



INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Problema da Mochila Multicritério

Aspectos algorítmicos e Implementação Informática

Eduardo Pombal Luila

Dissertação para obtenção do grau de Mestre em

Engenharia e Gestão Industrial

Júri

Presidente: Prof. Dr. Paulo Vasconcelos Dias Correia

Orientador: Prof. Dr. José Rui Figueira

Vogal: Prof. Dr. Acácio Manuel de Oliveira Porta Nova

Maio 2008



Problema da Mochila Multicritério

Aspectos algorítmicos e Implementação Informática

Eduardo Pombal Luila

Agradecimentos

Agradeço ao Professor Dr. José Rui Figueira pela orientação, pelo incentivo e apoio incondicional disponibilizado ao longo de todo o período de realização desta dissertação. Também gostaria de agradecer ao Professor João Lourenço pela ajuda prestada no arranque da tese, sobre tudo na escolha do orientador e do tema da tese. Os meus agradecimentos são extensíveis a todos aqueles que directa ou indirectamente contribuíram para, a concretização deste trabalho.

Resumo

A maior parte dos artigos publicados sobre o problema da mochila foram dedicados para o único critério. Nesta tese abordamos o problema da mochila tendo em consideração vários critérios. Existem poucos artigos publicados sobre este assunto, embora o problema seja aplicável em várias situações práticas como: controlo orçamental, selecção de projectos de investimento entre outros. O objectivo deste estudo foi de explorar eficientemente o algoritmo de etiquetagem numa abordagem multicritério, de forma a resolver instâncias, do problema da mochila, cada vez maiores e num menor espaço/intervalo de tempo, pois existe uma grande necessidade de encontrar algoritmos eficazes, que calculam as soluções não dominadas rapidamente. O propósito desta tese é implementar de modo exacto, um algoritmo para a resolução do problema da mochila multicritério 0 – 1 (com mais de 2 critérios), após a conversão do modelo do problema da mochila num problema do caminho mais longo multicritério numa rede acíclica. A metodologia proposta nesta tese usa eficientemente o algoritmo de etiquetagem e não necessita que o decisor realize qualquer tipo de julgamento sobre a relevância dos objectivos. A maior vantagem do uso deste algoritmo foi de calcular rapidamente as soluções não dominadas para certas instâncias do problema da mochila multicritério, conforme mostram os resultados computacionais. Contudo, devido a determinadas limitações computacionais este método revelou-se incapaz de resolver instâncias mais exigentes.

Palavras chave: Problema da Mochila, Optimização Combinatória Multiobjectivo, Algoritmo de Etiquetagem.

Abstract

Most of the existing papers regarding the Knapsack problem deal with a single criterion model. In this thesis we study the knapsack problem from a multiple criteria perspective. There are not many articles published on this topic, though this model describes several real-life applications such as, capital budgeting problems, project selection problems and others. The motivation for this study was to explore efficiently the labeling algorithm in a multiple criteria approach, that enable us to obtain non-dominated solutions quickly and to solve bigger instances than usually, because there is an emerging need for new algorithms able to compute non-dominated solutions quickly. The aim of this thesis is to implement an exact algorithm to solve the multiple criteria 0 – 1 knapsack problem (more than two criteria). Beforehand, we convert the multiple criteria knapsack model into a multiple criteria longest path over an acyclic network. The methodology proposed in this thesis efficiently uses the labelling algorithm and the decision maker does not have to judge the objectives about their priority. The advantage of this approach is to compute non-dominated solutions for multiple criteria 0 – 1 knapsack problems quickly for certain instances, as shown by the computational results. However, this exact approach is not able to solve hard instances.

Keywords: Knapsack Problem, Multiobjective Combinatorial Optimization, Labeling Algorithm.

Índice

1. Introdução	1
2. Problema da Mochila	4
2.1 Introdução	5
2.2 Problema da Mochila Monocritério	6
2.2.1 Problema da Mochila Inteiro	7
2.2.2 Problema da Mochila Inteiro 0 – 1	7
2.2.3 Problema da Mochila Restrito	8
2.2.4 Problema da Soma de Subconjuntos	9
2.2.5 Problema da Troca	9
2.2.6 Problema da Mochila Quadrático	10
2.2.7 Problema de Múltiplas Mochilas 0 – 1	11
2.2.8 Problema de Afectação Generalizada	11
2.2.9 Problema do Empacotamento	12
2.3 Problema da Mochila Multicritério 0 – 1	14
3. Optimização Combinatória Multiobjective	17
3.1 Introdução	18
3.2 Definições	19
3.4 Soluções não dominadas	20
3.4 Soluções Eficientes	21
3.5 Metodologias para resolver problemas multiobjective	22
3.6 Métodos para Optimização combinatória Multiobjective	24
4. Resolução do problema da mochila Multicritério 0 - 1	25
4.1 Definições	26
4.2 Formulação do problema da mochila como um problema do caminho mais longo	26
4.3 Implementação do algoritmo de Etiquetagem	29
5. Experiencias Computacionais e Resultados	34
5.1 Estrutura de dados	35
5.2 Dados de entrada e saída	35
5.3 Análise de Resultados	37
5.3.1 Instâncias bicritério	41
5.3.1 Instâncias multicritério	44
6. Conclusão, Recomendações e Desenvolvimentos futuros	49
7. Bibliografia	51
Anexos	53

Lista de Figuras

3.1	Representação do conceito de dominância no espaço dos objectivos	20
3.2	Conjunto eficiente e conjunto não dominado	21
4.1	Um arco de chegada	30
4.2	Um arco de chegada	30
4.3	Dois arcos de chegada	30
4.4	Parte do exemplo ilustrado mostrado a seguir	31
4.5	Modelo de rede	32
5.1	Ficheiro de entrada	34
5.2	Ficheiro de saída	35
5.3	Número de nós e arcos	37
5.4	Relação entre o número de etiquetas dominadas e não dominadas	38
5.5	Tempo de execução do algoritmo	38
5.6	Memória usada pelo programa	41
5.7	Memória usada pelo programa	41
5.8	Conjunto dos parâmetros de saída	41
5.9	Tempo de execução do algoritmo	42
5.10	Números de nós e arcos	43
5.11	Relação entre o número de soluções não dominadas e o número de critérios	44
5.12	Relação entre a memória usada e o número de critérios	44
5.13	Relação entre o tempo de execução e o número de critérios	45

Lista de Tabelas

5.1	Resultado de instâncias com 3 critérios e 35 objectos	35
5.2	Resultado de instâncias bicritério com diferentes números de objectos	40
5.3	Resultados de instâncias multicritério com 25 objectos	43
5.4	Resultado de instâncias com 3 critérios	45
5.5	Resultado de instâncias com 4 critérios	45
5.6	Resultado de instâncias com 5 critérios	46
5.7	Resultado de instâncias com 10 critérios	46

CAPÍTULO 1

Introdução

A resolução de muitos dos problemas que surgem no dia-a-dia consiste em escolher entre várias alternativas viáveis; ou seja, tomar decisões. A maioria destes problemas é de difícil resolução, uma vez que envolvem múltiplos critérios, geralmente conflituosos entre si. No entanto, em situações de grande pressão, opta-se usualmente por se utilizar modelos muito simples, em que apenas um dos critérios de decisão assume relevância, colocando de lado os restantes aspectos do problema, reduzindo-o a um problema de optimização do objectivo associado ao critério “relevante” geralmente o que se encontra associado à questão económica.

No início da década de 70, com a complexidade crescente do ambiente sócio-económico que caracteriza as sociedades tecnológicas modernas, verificou-se que quase todos os problemas envolviam vários critérios, geralmente conflituosos entre si, tornando-se difícil a formulação, modelação e obtenção da solução para o problema. As preocupações dos decisores não se limitam apenas ao campo económico em sentido restrito, mas também a outros campos como o político, o social, o meio ambiente, o estético entre outros.

A solução para alguns desses problemas consiste em transforma-los num problema da mochila multicritério que é um problema de optimização combinatória multiobjectivo. Apesar de inúmeros investigadores se terem dedicado ao estudo do problema combinatório da mochila, a determinação de uma solução óptima permanece um problema de difícil resolução. Para além disso, a maior parte dos estudos desses problemas foram dedicados à resolução do problema monocritério, que não leva em conta muitos aspectos relevantes que devem ser representados por vários critérios. No entanto, os problemas de decisão reais requerem, em geral, uma análise considerando explicitamente vários critérios, conforme foi dito acima. Por conseguinte, a abordagem multicritério parece ser mais realista.

Os problemas da mochila caracterizam uma classe de problema de programação linear inteira e são classificados na literatura, segundo a sua complexidade de resolução, como problemas *NP-Difícil*. Nesta tese, resolvemos uma das variantes do problema clássico da mochila, ou seja o problema da mochila com vários critérios e com variáveis binárias, conhecido como problema da mochila multicritério 0 – 1, por ser um dos problemas da mochila que é pouco estudado e aplicável em várias situações práticas.

A principal motivação deste estudo deve-se do facto, de pretendermos fazer um estudo para o problema da mochila com mais de dois critérios. O problema da mochila para dois critérios já foi testado em Captivo *et al.* (2003). O nosso algoritmo resolve, no entanto, o problema da mochila monocritério, bicritério e multicritério.

O objectivo desta tese é desenvolver um algoritmo exacto para a resolução do problema da mochila com mais de dois critérios, baseado na conversão do problema da mochila multicritério num problema do caminho mais longo multicritério numa rede acíclica, onde se aplica um algoritmo de etiquetagem que calcula todas as soluções não dominadas, por enumeração completa. Todavia, a determinação completa destas soluções é um processo computacionalmente moroso, tornando-se mesmo inviável a partir de determinadas dimensões (um problema com apenas 20 itens pode possuir muitas soluções). Assim, a via de resolução por enumeração completa é de utilidade muito reduzida, senão mesmo nula, para problemas de grande dimensão. Os métodos exactos existentes na literatura, até mesmo para o caso bicritério, ainda não podem ser aplicados com sucesso à resolução de problemas de grandes dimensões, o que tem motivado o desenvolvimento de métodos aproximados.

O problema da mochila pela sua expressão matemática, aparenta ser um problema de fácil resolução, o que não é verdade, por isso é conhecido como *NP-Difícil*. Tivemos muitas dificuldades em programar de raiz o nosso algoritmo e torná-lo eficiente em termos de busca das soluções, comparação, cálculo e na escolha e no uso adequado das estruturas de dados.

Usamos estruturas de dados dinâmicas, ou seja, apontadores (ponteiros) de estruturas, que é normalmente referenciado, nos livros de estrutura de dados como um dos aspectos mais exigentes da linguagem C. A pouca informação, existente na literatura sobre os problemas da mochila multicritério, contribuiu em certa parte, para as dificuldades que tivemos na concretização deste trabalho.

O problema da mochila com mais de dois critérios nunca tinha sido resolvido, nesta tese apresentamos pela primeira vez a sua resolução. Estudamos o comportamento e o desempenho do algoritmo utilizado para a resolução deste problema e os limites do mesmo, este estudo foi feito, mediante os resultados obtidos, em termos do número de nós, número de arcos, máximo de etiquetas usadas, número de soluções não dominadas, memória usada e o tempo de execução para várias instâncias multicritério geradas aleatoriamente.

Organização da tese

Esta dissertação encontra-se dividida em vários capítulos, os quais traduzem os diferentes aspectos do trabalho realizado subjacente à tese. Desta forma, e resumidamente, podem-se sintetizar aqueles aspectos no seguinte parágrafo:

No Capítulo 2, descrevemos o conceito do problema da mochila e introduzimos o problema em estudo nesta tese. No Capítulo 3, introduzimos alguns conceitos fundamentais sobre problemas de optimização combinatória e apresentamos metodologias que podem ser usadas para optimização combinatória multiobjectivo. No Capítulo 4, descrevemos o algoritmo de etiquetagem e a sua respectiva implementação. No Capítulo 5, mostramos os resultados

obtidos por aplicação do algoritmo de etiquetagem e o comportamento computacional do algoritmo. Finalmente no Capítulo 6, sintetizamos as principais valências, as maiores limitações e os aspectos que merecem ser desenvolvidos no futuro.

CAPÍTULO 2

Problema da Mochila

Neste capítulo, descrevemos o problema da mochila, apresentamos também uma breve história e algumas aplicações práticas. Falamos do problema da mochila numa perspectiva monocritério e multicritério. No monocritério, descrevemos as suas variantes e extensões. Para a vertente multicritério, introduzimos o problema da mochila multicritério $0 - 1$, que é o problema em estudo neste trabalho.

2.1. Introdução

Suponhamos que dispomos de um conjunto de objectos, com um peso e valor conhecidos, e de uma mochila com capacidade limitada. Determinar o subconjunto destes objectos cujo peso não excede aquela capacidade e que maximiza o seu valor total, corresponde a resolver o problema da mochila.

Os problemas da mochila são dos mais importantes em programação linear inteira e têm sido estudados intensivamente nos últimos anos por vários investigadores (ver, por exemplo, Martello e Toth, 1990 e Pisinger, 1995). Os problemas da mochila são aplicáveis em problemas de embalagem, de carregamento de equipamentos, de corte de materiais, de controlo orçamental e de selecção de projectos de investimento. O problema da mochila também é interessante por constituir um subproblema de modelos mais vastos, como por exemplo, os de constituição de tripulações de voo, de planeamento da produção, de problemas de partição e de concepção de circuitos electrónicos.

A primeira resolução do problema da mochila, por técnicas mais inteligentes, data dos anos 50, por aplicação da função recursiva (programação dinâmica) de Bellman. A partir de então, foram propostos inúmeros melhoramentos: a definição do limite superior para o valor óptimo da função objectivo, por Dantzig em 1957; a resolução do problema pela técnica de partição e avaliação sucessivas, por Kolesar em 1967; a resolução de problemas de grandes dimensões, igualmente pela técnica de partição e avaliação sucessivas, por Harowitz e Sahni, nos anos 70; o primeiro procedimento de redução da dimensão do problema (por fixação do valor de algumas variáveis) de Ingargiola e Korsh em 1973; um novo limite superior, por Martello e Toth, em 1977; o algoritmo de Balas e Zemel, em 1980, baseado na ordenação de apenas um subconjunto de itens; um novo algoritmo de Martello e Toth que permite resolver instâncias difíceis (Pisinger, 1995; Martello e Toth, 1998). (ver Gomes da Silva, 2003).

Toda a classe de problemas da mochila pertence à família dos problemas *NP - Difícil*, porém, utilizando algoritmos de programação dinâmica, diversos problemas desta classe podem ser resolvidos em tempo pseudo-polinomial. O tempo de execução, nestes casos, está directamente relacionado com a dimensão das instâncias, com número de critérios e com tamanho da mochila. Estes resultados surpreendentes vêm de várias décadas de pesquisas que têm exposto as propriedades estruturais especiais do problema da mochila, que tornam o problema relativamente fácil de resolver.

2.2. Problema da Mochila Monocritério

O problema da mochila monocritério pode ser definido da seguinte forma: dada uma instância do problema da mochila com um conjunto N de objectos, contendo n objectos j com o valor c_j e peso w_j , e de capacidade W (usualmente todos estes valores são números inteiros positivos). O objectivo é seleccionar o subconjunto de N cujo peso não excede a capacidade W e que maximiza o seu valor total. O Problema da Mochila Monocritério pode formular-se como se segue:

$$\text{Maximizar } \sum_{j=1}^n c_j x_j \quad (2.1)$$

$$\text{Sujeito a } \sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$x_j \in \{0, 1\} ; j = 1, \dots, n \quad (2.3)$$

No problema da mochila monocritério, existe apenas, em geral, uma solução que otimiza o critério envolvido. Assim a noção de solução óptima faz sentido.

Esta é representada pelo vector $x^* = (x_1^*, x_2^*, \dots, x_n^*)$, e o valor da solução óptima por z^* . O conjunto X^* representa o conjunto das soluções óptimas, isto é, o conjunto de objectos que corresponde ao vector de solução óptima.

Diversas aplicações práticas do problema da mochila têm restrições, como pedidos de urgências, prioridades, pacotes com pouco peso mas com muito volume, entre outros. Estas aplicações conduzem a várias extensões e variantes do modelo básico do problema da mochila que são essenciais principalmente para optimização prática desses problemas; algumas variantes do problema da mochila tornaram-se *standard* nos problemas em que são aplicáveis.

Descrevemos a seguir, alguns dos problemas que podem ser classificados como problema da mochila monocritério, segundos os trabalhos de Martello e Toth (1990) e Lin (1998) e apresentamos também as respectivas modelações matemáticas.

2.2.1. Problema da Mochila Inteiro

O problema modelado a seguir é conhecido, na literatura, como problema da mochila com variáveis não negativas ou simplesmente problema da mochila. Este problema da mochila é caracterizado por não ter limitações nas quantidades de objectos seleccionados.

Variável de decisão:

x_j : Quantidade de objecto do tipo j seleccionada, $j = 1, \dots, n$.

Modelação matemática

$$\text{Max } z(x_1, \dots, x_n) = \sum_{j=1}^n c_j x_j \quad (2.1)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$x_j \geq 0 \quad j = 1, \dots, n \quad (2.4)$$

Este modelo pode ser associado ao problema de corte de uma barra, que deve ser cortada ao longo do seu comprimento, sem que uma quantidade máxima de objectos seja especificada.

