PERGAMON

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 30 (2003) 1865–1886

computers & operations research

www.elsevier.com/locate/dsw

Solving bicriteria 0–1 knapsack problems using a labeling algorithm

M. Eugénia Captivo[a], João Clímaco[b,c], José Figueira[b,c,d,*], Ernesto Martins[e], José Luis Santos[e,f]

[a] *DEIO-CIO, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, Bloco C2 1749-016 Lisboa, Portugal*

[b] *Faculdade de Economia, Universidade de Coimbra, Av. Dias da Silva, 165, 3004-512, Coimbra, Portugal*

[c] *INESC-Coimbra, Rua Antero de Quental, 199, 3000-033, Coimbra, Portugal*

[d] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75 775 Paris Cedex 16, Paris, France*

[e] *Departamento de Matemática, Universidade de Coimbra, Apartado 3008, 3001-454, Coimbra, Portugal*

[f] *CISUC, Departamento de Matemática, Universidade de Coimbra, Coimbra, Portugal*

Received 1 June 2000; received in revised form 1 February 2001

Abstract

This paper examines the performances of a new labeling algorithm to find all the efficient paths (or non-dominated evaluation vectors) of the bicriteria 0–1 knapsack problem. To our knowledge this is the first time a bicriteria 0–1 knapsack is solved taking advantage of its previous conversion into a bicriteria shortest path problem over an acyclic network. Computational experiments and results are also presented regarding bicriteria instances of up to 900 items. The algorithm is very efficient for the hard bicriteria 0–1 knapsack instances considered in the paper.

Scope and purpose

The knapsack problem is a well-known combinatorial optimization problem. While most of the existing papers in the literature studied the single criterion model, this paper deals with the knapsack problem with two criteria. Literature on this topic is scarce. There is a growing need for new algorithms able to compute non-dominated solutions quickly. The motivation of this study was to explore the use of efficient labeling algorithms to obtain non-dominated solutions of combinatorial optimization problems that can be formulated as network models. The purpose of the article is to implement a new algorithm for the bicriteria 0–1 knapsack problem. After converting the knapsack model into a bicriteria path problem over an acyclic network,

* Corresponding author. Faculdade de Economia, Universidade de Coimbra, Av. Dias da Silva, 165, 3004-512, Coimbra, Portugal. Tel.: +351-239-790590; fax: +351-239-403511.

E-mail addresses: maria.captivo@fc.ul.pt (M. Eugénia Captivo), jclimaco@inescc.pt (João Clímaco), figueira@fe.uc.pt, figueira@lamsade.dauphine.fr (J. Figueira) (E. Martins), zeluis@mat.uc.pt (Jose Luis Santos).

PII: S0305-0548(02)00112-0

the methodology proposes the use of a very efficient labeling algorithm. The scope of applications of the algorithm includes capital budgeting problems, transportation investments, and so on. The major advantage of our approach is to compute non-dominated solutions for bicriteria 0–1 knapsack problems faster than other approaches as shown by the computational results.

Keywords: Bicriteria knapsack problem; Bicriteria shortest path problem; Labeling algorithms

1. Introduction

Most of the existing books and papers on knapsack problems deal with a single criterion function (e.g. see Martello and Toth [1]). The classical knapsack problem seeks to select, from a finite set of items, the subset which maximizes a linear function of the items chosen, subject to a single inequality constraint. This model describes real-life applications: capital budgeting problems, cutting stock problems, loading problems, project selection problems and can be found as a subproblem of other more general models. Nevertheless, it is insufficient to take into account many other practical situations. The single criterion model cannot take into account many relevant aspects of reality that "must be" represented by several criteria. So, a multiple criteria model seems to be more realistic. More precisely, the paper introduces a new algorithm for the general multiple criteria 0–1 knapsack problem and presents results, in terms of the number of efficient paths, CPU time and total memory used, for a large set of bicriteria instances. There have been relatively few studies dealing with this model (Rosenblatt and Sinuany-Stern [2]; Kwak et al. [3]; Teng and Tzeng [4]). Knapsack models have also been used in modeling multiple criteria combinatorial optimization problems in the field of conservation biology (see Kostreva et al. [5]).

Recently, several approaches have been proposed to solve multiple criteria knapsack problems: branch-and-bound procedures by Visée et al. [6], dynamic programming by Klamroth and Wiecek [7], tabu search by Hansen [8], simulated annealing by Ulungu et al. [9] and hybrid meta-heuristics (tabu search and genetic algorithms) by Ben Abdelaziz et al. [10].

In this paper a different approach is used which takes advantage of the network structure of the problem, after converting the bicriteria knapsack model into a bicriteria shortest path model. The algorithm is inspired by the multiple criteria labeling shortest path algorithmic framework for acyclic networks (see [11]). But, the new algorithm is much more efficient because the characteristics inherent in the underlying network are taken into account. To our knowledge this algorithm is the first one designed and implemented to solve bicriteria knapsack problems formulated as a bicriteria shortest path model. In the experiments undertaken we had some difficulties coping with the insufficiency of memory requirements, preventing us from presenting results for more than two criteria. We are working on an implementation of an algorithm for more than two criteria and we hope to obtain new results soon.

Two other approaches to obtain the set of all non-dominated solutions are available in the literature. The branch-and-bound approach by Visée et al. [6], which was designed to take into account only two criteria, and the dynamic programming based approach by Klamroth and Wiecek [7], which was designed to take into account more than two criteria. This approach is also based on the transformation of a knapsack into a network model, but is only theoretical in nature and no results are presented.

The meta-heuristic based approaches by Hansen [8], Ulungu et al. [9] and Ben Abdelaziz et al. [10], generate a good approximation of the non-dominated set. However, approximative algorithms are not in the scope of this paper.

This paper is organized as follows. In Section 2, some definitions and the necessary background for carrying out the proposed approach are presented. In Section 3, the method is outlined and the following issues are discussed: (i) how to formulate a knapsack model as a shortest path problem, and (ii) the main characteristics of the labeling algorithm applied to the particular acyclic network generated from the knapsack model. In Section 4, computational experiments are reported and the efficiency of the new approach is discussed; a comparison with the branch-and-bound approach by Visée et al. [6] is also presented. Finally, in Section 5, some possible improvements are advanced as well as some suggestions for further research.

2. Definitions

The notation concerning the knapsack problem includes:

- n is the number of items;
- r is the number of criteria;
- v_j^k is the kth value (or the kth profit) of the jth item, for $k=1,\ldots,r$ (v_j will be used when $r=1$);
- w_j is the weight of the jth item;
- W is the total weight limitation of the knapsack (or the capacity of the knapsack).

The multiple criteria knapsack problem can be stated as follows:

$$\begin{aligned}
\max f_1(y) &= \sum_{j=1}^{n} v_j^1 y_j,\\
&\vdots\\
\max f_r(y) &= \sum_{j=1}^{n} v_j^r y_j, \\
\text{s.t.} \quad & \sum_{j=1}^{n} w_j y_j \leqslant W,\\
& y_j \in \{0,1\}, \quad j=1,\ldots,n,
\end{aligned} \tag{1}$$

where $y_j=1, j=1,\ldots,n$, if and only if item j is to be included in the knapsack, $W>0$, $v_j^k, w_j>0$ holds for all $j=1,\ldots n$ and $k=1,\ldots,r$.

To understand better the approach some additional notation and definitions from graph theory [12] and multiple criteria combinatorial optimization are necessary.

