pode observar pelo exemplo E, para os exemplos de maior dimensão o tempo de execução do algoritmo *Greedy* torna-se muito elevado e as versões restritas surgem mais atrativas. Assim, para os exemplos de maior dimensão podem usar-se os algoritmos restritos pois estes permitem uma considerável poupança de tempo na busca das soluções, poupança essa que acaba por ser mais importante do que o aumento verificado nos custos dessas soluções.

Esta conclusão torna-se mais evidente se a solução obtida por um destes algoritmos for posteriormente melhorada, usando um outro algoritmo (ver [3]).

8. Conclusões

Esta tese apresentou o problema da Gestão Ótima da Diversidade de Cablagens e a sua aplicação na indústria automóvel.

Dado que, para problemas de grande dimensão, encontrar uma solução pode tornar-se um processo muito moroso, procurou-se estudar algumas caraterísticas do PGODC que tornassem esse processo mais rápido. Assim, estudaram-se algumas caraterísticas combinatórias do problema e concluiu-se que, na prática, para o resolver não é necessário guardar informação sobre todos os arcos na procura de medianas, mas apenas os que não podem ser obtidos por transitividade. Este resultado permite poupar bastante espaço de memória aquando da resolução do problema.

Pelo facto de o PGODC ser classificado como NP-difícil, apresentou-se um algoritmo *Greedy*, que se aplicou na resolução do mesmo. Apresentaram-se, também, alguns resultados úteis na utilização deste algoritmo, para o caso particular em que o grafo inclui todos os nós, a procura dos nós é unitária (ou constante positiva) e o custo de cada fio é unitário (ou constante positivo). Assim, concluiu-se que, neste caso, o segundo nó a ser escolhido pelo algoritmo *Greedy* pertence ao nível 1, o terceiro pertence ao nível 2 e não é antecessor do nó já escolhido no nível 1, e que após p iterações do algoritmo, sendo k o maior nível onde foi escolhido um nó, na iteração p+1 apenas é necessário considerar os primeiros k+1 níveis. Mais uma vez, estes resultados revelam-se muito importantes pois permitem poupar bastante tempo na resolução do problema, dado que o número de nós a analisar é menor.

Até aqui, o PGODC estava associado a um grafo orientado de pequena dimensão. Mas como na realidade os problemas que surgem são bem mais complexos, apresentou-se no Capítulo 6 a decomposição do PGODC, bem como um exemplo real de um problema resolvido através do algoritmo *Greedy*. Aqui a resolução do problema processou-se em duas fases: primeiro decompôs-se o problema em vários subproblemas, resolvendo-se cada um deles através do algoritmo *Greedy*, e em segundo aplicou-se o algoritmo *Greedy* para escolher o número medianas a selecionar de cada subproblema. Portanto, o algoritmo *Greedy* surge aqui sob duas formas distintas. Esta forma de resolver o PGODC permite poupar bastante tempo e fornece

exatamente a mesma solução, em relação à aplicação do algoritmo Greedy ao problema apenas uma vez ([4]).

Por fim, dado que para o caso em que o grafo pode não incluir todos os nós e os custos e as procuras não são constantes, não existem resultados teóricos comprovados, estudaram-se estatisticamente 5 exemplos reais do PGODC de uma empresa de cablagens. Estes cinco exemplos são bastante diferentes uns dos outros, sendo que o número de componentes de cada um dos grafos correspondentes varia entre 8 e 60, o número de níveis por componente varia entre 9 e 18 e o número de nós de cada grafo varia entre 3072 e 51840. Usou-se o algoritmo *Greedy* para obter soluções para cada um dos exemplos, para 4 valores de medianas: 50, 100, 150 e 200. Com a amostra obtida, numa primeira fase, estudou-se a localização das medianas nas soluções, concluindo-se que a maioria das medianas estão localizadas nos primeiros níveis do grafo e que nos últimos níveis do grafo estão localizadas uma quantidade mínima ou nula de medianas. Esta conclusão leva a pensar que algumas das caraterísticas estudadas para um caso particular no Capítulo 5, se poderão manter em alguns casos. Numa segunda fase, estudou-se a existência de correlação entre o número de medianas selecionadas por componente e o custo dessa componente, concluindo-se que existe uma correlação positiva forte entre as duas variáveis, quanto maior é o custo da componente maior é o número de medianas que se instalam na mesma, e consequentemente que é possível prever, com alguma exatidão, o número de medianas que são selecionadas de cada componente, a partir do conhecimento do custo da componente. Verificou-se num exemplo, usando um modelo de regressão linear, que a previsão do número de medianas por componente está bastante próxima do número de medianas calculadas pelo algoritmo *Greedy*.

Posteriormente, com a motivação trazida pelos resultados do Capítulo 5 e pelo resultado do estudo da localização das medianas na solução, foi feito um estudo estatístico, utilizando o algoritmo *Greedy* e duas outras versões do algoritmo com algumas restrições nas procuras: o algoritmo "*Greedy* k+1" que, após a instalação de uma mediana no nível k, restringe a procura de uma nova mediana até ao nível k+1 e o algoritmo "*Greedy* Metade" que restringe a procura de uma nova mediana até metade dos níveis do grafo.

Deste estudo concluiu-se que, para exemplos de grande dimensão, pode ser bastante vantajoso utilizar o algoritmo *Greedy* com restrições, nomeadamente o "*Greedy* k+1", pois o tempo de execução deste algoritmo pode ser bastante inferior ao tempo de execução do algoritmo *Greedy* propriamente dito. A única desvantagem é que os custos das soluções obtidas desta forma são um pouco mais elevados, mas para problemas de grande dimensão, extremamente morosos de resolver, o algoritmo *Greedy* k+1 pode ser uma excelente alternativa.