2.2.2. Problema da Mochila 0 – 1

Durante as últimas décadas, o problema da mochila 0 – 1 tem sido estudado por diferentes abordagens, tais como: em programação dinâmica e enumeração implícita. É, talvez, o mais importante problema da mochila e um dos mais estudados problemas de programação discreta, devido basicamente a três factores:

1. Pode ser visto como o mais simples problema de programação linear inteira.
2. Aparece como um subproblema em muitos problemas complexos.
3. É aplicável numa vasta gama de situações práticas.

No problema da mochila 0 – 1, temos a situação em que um único exemplar de cada objecto pode ser seleccionado. Neste caso as variáveis de decisão são:

$$x_j = \begin{cases} 1 & \text{se o objecto } j \text{ é incluído na mochila} \\ 0 & \text{caso contrário} \end{cases} \quad j = 1, \dots, n \quad (2.5)$$

Modelação matemática

$$\text{Max } z(x_1, \dots, x_n) = \sum_{j=1}^n c_j x_j \quad (2.1)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$x_j \in \{0,1\} ; j = 1, \dots, n \quad (2.3)$$

Este problema é um caso particular do problema em estudo neste trabalho (problema da mochila multicritério 0 – 1).

2.2.3. Problema da Mochila Restrito

Alguns problemas da mochila podem apresentar condições adicionais, como por exemplo, a limitação da quantidade de objectos a serem seleccionados. Neste caso, o problema passa a ser chamado de problema da mochila restrito.

Além dos dados necessários utilizados nos modelos anteriores, tem que se definir:

d_j : quantidade máxima de objectos do tipo j que pode ser seleccionada, $j = 1, \dots, n$;

Modelação matemática:

$$\text{Max } z(x_1, \dots, x_n) = \sum_{j=1}^n c_j x_j \quad (2.1)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$0 \leq x_j \leq d_j ; j = 1, \dots, n \quad (2.6)$$

O problema da mochila restrito pode ser visto como uma generalização do problema da mochila 0 – 1, onde $d_j = 1$ para $j = 1, \dots, n$.

2.2.4. O Problema da Soma de Subconjuntos

O problema da soma dos subconjuntos consiste em seleccionar um subconjunto de objectos cuja soma dos pesos dos objectos escolhidos se aproxime ao máximo de W , sem excedê-lo.

Modelação matemática:

$$\text{Max } z(x_1, \dots, x_n) = \sum_{j=1}^n w_j x_j \quad (2.7)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$x_j \in \{0,1\} ; j = 1, \dots, n \quad (2.3)$$

Este problema é um caso particular do problema da mochila 0 – 1, onde $w_j = c_j$ para $j = 1, \dots, n$. O problema da soma de subconjuntos, também é conhecido como problema da mochila de valor independente e surge em situações onde a quantidade desejada deve ser alcançada, de modo que a diferença entre os valores da capacidade da mochila W e a soma total dos pesos dos objectos seleccionados seja minimizado, sem que W seja ultrapassado.

2.2.5. Problema da Troca

O problema da troca geralmente aparece na literatura como um problema de minimização. O problema consiste em seleccionar um número x_j ($i = 1, \dots, n$) de objectos de cada tipo j de maneira que o peso total seja W e o número total de objectos seleccionados seja mínimo.

Modelação matemática:

$$\text{Min } z(x_1, \dots, x_n) = \sum_{j=1}^n x_j \quad (2.8)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j = W \quad (2.9)$$

$$x_j \geq 0 \text{ e inteiro}; j = 1, \dots, n \quad (2.4)$$

Este problema é chamado de problema da troca, porque pode ser interpretado como o problema de uma caixa (supermercado, banco, etc.) que deve devolver uma determinada quantia W usando para isto um número mínimo de moedas de valores específicos w_j ($j = 1, \dots, n$). Neste caso, para cada valor, um número ilimitado de moedas está disponível. É importante notar que a condição de igualdade imposta pode fazer com que não exista uma solução para o problema.

2.2.6. Problema da Mochila Quadrático

O problema da mochila quadrático é um dos problemas mais estudados na área de problemas da mochila não lineares. Um problema da mochila não linear, em geral, é um problema com função objectivo não linear ou que envolve restrições não lineares. Pode em geral ser formulado matematicamente da seguinte forma:

Modelação matemática:

$$\text{Max } z(x_1, \dots, x_n) = f(x) \quad (2.10)$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.11)$$

$$x_j \geq 0 \text{ e inteiro; } j = 1, \dots, n \quad (2.4)$$

Onde $f(x)$ representa uma função quadrática de x na forma $xQx^T + cx^T$ com $x = (x_1, x_2, \dots, x_n)$, $c = (c_1, c_2, \dots, c_n)$ e $Q = (q_1, q_2, \dots, q_n)$ para algum j com $q_j \neq 0, j = 1, \dots, n$.

Até aqui, os problemas da mochila consistiram em carregar apenas uma mochila. Os próximos problemas, também classificados por Martello e Toth (1990) e Lin (1998) como problemas da mochila, consistem em carregar várias mochilas.

2.2.7. Problema de Múltiplas Mochilas 0 – 1

O problema de múltiplas mochilas 0 – 1 consiste em carregar um conjunto de n mochilas ($n \leq m$) cujas capacidades são dadas por: $W_j, j = 1, \dots, n$. Os outros dados do problema são os mesmos do problema para uma única mochila.

O problema consiste em seleccionar subconjuntos disjuntos de objectos de modo que o valor das utilidades dos objectos seleccionados seja máximo, cada subconjunto pode ser incluído numa mochila diferente, cuja capacidade não seja menor que o peso total dos objectos no subconjunto.

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1 & \text{se o objecto } i \text{ é incluído na mochila } j \\ 0 & \text{caso contrário} \end{cases} \quad (2.12)$$

Modelação matemática

$$\text{Max } z(x_1, \dots, x_n) = \sum_{j=1}^n \sum_{i=1}^m c_i x_{ij} \quad (2.13)$$

Sujeito a:

$$\sum_{i=1}^m w_j x_{ij} \leq W_j, \quad j = 1, \dots, n \quad (2.14)$$

$$\sum_{j=1}^n x_{ij} \leq 1, \quad i = 1, \dots, m \quad (2.15)$$

$$x_{ij} = 0 \text{ ou } 1; \quad i = 1, \dots, m \text{ e } j = 1, \dots, n \quad (2.16)$$

Quando $n = 1$, o problema de múltiplas mochilas 0 – 1 reduz-se a um problema da mochila 0 – 1 simples, pois a restrição (2.15) torna-se redundante.

2.2.8. Problema de Afectação Generalizada

O problema de designação afectação pode ser descrito utilizando a terminologia aplicada para os problemas da mochila. O problema consiste em incluir cada objecto exactamente numa mochila, visando maximizar o valor total, sem incluir em nenhuma mochila um peso total que ultrapasse sua capacidade. Considere os seguintes dados:

- c_{ij} : Valor fornecido pela inclusão do objecto i na mochila $j, i = 1, \dots, m$ e $j = 1, \dots, n$;

- w_{ij} : Peso do objecto i se é incluído na mochila j , $i = 1, \dots, m$ e $j = 1, \dots, n$.

Modelação matemática

$$\text{Max } z(x_1, \dots, x_n) = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (2.17)$$

Sujeito a:

$$\sum_{j=1}^m c_{ij} x_{ij} \leq W_j ; \quad j = 1, \dots, n \quad (2.18)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, \dots, m \quad (2.19)$$

$$x_{ij} = 0 \text{ ou } 1; \quad i = 1, \dots, m \text{ e } j = 1, \dots, n \quad (2.16)$$

Este problema da mochila é, frequentemente, descrito na literatura como o problema de afectação óptima de m tarefas a n processadores, dado o valor c_{ij} e quantidade de recursos w_{ij} , correspondente a inclusão da tarefa i ao processador j e a quantidade total de recursos W_j suportados por cada processador j .

2.2.9. Problema de Empacotamento

O problema de empacotamento pode ser descrito usando a terminologia do problema da mochila onde:

- w_i : Peso do objecto i , $i = 1, \dots, m$;
- W : Capacidade de cada caixa.

Associa-se cada objecto a uma caixa, tal que o peso total dos objectos em cada caixa não exceda W e o número de caixas usada seja mínimo.

Variáveis de decisão:

$$y_j = \begin{cases} 1 & \text{se a caixa } j \text{ é usada} \\ 0 & \text{caso contrário} \end{cases} \quad (2.20)$$

$$x_{ij} = \begin{cases} 1 & \text{se o objecto } i \text{ é incluído na caixa } j \\ 0 & \text{caso contrário} \end{cases} \quad (2.21)$$

Modelação matemática

$$\text{Min } z(x_1, \dots, x_n) = \sum_{j=1}^n y_j \quad (2.8)$$

Sujeito a:

$$\sum_{i=1}^m w_j x_{ij} \leq W_j y_j, \quad j = 1, \dots, n \quad (2.22)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, \dots, m \quad (2.19)$$

$$y_j = 0 \text{ ou } 1, j = 1, \dots, n \quad (2.23)$$

$$x_{ij} = 0 \text{ ou } 1, i = 1, \dots, m \quad (2.16)$$

2.3. Problema da Mochila Multicritério 0 – 1

A maior parte dos estudos sobre a resolução do problema da mochila foram dedicados à resolução do problema da mochila com apenas um único critério (monocritério). No entanto, os problemas de decisão reais requerem, em geral, uma análise considerando explicitamente vários critérios. São exemplos de aplicação do problema da mochila multicritério a selecção de projectos de investimento com decisores e critérios múltiplos (ver Kwark *et al.*, 1996); a selecção de projectos de investimento que não são independentes entre si, em vias de comunicação (ver Teng e Tzeng, 1996) e a geração da fronteira eficiente discreta na selecção de projectos de investimento (ver Rosenblatt e Sinnuany-Stern, 1989).

Nos problemas com mais do que um critério não existe, em geral, uma solução que optimize simultaneamente todos os critérios envolvidos. Assim, a noção de solução óptima deixa de fazer sentido. Em contrapartida, existem soluções admissíveis com características particulares, designadas por soluções não dominadas, em que a melhoria do valor de um dos critérios é sempre feita à custa da degradação do valor de pelo menos um dos restantes, ou seja, os critérios estão em conflito.

O problema da mochila multicritério 0 – 1 é uma variante do problema da mochila multicritério que é o objecto de estudo neste trabalho, trata-se de um problema que ainda não está completamente explorado na literatura, e os métodos disponíveis para a sua resolução resolvem apenas problemas com não mais de dois critérios e ainda não podem ser aplicados com sucesso à resolução de problemas de grandes dimensões.

O Problema da Mochila Multicritério 0 – 1 pode ser formulado da seguinte forma:

$$\text{Max } z_1(x_1, \dots, x_n) = \sum_{j=1}^n c_j^1 x_j \quad (2.24)$$

$$\text{Max } z_2(x_1, \dots, x_n) = \sum_{j=1}^n c_j^2 x_j$$

\vdots

$$\text{Max } z_m(x_1, \dots, x_n) = \sum_{j=1}^n c_j^m x_j$$

Sujeito a:

$$\sum_{j=1}^n w_j x_j \leq W \quad (2.2)$$

$$x_j \in \{0,1\} ; j = 1, \dots, n \quad (2.3)$$

Variável de decisão:

$$x_j = \begin{cases} 1 & \text{se o objecto } j \text{ é incluído na mochila} \\ 0 & \text{caso contrário} \end{cases} \quad j = 1, \dots, n \quad (2.5)$$

Onde:

W : Capacidade da Mochila;

w_j : Peso do objecto j ;

c_j^i : Valorização do objecto j segundo o critério i , com $i = 1, \dots, m$;

c_j^i, w_j e W São valores inteiros positivos, $w_j \leq W$ e $\sum_{j=1}^n w_j > W$.

O problema pode ser escrito simplificadamente da seguinte maneira:

$$\begin{aligned} \text{Max } z_1(x) &= c^1 x \\ \text{Max } z_2(x) &= c^2 x \\ &\vdots \\ \text{Max } z_m(x) &= c^m x \end{aligned} \quad (2.25)$$

Sujeito a:

$$wx \leq W \quad (2.26)$$

$$x_j \in \{0,1\}^n \quad (2.27)$$

Em que:

$$c^1 = (c_1^1, \dots, c_n^1), c^2 = (c_1^2, \dots, c_n^2), \dots, c^m = (c_1^m, \dots, c_n^m) \quad (2.28)$$

$$w = (w_1, \dots, w_n) \quad (2.29)$$

$$x^T = (x_1, \dots, x_n)^T \quad (2.30)$$

É característica dos problemas multicritério o aumento significativo do número de soluções não dominadas com o aumento do número de critérios, o que dificulta a sua resolução. Existem várias abordagens para analisar problemas deste tipo, como, por exemplo, a enumeração completa de todas as soluções não dominadas, a pesquisa de um conjunto reduzido de soluções não dominadas, ou a pesquisa interactiva de soluções mediante um protocolo do tipo

questão-resposta, onde se incorporam progressivamente e de modo interactivo as preferências do decisor. Durante o processo de decisão que, em geral, consiste na escolha da "melhor solução possível", o decisor pode não ser capaz, ou não estar interessado, em analisar todas essas soluções, nestes casos o grande esforço computacional requerido pode ser inútil.

Por outro lado, existe o esforço computacional subjacente à determinação de soluções não dominadas, em instâncias de dimensões elevadas, ou mesmo, médias. Estas duas dificuldades estão de facto presentes nos problemas da mochila com um elevado número de itens.

Neste trabalho, abordamos o problema da mochila multicritério numa perspectiva não interactiva, porém, com recurso exclusivo a um método exacto. Devido a qualidades das soluções obtidas e pelo tempo de execução que é relativamente pequeno para instâncias pequenas e médias.

Mais a frente, falaremos embora não detalhadamente, de outros métodos exactos e aproximados que podem ser utilizados para a resolução deste tipo de problema.

CAPÍTULO 3

Optimização Combinatória Multiobjectivo

O propósito deste capítulo é introduzir alguns conceitos fundamentais sobre Problemas de Optimização Combinatória Multiobjectivo para ajudar a resolver o problema da mochila multicritério. Inicialmente fizemos a descrição do conceito de optimização combinatória multiobjectivo e apresentamos alguns conceitos adicionais como eficiência e dominância. Finalmente apresentamos metodologias que podem ser usadas para Optimização Combinatória Multiobjectivo.

3.1. Introdução

Muitos problemas do mundo real apresentam vários objectivos a serem cumpridos, que na maioria dos casos estão em conflito entre si, ou seja, a melhoria de algum(ns) objectivo(s) causa(m) consequentemente a deterioração de outro(s).

Um exemplo de problema com objectivos conflituosos seria o projecto de uma ponte onde se deseja minimizar o peso (custo) da estrutura e maximizar as frequências naturais de vibração (melhor desempenho dinâmico): à medida que se reduz o peso da ponte também diminuem as frequências naturais de vibração.

Portanto, não existe uma solução óptima única mas sim um conjunto de soluções. Essas soluções são óptimas, porque não existem outras soluções no espaço de pesquisa melhores do que elas, quando todos os objectivos são simultaneamente considerados, sendo conhecidas como soluções *óptimas de Pareto*.

Outro exemplo comum de problema combinatório multicritério é a compra de um computador. A aquisição óptima é aquela que fornece o custo mínimo enquanto maximiza o desempenho do equipamento. Os dois objectivos estão em conflitos entre si, porque uma vez que existem computadores com elevado custo e desempenho até aqueles com baixo custo e desempenho. Um computador com o mais alto desempenho pelo menor custo, embora ideal, não existe no mundo real.

Assim, nenhuma solução que tenha menor custo e desempenho pode ser considerada como superior a outra com maior custo e desempenho. Contudo, entre todas as configurações de equipamentos existem algumas que são superiores a outras, isto é, apresentam desempenho maior ou equivalente por um custo menor ou igual. Estas configurações (soluções) que superam outras são denominadas soluções não dominadas, enquanto que as configurações que são superadas por pelo menos uma outra são chamadas de soluções dominadas.

Deste modo, é muito interessante uma ferramenta que encontre o conjunto das soluções não dominadas para que o projectista escolha entre as alternativas, aquela que melhor se enquadra as necessidades do projecto. Esta é a tarefa da Optimização Combinatória Multiobjectivo.

Optimização combinatória é um campo que se encontra intensivamente estudado por vários investigadores, devido as suas aplicações práticas nos problemas reais, e tem prosperado nas últimas décadas. Ao contrário da optimização combinatória multiobjectivo, em que apenas nos anos mais recentes, aproximadamente desde 1990, houve um profundo interesse neste tópico, desde tese de doutoramento, desenvolvimento de metodologias específicas e um número considerável de artigos neste campo estão a ser publicados.

Constituem problemas de Optimização Combinatória Multiobjectivo algumas variedades de planeamento espacial, estudos de *lay-out*, empacotamento de caixas, posicionamento de satélites, alocação de trabalhadores ou máquinas a tarefas, projectos de computadores, escalonamento de equipas de distribuição, entre outros.