Let $\mathscr{G}=(\mathscr{Q},\mathscr{A})$ be a directed and connected graph, where $\mathscr{Q}$ is the set of nodes and $\mathscr{A}\subseteq\mathscr{Q}\times\mathscr{Q}$ is the set of arcs. The arc linking the nodes i and j is denoted by (i,j), and the values concerning the r criteria associated with the arc (i,j) by $c^1(i,j),\ldots,c^r(i,j)$. A path p from a starting node s to an end node t in $\mathscr{G}$ is a sequence of arcs and nodes from s to t, where the head node of a given arc coincides with the tail node of the next arc in the path. Let $f_k(p)$ denote the value of a path p with respect to criterion k, for all $k=1,\ldots,r$. The objective is to minimize each criterion k. A path p^{e} is said to be efficient if and only if there does not exist any path p in $\mathscr{G}$ such that $f_k(p) \leqslant f_k(p^{\mathrm{e}})$, for all $k=1,\ldots,r$, with at least one inequality holding strictly. The multiple criteria path problem concerns the computation of all efficient paths from a starting node s to an end node t in $\mathscr{G}$. Let X^{e} denote the set of all efficient paths from s to t in $\mathscr{G}$. Note that it is important to give a short definition of supported and unsupported efficient paths (or solutions). An efficient path is said to be supported if it can be obtained by using a single linear criterion function, which results from an aggregation of the r linear criteria functions. Otherwise, it is called an unsupported efficient path.

3. Outline of the method

To take advantage of the particular structure of network problems, the multiple criteria knapsack model is transformed into a multiple criteria shortest path problem over an acyclic network. This makes it possible to use a special labeling algorithm to search for all the efficient solutions. This transformation does not improve the theoretical complexity of multiple criteria knapsack problems which is known to be NP-hard as are multiple criteria shortest path problems [13,11]. The major advantage of our approach is to compute non-dominated solutions for bicriteria knapsack problems faster than other existing approaches.

3.1. Formulating the knapsack model as a shortest path problem

This subsection shows how to formulate the single knapsack problem as a shortest path model (see [12]). There is a one-to-one correspondence between the set of feasible solutions of the knapsack problem and the set of paths from s to t in $\mathscr{G}$. Both the path in $\mathscr{G}$ and the solution of the knapsack have the same (but symmetric) value or profit. The basic ideas of this technique can be stated as follows:

1. *Determine the set of nodes by using a layer technique.* Each intermediate layer has several nodes. The first layer (*layer* 0) includes the single node s. Then, layer j can be directly obtained from layer $j-1$, for all $j=1,\ldots,n$ where each layer, from $j=1$ to n, has at most $W+1$ nodes, $j^0,\ldots,j^W$. Finally, the last layer (*layer* $n+1$) has the single node t. Thus, $|\mathscr{Q}| \leqslant (W+1)n+2$.
2. *Define the set of arcs and the arc costs.* The node s has always two outgoing arcs $(s,1^0)$ with $c(s,1^0)=0$, and $(s,1^{w_1})$ with $c(s,1^{w_1})=-v_1$. The first arc represents the decision of not including item 1 in the knapsack, and the second arc represents the decision to include it. Concerning the layers from $j=1$ to $j=n-1$, each node j^a, for $a=0,\ldots,W$, has at the most two outgoing arcs:
   - The arc $(j^a,(j+1)^a)$ with $c(j^a,(j+1)^a)=0$, meaning that item $(j+1)$ is not included in the knapsack.

- The arc $(j^a, (j+1)^{a+w_{j+1}})$ with $c(j^a, (j+1)^{a+w_{j+1}}) = -v_{j+1}$, if $a + w_{j+1} \leqslant W$, meaning that item $(j+1)$ is included in the knapsack, if and only if it has sufficient capacity to contain this item. Finally, all nodes belonging to layer n are connected to the node t. Thus, there are at the most $W+1$ dummy arcs with cost $c(n^a, t) = 0$, for all $a = 0, \dots, W$.

To built the network efficiently we proceed as follows:

1. Let T and V be two lists of superscripts related to the nodes in layers j and $j+1$, respectively.
2. In the first layer, define the node s and the outgoing arcs from s, that is, $(s, 1^0)$ and $(s, 1^{w_1})$. The list T is initialized with two superscripts, 0 and w_1 ($T \leftarrow \{0, w_1\}$), while V is an empty list ($V \leftarrow \{\}$).
3. Then, build the network, layer by layer, until layer n. At each iteration, list T contains all superscripts related to the nodes in layer j and successively a new list, V, is built with the superscripts of all nodes in layer $j+1$. The set of the arcs between the nodes with the superscripts in T and the nodes with the superscripts in V are thus created simultaneously. It should be noted that the elements in T follow the *FIFO* rule. The procedure stops at iteration $n-1$.
4. Finally, dummy arcs are added to the network by connecting all the nodes with superscripts belonging to T with the last node, t.

Some properties of the network generated by the procedure given in Fig. 1:

1. It has no cycles.
2. There is a one-to-one correspondence between the set of feasible solutions of the knapsack problem and the set of paths from s to t in $\mathscr{G}$.
3. The shortest path from s to t represents the optimal solution for the knapsack model and the value of this solution is the negative "cost" of the shortest path in the network.

The multiple criteria generalization of the model is obvious. It is only necessary to assign a vector with r components to each arc of the network, say: $(c^1(i,j), \dots, c^r(i,j))$.

To illustrate, consider the following bicriteria knapsack example:

$$\max f_1(y) = 8y_1 + 9y_2 + 3y_3 + 7y_4 + 6y_5$$

$$\max f_2(y) = 3y_1 + 2y_2 + 10y_3 + 6y_4 + 9y_5$$

$$\text{s.t. } 3y_1 + 2y_2 + 2y_3 + 4y_4 + 3y_5 \leqslant 9$$

with $y_j \in \{0,1\}$ for $j = 1, \dots, 5$.

The following network model (where $|\mathscr{Q}| = 32$ and $|\mathscr{A}| = 49$) is obtained (Fig. 2):

3.2. *A new algorithm for the multiple criteria knapsack problem*

The algorithm described in this subsection (see Fig. 3) is a particular implementation of the labeling algorithms for multiple criteria shortest path problems on acyclic networks. For more details about

```
*Building a network shortest path model.*
{ Converting a knapsack model into a shortest path problem. }
(1)  begin
(2)       Set $\mathcal{Q} \leftarrow \{s\}$, $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
(3)       $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{1^0, 1^{w_1}\}$;
(4)       Set $\mathcal{A} \leftarrow \{(s, 1^0), (s, 1^{w_1})\}$;
(5)       $c(s, 1^0) \leftarrow 0$ and $c(s, 1^{w_1}) \leftarrow -v_1$;
(6)       for $(j = 1)$ to $(n - 1)$ do
(7)       begin
(8)            while $(T \neq \{\})$ do
(9)            begin
(10)                Let $a$ be the first element in $T$ and remove it from $T$;
(11)                if $(a \notin V)$ then
(12)                begin
(13)                     $V \leftarrow V \cup \{a\}$;
(14)                     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j + 1)^a\}$;
(15)                end
(16)                $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{\left(j^a, (j + 1)^a\right)\right\}$;
(17)                $c\left(j^a, (j + 1)^a\right) \leftarrow 0$;
(18)                if $(a + w_{j+1} \leq W)$ then
(19)                begin
(20)                     if $(a + w_{j+1} \notin V)$ then
(21)                     begin
(22)                          $V \leftarrow V \cup \{a + w_{j+1}\}$;
(23)                          $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j + 1)^{a+w_{j+1}}\}$;
(24)                     end
(25)                     $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{\left(j^a, (j + 1)^{a+w_{j+1}}\right)\right\}$;
(26)                     $c\left(j^a, (j + 1)^{a+w_{j+1}}\right) \leftarrow -v_{j+1}$;
(27)                end
(28)           end
(29)           $T \leftarrow V$ and $V \leftarrow \{\}$;
(30)      end
(31)      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{t\}$;
(32)      while $(T \neq \{\})$ do
(33)      begin
(34)           Let $a$ be the first element in $T$ and remove it from $T$;
(35)           $\mathcal{A} \leftarrow \mathcal{A} \cup \{(n^a, t)\}$;
(36)           $c(n^a, t) \leftarrow 0$;
(37)      end
(38) end
```

Fig. 1. Network converting algorithm.

labeling algorithms for general networks see Martins and Santos [11] (http://www.mat.uc.pt/~eqvm/eqvm.html).