Para exemplos de pequena dimensão, é mais rentável usar o algoritmo *Greedy* pois os tempos de execução dos três algoritmos são semelhantes e com o algoritmo *Greedy* obtém-se uma solução mais barata.

Bibliografia

[1] Agra, A.; Cardoso, D.; Cerdeira, J.; Miranda, M. e Rocha, E., 2009. Solving Huge size instances of the Optimal Diversity Management Problem. Journal of Mathematical Sciences.

[2] Agra, A.; Cardoso, D.; Cerdeira, J.; Miranda, M. e Rocha, E., 2007. The minimum weight spanning star forest model of the optimal diversity management problem. Cadernos de Matemática. Universidade de Aveiro.

[3] Agra, A.; Cerdeira, J; e Requejo, C. Using Decomposition to improve Greedy solutions of the optimal diversity management problem. Submetido para publicação.

[4] Agra, A. e Requejo, C., 2009. The Linking Set Problem: A polynomial special case of the multiple-choice Knapsack problem. Journal of Mathematical Sciences.

[5] Avella, P.; Boccia, M.; Martino, Di C.; Oliviero, G.; Sforza, J. e Vasil'ev, I., 2005. A decomposition approach for a very large scale optimal diversity management problem. 4OR 3: 23-37.

[6] Avella, P.; Sassano, A. e Vasil'ev, I., 2007. Computational study of large-scale p-median problems. Mathematical Programming, 109(1):89-114.

[7] Briant, O., 2000. Etude theórique et numérique du problème de la gestion de la diversité. Ph.D. thesis, INP Grenoble, France.

[8] Briant, O. e Naddef, D., 2004. The optimal diversity management problem. Operations Research 52(4):515-526.

[9] Cardoso, D.; Rostam, M. e Szymanski, J., 2008. Matemática discreta. Escolar Editora.

[10] Cardoso, D; Cerdeira, J., 2012. The minimum weight t-composition of an integer. Journal of Mathematical Sciences, Vol. 182, No. 2, (2012): 210-215.

[11] Dias, A., 2008. O problema da p-mediana aplicado ao problema da gestão ótima da diversidade.

[12] FICO Xpress Optimization Suite, Xpress-Optimizer reference manual. Technical Report Release 22.01, 2011.

[13] Kellerer, H; Pferschy, U; and Pisinger, D., 2004. Knapsack Problems. Springer-Verlag.

[14] Mirchandani, P.B. e Francis, R.L., 1990. Discrete Location Theory. John Wiley & Sons.

[15] Reese, J., 2006. Solution methods for the p-median problem: an annotated bibliography. Networks, 48:125-142.

Apêndice

Lema 1: $2k-1 \geq \frac{2k+1}{2}$, para $k \geq 2$.

Prova:

Prove-se por indução.

Para $k = 2$ tem-se $4-1 \geq \frac{4+1}{2}$, ou seja, $3 \geq \frac{5}{2}$.

Para $k > 2$, suponha-se que $2k-1 \geq \frac{2k+1}{2}$ e mostre-se que:

$2(k+1)-1 \geq \frac{2(k+1)+1}{2}$.

Ora, $2(k+1)-1 = 2k+2-1 \geq \frac{2k+1}{2}+2 = \frac{2k+1+4}{2} =$

$$= \frac{2(k+1)+2}{2} \geq \frac{2(k+1)+1}{2}.$$

Lema 2*:* Seja $C(k) = (2k-1)(2^{n-k-1}),\ n\epsilon\mathbb{N},\ k\epsilon\{2,\dots,n\}$.

Então, $C(k) \geq C(k+1)$, para $k \geq 2$.

Prova:

Para $k > 2$ e para $n \geq 4$:

$C(k) = (2k-1)(2^{n-k-1})$.

$C(k+1) = (2(k+1)-1)\left(2^{n-(k+1)-1}\right) = (2k+1)(2^{n-k-2})$.

Assim, $C(k) \geq C(k+1) \Leftrightarrow (2k-1)(2^{n-k-1}) \geq (2k+1)(2^{n-k-2}) \Leftrightarrow$

$$\Leftrightarrow 2k-1 \geq \tfrac{2k+1}{2} \text{ (Lema 1).}$$

Lema 3: Seja $k < n$ um inteiro não negativo, então para $t \geq 2$,

$2^{n-(k+1)} \geq t \times 2^{n-(k+t)}$.

Prova(por indução):

Para $t = 2$, tem-se:

$2^{n-(k+1)} \geq 2 \times 2^{n-(k+2)} \Leftrightarrow 2^{n-k-1} \geq 2^{n-k-2+1} \Leftrightarrow 2^{n-k-1} \geq 2^{n-k-1}$

Suponha-se que $2^{n-(k+1)} \geq t \times 2^{n-(k+t)}$ e prove-se que:

$2^{n-(k+1)} \geq (t+1) \times 2^{n-(k+t+1)}$.

Assim, $(t+1) \times 2^{n-(k+t+1)} = t \times 2^{n-(k+t+1)} + 2^{n-(k+t+1)} =$

$$= \frac{t \times 2^{n-(k+t)}}{2} + 2^{n-(k+t+1)} \leq \frac{2^{n-(k+1)}}{2} + 2^{n-(k+t+1)} =$$

$$= 2^{n-(k+1)-1} + 2^{n-(k+1)-t} = 2^{n-(k+1)}(2^{-1} + 2^{-t}) \leq 2^{n-(k+1)}.$$