3.2. Definição

Problemas combinatórios são caracterizados pela possibilidade de combinação de diversas variáveis. É comum existirem diversos critérios a serem utilizados na avaliação de uma solução para um problema combinatório, o que torna o problema multiobjectivo (também conhecido como multicritério).

Nos problemas de optimização, em particular, quando o espaço de decisões é composta por variáveis discretas, fica caracterizado o seguinte problema de optimização combinatória multiobjectivo:

$$\begin{aligned} &\text{Maximizar } f_1(x) = c_1x \\ &\text{Maximizar } f_2(x) = c_2x \\ &\quad \vdots \\ &\text{Maximizar } f_h(x) = c_hx \end{aligned} \tag{3.1}$$

Sujeito a:

$$x \in X = \{x \in \mathbb{R}^n : x \geq 0, Ax = b, b \in \mathbb{R}^m\} \tag{3.2}$$

Em que:

h : número de funções objectivo (critérios);

n : número de variáveis do problema de decisão;

m: número de restrições do problema;

X: região admissível no espaço das decisões (ou das variáveis);

x: vector das variáveis do problema ($x = [x_1, x_2, \dots, x_n]^T$ é solução admissível),

A: matriz dos coeficientes tecnológicos ($m \times n$).

b: vector dos termos independentes (recursos disponíveis ou requerimentos), o que não implica perda de generalidade, pois mediante operações convencionais é sempre possível transformar qualquer problema num de maximização.

A região admissível no espaço das funções objectivas (o conjunto de todas as imagens dos pontos em X), pode ser definida da seguinte forma:

$$Z = \{z \in \mathbb{R}^h : z = (f_1(x), f_2(x), \dots, f_h(x)), x \in X\} \tag{3.3}$$

Na resolução de problemas com apenas um objectivo, procura-se encontrar a solução óptima, ou seja, a solução admissível que optimize a função objectivo, cujo valor é único, mesmo que existam soluções óptimas alternativas. Logicamente, em problemas com múltiplos objectivos, esse conceito não é aplicável, uma vez que uma solução admissível que optimize

um dos objectivos, não optimiza, em geral, os restantes objectivos, quando estes estão em conflito.

Na resolução de problemas multiobjectivo, pretende-se encontrar uma “melhor” solução de compromisso para o decisor, que possa constituir uma solução final do problema de decisão.

3.3. Soluções não dominadas

O conceito de solução não dominada surge, devido a determinação de uma solução óptima permanece um problema de difícil resolução em problemas com mais de um critério, e como não existe, em geral, uma solução que maximize simultaneamente os critérios envolvidos, a noção de óptimo dá lugar ao conceito de solução não dominada. Este conceito está explicado na definição seguinte.

Uma solução admissível diz-se dominada por outra, se ao passar-se da primeira para a segunda existir “melhoria” de pelo menos um dos objectivos, permanecendo inalterados os restantes. Por outro lado, uma solução não dominada caracteriza-se por não existir uma outra solução admissível que melhore simultaneamente todos os objectivos, isto é, a melhoria num objectivo é alcançada à custa de piorar, pelo menos, um dos outros. Matematicamente, tem-se:

- I. Sejam $x = (x_1, x_2, \dots, x_n)^T$ e $y = (y_1, y_2, \dots, y_n)^T$ duas soluções admissíveis de um problema multiobjectivo ($x, y \in X$). A solução x domina a solução y se e só se:

$$f_j(x_1, x_2, \dots, x_n) \geq f_j(y_1, y_2, \dots, y_n) \text{ para } \forall j \quad (3.4)$$

$$f_j(x_1, x_2, \dots, x_n) > f_j(y_1, y_2, \dots, y_n) \text{ para algum } j \text{ com } j = 1, 2, \dots, h \quad (3.5)$$

- II. Seja x uma solução admissível de um problema multiobjectivo ($x \in X$). A solução x diz-se não dominada se e só se não existir outra solução admissível y ($y \in X$) que domine x .

Uma forma de visualizar o conceito de dominância, é apresentado na figura 3.1, onde está representado no espaço dois objectivos: a região dominada por uma solução x , a região que domina esta solução e a região que não é comparável a x , para um problema de dois critérios.

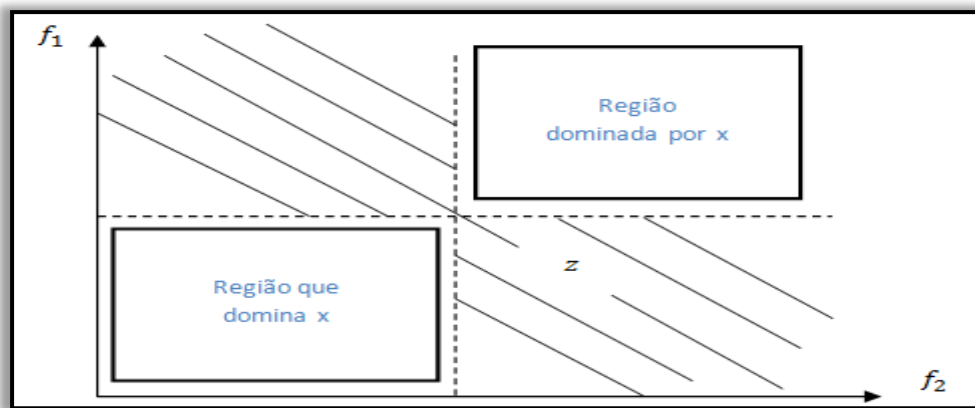


Figura 3.1 - Representação do conceito de dominância no espaço dos objectivos

Existem múltiplas técnicas que podem ser utilizadas na determinação de soluções não dominadas (ver Steuer, 1986). Uma delas consiste na optimização de uma função soma ponderada dos critérios.

3.4. Soluções Eficientes

O conceito de eficiência é geralmente utilizado para pontos no espaço de decisão, enquanto o conceito de não dominância é utilizado para a respectiva imagem no espaço das funções objectivo. Ou seja, uma solução não dominada é a imagem de uma solução eficiente. Matematicamente, tem-se:

- a) Uma solução admissível x_1 é **eficiente** ($x_1 \in X$) se e só se não existe outra solução admissível x_2 ($x_2 \in X$) tal que $f(x_2) \geq f(x_1)$ e $f(x_2) \neq f(x_1)$, em que $f = (f_1, f_2, \dots, f_h)$. Caso contrário, x_1 é ineficiente. Ao conjunto das soluções eficientes, dá-se o nome de fronteira eficiente ou conjunto de soluções não inferiores.
- b) Seja $z_1 \in Z$. Então z_1 é **não dominado** se e só se não existe outro $z_2 \in Z$, tal que $z_2 \geq z_1$ e $z_2 \neq z_1$. Caso contrário, z_1 é um vector de objectivos **dominados**.

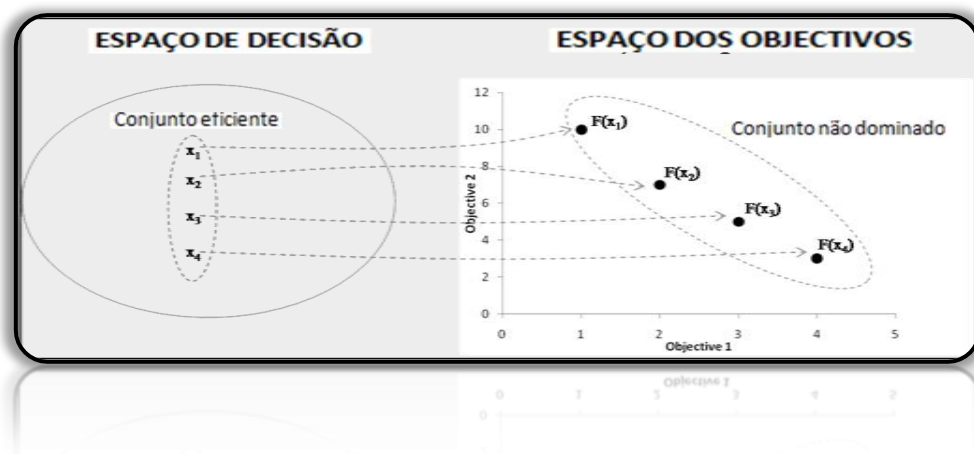


Figura 3.2 - Conjunto eficiente e conjunto não dominado

Uma solução de um problema multiobjectivo deverá ser eficiente, de tal modo que seja aceitável como solução final do processo de decisão. Desta forma, é apenas sobre o conjunto das soluções eficientes que deve recair a atenção do analista e do decisor.

Entre quaisquer duas soluções não dominadas verifica-se uma melhoria em pelo menos um dos objectivos que se encontra sempre associado a um sacrifício em pelo menos um dos outros objectivos. Isto é, verifica-se sempre uma compensação (“trade-off”) entre objectivos no conjunto das soluções não dominadas.

3.5. Metodologias para resolver problemas Multiobjectivo

Nos últimos anos tem sido dedicado um esforço considerável ao desenvolvimento de métodos para o cálculo de soluções não dominadas, tendo sempre em atenção o contributo que o decisor pode fornecer na procura de tais soluções, através do seu sistema de preferências. Desta forma, excluindo o caso trivial em que existe uma solução que otimiza em simultâneo todos os objectivos, podem-se considerar 3 abordagens para resolver um problema multiobjectivo, consoante o sistema de preferências do decisor é incorporado *a priori*, *a posteriori* ou progressivamente, no processo de decisão:

- Métodos em que é feita uma agregação *a priori* de preferências,
- Métodos geradores das soluções eficientes (não há articulação de preferências),
- Métodos de “articulação progressiva de preferências” (interactivos).

Nos do primeiro tipo, o decisor começa por indicar as suas preferências, a partir das quais é possível transformar o problema inicial num problema monocritério, por exemplo, através da construção de uma função utilidade. Desta forma, apesar de existir modelação multicritério do problema, pretende-se otimizar a função utilidade, cuja solução óptima será a solução final. A maior dificuldade deste processo está no facto de não ser possível, na maioria dos casos, obter uma representação matemática da função utilidade por parte do decisor. Ou seja, é difícil obter os parâmetros para construir uma função utilidade que agregue, numa única dimensão, todos os critérios em análise.

Nos do segundo tipo, são calculadas todas as soluções eficientes do problema (ou parte), que depois são colocadas à disposição do decisor para serem avaliadas. As críticas de que são alvo estes métodos devem-se ao elevado esforço computacional necessário para o cálculo exaustivo das soluções eficientes.

Nos métodos do terceiro tipo, o decisor expressa as suas preferências através de um processo de diálogo com a componente procedimental, de forma a conduzir a pesquisa para a zona da região admissível onde se localizam as soluções que melhor correspondem ao seu sistema de preferências. Na prática, este tipo de métodos tem mostrado ser o mais eficaz na pesquisa de uma solução final, através da utilização de processos de interacção Homem-máquina, alternando fases de cálculo com fases de diálogo. A intervenção humana no processo de pesquisa da solução é uma das características que distingue os métodos de programação multiobjectivo dos tradicionais da programação matemática (com um só objectivo).

Os métodos interactivos, aproveitando a intervenção do decisor, reduzem a zona de pesquisa, de forma quer a minimizar o esforço computacional, quer o esforço do decisor no processamento da informação. As críticas deste método devem-se que nem sempre o agente decisor está disponível durante o processo de resolução do problema.

Os métodos que utilizam o cálculo exaustivo para identificar o conjunto de soluções eficientes exigem um elevado esforço computacional e não têm em atenção a experiência humana, que poderia servir de apoio na resolução do problema, visto existirem múltiplos objectivos em conflito, e consequentemente, um grande grau de subjectividade no problema, mas este método calcula soluções bastante eficientes com tempos de execução relativamente pequenos para problemas com instâncias baixas e médias com um número razoável de itens. (ver Carlos Barrico, 1998)

3.6. Métodos para Optimização Combinatória

Multiobjectivo

Os métodos utilizados para resolver problemas de optimização combinatória multiobjectivo classificam-se como exactos ou aproximados. Nos métodos exactos as soluções obtidas são efectivamente não dominadas, ao contrário das apresentadas pelos métodos aproximados, onde são apenas potencialmente não dominadas.

Nos métodos exactos são utilizadas as técnicas soma ponderada, ε -restrição, minimização da distância a um ponto de referência para obtenção das soluções não dominadas (Steuer, 1986) e outros.

Em problemas com muitos objectivos e instâncias muito grandes, os algoritmos exactos levam muito tempo a resolver, por vezes não são muito eficientes; nestas circunstâncias, é importante encontrar métodos que levam menos tempo de execução, ou que pelo menos se aproximam da solução óptima. Estes métodos são denominados aproximados, as quais pertencem as heurísticas e as meta-heurísticas, como sejam os algoritmos genéticos, os métodos evolutivos, o *simulated annealing*, as redes neuronais, a pesquisa tabu, entre outros.

Nestes métodos pretende-se estabelecer um compromisso entre a qualidade das soluções obtidas e o tempo necessário para as calcular.

Recentemente, foram propostos métodos para a geração de todas as soluções não dominadas para o caso bicritério, nomeadamente o método de Visée *et al.* (1998) e de Captivo *et al.* (2003).

O método proposto por Visée *et al.* (1998) encontra-se estruturado em duas fases. Na primeira fase, determinam-se todas as soluções suportadas extremas e não extremas da envolvente convexa da região admissível, com recurso a um problema soma ponderada dos critérios. Na segunda fase, são analisadas, com recurso à técnica de partição e avaliação sucessivas, todas as soluções suportadas adjacentes com o objectivo de obter o conjunto das soluções não suportadas.

O método proposto por Captivo *et al.* (2003) converte o problema da mochila bicritério numa rede acíclica sobre a qual aplica um algoritmo de etiquetagem. O problema é transformado num problema de caminho mais longo bicritério. Este método calcula indistintamente as soluções suportadas e as soluções não suportadas.

Neste trabalho o método utilizado para resolver o problema da mochila multicritério 0 – 1 que é um problema de optimização combinatória multiobjectivo, é uma adaptação exacta do método proposto por Captivo *et al.* (2003) numa abordagem multicritério, por enumeração completa de todas as soluções não dominadas; esta técnica será descrita detalhadamente no capítulo seguinte.

CAPÍTULO 4

Resolução do problema da mochila multicritério 0 – 1

De modo a obter soluções eficientes não dominadas, calculadas rapidamente, para certas instâncias do problema da mochila multicritério, apresentamos o algoritmo de etiquetagem que calcula indistintamente as soluções suportadas e as soluções não suportadas, a partir da transformação do modelo do problema da mochila multicritério para o problema do caminho mais longo multicritério numa rede acíclica. Neste capítulo, descrevemos o algoritmo de etiquetagem e a sua respectiva implementação, e o problema da mochila multicritério é abordado como um problema do caminho mais longo. Os algoritmos apresentados neste capítulo e os seus respectivos procedimentos foram adaptados do artigo de Captivo *et al.* (2003).

4.1. Definições

Considere $\mathcal{G} = (\mathcal{L}, \mathcal{A})$ um grafo orientado e conexo, onde \mathcal{L} é o conjunto dos nós e $\mathcal{A} \subseteq \mathcal{L} \times \mathcal{L}$ é o conjunto dos arcos. O arco que liga o nó i ao j é representado por (i, j) , os valores associados ao arco (i, j) , referentes ao critério r são representados por $c^1(i, j), \dots, c^r(i, j)$.

O caminho p que parte do nó inicial s para o nó final t em \mathcal{G} , é uma sequência de arcos e nós de s para t . $f_k(p)$ representa o valor do caminho p relativo ao critério k , para $k = 1, \dots, r$. O objectivo é maximizar cada critério k . O caminho p^e diz-se eficiente se não existir nenhum caminho p em \mathcal{G} tal que $f_k(p) \geq f_k(p^e)$, para $k = 1, \dots, r$, com pelo menos uma desigualdade estritamente mantida. O problema do caminho mais longo multicritério consiste neste caso, numa computação de todos os caminhos eficientes do nó s para o nó t no grafo \mathcal{G} .

Considere X^e um conjunto de caminhos eficientes de s para t em \mathcal{G} . Note que é importante dar uma pequena definição de caminhos (ou soluções) eficientes suportados e não suportados. Um caminho eficiente diz-se suportado se pode ser obtido usando uma única função critério linear, que resulta da agregação de r funções critério lineares. Caso contrário é chamado de caminho eficiente não suportado. Como vimos atrás o nosso algoritmo não faz distinção entre uns e outros.

4.2. Formulação do modelo do problema da mochila como um problema do caminho mais longo

Esta secção mostra a transformação do problema da mochila como um problema do caminho mais longo. A maior vantagem desta abordagem é de calcular as soluções não dominadas mais rapidamente do que os outros métodos existentes, mas não melhora obviamente a complexidade teórica do problema da mochila multicritério que é conhecido como NP- Difícil.

Existe uma correspondência, de um para um, entre o conjunto das soluções do problema da mochila e o conjunto dos caminhos de s para t em \mathcal{G} . O caminho na rede \mathcal{G} e a solução do problema da mochila têm o mesmo valor e estrutura de solução.