The new algorithm proposed here takes into account the following characteristics of the network model which results from the multiple criteria 0–1 knapsack problem:

- This network is also generated, layer by layer; only knowledge of the nodes belonging to layer $j - 1$ is required to construct the set of nodes of layer j and the arcs connecting these two

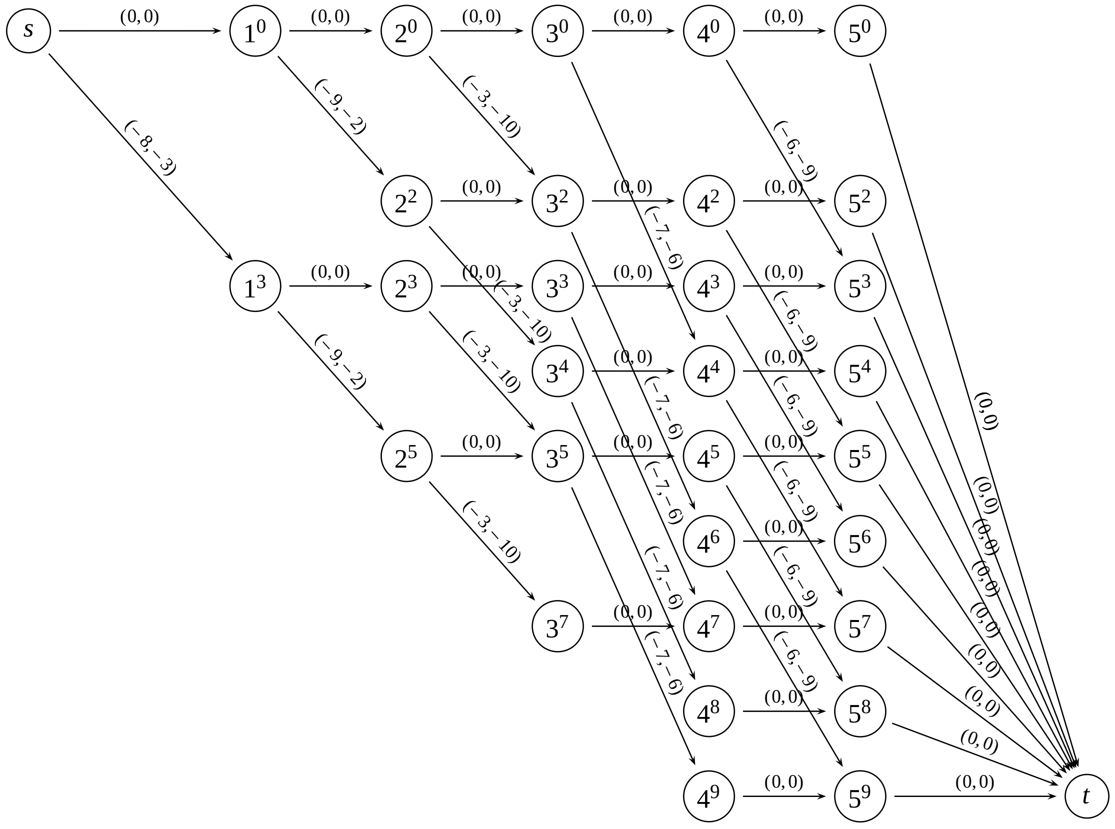

Fig. 2. Network model.

consecutive layers. Thus, all the other nodes and arcs are not taken into account in this procedure. With this technique a lot of memory space is saved which is not the case for general networks.

- Each node j^a, for $a=0,\dots,W$, has at most two incoming arcs $((j-1)^a, j^a)$ and $((j-1)^{a-w_j}, j^a)$. In this way, the labels of j^a can be easily obtained from the labels of nodes $(j-1)^a$ and $(j-1)^{a-w_j}$. So, the whole set of labels of j^a is generated in one iteration. In subsequent iterations it is not necessary to update the set of non-dominated labels for each node as it is usually the case when applying labeling algorithms to general networks.

In our algorithm, as well as in labeling algorithms for multiple criteria shortest path problems, all the labels, for each node, can be lexicographically ordered, but in the particular case of bicriteria problems, the following property exists: the values concerning the first criterion are placed in non-decreasing order, while the values of the second one are placed in non-increasing order. So, to see if a new given label is dominated or not, it is only necessary to compare this new label with the last non-dominated label determined. For more than two criteria the new label must be compared with all the labels already determined. This operation is very time consuming.

The main steps of the algorithm can now be described as follows:

1. Let T and V be two lists of superscripts related to the nodes in layers $j-1$ and j, respectively.

```
Computing efficient paths in the network.
{ A new labeling algorithm to determine efficient paths. }
(1)  begin
(2)      Set $S(1^0) \leftarrow \{(0,\ldots,0)\}$ and $S(1^{w_1}) \leftarrow \{(-v_1^1,\ldots,-v_1^r)\}$;
(3)      Set $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
(4)      for $(j = 2)$ to $n$ do
(5)      begin
(6)          for $(a = 0)$ to $W$ do
(7)          begin
(8)              if $(a \in T)$ then
(9)              begin
(10)                 $V \leftarrow V \cup \{a\}$;
(11)                 if $(a - w_j \in T)$ then { two incoming arcs are defined }
(12)                     $S(j^a) \leftarrow$ Set of ND labels of$\Big(S\big((j-1)^a\big) \cup$
(12a)                    $\{(-v_j^1,\ldots,-v_j^r)\} + S\big((j-1)^{a-w_j}\big)\Big)$;
(13)                 else { only the arc $\big((j-1)^a, j^a\big)$ is defined }
(14)                     $S(j^a) \leftarrow S\big((j-1)^a\big)$;
(15)             end
(16)             else
(17)             begin
(18)                 if $(a - w_j \in T)$ then { only the arc $\big((j-1)^{a-w_j}, j^a\big)$ is
(18a)                defined }
(19)                 begin
(20)                     $S(j^a) \leftarrow \{(-v_j^1,\ldots,-v_j^r)\} + S\big((j-1)^{a-w_j}\big)$;
(21)                     $V \leftarrow V \cup \{a\}$;
(22)                 end
(23)                 { else The node $j^a$ does not exist }
(24)             end
(25)         end
(26)         $T \leftarrow V$ and $V \leftarrow \{\}$;
(27)     end
(28)     $S(t) \leftarrow$ Set of ND labels of $\bigcup_{a=0}^{W} S(n^a)$;
(29) end
```

Fig. 3. The new algorithm.

2. Let $S(j^a)$ be the set of non-dominated (ND) labels concerning the set of all paths from s to j^a.
3. Initialize sets $S(1^0)$ and $S(1^{w_1})$ with the labels $(0,\ldots,0)$ and $(-v_1^1,\ldots,-v_1^r)$, respectively. These two sets contain all the labels of layer 1.
4. Then, for each of the remaining layers j ($j=2,\ldots,n$), build the set $S(j^a)$ in the following manner:
   (a) If j^a has only one incoming arc, then consider two different cases (see Figs. 4a and b):
      (i) If the arc is $((j-1)^a, j^a)$, i.e., a belongs to T but $a - w_j$ does not (see Fig. 4a). Then, $S(j^a)$ is equal to $S((j-1)^a)$ (see the computation of $S(4^2)$ from $S(3^2)$ in Fig. 5).
      (ii) If the arc is $((j-1)^{a-w_j}, j^a)$, i.e., a does not belong to T but $a - w_j$ does (see Fig. 4b). Then, the labels in $S(j^a)$ can be obtained by summing the vector $(-v_j^1,\ldots,-v_j^r)$ to each of the labels of set $S((j-1)^{a-w_j})$ (see the computation of $S(4^6)$ from $S(3^2)$ in Fig. 5).

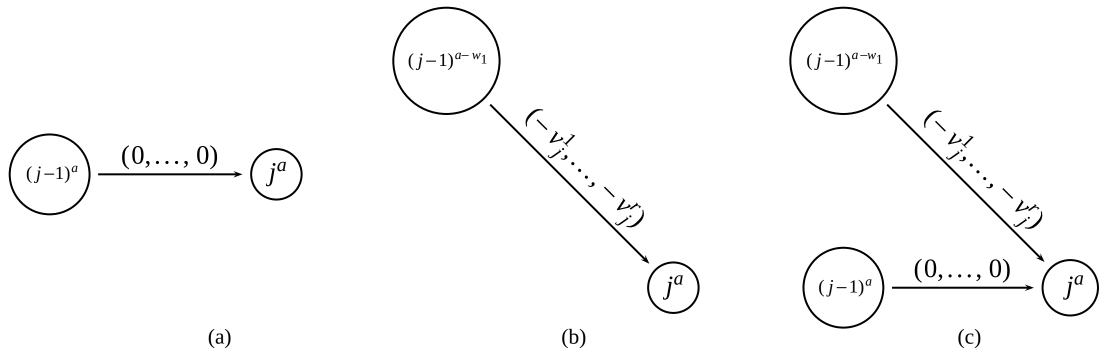

Fig. 4. Different cases of incoming arcs.