Esta técnica pode ser implementada como se segue:

1. Determinar o conjunto dos nós usando a técnica das camadas.

Cada camada intermédia tem vários nós. A primeira camada (*camada 0*) tem um único nó, s . Então, a camada j pode ser obtida directamente da camada $j - 1$, para $j = 1, \dots, n$, onde cada

camada $j = 1$ para n , tem no máximo $W + 1$ nós, j^0, \dots, j^W . Finalmente, a última camada (camada $n + 1$) tem um único nó, t . Assim $|\mathcal{L}| \leq (W + 1)n + 2$.

2. Definir o conjunto dos arcos e os valores dos arcos.

Do nó s saem sempre dois arcos $(s, 1^0)$ com $c(s, 1^0) = 0$ e $(s, 1^{w_1})$ com $c(s, 1^{w_1}) = v_1$. O primeiro arco representa a decisão de não incluir o objecto 1 na mochila, e o segundo arco representa a decisão de incluí-lo. Entretanto, para as camadas de $j = 1$ para $j = n - 1$, cada nó j^a , para $a = 0, \dots, W$, tem no máximo, dois arcos de saída.

- O arco $(j^a, (j + 1)^a)$ com $c(j^a, (j + 1)^a) = 0$ significa que o objecto $(j + 1)$ não é incluído na mochila.
- O arco $(j^a, (j + 1)^{a+w_{j+1}})$ com $c(j^a, (j + 1)^{a+w_{j+1}}) = v_{j+1}$, se $a + w_{j+1} \leq W$ então o objecto $(j + 1)$ é incluído na mochila, se e só se a mochila tem capacidade suficiente para conter este objecto. Finalmente todos os nós pertencentes à camada n estão ligados ao nó t . Assim há mais $W + 1$ arcos com $c(n^a, t) = 0$, para $a = 0, \dots, W$.

Transformação do modelo do problema da mochila para o problema do caminho mais longo

BEGIN;

$Q \leftarrow \{s\}, T \leftarrow \{0, w_1\}$ e $V \leftarrow \{\};$

$Q \leftarrow Q \cup \{1^0, 1^{w_1}\};$

$A \leftarrow \{(s, 1^0), (s, 1^{w_1})\};$

$c(s, 1^0) \leftarrow 0$ e $c(s, 1^{w_1}) \leftarrow v_1;$

FOR ($j = 1$) **TO** ($n - 1$) **DO**

BEGIN

WHILE ($T \neq \{\}$) **DO**

BEGIN

Considere a como o primeiro elemento em T e remove – lo de T

IF ($a \notin T$) **THEN**

BEGIN

$V \leftarrow V \cup \{a\};$

$Q \leftarrow Q \cup \{(j + 1)^a\};$

END

$A \leftarrow A \cup \{(j^a, (j + 1)^a)\};$

$c(j^a, (j + 1)^a) \leftarrow 0;$

IF ($a + w_{j+1} \leq W$) **THEN**

BEGIN

IF ($a + w_{j+1} \notin V$) **THEN**

BEGIN

```

                                 $V \leftarrow V \cup \{a + w_{j+1}\};$ 
                                 $Q \leftarrow Q \cup \{(j+1)^{a+w_{j+1}}\};$ 
                                END
                                 $A \leftarrow A \cup \{(j^a, (j+1)^{a+w_{j+1}})\};$ 
                                 $c(j^a, (j+1)^{a+w_{j+1}}) \leftarrow v_{j+1};$ 
                                END
                                END
                                 $T \leftarrow V \text{ e } V \leftarrow \{\};$ 
END
 $Q \leftarrow Q \cup \{t\};$ 
WHILE ( $T \neq \{\}$ ) DO
BEGIN
        Considere  $a$  como o primeiro elemento em  $T$  e remove – lo de  $T$ 
         $A \leftarrow A \cup \{(n^a, t)\};$ 
         $c(n^a, t) \leftarrow 0;$ 
END
END

```

Para o algoritmo esquematizado acima procedemos da seguinte maneira:

1. Considere T e V duas listas para conter os nós das camadas j e $j + 1$, respectivamente.
2. Na primeira camada, definir o nó s e os seus arcos de saída $(s, 1^0)$ e $(s, 1^{w_1})$. A lista T é inicializada com 0 e w_1 ($T \leftarrow \{0, w_1\}$), Enquanto V é uma lista vazia ($V \leftarrow \{\}$).
3. Então, construir a rede, camada à camada, até à camada n . Em cada iteração a lista T conterá todos os nós referentes à camada j e sucessivamente uma nova lista, V , será preenchida com todos os nós referentes à camada $j + 1$. O conjunto dos arcos entre os nós sobrescritos em T e com os nós sobrescritos em V são criados simultaneamente. Nota-se que, os elementos em T seguem a regra *FIFO*. Este procedimento pára na iteração $n - 1$.
4. Finalmente os arcos fictícios são adicionados à rede por conexão de todos os nós que pertencem a T com o ultimo nó, t .

4.3. Implementação do algoritmo de Etiquetagem

O algoritmo descrito nesta secção é uma implementação particular do algoritmo de etiquetagem para o problema do caminho mais longo multicritério numa rede acíclica.

Este algoritmo leva em conta as seguintes características do modelo de rede que resulta do problema da mochila multicritério 0 – 1:

1. A rede também é gerada camada à camada, apenas o conhecimento dos nós que pertencem à camada $j - 1$ é requerido para construir o conjunto de nós da camada j e os arcos que ligam estas duas camadas consecutivas. Assim todos os outros nós e arcos não são levados em conta neste procedimento. Esta técnica utiliza pouco espaço de memória.
2. Para cada nó j^a , $a = 0, \dots, W$, chegam no máximo dois arcos $((j - 1)^a, j^a)$ e $((j - 1)^{a-w_j}, j^a)$. Desta forma, as etiquetas de j^a podem ser facilmente obtidas das etiquetas dos nós $(j - 1)^a$ e $(j - 1)^{a-w_j}$. Então o conjunto inteiro das etiquetas de j^a é gerada apenas numa iteração. As iterações subsequentes não são necessárias para o actualizar o conjunto de etiquetas não dominadas para cada nó, como é usual no caso quando se aplica o algoritmo de etiquetagem nas redes gerais.

Quando se gera uma nova etiqueta para um determinado nó, é necessário saber se esta etiqueta é dominada ou não. Então compara-se esta etiqueta com todas as etiquetas do nó, para o caso multicritério. Este processo pode consumir muito tempo.

Algoritmo de Etiquetagem para determinar caminhos eficientes

BEGIN;

$s(1^0) \leftarrow \{(0, \dots, 0)\}$ e $s(1^{w_1}) \leftarrow \{(v_1^1, \dots, v_1^r)\}$

$T \leftarrow \{0, w_1\}$ e $V \leftarrow \{\}$

FOR ($j = 2$) **TO** n **DO**

BEGIN

FOR ($a = 0$) **TO** W **DO**

BEGIN

IF ($a \in T$) **THEN**

BEGIN

$V \leftarrow V \cup \{a\};$

IF ($a - w_j \in T$) **THEN** {Dois arcos de chegada são definidos}

$S(j^a) \leftarrow$ Conjuntos de etiquetas não dominadas de

$(S((j - 1)^a) \cup \{(v_j^1, \dots, v_j^r)\}) + S((j - 1)^{a-w_j});$

ELSE {Apenas o arco $((j - 1)^a, j^a)$ é definido }

```

                                 $S(j^a) \leftarrow S((j-1)^a);$ 
END
ELSE
    BEGIN
        IF  $(a - w_j \in T)$  THEN { Apenas o arco  $((j-1)^{a-w_j}, j^a)$  é
                                definido }
            BEGIN
                 $S(j^a) \leftarrow \{(v_j^1, \dots, v_j^r)\} + S((j-1)^{a-w_j});$ 
                 $V \leftarrow V \cup \{a\};$ 
            END
        {ELSE O nó  $j^a$  não existe }
    END
END
     $T \leftarrow V \text{ e } V \leftarrow \{\};$ 
END
 $S(t) \leftarrow \text{Conjunto de etiquetas não dominadas de } \bigcup_{a=0}^W S(n^a);$ 
END

```

Os principais passos do algoritmo acima podem ser descritos como se segue:

1. Considere T e V duas listas para conter os nós das camadas $j-1$ e j , respectivamente.
2. Definir $S(j^a)$ como o conjunto de etiquetas não dominadas referente ao conjunto de todos os caminhos de s para j^a .
3. Inicializar os conjuntos $S(1^0)$ e $S(1^{w_1})$ com as etiquetas $(0, \dots, 0)$ e (v_1^1, \dots, v_1^r) , respectivamente. Este dois conjuntos contêm todas as etiquetas da camada 1.
4. Para cada uma das restantes camadas j ($j = 2, \dots, n$), tem-se que construir o conjunto $S(j^a)$ da seguinte maneira:

a) Se na camada j^a chegar apenas um arco, então procede-se mediante uma das duas diferentes possibilidades (ver Figuras 4.1 e 4.2):

- i. Se o arco for $((j-1)^a, j^a)$, isto é, a pertence a T mas $a - w_j$ não pertence. (ver Figura 4.1). Então $S(j^a)$ é igual a $S((j-1)^a)$ (ver o cálculo do $S''(4^3)$ a partir do $S(3^3)$ na Figura 4.4).
- ii. Se o arco for $((j-1)^{a-w_j}, j^a)$, isto é, a não pertence a T mas $a - w_j$ pertence (ver Figura 4.2). Então, as etiquetas em $S(j^a)$ pode ser obtidas pela soma do vector (v_1^1, \dots, v_1^r) com cada etiqueta do conjunto $S((j-1)^{a-w_j})$ (ver o cálculo do $S(4^6)$ a partir do $S(3^3)$ na Figura 4.4).

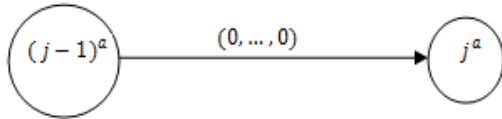


Figura 4.1 - Um arco de chegada

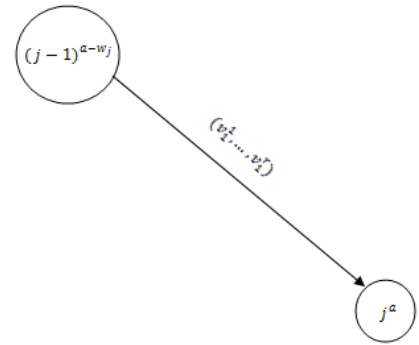


Figura 4.2 - Um arco de chegada

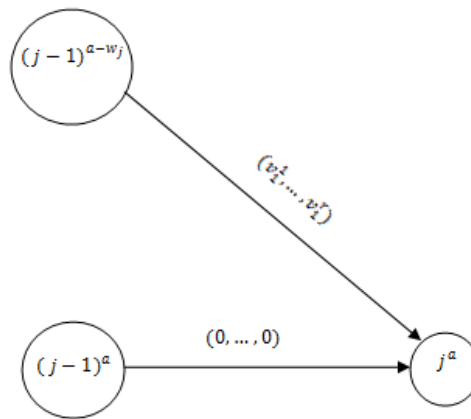


Figura 4.3 - Dois arcos de chegada

- b) Se em j^a chegam dois arcos, isto é, a e $a - w_j$ pertencem a T (ver Figura 4.3). Então, constrói-se $S(j^a)$ por escolha das etiquetas não dominadas, simultaneamente de etiquetas em $S((j-1)^a)$, e de etiquetas obtidas por soma do vector (v_j^1, \dots, v_j^r) para cada etiqueta do conjunto $S((j-1)^{a-w_j})$. (ver na Figura 4.4 a obtenção da etiqueta $S(4^3)$).

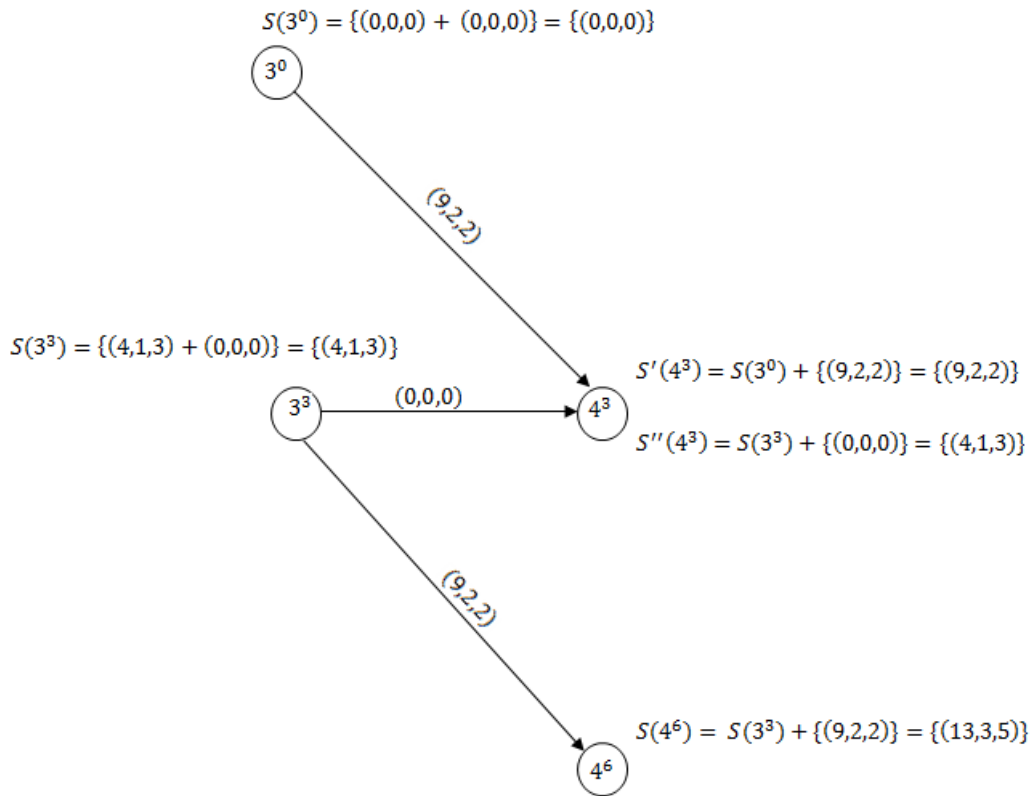


Figura 4.4 – Parte do exemplo ilustrativo mostrado a seguir.

Na Figura 4.4 se a etiqueta $S''(4^3)$ domina-se a etiqueta $S'(4^3)$, então $S(4^3)$ seria igual a $S''(4^3)$, caso contrário $S(4^3)$ seria igual $S'(4^3)$. Como $S'(4^3)$ não domina $S''(4^3)$, e vice-versa, então $S(4^3) = S''(4^3) \cup S'(4^3) = \{(9,2,2), (4,1,3)\}$.

Para clarificar todo procedimento da implementação algoritmo de etiquetagem, propomos o seguinte exemplo:

$$\text{Max } z_1(x) = 3x_1 + 4x_3 + 5x_3 + 9x_4$$

$$\text{Max } z_2(x) = 4x_1 + x_3 + 6x_3 + 2x_4$$

$$\text{Max } z_3(x) = x_1 + 3x_3 + 2x_3 + 2x_4$$

Sujeito a:

$$2x_1 + 3x_3 + 2x_3 + 3x_4 \leq 6$$

Onde $x_i \in \{0,1\}$ para $j = 1, \dots, 4$

O seguinte modelo de rede (onde o número de nós $|\mathcal{L}| = 19$ e o número de arcos $|\mathcal{A}| = 27$), foi obtido por aplicação do algoritmo de etiquetagem:

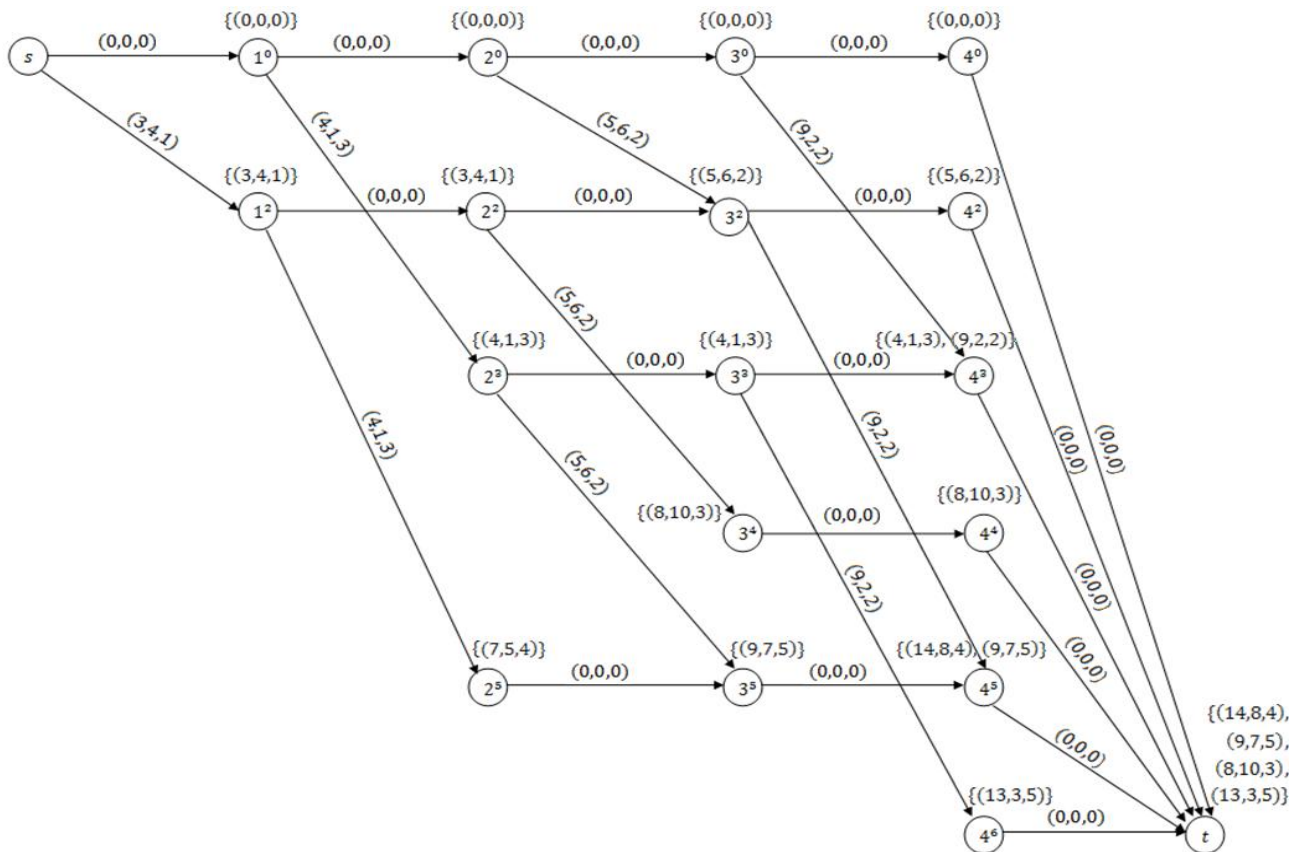


Figura 4.5 - Modelo de rede

A rede inicia-se no nó s e termina no nó t , os valores que aparecem em cima dos arcos representam os seus respectivos valores, os arcos que ligam a penúltima camada à última camada são normalmente todos nulos. As etiquetas de cada nó encontram-se em cima dos mesmos e são todas não dominadas, que posteriormente são utilizadas para a construção da camada seguinte.

A rede foi construída camada à camada, a partir de dois nós e arcos consecutivos não levando em conta os outros arcos e nós. O exemplo ilustrado tem 4 soluções não dominadas:

$$S(t) = \{(14,8,4), (9,7,5), (8,10,3), (13,3,5)\}.$$

CAPÍTULO 5

Experiências Computacionais e Resultados

Este capítulo mostra os resultados obtidos por aplicação do algoritmo de etiquetagem e o comportamento computacional do algoritmo, para certas instâncias do problema da mochila bicritério e multicritério geradas aleatoriamente. Todas as instâncias utilizadas nesta tese foram obtidas a partir dum gerador de instâncias, com base na geração de números aleatórios do gerador proposto por Klingman (1974) para problemas de fluxos em redes. Este gerador gera aleatoriamente instâncias do problema da mochila para vários critérios e objectos mediante a escolha do utilizador.

5.1 Estrutura de dados

Nesta tese, usamos estruturas de dados dinâmicas, ou seja, o programa inicialmente cria um apontador de estrutura, aloca-lhe memória de forma eficiente, consoante a dimensão da instância de entrada. O programa também aloca memória eficientemente nas iterações consecutivas do processo de resolução, desta maneira conseguimos fazer uma melhor gestão da memória disponível. O mesmo não sucederia se usássemos vector, que era a melhor alternativa em termos de facilidades de uso. Os acessos aos elementos do apontador de estrutura, são feitos da mesma forma que no vector, rápidos e eficazes.

5.2 Dados de entrada e saída

Vamos mostrar a seguir, por meio de um exemplo como se procede para inserir os dados de entrada e a leitura dos dados de saída. Dado o seguinte problema da mochila multicritério, temos:

$$\text{Max } z_1(x) = 3x_1 + 4x_3 + 5x_3 + 9x_4$$

$$\text{Max } z_2(x) = 4x_1 + x_3 + 6x_3 + 2x_4$$

$$\text{Max } z_3(x) = x_1 + 3x_3 + 2x_3 + 2x_4$$

Sujeito a:

$$2x_1 + 3x_3 + 2x_3 + 3x_4 \leq 6$$

Onde $x_i \in \{0,1\}$ para $j = 1, \dots, 4$

Os dados de entrada têm que ser escritos num ficheiro de texto, com extensão txt ou knp, e dispostos como mostra a Figura 5.1.

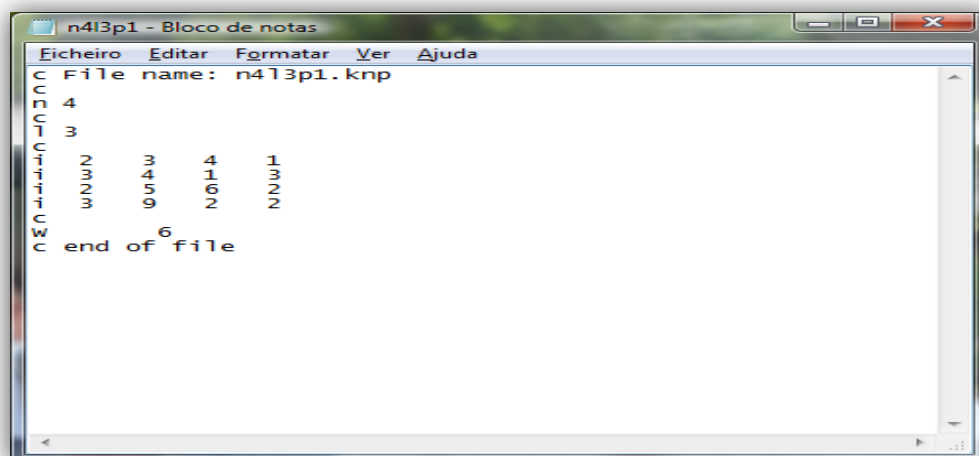
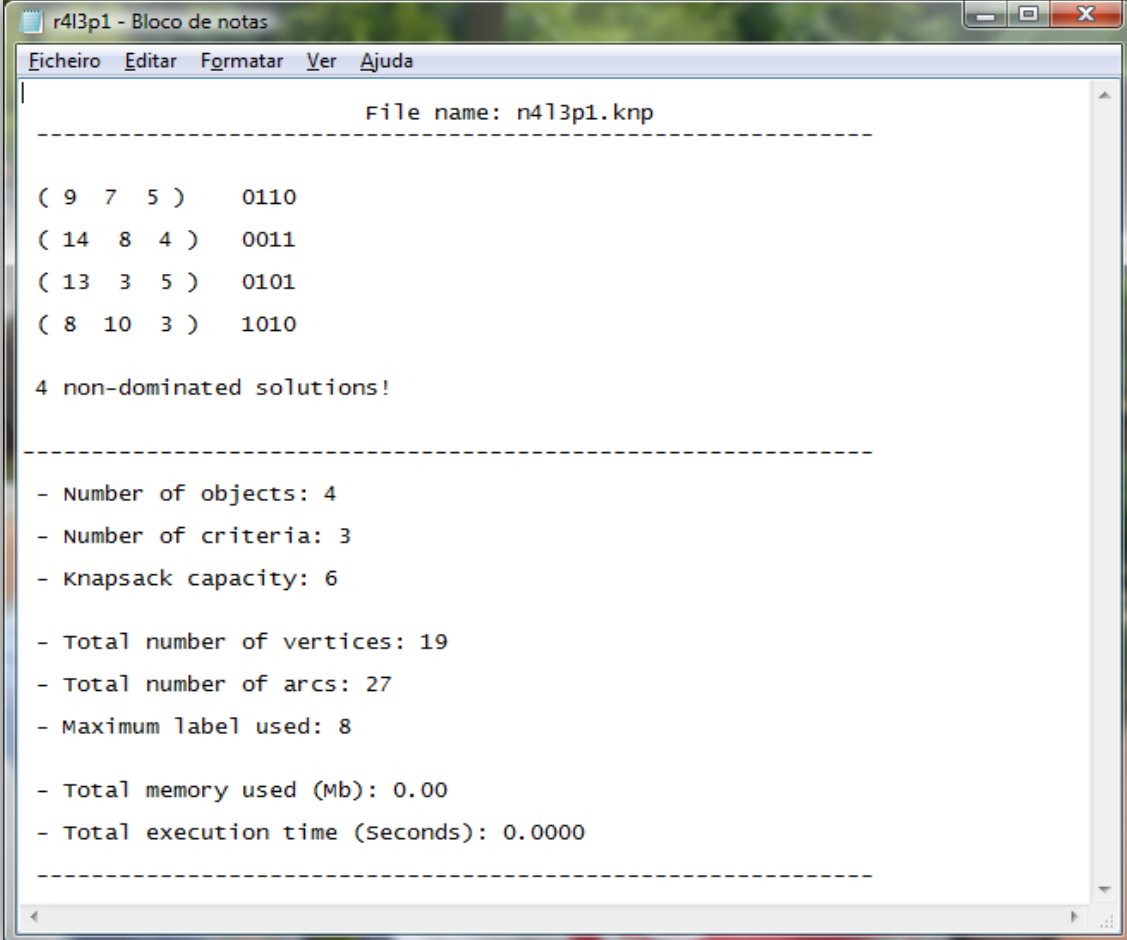


Figura 5.1 - Ficheiro de entrada

Onde n representa o número de objectos (variáveis), l o número de critérios (funções objectivo), w a capacidade da mochila (termo independente da restrição) e os i as matrizes transpostas dos coeficientes da restrição e das funções objectivo.

Os dados de saída são impressos num ficheiro do tipo *txt* ou *knp*; este ficheiro deve ser previamente indicado pelo utilizador, neste caso é o ficheiro *r4l3p1.knp*, conforme mostra a Figura 5.2.



```
File name: r4l3p1.knp
-----
( 9  7  5 )    0110
( 14  8  4 )    0011
( 13  3  5 )    0101
(  8 10  3 )    1010

4 non-dominated solutions!
-----
- Number of objects: 4
- Number of criteria: 3
- Knapsack capacity: 6

- Total number of vertices: 19
- Total number of arcs: 27
- Maximum label used: 8

- Total memory used (Mb): 0.00
- Total execution time (seconds): 0.0000
-----
```

Figura 5.2 - Ficheiro de saída

O problema tem 4 soluções não dominadas, como se pode ver na Figura 5.2; os números binários, que estão a seguir à essas soluções não dominadas, representam as indicações dos objectos incluídos ou não incluídos na mochila. Por exemplo, a solução (9 7 5), indica que $z_1(x) = 9$, $z_2(x) = 7$ e $z_3(x) = 5$; no entanto o número binário 0110 indica que os objectos x_1 e x_4 não são incluídos na mochila, enquanto os objectos x_2 e x_3 são incluídos. Neste mesmo ficheiro, por baixo do número de soluções não dominadas, são apresentados alguns parâmetros importantes como, os dados referentes à instância (número de objectos, número de critérios e capacidade da mochila), os dados essenciais sobre a rede acíclica (número de nós, número de arcos e máximo de etiquetas usadas por camadas) e o desempenho do algoritmo a nível de memória usada bem como o seu tempo de execução.

5.3 Análise de Resultados

Nesta secção analisamos os parâmetros de saída a nível dos dados da rede acíclica e do desempenho do algoritmo. Para correr o algoritmo que foi previamente programado em linguagem C, usamos um computador com um processador Intel Core 2 Duo a 1.80 GHz com 2GB de RAM. Inicialmente resolvemos 30 instâncias de 3 critérios e 35 objectos, geradas aleatoriamente a partir do gerador de Klingman (1974), e analisamo-los. Os resultados obtidos pela aplicação do algoritmo às 30 instâncias estão apresentados na Tabela 5.1.

Instância	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	872	20052	38428	126786	2,15	53,09	83
2	902	20709	39637	106327	1,83	39,53	117
3	904	21682	41567	201963	3,76	77,19	131
4	1024	23591	45190	242342	6,22	101,30	169
5	847	18488	35301	148230	2,90	50,47	144
6	976	22682	43480	223412	5,13	87,39	179
7	946	22165	42474	192861	4,26	69,48	183
8	819	20095	38581	145699	2,67	49,71	162
9	895	21319	40881	223227	6,25	86,39	183
10	865	20471	39238	170601	2,93	60,14	199
11	841	19871	38091	94283	1,59	37,75	126
12	684	16357	31384	212896	5,05	72,02	183
13	1060	24435	46802	153255	4,50	65,12	169
14	960	22422	42954	119474	2,18	45,19	109
15	736	17250	33025	85412	1,37	35,36	62
16	909	22243	42708	300532	7,35	122,81	235
17	885	21259	40753	110995	2,14	44,87	130
18	882	21313	40915	340199	10,00	130,97	332
19	881	21387	41031	154692	2,65	56,66	169
20	1007	22920	43913	355092	8,66	121,59	192
21	783	18274	35003	163209	4,23	64,35	199
22	900	21253	40802	188967	3,92	57,62	196
23	850	20852	40060	166753	3,06	71,92	130
24	976	21024	40183	211954	4,56	75,01	160
25	896	21854	41929	124140	2,75	47,56	151
26	848	20425	39195	110618	1,76	36,51	64
27	971	22286	42692	197161	4,10	60,63	185
28	924	22159	42556	305307	6,86	125,00	275
29	850	20301	38979	254452	6,79	77,94	190
30	793	19304	37051	80893	1,20	30,43	42
Médias	890	20948	40160	183724	4,09	68,47	162

Tabela 5.1 - Resultados de instâncias com 3 critérios e 35 objectos

Embora as diferentes instâncias tenham objectivos e restrições diferentes, com capacidades da mochila também diferentes, mas próximas umas das outras, a média do número de nós e arcos na rede acíclica é relativamente próxima dos valores máximos e mínimos. Para instâncias desta natureza teremos os valores do número de nós e arcos próximos dos seus valores médios.

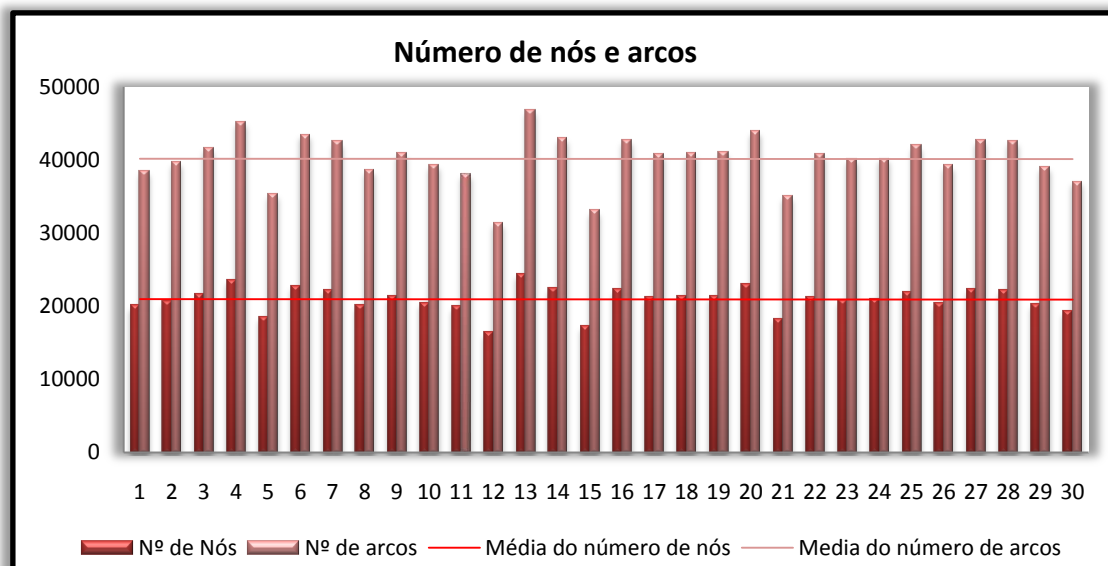


Figura 5.3 - Número de nós e arcos

Normalmente, de um nó saem dois arcos. Em apenas duas situações sai um único arco, quando a criação de um novo nó faz com que se ultrapasse a capacidade da mochila, consequentemente o novo nó e o arco são excluídos da rede ou quando estamos perante o nó final, onde chegam apenas um único arco de cada nó pertencente à penúltima camada. Então podemos dizer que o número de arcos é aproximadamente igual a dobro do número de nós, como podemos ver em cada instância do gráfico da Figura 5.3.

O máximo de etiquetas usadas representa a máxima dimensão do conjunto de etiquetas dominadas e não dominadas por camadas, este máximo de etiquetas pertence quase sempre à penúltima camada $n - 1$, então é a partir deste conjunto onde verifica-se as etiquetas não dominadas que passam para o nó final n e que são nomeadamente as soluções do problema.