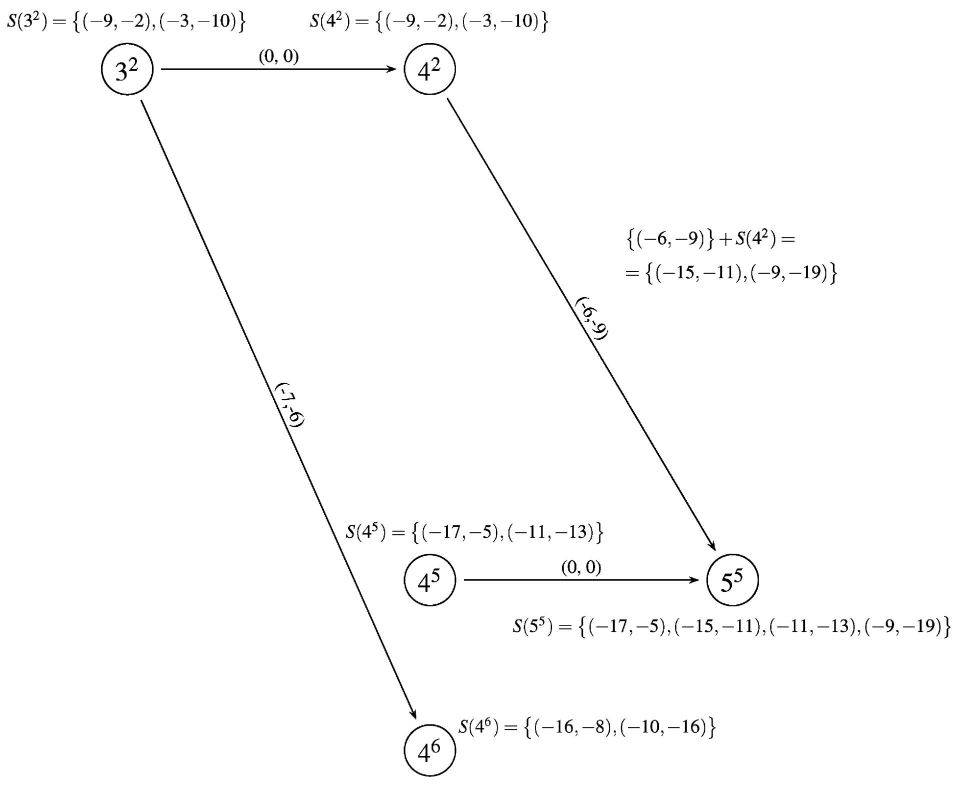
$S(3^2)=\{(-9,-2),(-3,-10)\}$
$S(4^2)=\{(-9,-2),(-3,-10)\}$
3^2
(0, 0)
4^2
$\{(-6,-9)\}+S(4^2)=$
$=\{(-15,-11),(-9,-19)\}$
(-6,-9)
(-7,-6)
$S(4^5)=\{(-17,-5),(-11,-13)\}$
4^5
(0, 0)
5^5
$S(5^5)=\{(-17,-5),(-15,-11),(-11,-13),(-9,-19)\}$
$S(4^6)=\{(-16,-8),(-10,-16)\}$
4^6

Fig. 5. An illustrative example.

(b) If j^a has two incoming arcs, i.e., a and $a-w_j$ belonging to T (see Fig. 4c). Then, build $S(j^a)$ by choosing the ND labels simultaneously from the labels in $S((j-1)^a)$, and from the labels obtained by summing the vector $(-v_j^1,\dots,-v_j^r)$ to each of the labels of set $S((j-1)^{a-w_j})$.

To clarify the last case of this procedure an example will be presented. Fig. 3 shows the algorithm described above.

Now, case (4b) is clarified by using the bicriteria example (see the computation of $S(5^5)$ from $S(4^2)$ and $S(4^5)$ in Fig. 5):

1. In this example $S(4^2)=\{(-9,-2),(-3,-10)\}$ and $S(4^5)=\{(-17,-5),(-11,-13)\}$. The labels of each set are placed in non-decreasing order of the first criterion.
2. Determine the set $S(5^5)$. In this example, $c^1(4^2,5^5)=-6$, $c^2(4^2,5^5)=-9$, and $c^1(4^5,5^5)=c^2(4^5,5^5)=0$.
3. Now determine the labels of $S(5^5)$ in the following way:
   - First, compare the first label of $S(4^5)$ with the label obtained from the first element of $S(4^2)$ plus $(c^1(4^2,5^5),c^2(4^2,5^5))$, i.e., compare $(-17,-5)$ with $(-9,-2)+(-6,-9)$. The lexicographic minimum element between $(-17,-5)$ and $(-15,-11)$ is $(-17,-5)$. So, this is the first label of the set $S(5^5)$. The label $(-17,-5)$ can be removed from $S(4^5)$. The first element of $S(4^5)$ is now $(-11,-13)$.
   - Second, proceed similarly by comparing $(-11,-13)$ with $(-9,-2)+(-6,-9)=(-15,-11)$. The lexicographic minimum is $(-15,-11)$. In order to know if $(-15,-11)$ is dominated or not, compare this label with the last element in $S(5^5)$. Label $(-15,-11)$ is also non-dominated. So, add $(-15,-11)$ at the end of $S(5^5)$. Label $(-9,-2)$ is removed from $S(4^2)$. The first element of $S(4^2)$ is now $(-3,-10)$.
   - Third, compare the first label of $S(4^5)$ with the first label of $S(4^2)$ plus $(c^1(4^2,5^5)$, $c^2(4^2,5^5))$, i.e., $(-11,-13)$ with $(-3,-10)+(-6,-9)$. The lexicographic minimum element between $(-11,-13)$ and $(-9,-19)$ is $(-11,-13)$. When comparing this vector with the last label in the new set $S(5^5)$ it is easy to see that $(-11,-13)$ is also non-dominated. Then, the label $(-11,-13)$ is added to $S(5^5)$ and it is removed from $S(4^5)$, which becomes an empty set.
   - Finally, compare the vector $(-3,-10)+(-6,-9)=(-9,-19)$ with the last label of $S(5^5)$. Label $(-9,-19)$ is also non-dominated. So, $S(5^5)=\{(-17,-5),(-15,-11),(-11,-13),(-9,-19)\}$ is obtained.

The illustrative example has seven non-dominated solutions: $S(t)=\{(-24,-11),(-23,-14),(-22,-17),(-19,-18),(-18,-21),(-17,-22),(-16,-25)\}$. The solutions for the bicriteria knapsack example can be easily rebuilt by using a backtracking technique.

The network is built layer by layer. During the optimization process, ranking algorithms [14–16] cannot be used because they start by computing the shortest tree using the entire network. A previous study had used the ranking shortest paths algorithm of Martins and Santos [16]. The algorithm appears inefficient because it quickly reaches the maximum memory available on a SunSparc Station with 48 Mbytes of RAM. So, it was not possible to solve problems with more that 20 items.