Entretanto, ao analisarmos o número de soluções dominadas (máximo de etiquetas usadas menos o número de soluções (etiquetas) não dominadas) *versus* o número de soluções não dominadas para cada instância da Tabela 5.1, verificamos que existe uma diferença enorme entre os dois valores. O gráfico da Figura 5.4 mostra que aproximadamente 0.1% do conjunto das etiquetas dominadas e não dominadas representa o conjunto da solução final do problema. Então podemos concluir que o programa perde muito tempo a procurar as poucas soluções não dominadas finais num conjunto muito vasto de etiquetas, que na sua grande maioria é composta por etiquetas dominadas.

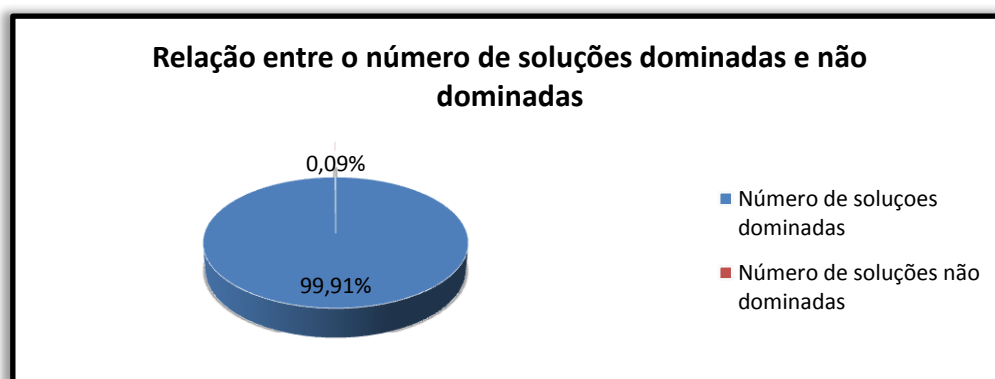


Figura 5.4 - Relação entre o número de etiquetas dominadas e não dominadas

Na Tabela 5.1 constata-se que algoritmo atingiu o tempo máximo de 10.00s e o tempo mínimo de 1.20s, o tempo médio de execução foi 4,09s. Face ao número de objectos e o número de critérios utilizados, consideramos estes tempos bons, porque são relativamente baixos. Mas para instâncias de dimensão média ou com um número elevado de critérios, o tempo de execução do algoritmo é muito elevado. Para uma melhor visualização do tempo de execução das diferentes instâncias da Tabela 5.1 construímos o seguinte gráfico.

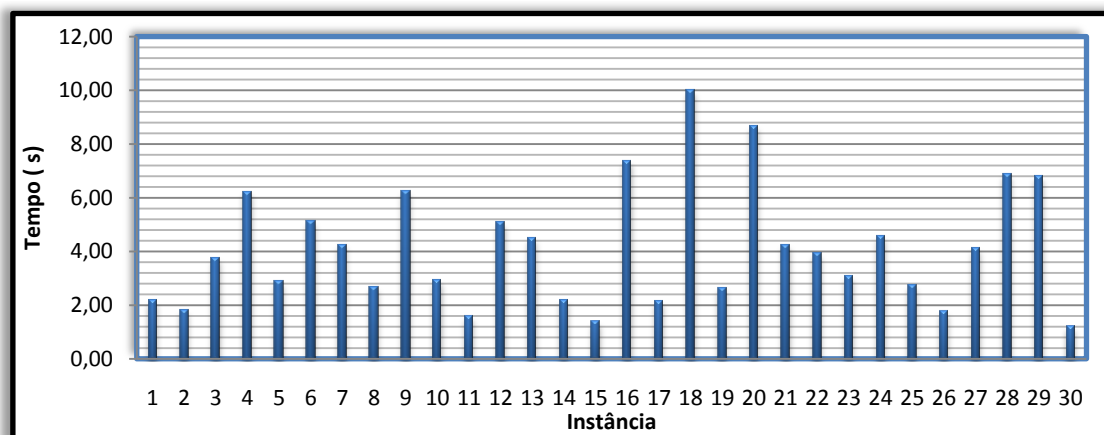


Figura 5.5 - Tempo de execução do algoritmo

A memória usada indica a memória ocupada e alocada pelas variáveis durante a execução do programa. Analisando a Figura 5.6, constatamos que as instâncias 4, 16, 18, 20 e 28 utilizam quantidades de memória consideráveis. A memória usada é um dos factores que condiciona fortemente o desempenho do nosso algoritmo, sendo uma clara limitação do mesmo. Como teremos oportunidade de ver mais à frente, instâncias de grandes dimensões ou com um elevado número de critérios, são normalmente mais exigentes em termos de memória por conseguinte, e tendo em consideração as limitações computacionais do computador utilizado, o nosso algoritmo não é capaz de resolver estas instâncias.

O valor médio da memória usada foi de 68.47Mb para as 30 instâncias de 35 objectos e 3 critérios o que é um pouco preocupante, porque à medida que se aumenta a dimensão da instância (número de objectos) ou o número de critério, o consumo de memória aumenta significativamente.

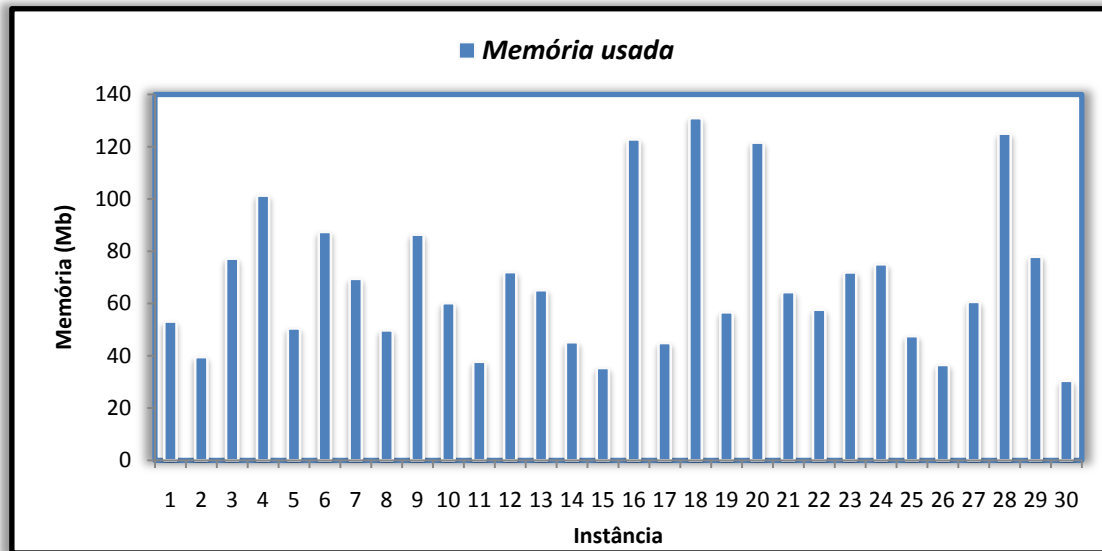


Figura 5.6 – Memória usada pelo programa

5.3.1 Instâncias bicritério

Nesta subsecção analisamos o comportamento computacional do algoritmo para instâncias bicritério com diferentes números de objectos. Consideramos apenas as instâncias bicritério porque o espaço de memória usado no caso bicritério é inferior ao dos casos com mais do que dois critérios, permitindo-nos resolver problemas com mais objectos.

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	122	40	53	16	0,00	0,00	2
10	254	634	987	250	0,00	0,02	4
15	373	2546	4470	1076	0,01	0,18	7
20	515	5501	10397	3790	0,04	0,77	11
25	632	9708	18069	10393	0,11	2,64	18
30	752	14720	27972	15076	0,23	4,77	17
35	878	20725	39734	46631	0,61	16,36	27
40	996	27635	53312	97893	1,49	39,03	40
45	1119	35293	68374	163769	2,37	66,58	46
50	1282	45722	88339	304370	5,36	146,05	54
55	1375	54436	106128	433838	9,42	244,13	56
60	1488	64938	126865	549850	12,81	318,00	57
65	1540	76333	149353	656433	17,60	399,07	64
70	1774	93004	191579	846352	26,68	500,43	69

Tabela 5.2 - Resultados de instâncias bicritério com diferentes números de objectos

Para problemas bicritério o nosso algoritmo consegue resolver instâncias até 64 objectos. Esta limitação do nosso algoritmo deve-se ao cálculo dos objectos que são incluídos na mochila. Se retirarmos esta componente de cálculo, o algoritmo consegue resolver instâncias até 70 objectos.

Como podemos observar, o cálculo dos objectos não é o principal responsável pela incapacidade do nosso algoritmo para resolver instâncias maiores. A razão da incapacidade de resolução de problemas bicritério com mais de 70 objectos prende-se essencialmente com a memória disponível do computador, para correr o programa. Se este exemplo fosse executado num computador mais robusto em termos de memória RAM, então este limite seria ultrapassado.

Nota-se que na Figura 5.7, à medida que o número de objectos aumenta, a memória usada aumenta exponencialmente, tendendo para a memória limite rapidamente. Este facto vem confirmar o anteriormente dito sobre as limitações impostas pela memória computacional.

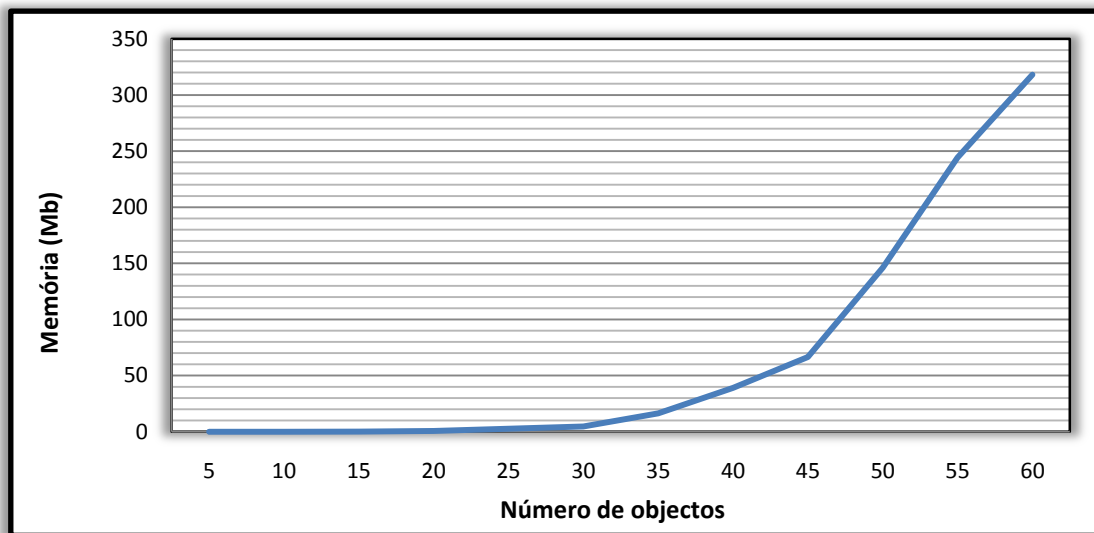


Figura 5.7 - Memória usada pelo programa

A figura abaixo foi construída do seguinte modo: como os parâmetros do gráfico abaixo são funções crescentes, então atribuímos 100% ao valor máximo para cada função e 0% ao valor mínimo, as percentagens intermédias foram obtidas a partir de interpolação linear.

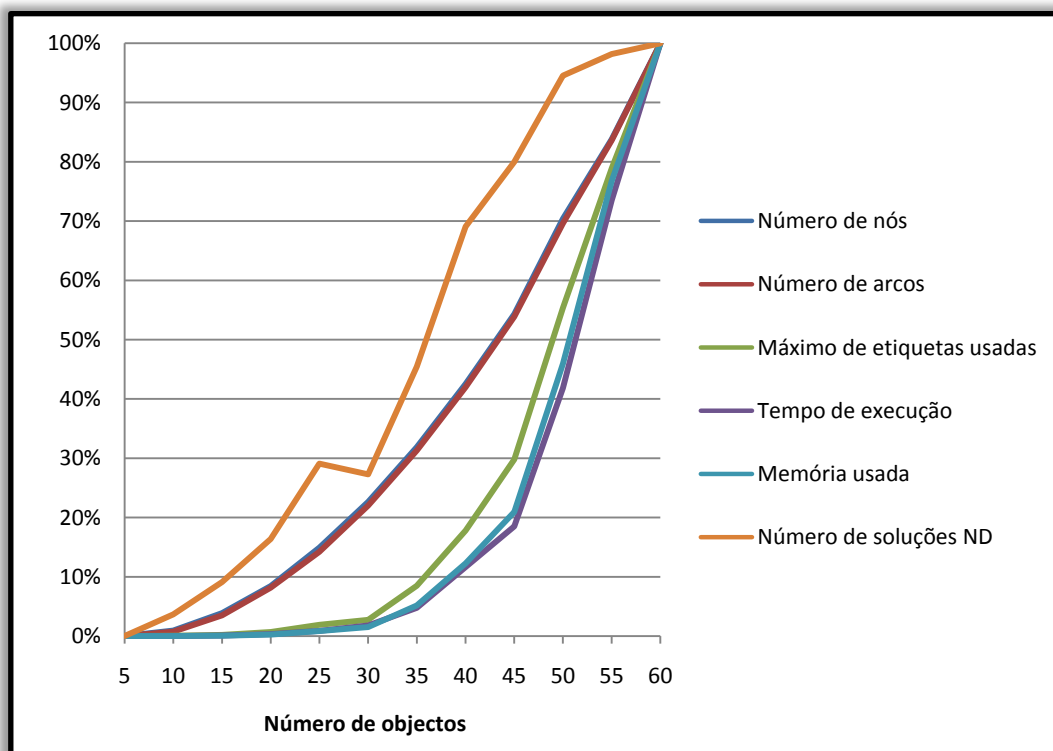


Figura 5.8 – Conjunto dos parâmetros de saída

Verificamos que no gráfico da Figura 5.8 o número de nós, o número de arcos, máximo de etiquetas usadas, o tempo de execução, a memória usada e o número de soluções não dominadas cresce com o aumento do número de objectos. Contudo, o número de nós, o

número de arcos e o número de soluções eficientes (número de soluções não dominadas) crescem mais rapidamente do que o máximo de etiquetas usadas, a memória usada e o tempo de execução. Convém referir que a quantidade de memória utilizada depende do número de etiquetas entre duas camadas consecutivas. Este facto explica a não resolução de problemas de grandes dimensões.

Na Figura 5.9, à medida que cresce o número de objectos (dimensão da instância) mantendo constante o número de critérios, o tempo de execução aumenta lentamente. No gráfico da Figura 5.8, o tempo de execução cresce mais devagar em relação aos outros parâmetros, na medida que aumenta o número de objectos. Então o algoritmo calcula rapidamente as soluções eficientes não dominadas para o caso bicritério.

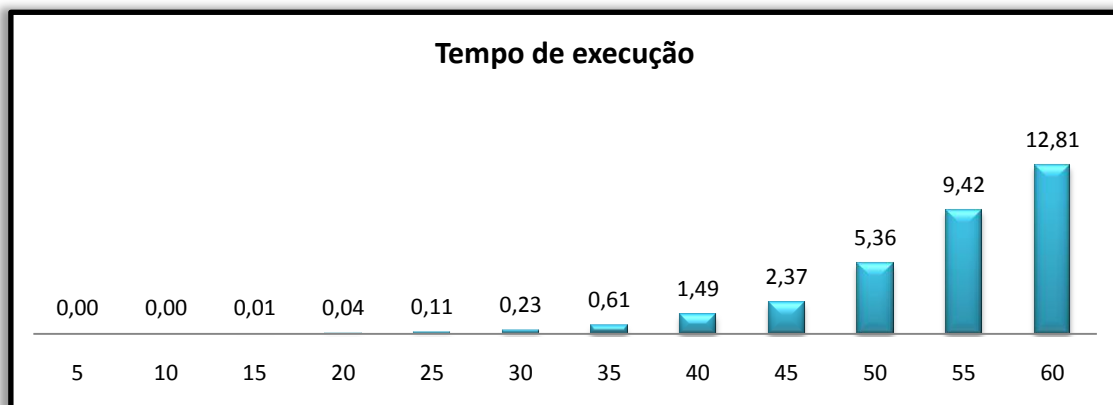


Figura 5.9 – Tempo de execução do algoritmo

5.3.2 Instâncias multicritério

Nesta subsecção, avaliamos o desempenho do algoritmo na medida em que aumenta o número de critérios e as consequências deste aumento. Existe um outro aspecto a ter em conta na resolução de instâncias multicritério, que é o tempo necessário para calcular as soluções não dominadas. De modo a analisar estes factos, geramos aleatoriamente instâncias multicritério de 25 objectos. As médias dos resultados obtidos dessas instâncias multicritério encontram-se apresentadas na Tabela 5.3.

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	632	9708	18069	10393	0,11	2,64	18
3	632	9708	18217	29690	0,39	7,81	70
4	638	9740	18272	62045	1,84	17,69	235
5	633	9663	18102	98406	6,61	28,83	455
6	626	9544	22374	151233	14,78	44,03	890
7	632	9644	18102	209541	44,81	70,27	1622
8	618	9473	19768	255478	84,60	96,20	2144
9	616	9509	17848	362159	184,63	131,13	3162
10	636	9432	17710	333400	201,47	136,71	2905
11	620	9025	18910	487761	339,74	179,98	4233
12	622	8955	17464	512673	477,54	198,38	4344
13	649	10512	19746	664703	583,68	259,42	5237
14	656	9909	18573	662980	605,46	299,23	6942
15	624	9594	18007	692452	799,53	358,72	7351

Tabela 5.3 - Resultados de instâncias multicritério com 25 objectos

Observando a Tabela 5.3, constatamos que com o aumento do número de critérios, o número de nós e o número de arcos têm pequenas variações entre os seus valores (ver Figura 5.9); isto deve-se ao facto de o número de nós e de arcos tender a aumentar com o aumento do número de objectos e não com o aumento do número de critérios. Como as instâncias resolvidas têm todas o mesmo número de objectos, isto é, todas as instâncias com 25 objectos, então existem pequenas variações entre as diferentes instâncias relativamente ao número de nós e arcos.

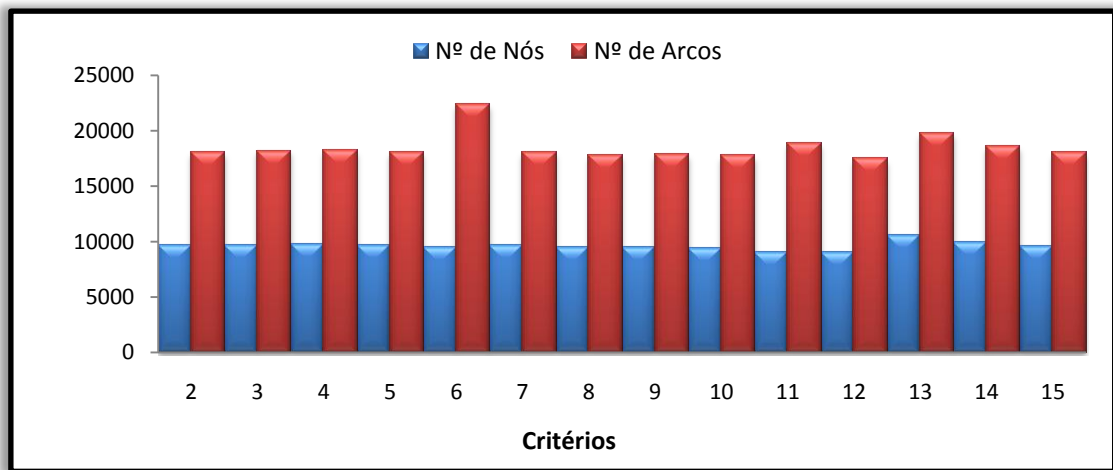


Figura 5.10 - Números de nós e arcos

O aumento de mais um ou dois critérios num problema da mochila multicritério não é uma condição suficiente para que o número de soluções do problema aumente. Mas à medida que aumentamos a dimensão do problema da mochila em termos do aumento do número de critérios, a quantidade das soluções do problema aumenta muito rapidamente (ver Figura 5.10). Pode acontecer, em alguns problemas com menos critérios, que exista um maior número de soluções eficientes do que noutros, com mais critérios. Esta situação pode-se verificar entre as instâncias com 9 critérios e 10 critérios da Tabela 5.3.

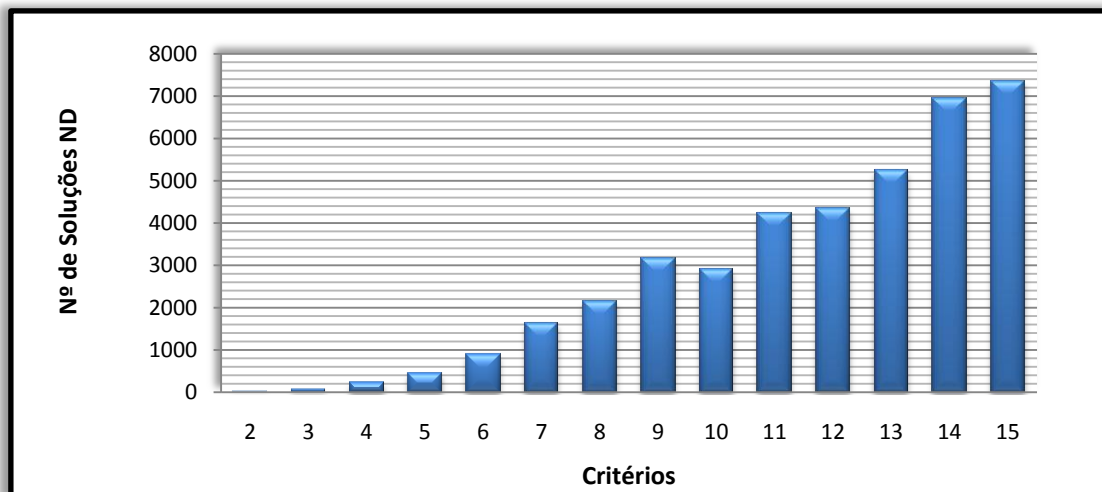


Figura 5.11 Relação entre o número de soluções ND e o número de critérios

Na Tabela 5.3 só nos foi possível resolver instâncias com 15 critérios, porque para instâncias com 25 objectos e com mais de 15 critérios, o programa excedeu a memória disponível que o computador disponibilizou para correr os problemas. Ou seja, só com o aumento do número de critérios o computador atinge rapidamente o limite de memória disponibilizado pelo computador. A Figura 5.11 mostra a evolução da memória usada quando se aumenta o número de critérios.

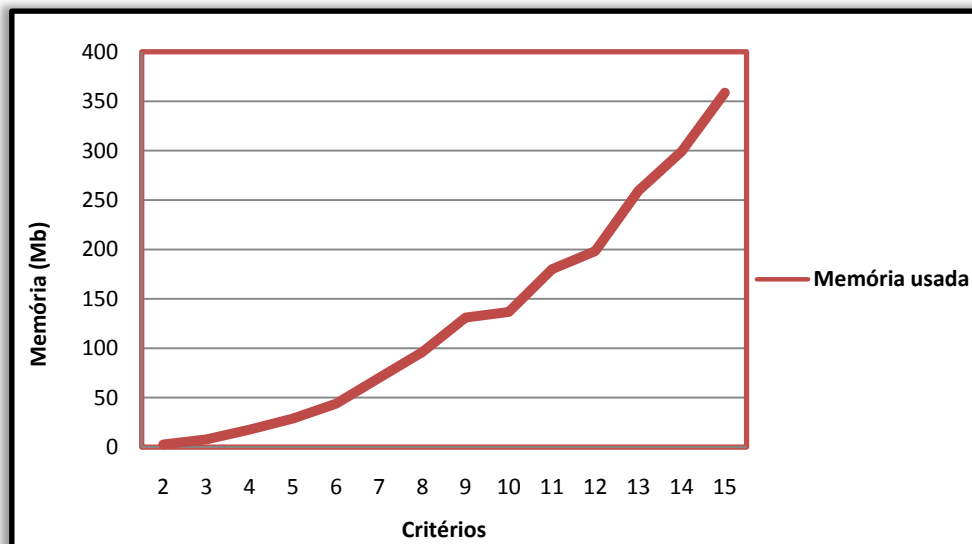


Figura 5.12 - Relação entre a memória usada e o número de critérios

O número elevado de critérios é um factor importante a ter em conta nos problemas da mochila, pelas razões acima citadas, mas não é o único. Um outro aspecto importante é o tempo de execução (o tempo que um algoritmo leva para calcular todas as soluções não dominadas). Este tempo está directamente relacionado com o tamanho das instâncias e com o número de critérios.

Na Tabela 5.3, mantemos constante a dimensão da instância, ou seja, consideramos todas as instâncias com 25 objectos mas aumentamos gradualmente o número de critérios. Constatamos ainda que, quanto maior for o número de critérios, maior é o período de tempo necessário para encontrar todas as soluções não dominadas do problema (ver a Figura 5.12).

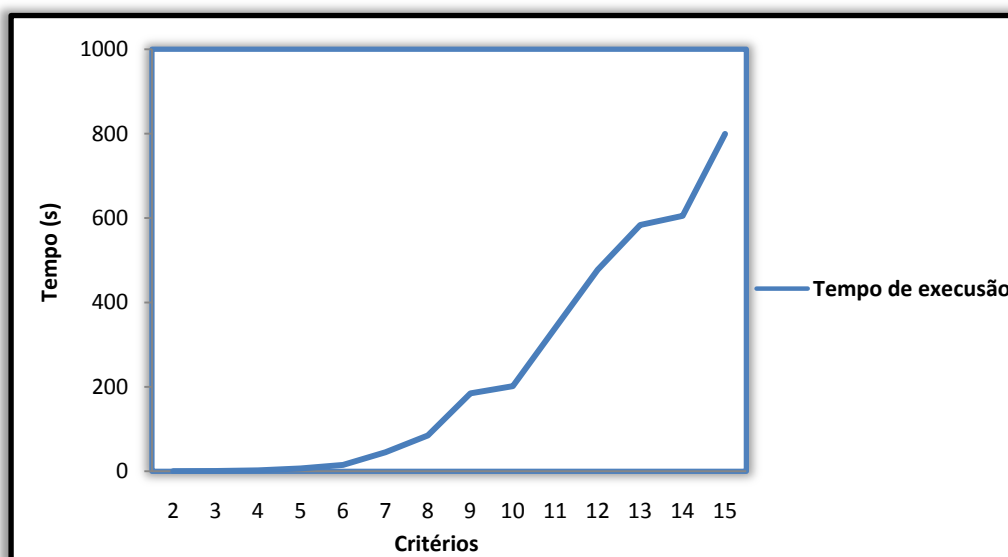


Figura 5.13 - Relação entre o tempo de execução e o número de critérios

As tabelas apresentadas a seguir mostram os resultados de várias instâncias com 3, 4, 5 e 10 critérios e o limite do nosso algoritmo em resolver essas instâncias multicritério.

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	130	41	55	16	0,00	0,00	3
10	240	539	838	246	0,02	0,03	15
15	342	2072	3621	2265	0,03	0,31	18
20	482	4948	9057	10132	0,11	2,15	48
25	574	8459	15839	28880	0,42	8,39	74
30	746	13806	26182	65245	1,40	24,86	108
35	890	20948	40160	183724	4,09	68,47	162
40	944	25435	49023	360457	11,29	134,16	232
45	1100	33655	65140	765385	32,93	326,23	408

Tabela 5.4 - Resultados de instâncias com 3 critérios

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	154	42	56	16	0,00	0,00	6
10	294	762	1168	392	0,02	0,04	19
15	403	2724	4792	2782	0,03	0,51	37
20	532	6024	11085	11143	0,17	3,06	70
25	684	10740	20168	44234	1,39	13,46	305
30	841	16671	31691	168561	6,68	50,40	526
35	990	23380	44811	486363	26,08	168,92	711
40	1110	30757	59303	1232942	113,94	485,99	1072

Tabela 5.5 - Resultados de instâncias com 4 critérios

Para problemas com 3 critérios, o nosso algoritmo consegue resolver instâncias até 45 objectos, conforme mostra a Tabela 5.4. Enquanto que, para problemas com 4 critérios o algoritmo consegue resolver instâncias com 40 objectos no máximo (ver a Tabela 5.5). Este limite foi diminuindo à medida que resolvíamos problemas com maior número de critérios (ver Tabela 5.6, Tabela 5.7 e anexos), pelas razões citadas neste capítulo.

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	92	41	55	16	0,00	0,00	2
10	225	569	884	248	0,02	0,04	11
15	376	2404	4183	2922	0,05	0,56	93
20	540	5592	10200	19165	0,41	5,10	109
25	698	9952	18583	80910	3,84	26,41	433
30	894	15990	30264	345509	23,13	105,85	1132
35	975	21666	41450	1037802	86,49	368,71	1437

Tabela 5.6 - Resultados de instâncias com 5 critérios

º de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	146	39	53	16	0,00	0,01	3
10	336	845	1288	489	0,02	0,09	27
15	449	3044	5352	7500	0,39	1,83	198
20	506	5775	10670	42629	3,39	16,37	346
25	696	10962	20623	289381	136,89	120,76	2738
30	806	16216	30892	1159752	451,76	585,62	3247

Tabela 5.7 - Resultados de instâncias com 10 critérios

CAPÍTULO 6

Conclusão e Desenvolvimentos Futuros

Muitos dos problemas reais que podem ser modelados através do recurso à teoria de optimização em redes, são de natureza multiobjectivo. Os objectivos geralmente envolvidos nestes problemas são os seguintes: custo, tempo, distância, acessibilidade, satisfação, protecção do ambiente, minimização do risco, segurança, entre outros.

Na resolução de problemas com apenas um objectivo, procura-se encontrar a solução óptima, ou seja, a solução admissível que optimize a função objectivo, cujo valor é único. Na resolução de problemas multiobjectivo, esse conceito não se aplica, uma vez que uma solução admissível que optimize um dos objectivos, não otimiza, em geral, os restantes objectivos, quando estes estão em conflito. Desta forma, a noção de solução óptima é substituída pela noção de solução não dominada. De uma maneira geral, existem três tipos de metodologias que podem ser utilizadas para resolver problemas de programação multiobjectivo, conforme as preferências do decisor são introduzidas *a priori*, *a posteriori* ou *progressivamente*, no processo de decisão.

A metodologia usada nesta tese, incorpora as preferências do decisor *a posteriori*, determinam-se inicialmente todas as soluções não dominadas do problema, as quais são posteriormente apresentadas ao decisor para que este proceda à sua selecção. A vantagem desta abordagem é que o decisor não necessita de realizar qualquer tipo de julgamento sobre a relevância dos objectivos. Para além desta, o conhecimento das diversas soluções permitem ao decisor aprender mais sobre as contrapartidas entre as diversas soluções. Quanto às desvantagens, existe um esforço computacional subjacente à determinação de todas as soluções não dominadas em certas instâncias e pode tornar-se complicado para o decisor, escolher uma solução no conjunto das soluções não dominadas, que pode ser muito vasto, e onde muitas vezes as soluções possuem características muito semelhantes.

As experiências computacionais e os resultados apresentados nesta tese mostraram que a abordagem exacta do nosso algoritmo, se revelou bastante eficiente para resolver instâncias multicritério de pequenas dimensões. Em contrapartida a memória usada foi um dos factores que condicionou fortemente o desempenho do nosso algoritmo, sendo uma clara limitação do mesmo, para correr instâncias de grandes dimensões.

Para problemas bicritério, o algoritmo só conseguiu resolver instâncias até 64 objectos. Ao retirarmos a componente de cálculo dos objectos que são incluídos na mochila, o algoritmo conseguiu resolver instâncias até 70 objectos. A razão da incapacidade de resolução de problemas bicritério com mais de 70 objectos prendeu-se essencialmente com a memória disponível do computador, pois à medida que o número de objectos aumentava, a memória usada aumentava exponencialmente, tendendo para a memória limite rapidamente.

Nos problemas multicritério, constatamos que quanto maior for o número de critérios, maior será a quantidade das soluções do problema, que, no entanto, aumentavam muito rapidamente com o aumento da dimensão do problema da mochila em termos do aumento do número de critério. O computador também atingia rapidamente a memória limite à medida que aumentava o número de critérios. Estes factos vêm confirmar as limitações impostas pela memória computacional.

Desenvolvimentos Futuros

Relativamente a métodos para resolver problemas deste tipo, apesar de já se ter um grau de desenvolvimento elevado, mas existindo poucos trabalhos nesta área, qualquer melhoria no algoritmo utilizado é vista com bons olhos. No que respeita aos algoritmos de busca, de comparação e nas estruturas de dados utilizadas, pode até haver introdução de novas ideias na forma de cálculo das soluções não dominadas.

O conjunto das soluções não dominadas no nosso algoritmo é identificado por comparação das soluções obtidas, armazenadas numa única lista. Se este modo permanecer para algoritmos futuros, então convém utilizar estruturas de dados mais complexas e métodos mais eficientes de busca dessas soluções, de modo a evitar que o algoritmo perca muito tempo em procurar as possíveis soluções não dominadas, num conjunto muito vasto de soluções que na sua maioria pode não ser construída por soluções não dominadas.

Os algoritmos que venham propostos futuramente devem fazer uma boa gestão de memória computacional, pois este revelou-se neste trabalho, como uma grande limitação para a resolução de problemas de grandes dimensões.

Um outro aspecto a considerar em desenvolvimentos futuros é a possibilidade desta abordagem ser testada em aplicações reais, uma vez que os testes aqui efectuados apenas utilizaram exemplos puramente académicos.

Bibliografia

Barrico, C. (1998) "*Uma Abordagem ao problema de caminho mais curto multiobjectivo - Aplicação ao problema de encaminhamento em redes integradas de comunicações*", Dissertação de Mestrado, Faculdade de Ciências e Tecnologia, Universidade de Coimbra.

Captivo, M., Clímaco, J., Figueira, J., Martins, E., Santos, J.L. (2003) "*Solving bicriteria 0 – 1 knapsack problems using a labeling algorithm*", Computers & Operations Research, 30, pp. 1865 - 1886.

Gomes da Silva, C., Figueira, J., Clímaco, J. (2003) "*An interactive procedure dedicated to the bi-criteria knapsack problem*". (In Portuguese), Research Report Nº 4, INESC -Coimbra, Portugal.