4. Experiments and results

This section deals with the computational behavior of the algorithm on certain sets of randomly generated instances. In this study, we considered only bicriteria 0–1 knapsack instances because multiple criteria instances are much more demanding of memory. The design of the computational experiments was inspired by the framework proposed in Martello and Toth [1] for the classical single criterion 0–1 knapsack problem. Also, for the bicriteria 0–1 knapsack problems, the results depend

Table 1
Average results for uncorrelated instances with: $w_j, v_j^1, v_j^2 \in [1, 1000]$, $n = 50$, and $W = (P/100)\sum_{j=1}^{n} w_j$

P	W	No. of nodes	No. of arcs	Total number of labels created	$\lvert X^e \rvert$	CPU time (s)	Total memory used (Mbytes)
10	2435.15	78,143.70	135,108.80	235,987.25	13.15	0.03	1.55
20	4870.10	173,174.05	323,549.69	846,833.69	24.35	0.08	5.11
30	7305.20	260,064.84	496,824.84	1,702,653.25	35.75	0.15	10.54
40	9740.35	336,412.56	649,260.81	2,648,163.00	43.65	0.18	17.53
50	12,175.15	401,911.31	780,121.12	3,537,664.50	43.60	0.21	25.33
60	14,610.45	456,660.44	889,534.19	4,265,977.50	43.00	0.19	33.17
70	17,045.50	499,687.19	975,507.19	4,770,651.50	36.10	0.22	40.19
80	19,480.70	530,599.94	1,037,266.12	5,053,485.00	25.60	0.35	45.61
90	21,915.75	549,376.31	1,074,775.25	5,170,086.00	12.80	0.27	49.04

on the instance type. For the uncorrelated instances, the values v_j^1, v_j^2 and w_j are uniformly random generated in the range $[1, U]$ (U represents an upper bound for the values v_j^1, v_j^2 and w_j). For each data set the value for W is computed as the nearest integer value of $(P/100)\sum_{j=1}^{n} w_j$ (where P is a percentage of $\sum_{j=1}^{n} w_j$).

Six types of correlated instances were generated as follows:

1. Weakly correlated instances, where w_j is correlated with v_j^1, i.e., $w_j \in [111, 1000]$, and $v_j^1 \in [w_j - 100, w_j + 100]$. The value of v_j^2 is generated in the range $[1, 1000]$.
2. Weakly correlated instances, where w_j is correlated with v_j^2, i.e., $w_j \in [111, 1000]$, and $v_j^2 \in [w_j - 100, w_j + 100]$. The value of v_j^1 is generated in the range $[1, 1000]$.
3. Weakly correlated instances, where v_j^1 is correlated with v_j^2, i.e., $v_j^1 \in [111, 1000]$, and $v_j^2 \in [v_j^1 - 100, v_j^1 + 100]$. The value of w_j is generated in the range $[1, 1000]$.
4. Weakly correlated instances, where v_j^2 is correlated with v_j^1, i.e., $v_j^2 \in [111, 1000]$, and $v_j^1 \in [v_j^2 - 100, v_j^2 + 100]$. The value of w_j is generated in the range $[1, 1000]$.
5. Strongly correlated instances, where w_j is correlated with v_j^1, i.e., $w_j \in [1, 1000]$, and $v_j^1 = w_j + 100$. The value of v_j^2 is generated in the range $[1, 1000]$.
6. Strongly correlated instances, where w_j is correlated with v_j^2, i.e., $w_j \in [1, 1000]$, and $v_j^2 = w_j + 100$. The value of v_j^1 is generated in the range $[1, 1000]$.

For the whole set of instances considered in Tables 2–12, $P = 50$.

The goals of the numerical experiments were as follows:

1. To identify the hardest uncorrelated instances and the most interesting instances in terms of the number of efficient solutions (see Table 1).
2. To understand the size of the network generated by a given knapsack instance.
3. To analyze the performances of the algorithm (in terms of the CPU computing time) when searching for all efficient solutions for all the instances (both uncorrelated and correlated).

4. To analyze the total number of labels used and consequently the memory capacity requirements.
5. To compare our algorithm with the branch-and-bound approach by Visée et al. [6].

In this study, 20 different instances were generated for each problem. Every instance was built with the help of the C version procedure `random.h` of NETGEN [17]. The random seed-numbers 100, 1000, and 2000 were used on a PC to get the values (concerning the first instance) of v_j^1, v_j^2, and w_j, respectively. The remaining instances were generated by successively increasing those seed-numbers by 10. The code was executed on a Pentium II (two processors, 256 MHz, 64 Mbytes of RAM) at the CISUC (University of Coimbra, Portugal). The software can be directly obtained from José Luis Santos (`zeluis@mat.uc.pt`). The knapsack generator is available from José Figueira (`figueira@fe.uc.pt`).

The results shown in Tables 1–8 concern the average obtained from a set of 20 different instances. The memory space available is around 2 Gbytes. The results concerning the comparison with the branch-and-bound approach, presented in Tables 9–12, were obtained in the same computer.

Concerning the experimental computations, some comments can be made.

- Problems generated with a percentage close to 50% of the total sum of w_j are bigger (considering the cardinality of X^e) than problems generated with a percentage far from this central value. Table 1 gives us an idea of this fact. The average was determined from a set of 20 instances. When P (P is a percentage of $\sum_{j=1}^{n} w_j$) is very low, the number of solutions is small. Consequently, the efficient solutions are also in a small number. When P is very high, the number of items belonging to a given solution is also high, i.e., almost all the items belong to every solution. The number of solutions is very large, but efficient solutions are scarce. There are a small number of efficient solutions that dominate all the other solutions.
- Problems generated with a percentage P, close to $80\%, 90\%$ of the total sum of w_j are more difficult to solve (in terms of CPU time and memory capacity requirements) than the remaining problems (see Table 1).
- The storage memory capacity limits the computational experimentation. So, uncorrelated bicriteria knapsack instances with more than 210 items (when the data are generated in the range $[1, 1000]$), and more than 320 (when the data are generated in the range $[1, 300]$) items could not be solved (see Table 2).
- Concerning the uncorrelated instances, we observe that, for a given number of items, the number of efficient solutions is similar when the data are generated either in the range $[1, 300]$ or in the range $[1, 1000]$ (see Table 2).
- An interesting result occurs when uncorrelated instances are compared with weakly correlated instances of Types 1 and 2. Weakly correlated instances are more difficult to solve than uncorrelated instances (cf. Table 2 with Tables 3 and 4). The number of efficient solutions concerning these correlated instances is higher than the number of efficient solutions related to the uncorrelated instances.
- It should be noted that the number of efficient solutions is very low when the values of v_j^1 and v_j^2 are correlated (see Tables 5 and 6).
- Strongly correlated instances are easier to solve than weakly correlated and uncorrelated instances (*cf.* Tables 7 and 8 with Tables 2–6).

Table 2
Average results for uncorrelated instances with: $w_j, v_j^1, v_j^2 \in [1, U]$ and $W = (50/100)\sum_{j=1}^{n} w_j$

U	n	W	No. of nodes	No. of arcs	Total number of labels created	$\|X^e\|$	CPU time (s)	Total memory used (Mbytes)
	10	2363.40	1126.50	1611.35	1175.00	4.00	0.00	0.35
	20	4908.25	37,293.70	66,893.85	77,756.95	9.75	0.04	2.21
	30	7445.60	121,021.60	228,378.95	466,598.75	19.65	0.11	6.49
	40	9769.60	241,556.95	464,428.50	1,461,549.62	30.45	0.19	13.59
	50	12,175.15	401,911.31	780,121.12	3,537,664.50	43.60	0.21	25.33
	60	14,629.80	601,491.38	1,174,242.25	7,195,329.00	60.40	0.34	42.79
	70	17,233.90	844,104.38	1,654,160.38	13,434,125.00	83.60	0.52	68.62
1000	80	19,881.10	1,126,660.25	2,213,888.75	22,916,352.00	101.95	0.90	99.65
	90	22,433.25	1,443,420.38	2,842,250.00	36,360,640.00	126.30	1.05	138.43
	100	24,932.25	1,794,925.25	3,540,228.00	54,559,704.00	147.90	1.50	186.47
	120	30,059.85	2,620,489.50	5,181,052.50	109,194,928.00	200.70	2.57	306.54
	140	34,861.30	3,577,586.00	7,085,609.50	193,496,672.00	246.60	4.11	458.18
	160	39,764.45	4,689,653.00	9,299,918.00	320,856,384.00	317.20	6.38	680.47
	180	44,767.10	5,959,491.00	11,829,573.00	504,200,544.00	403.85	9.44	957.19
	190	47,272.25	6,651,830.00	13,209,230.00	620,352,640.00	449.75	13.57	1,109.84
	200	49,674.45	7,371,979.00	14,644,720.00	753,986,624.00	477.70	13.02	1,274.12
	210	52,083.60	8,129,389.00	16,154,717.00	907,709,440.00	527.50	24.02	1,465.92
	10	755.45	921.10	1,368.15	994.55	3.80	0.00	0.14
	20	1,473.95	14,020.55	25,517.10	33,629.90	10.05	0.01	0.79
	30	2,260.70	40,920.70	77,562.30	181,812.25	20.35	0.04	2.32
	40	3,032.55	79,682.75	153,465.16	559,105.12	31.10	0.07	4.92
	50	3,743.10	128,777.50	250,188.84	1,283,317.25	42.05	0.14	8.71
	60	4,522.85	191,145.05	373,338.41	2,562,555.75	60.85	0.16	14.58
	70	5,271.40	264,406.69	518,350.84	4,626,435.50	84.55	0.21	22.40
	80	6,053.60	350,354.91	688,664.62	7,710,385.50	108.25	0.49	32.83
	90	6,795.75	446,267.41	878,989.19	12,085,538.00	125.90	0.40	45.36
	100	7,577.25	554,811.31	1,094,504.38	18,136,050.00	147.20	0.82	61.26
300	120	9,112.35	805,540.12	1,592,883.38	36,908,744.00	209.75	0.86	104.87
	140	10,628.80	1,100,999.25	2,180,763.00	67,121,896.00	271.40	1.10	159.35
	160	12,074.45	1,435,843.62	2,847,554.00	111,260,800.00	344.65	2.25	231.04
	180	13,602.10	1,823,631.00	3,620,068.75	175,067,696.00	421.85	3.30	317.58
	200	15,116.95	2,255,463.25	4,480,701.00	261,069,984.00	503.70	3.75	426.84
	220	16,651.20	2,734,668.25	5,436,041.00	376,112,160.00	590.25	6.88	563.27
	240	18,230.35	3,266,026.00	6,495,597.50	526,541,632.00	679.50	5.40	723.63
	260	19,700.65	3,829,434.00	7,619,472.00	713,292,608.00	786.15	12.30	908.26
	280	21,209.90	4,443,209.50	8,844,003.00	947,385,664.00	895.50	16.37	1,124.06
	300	22,794.35	5,113,660.00	10,181,734.00	1,240,443,264.00	1,001.20	22.89	1,378.52
	320	24,286.40	5,814,675.50	11,580,778.00	1,374,251,776.00	1,132.85	28.70	1,652.17

Table 3
Average results for weakly correlated instances of Type 1

n	W	No. of nodes	No. of arcs	Total number of labels created	$\lvert X^e\rvert$	CPU time (s)	Total memory used (Mbytes)
10	2836.50	1116.45	1605.15	1183.20	5.30	0.00	0.41
20	5558.15	39,449.15	70,511.15	117,016.00	16.70	0.05	3.42
30	8308.25	129,448.50	244,219.59	884,646.81	33.75	0.10	12.57
40	11,155.90	266,245.56	511,701.69	3,144,457.00	53.80	0.20	29.99
50	13,855.15	443,670.16	860,922.12	7,860,288.00	79.75	0.28	57.96
60	16,627.70	665,527.06	1,298,934.00	16,621,605.00	111.05	0.48	101.07
70	19,401.00	930,063.38	1,822,341.62	31,096,588.00	143.70	0.80	160.48
80	22,156.50	1,236,558.25	2,429,748.25	53,032,964.00	179.15	1.59	229.50
90	24,921.00	1,584,363.75	3,119,738.50	84,201,704.00	221.80	2.20	322.64
100	27,791.45	1,978,891.25	3,902,998.75	127,081,920.00	260.05	3.25	430.77
120	33,341.65	2,886,468.50	5,706,975.50	256,987,216.00	362.75	5.35	731.43
140	38,745.55	3,951,615.25	7,826,393.00	467,984,224.00	480.35	9.74	1,137.81
160	44,204.75	5,183,585.00	10,279,378.00	783,581,824.00	595.05	16.54	1,656.98
170	47,048.05	5,871,005.00	11,648,514.00	993,145,536.00	643.65	23.22	1,977.08

Table 4
Average results for weakly correlated instances of Type 2

n	W	No. of nodes	No. of arcs	Total number of labels created	$\lvert X^e\rvert$	CPU time (s)	Total memory used (Mbytes)
10	2836.50	1116.45	1605.15	1162.15	5.65	0.00	0.40
20	5558.15	39,449.15	70,511.15	109,515.80	14.40	0.03	3.20
30	8308.25	129,448.50	244,219.59	785,160.62	28.70	0.18	11.04
40	11,155.90	266,245.56	511,701.69	2,818,977.50	47.90	0.19	27.94
50	13,855.15	443,670.16	860,922.12	7,140,872.00	70.95	0.39	52.64
60	16,627.70	665,527.06	1,298,934.00	14,996,848.00	97.80	0.52	89.12
70	19,401.00	930,063.38	1,822,341.62	27,631,044.00	127.40	0.89	135.99
80	22,156.50	1,236,558.38	2,429,748.25	46,638,620.00	162.80	1.52	200.85
90	24,921.00	1,584,363.75	3,119,738.50	74,155,824.00	200.85	2.10	286.68
100	27,791.45	1,978,891.62	3,902,998.75	113,421,400.00	246.70	2.79	404.64
120	33,341.65	2,886,468.50	5,706,975.50	234,624,976.00	344.00	5.42	680.58
140	38,745.55	3,951,615.25	7,826,393.00	429,999,712.00	451.70	9.25	1,050.89
160	44,204.75	5,183,585.00	10,279,378.00	724,166,592.00	572.65	16.82	1,531.44
170	47,048.05	5,871,005.00	11,648,514.00	918,991,872.00	634.05	29.00	1,826.98

- Unexpected results occur when strongly correlated instances are solved. Strongly correlated instances of Type 5 (see Table 7) are easier to solve than strongly correlated instances of Type 6 (see Table 8). In fact, the strongly correlated instance 2 of Type 6 with 600 items is a pathological case. It demands excessive time and memory. The number of labels estimated at the last iteration

Table 5
Average results for weakly correlated instances of Type 3

n	W	No. of nodes	No. of arcs	Total number of labels created	$\|X^e\|$	CPU time (s)	Total memory used (Mbytes)
10	2400.45	1129.80	1616.85	1131.95	1.15	0.00	0.35
20	4856.45	36,933.95	66,179.20	41,247.00	1.40	0.03	1.21
30	7445.70	121,022.95	228,381.66	148,925.09	1.95	0.05	2.11
40	9770.05	241,565.34	464,445.25	325,160.66	1.75	0.12	3.02
50	12,175.60	401,922.41	780,143.31	587,161.31	2.50	0.08	4.08
60	14,629.80	601,491.38	1,174,242.25	951,950.81	3.45	0.11	5.34
70	17,233.90	844,104.38	1,654,160.38	1,448,895.62	3.75	0.13	6.81
80	19,881.10	1,126,660.25	2,213,888.75	2,091,244.00	3.75	0.30	8.40
90	22,433.25	1,443,420.38	2,842,250.00	2,885,715.75	4.65	0.15	10.27
100	24,932.25	1,794,925.25	3,540,228.00	3,858,098.00	5.55	0.18	12.19
150	37,265.00	4,111,134.75	8,147,887.00	12,491,885.00	9.60	0.39	26.51
200	49,674.45	7,371,979.00	14,644,720.00	30,639,232.00	13.60	0.66	49.11
250	61,769.50	11,533,629.00	22,943,804.00	62,602,060.00	17.85	1.42	79.01
300	74,109.35	16,645,086.00	33,142,026.00	113,651,760.00	24.35	2.28	118.74
350	86,409.65	22,674,196.00	45,175,632.00	190,489,072.00	29.50	3.55	172.43
400	98,482.25	29,584,598.00	58,972,276.00	300,213,792.00	34.20	3.94	240.65
450	110,663.75	37,423,768.00	74,626,256.00	451,934,016.00	44.25	5.64	319.56
500	122,737.65	46,151,184.00	92,056,928.00	652,767,936.00	49.50	8.26	419.88
600	147,143.95	66,402,648.00	132,511,024.00	1,241,959,040.00	67.70	16.12	658.84
700	171,388.16	90,273,920.00	180,205,088.00	>2,147,483,647.00	84.65	27.21	1,007.18
800	195,507.91	117,724,592.00	235,058,208.00	>2,147,483,647.00	108.85	43.31	1,435.93

is around 4,000,000,000 while the average number of labels for the remaining 19 instances is 1,609,998,720 (see Table 8).