Klingman, J., Napier A., and Stutz, J. (1974) "*NETGEN – A program for generating large scale (un) capacitated assignment, transportation and minimum cost flow network problems*", Management Science, 20, pp 814 - 822.

Kwark, W., Lee, H., Lee, C. (1996) "*Capital Budgeting With Multiple Criteria and Multiple Decision Makers*", Review of Quantitative Finance and Accounting, 7, pp. 97 - 112.

Martello, S., Toth, P. (1990) "*Knapsack problems - Algorithms and computer implementation*", John Wiley & Sons, Inc.

Pisinger, D. (1995) "*Algorithms for knapsack problems*", Ph.D. thesis, Dept. of Computer Science, University of Copenhagen.

Rosenblatt, M., Sinuany-Stern, Z. (1989) "*Generating the Discrete Efficient Frontier to the Capital Budgeting Problem*", Operations Research, Vol. 37, Nº 3, pp. 384 - 394.

Steuer, R. (1986) "*Multiple Criteria Optimization, Theory, Computation and Application*", John Wiley & Son, Inc.

Teng, J., Tzeng G. (1996) "*A Multiobjective Programming Approach for Selecting Non-independent Transportation Investment Alternatives*", Transportation Research, B. Vol. 30, N.o4, pp. 291-307.

Visée, M., Teghem, J., Ulungu, E.L. (1998) "*Two-Phases method and branch and bound procedures to solve the bi-objective knapsack problem*", Journal of Global Optimization, 12, pp. 139 - 155.

Anexos

A.1. Tabela de resultados de instâncias com 6, 7, 8 e 9 critérios.

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	176	42	56	16	0,02	0,01	3
10	284	688	1092	427	0,00	0,05	27
15	338	2462	4356	3928	0,06	0,75	55
20	448	5111	9423	26269	0,80	6,48	85
25	584	9275	17451	110087	6,16	36,25	485
30	826	15729	29850	628480	115,18	226,83	2098

Tabela A1.1 - Resultados de instâncias com 6 critérios

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	65	37	52	17	0,00	0,00	3
10	276	591	925	413	0,02	0,06	26
15	354	2371	4157	4097	0,09	0,86	81
20	488	5573	10244	36898	1,25	9,98	330
25	689	10512	19746	152762	13,07	52,68	477
30	756	15166	28861	347074	30,39	144,23	867

Tabela A.1.2 - Resultados de instâncias com 7 critérios

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	142	32	44	16	0,00	0,00	3
10	236	692	1082	417	0,02	0,07	18
15	456	3037	5327	4281	0,09	1,08	166
20	594	6421	11775	34018	1,53	9,49	300
25	652	10133	19013	197185	32,71	73,48	1345
30	746	13806	26182	763959	337,02	435,90	5721

Tabela A.1.3 - Resultados de instâncias com 8 critérios

Nº de objectos	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
5	162	45	56	16	0,02	0,00	3
10	252	564	875	369	0,00	0,05	7
15	386	2474	4336	7290	0,41	1,66	222
20	488	6024	11085	38043	4,06	15,20	698
25	636	9599	18029	1142984	1069,23	428,17	10842
30	756	15166	28861	593309	189,71	296,88	2375

Tabela A.1.4 - Resultados de instâncias com 9 critérios

A2. Instâncias de 3 critérios

A2.1 Instâncias com 25 objectos

Instâncias	Capacidade da mochila	Nº de nós	Nº de arcos	Nº máximo de etiquetas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	612	9213	17238	24859	0,41	5,88	83
2	685	10406	19500	31489	0,37	7,33	54
3	760	11368	21317	29663	0,34	7,00	106
4	621	9654	18086	31546	0,44	9,21	77
5	646	10104	19048	35692	0,59	8,47	144
6	613	9622	18070	24020	0,28	6,61	50
7	609	9448	17711	14288	0,17	4,27	38
8	708	10575	19839	23219	0,26	6,08	44
9	501	8245	15492	14917	0,19	4,93	25
10	626	10115	19013	41280	0,61	10,07	93
11	740	11166	20934	24021	0,34	7,15	64
12	516	8155	15306	21392	0,27	6,38	39
13	606	9202	17274	28397	0,44	6,69	148
14	525	8361	15682	17238	0,19	5,58	22
15	521	8025	15067	19531	0,27	5,01	61
16	517	8024	15045	18183	0,28	5,59	93
17	607	9277	17422	24259	0,31	6,80	56
18	636	9691	18160	20348	0,23	5,45	76
19	705	10848	20374	21686	0,25	6,51	41
20	501	7946	14935	26066	0,44	7,90	89
21	594	9558	17973	28761	0,41	8,04	88
22	740	11109	20849	31028	0,44	9,82	83
23	586	9025	16910	16081	0,20	5,36	35
24	656	9909	18573	44008	0,72	9,68	161
25	552	8755	16464	15567	0,20	4,86	35
Médias	615	9512	17851	25102	0,35	6,83	72

Tabela A.2.1 - Resultados de instâncias com 3 critérios e 25 objectos

A2.2 Instâncias com 30 objectos

Instâncias	Capacidade da mochila	Nº de nós	Nº de arcos	Nº máximo de etiquetas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	832	15958	30335	50110	0,80	17,90	79
2	803	15585	29605	197070	4,46	47,78	210
3	750	15092	28730	55205	0,69	16,67	44
4	728	14534	27656	50410	0,66	15,03	61
5	725	14378	27332	84551	1,20	28,72	84
6	699	13777	26187	77330	1,25	21,28	69
7	620	12111	23032	76441	1,47	22,30	97
8	844	16471	31308	69781	1,56	24,78	193
9	821	15874	30139	56359	0,94	17,49	132
10	598	11890	22581	22154	0,28	7,78	38
11	819	16374	31161	62249	1,03	21,14	101
12	720	14654	27919	67394	1,15	27,83	86
13	744	14802	28173	62675	0,69	18,01	73
14	894	16498	31312	112091	1,72	35,49	103
15	744	15024	28596	78567	1,06	24,84	110
16	645	12745	24225	73705	1,33	23,85	107
17	863	16286	30973	103604	1,84	33,09	139
18	714	14420	27481	61134	0,94	24,14	67
19	814	14331	27130	91429	2,34	31,24	175
20	735	15106	28770	65737	0,87	22,40	51
21	762	14943	28423	43733	0,50	13,23	38
22	836	15634	29663	112942	2,14	37,61	185
23	813	15933	30333	84357	1,40	26,89	91
24	740	14461	27526	62943	0,73	21,98	44
25	684	13731	26145	82144	1,31	26,96	137
Médias	758	14824	28189	76165	1,29	24,34	101

Tabela A.2.2 - Resultados de instâncias com 3 critérios e 30 objectos

A2.3 Instâncias com 35 objectos

Instâncias	Capacidade da mochila	Nº de nós	Nº de arcos	Nº máximo de etiquetas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	872	20052	38428	126786	2,15	53,09	83
2	902	20709	39637	106327	1,83	39,53	117
3	904	21682	41567	201963	3,76	77,19	131
4	1024	23591	45190	242342	6,22	101,30	169
5	847	18488	35301	148230	2,90	50,47	144
6	976	22682	43480	223412	5,13	87,39	179
7	946	22165	42474	192861	4,26	69,48	183
8	819	20095	38581	145699	2,67	49,71	162
9	895	21319	40881	223227	6,25	86,39	183
10	865	20471	39238	170601	2,93	60,14	199
11	841	19871	38091	94283	1,59	37,75	126
12	684	16357	31384	212896	5,05	72,02	183
13	1060	24435	46802	153255	4,50	65,12	169
14	960	22422	42954	119474	2,18	45,19	109
15	736	17250	33025	85412	1,37	35,36	62
16	909	22243	42708	300532	7,35	122,81	235
17	885	21259	40753	110995	2,14	44,87	130
18	882	21313	40915	340199	10,00	130,97	332
19	881	21387	41031	154692	2,65	56,66	169
20	1007	22920	43913	355092	8,66	121,59	192
21	783	18274	35003	163209	4,23	64,35	199
22	900	21253	40802	188967	3,92	57,62	196
23	850	20852	40060	166753	3,06	71,92	130
24	976	21024	40183	211954	4,56	75,01	160
25	896	21854	41929	124140	2,75	47,56	151
Médias	892	20959	40173	152110	4,08	68,94	164

Tabela A.2.3 - Resultados de instâncias com 3 critérios e 35 objectos

A2.4 Instâncias com 40 objectos

Instâncias	Capacidade da mochila	Nº de nós	Nº de arcos	Nº máximo de etiquetas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	990	27558	53151	223323	4,65	110,48	78
2	1018	28206	54412	267127	8,46	157,59	257
3	930	25930	50020	428073	17,11	185,23	365
4	1005	27675	53367	261125	6,32	132,11	115
5	847	29031	56067	618756	17,03	289,92	276
6	942	26427	50980	324774	7,00	133,78	156
7	1060	30106	58138	231424	4,51	107,69	186
8	902	25565	49350	164486	2,73	83,67	79
9	848	23202	44728	318942	10,05	138,92	200
10	872	23809	45888	169547	4,59	62,51	134
11	1038	28753	55434	333008	9,14	147,83	143
12	1059	28108	54129	475160	18,25	209,90	242
13	898	25872	49970	199572	5,46	86,79	233
14	946	26331	50786	199899	3,59	94,26	118
15	1088	27973	53844	526155	14,71	200,71	287
16	1056	29954	57838	264133	5,32	109,43	75
17	1025	28876	55769	410759	13,29	154,57	277
18	920	25782	49743	266522	5,96	139,64	110
19	962	26063	50223	332468	12,25	118,80	370
20	1107	30043	57919	333307	9,55	146,81	142
21	943	26113	50383	541726	19,70	238,32	315
22	1120	30470	58749	422239	15,87	193,35	281
23	1110	30757	59303	393326	12,39	157,21	344
24	1060	29810	57539	372883	8,97	164,41	179
25	926	26068	50300	580242	23,91	240,52	311
Médias	987	27539	53121	346359	10	152	211

Tabela A.2.4 - Resultados de instâncias com 3 critérios e 40 objectos

A2.5 Instâncias com 45 objectos

Instâncias	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
1	1146	35867	69473	890557	50,72	448,89	323
2	1205	37782	73188	908541	43,51	422,24	448
3	1164	36087	69870	377961	11,79	170,81	228
4	1182	37478	72646	731637	29,03	322,71	260
5	1170	35772	69220	467517	15,71	237,92	315
6	900	28827	55866	500799	14,06	239,38	170
7	1232	38780	75136	445899	10,53	203,14	116
8	1218	37887	73388	1298057	71,08	477,00	652
9	750	26098	50043	748998	45,08	407,65	664
10	1109	35801	69401	923430	42,29	469,99	226
11	1062	33817	65548	778092	48,70	396,89	610
12	1198	36899	71434	816134	33,57	419,41	281
13	1100	36608	71077	768386	25,63	378,85	243
14	1306	40549	78514	909070	393,44	36,77	268
15	700	24776	47292	265964	11,50	146,74	353
16	1172	36673	71029	467635	18,61	264,84	368
17	1104	35071	67954	378447	11,40	199,96	276
18	1207	38337	74279	788260	45,35	130,97	332
19	1119	35729	69223	872163	30,75	361,34	366
20	1122	36566	70914	810021	31,51	424,85	169
21	981	33026	63936	891112	47,55	496,18	487
22	1037	32881	63704	339115	8,42	168,41	143
23	900	29910	57816	921717	44,94	472,95	469
24	1005	32064	61768	1089862	54,77	420,48	520
25	1105	34605	67014	1168772	105,30	464,95	897
Médias	1088	34716	67189	742326	50	327	367

Tabela A.2.5 - Resultados de instâncias com 3 critérios e 45 objectos

A3. Instâncias Multicritério

A3.1 Instâncias com 5 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	122	40	53	16	0,00	0,00	4
3	146	39	53	16	0,00	0,00	2
4	140	41	55	16	0,00	0,00	3
5	142	37	51	16	0,00	0,00	5
6	154	42	56	16	0,00	0,00	9
7	150	41	55	16	0,00	0,00	3
8	176	42	56	16	0,00	0,00	6
9	138	42	56	16	0,00	0,01	4
10	150	41	55	16	0,00	0,01	3
11	161	43	57	16	0,00	0,01	7
12	120	40	55	17	0,00	0,00	6
13	158	43	57	16	0,00	0,01	7
14	142	32	44	16	0,00	0,01	3
15	147	36	50	16	0,00	0,01	5

Tabela A3.1 - Resultados de instâncias multicritério com 5 objectos

A3.2 Instâncias com 10 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	254	634	987	250	0,00	0,02	4
3	336	845	1288	349	0,04	0,00	7
4	240	539	838	408	0,04	0,02	24
5	294	668	1034	325	0,04	0,02	17
6	294	762	1168	389	0,06	0,00	57
7	225	569	884	462	0,06	0,02	48
8	252	564	875	410	0,05	0,02	14
9	280	624	958	434	0,08	0,02	36
10	284	688	1092	473	0,08	0,00	41
11	288	717	1108	473	0,09	0,02	62
12	236	692	1082	493	0,10	0,02	46
13	293	661	57	16	0,10	0,01	50
14	270	629	947	457	0,11	0,00	35
15	248	491	752	503	0,14	0,02	10

Tabela A3.2 - Resultados de instâncias multicritério com 10 objectos

A3.3 Instâncias com 15 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	373	2546	4470	1076	0,01	0,18	7
3	449	3044	5352	1589	0,29	0,02	20
4	342	2072	3621	3349	0,60	0,05	35
5	412	2638	4645	4399	0,69	0,09	113
6	403	2724	4792	3884	0,73	0,08	121
7	376	2404	4183	7499	1,56	0,39	303
8	443	2990	5198	4816	1,21	0,09	119
9	386	2474	4336	7976	1,64	0,42	227
10	456	3037	5327	9649	2,14	0,76	364
11	450	2949	5140	9700	2,37	0,72	419
12	338	2462	4356	10739	2,65	1,53	487
13	437	2852	4988	7620	2,39	0,62	335
14	354	2371	4157	7186	2,09	0,87	234
15	424	2491	4267	9423	3,14	1,23	242

Tabela A3.3 - Resultados de instâncias multicritério com 15 objectos

A3.4 Instâncias com 20 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	515	5501	10397	3990	0,04	0,77	11
3	506	5775	10670	5904	0,06	1,32	32
4	482	4948	9057	19207	0,47	4,27	170
5	586	6190	11355	32889	1,23	6,88	594
6	532	6024	11085	24590	1,26	6,94	309
7	540	5592	10200	75204	8,56	20,63	904
8	540	6018	11013	45264	3,82	11,60	628
9	565	6039	11053	64339	7,11	17,95	814
10	594	6421	11775	122422	42,65	32,65	2031
11	596	6454	11849	118958	62,43	34,82	3061
12	488	5573	10244	142787	43,30	84,76	3916
13	523	5805	10671	74631	33,82	27,65	1602
14	448	5111	9423	82510	46,39	29,43	1689
15	577	5818	10574	150802	174,58	49,71	3511

Tabela A3.4 - Resultados de instâncias multicritério com 20 objectos

A3.5 Instâncias com 25 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	632	9708	18069	10393	0,11	2,64	18
3	698	9952	18583	28228	0,374	7,62	106
4	636	9599	18029	69041	1,372	19,60	247
5	696	10962	20623	97067	8,627	36,25	715
6	618	9538	17902	113734	6,177	45,32	581
7	652	10133	19013	164910	18,300	58,75	873
8	698	10427	19513	349234	101,603	111,76	3069
9	584	9275	17451	286063	73,414	110,82	1736
10	576	8459	15839	337776	127,047	136,71	2405
11	586	9025	16910	187561	39,734	109,98	993
12	552	8755	16464	312673	177,545	170,38	2344
13	689	10512	19746	664904	580,666	259,42	4237
14	656	9909	18573	622979	477,036	299,23	3942
15	624	9594	18007	662322	899,533	358,72	7351

Tabela A3.5 - Resultados de instâncias multicritério com 25 objectos

A3.6 Instâncias com 30 objectos

Critérios	Capacidade da mochila	Nº de nós	Nº de arcos	Máximo de etiquetas usadas	Tempo de execução (s)	Memória usada (Mb)	Nº de Soluções não dominadas
2	752	14720	27972	15076	0,30	4,47	17
3	806	16216	30892	66205	1,04	17,99	126
4	746	13806	26182	199862	9,33	71,31	417
5	740	14461	27526	537943	60,61	180,40	1405
6	845	16671	31691	572279	119,58	172,60	3237
7	684	13731	26145	389622	56,35	149,51	1070
8	756	15166	28861	779157	203,44	344,04	2272
9	756	15166	28861	593309	189,71	296,88	2375
10	684	13731	26145	988349	663,39	455,88	4282
11	684	13731	26145	1134232	952,34	548,70	4629
12	684	13731	26145	1453328	1937,03	717,97	7546
13	584	12042	22772	1008282	1313,73	589,42	8424
14	574	11865	22418	1029240	1425,01	647,56	8356

Tabela A3.6 - Resultados de instâncias multicritério com 30 objectos