- The graphs concerning the results provided in Tables 1–8 are similar to the one presented in Fig. 6, which is related to the data in the second part of Table 2. In general, the number of nodes, the number of arcs, the number of efficient solutions, the total number of labels created, the memory requirements and the CPU time grow with the number of the items. However, the number of nodes, the number of arcs and the number of efficient solutions grow more quickly than the total number of labels created, the memory requirements and the CPU time. It should be noted that the memory capacity used depends on the number of labels in two consecutive layers. This fact explains why problems of larger dimension cannot be solved.
- The code is very fast for the instances being considered here even when huge networks result. See, for example, Table 7 for $n = 900$. In such a case, the number of nodes is greater than $149,000,000$ and the number of arcs is greater than $297,000,000$. The CPU time is around 70 s.
- The results of a comparative study between our labeling approach (LA) and the branch-and-bound approach (BB) by Visée et al. [6] are presented in Tables 9–12. Because the method of Visée et al. [6] is a two phase based procedure, we add to each table two different columns, one of them containing the number of supported efficient solutions (computed in the first phase), $|X^{es}|$, and

Table 6
Average results for weakly correlated instances of Type 4

n	W	No. of nodes	No. of arcs	Total number of labels created	$\|X^e\|$	CPU time (s)	Total memory used (Mbytes)
10	2400.45	1129.80	1616.85	1132.70	1.15	0.00	0.35
20	4856.45	36,933.95	66,179.20	40,804.40	1.30	0.03	1.20
30	7445.70	121,022.95	228,381.66	147,886.16	2.25	0.06	2.09
40	9770.05	241,565.34	464,445.25	322,582.59	2.40	0.11	3.01
50	12,175.60	401,922.41	780,143.31	582,351.88	2.25	0.13	4.08
60	14,629.80	601,491.38	1,174,242.25	939,805.00	2.60	0.14	5.22
70	17,233.90	844,104.38	1,654,160.50	1,421,028.12	2.70	0.14	6.63
80	19,881.10	1,126,660.25	2,213,888.75	2,043,228.38	3.80	0.15	8.27
90	22,433.25	1,443,420.38	2,842,250.00	2,831,194.50	4.75	0.13	10.17
100	24,932.25	1,794,925.25	3,540,228.00	3,792,381.25	5.70	0.20	12.12
150	37,265.00	4,111,134.75	8,147,887.00	12,197,453.00	7.50	0.37	25.33
200	49,674.45	7,371,979.00	14,644,720.00	29,241,920.00	11.05	0.79	45.75
250	61,769.50	11,533,629.00	22,943,804.00	59,028,648.00	15.50	1.33	74.49
300	74,109.35	16,645,086.00	33,142,026.00	107,622,312.00	23.25	2.12	115.34
350	86,409.65	22,674,196.00	45,175,632.00	182,277,600.00	27.95	2.97	168.89
400	98,482.25	29,584,602.00	58,972,276.00	290,068,192.00	33.05	3.94	232.74
450	110,663.75	37,423,768.00	74,626,256.00	438,990,752.00	39.00	5.77	315.63
500	122,737.65	46,151,184.00	92,056,928.00	638,551,936.00	45.30	8.27	417.92
600	147,143.95	66,402,648.00	132,511,024.00	1,232,632,576.00	65.40	26.83	678.90
700	171,388.16	90,273,920.00	180,205,088.00	>2,147,483,647.00	79.95	26.44	1028.89
800	195,507.91	117,724,592.00	235,058,208.00	>2,147,483,647.00	100.35	44.92	1496.91

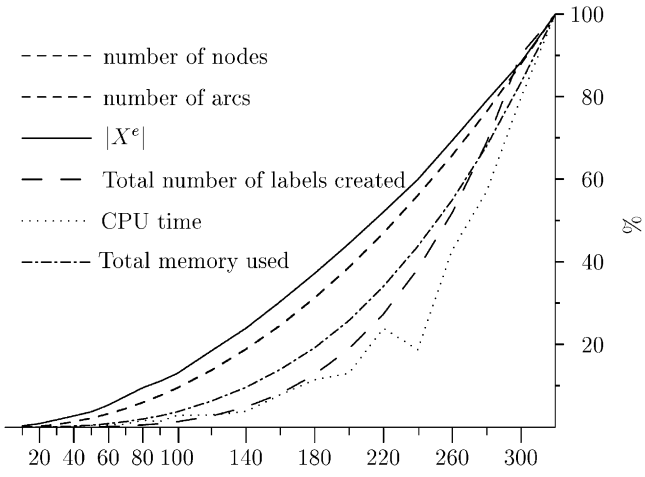

Fig. 6. Results for the uncorrelated instances ($U = 300$).

Table 7
Average results for strongly correlated instances of Type 5

n	W	No. of nodes	No. of arcs	Total number of labels created	$\|X^{e}\|$	CPU time (s)	Total memory used (Mbytes)
10	2400.45	1129.80	1616.85	1149.25	3.35	0.00	0.35
20	4856.45	36,933.95	66,179.20	48,615.20	5.45	0.03	1.40
30	7445.70	121,022.95	228,381.66	205,729.30	9.45	0.08	2.80
40	9770.05	241,565.34	464,445.25	497,843.75	11.65	0.12	4.43
50	12,175.60	401,922.41	780,143.31	969,781.88	13.80	0.12	6.36
60	14,629.80	601,491.38	1,174,242.25	1,649,708.38	16.10	0.18	8.69
70	17,233.90	844,104.38	1,654,160.50	2,624,917.75	19.55	0.20	11.79
80	19,881.10	1,126,660.25	2,213,888.75	3,922,242.75	21.90	0.18	15.13
90	22,433.25	1,443,420.38	2,842,250.00	5,566,755.50	25.35	0.18	18.76
100	24,932.25	1,794,925.25	3,540,228.00	7,591,037.50	27.70	0.30	22.81
150	37,265.00	4,111,135.25	8,147,888.00	24,745,222.00	37.40	0.72	47.72
200	49,674.45	7,371,979.00	14,644,720.00	57,953,332.00	50.50	1.06	82.22
250	61,769.50	11,533,629.00	22,943,804.00	112,346,176.00	61.20	1.45	126.30
300	74,109.35	16,645,086.00	33,142,026.00	193,848,096.00	68.75	2.74	179.54
350	86,409.65	22,674,196.00	45,175,632.00	306,942,592.00	78.30	4.87	240.93
400	98,482.25	29,584,598.00	58,972,276.00	455,952,192.00	85.05	5.97	312.61
450	110,663.75	37,423,768.00	74,626,256.00	648,949,568.00	95.00	8.50	397.16
500	122,737.65	46,151,184.00	92,056,928.00	891,502,592.00	105.80	11.83	494.80
600	147,143.95	66,402,648.00	132,511,024.00	1,549,186,304.00	119.40	20.90	710.68
700	171,388.16	90,273,920.00	180,205,088.00	>2,147,483,647.00	134.20	34.01	972.93
800	195,507.91	117,724,592.00	235,058,208.00	>2,147,483,647.00	148.20	50.47	1260.56
900	220,187.80	149,068,448.00	297,696,576.00	>2,147,483,647.00	159.85	71.53	1610.71

the other concerning the number of unsupported efficient solutions (computed during the second phase), $|X^{ue}|$. Some observations concerning the comparative results follow.

- In Table 9, we present the comparative results for a set of instances with 50 items and different values for the parameter P. We may observe that the BB approach is competitive (in terms of CPU time) with the LA algorithm when P is greater than 60, which means that we are solving uncorrelated instances with a small number of non-dominated solutions. For the same instances, LA is slower than BB because the networks built are huge, requiring excessive computations in the intermediate layers. It should be remarked that this situation is the only one where BB is competitive with LA.
- In Table 10, the comparative study was done for uncorrelated instances taking into account two different values for parameter U. We can conclude that LA is always faster than BB. The computations were stopped for $n = 200$ when $U = 1000$, and for $n = 140$ when $U = 300$. The only reason for this decision was related to the fact that our approach is always faster than BB so we did not feel that it was necessary to do more computations for the same type of problems. It should be noted that the CPU time for the BB algorithm it is not significantly changed with different U values, while the CPU time for the LA algorithm depends on the value of U.

Table 8
Average results for strongly correlated instances of Type 6

n	W	No. of nodes	No. of arcs	Total number of labels created	$\|X^e\|$	CPU time (s)	Total memory used (Mbytes)
10	2400.45	1129.80	1616.85	1146.45	3.75	0.00	0.35
20	4856.45	36,933.95	66,179.20	47,687.00	5.50	0.01	1.37
30	7445.70	121,022.95	228,381.66	198,618.16	7.65	0.01	2.71
40	9770.05	241,565.34	464,445.25	480,546.25	11.50	0.02	4.35
50	12,175.60	401,922.41	780,143.31	946,441.62	12.85	0.05	6.38
60	14,629.80	601,491.38	1,174,242.25	1,632,511.38	17.20	0.07	8.74
70	17,233.90	844,104.38	1,654,160.50	2,587,452.25	20.15	0.15	11.39
80	19,881.10	1,126,660.25	2,213,888.75	3,843,505.25	21.45	0.25	14.52
90	22,433.25	1,443,420.38	2,842,249.75	5,422,356.00	24.10	0.18	18.02
100	24,932.25	1,794,925.25	3,540,228.00	7,374,461.50	26.00	0.23	21.84
150	37,265.00	4,111,134.75	8,147,887.00	24,208,830.00	36.70	0.67	46.94
200	49,674.45	7,371,979.00	14,644,720.00	57,523,532.00	50.75	0.87	83.64
250	61,769.50	11,533,629.00	22,943,804.00	112,958,000.00	59.10	2.34	130.26
300	74,109.35	16,645,086.00	33,142,026.00	196,828,176.00	70.30	2.89	186.90
350	86,409.65	22,674,196.00	45,175,632.00	315,192,384.00	80.30	4.28	260.51
400	98,482.25	29,584,598.00	58,972,276.00	477,039,008.00	91.95	7.11	351.62
450	110,663.75	37,423,768.00	74,626,256.00	697,288,448.00	105.60	7.26	493.19
500	122,737.65	46,235,544.00	92,224,992.00	923,362,880.00	104.42	13.65	512.99
600[a]	147,143.95	66,552,872.00	132,810,600.00	1,609,998,720.00	120.74	25.26	746.21

[a]Means that only 19 instances were solved.

Table 9
Comparative results for uncorrelated instances (1)

P	$\|X^{es}\|$	$\|X^{eu}\|$	$\|X^e\|$	BB CPU time (s)	LA CPU time (s)
10	5.55	7.60	13.15	0.05	0.03
20	8.05	16.30	24.35	0.14	0.08
30	9.05	26.70	35.75	0.23	0.15
40	10.20	33.45	43.65	0.33	0.18
50	10.00	33.70	43.70	0.33	0.21
60	10.55	32.45	43.00	0.29	0.19
70	9.65	26.45	36.10	0.21	0.22
80	9.10	16.50	25.60	0.12	0.35
90	6.05	6.75	12.80	0.04	0.27

- In Table 11, we show the results concerning weakly correlated instances. The conclusions are similar to the comments pointed out for results in Table 10.
- In Table 12, we present the results for strongly correlated instances. In this particular case, our algorithm is not only much faster than the BB approach, but it can solve large instances which

Table 10
Comparative results for uncorrelated instances (2)

U	n	$\lvert X^{es}\rvert$	$\lvert X^{eu}\rvert$	$\lvert X^{e}\rvert$	BB CPU time (s)	LA CPU time (s)
	10	2.75	1.25	4.00	0.01	0.00
	20	4.60	5.15	9.75	0.01	0.04
	40	9.10	21.35	30.45	0.17	0.19
	60	11.65	48.75	60.40	0.61	0.34
	80	16.20	85.75	101.95	1.76	0.09
	100	18.75	129.15	147.90	3.78	1.50
1000	120	22.60	178.10	200.70	7.41	2.57
	140	24.70	221.90	246.60	12.08	4.11
	160	28.50	288.70	317.20	19.80	6.38
	180	31.55	372.30	403.85	28.91	9.44
	200	34.35	443.35	477.70	42.91	13.02
	10	2.60	1.20	3.80	0.00	0.00
	20	4.75	5.30	10.05	0.02	0.01
	30	7.00	13.35	20.35	0.07	0.04
	40	8.20	22.90	31.10	0.17	0.07
	50	10.95	31.10	42.05	0.35	0.14
300	60	12.50	48.35	60.85	0.66	0.16
	70	14.90	69.65	84.55	1.41	0.21
	80	16.30	91.95	108.25	1.93	0.49
	100	19.65	127.55	147.20	4.07	0.82
	120	22.75	187.00	209.75	8.79	0.86
	140	27.50	243.90	271.40	13.99	1.10

cannot be solved with the BB algorithm. The method of Visée et al. [6] is not able to solve all the instances with more than 50 items, while our algorithm can be used to solve problems with more than 600 items. For the BB approach, available memory is a significant limitation.

5. Conclusions

The results presented in this paper are only preliminary. Future research is needed. There are certain algorithmic improvements and many computational experiments that can be undertaken. In particular:

1. Perform some experiments for more than two criteria for instances with a smaller number of items.
2. Analyze the CPU time with a more advanced version of ranking algorithms [18,19] and or/other ranking paths algorithms.
3. Apply this approach to multiple criteria interactive procedures, where a specific type of local search is needed.
4. Test the model on real-world applications.

Table 11
Comparative results for weakly correlated instances

Type	n	$\|X^{es}\|$	$\|X^{eu}\|$	$\|X^{e}\|$	BB CPU time (s)	LA CPU time (s)
1	20	6.10	10.60	16.70	0.09	0.05
	40	10.20	43.60	53.80	1.11	0.20
	60	15.30	95.75	111.05	5.06	0.48
	80	18.85	160.30	179.15	14.63	1.59
2	20	5.25	9.15	14.40	0.06	0.03
	40	9.75	38.15	47.90	0.79	0.19
	60	13.90	83.90	97.80	3.98	0.52
	80	17.95	144.85	162.80	11.72	1.52
	100	23.05	223.65	246.70	29.70	2.79
3	100	3.40	2.15	5.55	0.07	0.18
	200	5.40	8.20	13.60	0.63	0.66
	300	8.00	16.35	24.35	2.64	2.28
	400	9.05	25.15	34.20	6.63	3.94
	500	11.85	37.65	49.50	15.56	8.26
	600	13.35	54.35	67.70	35.85	16.12
	700	14.60	70.05	84.65	59.68	27.21
4	100	3.30	2.40	5.70	0.09	0.20
	200	4.95	6.10	11.05	0.47	0.79
	300	7.50	15.75	23.35	2.58	2.12
	400	8.90	24.15	33.05	7.08	3.94
	500	10.05	35.25	45.30	12.75	8.27
	600	12.50	52.90	65.40	30.60	26.83
	700	14.45	65.50	79.95	51.49	26.44

Table 12
Comparative results for strongly correlated instances

Type	n	$\|X^{es}\|$	$\|X^{eu}\|$	$\|X^{e}\|$	BB CPU time (s)	LA CPU time (s)
1	10	2.60	0.75	3.35	0.00	0.00
	20	3.90	1.55	5.45	0.04	0.03
	30	5.15	4.30	9.45	0.21	0.08
	40	5.70	5.95	11.65	0.63	0.12
	50	6.00	7.80	13.80	3.47	0.12
	60*	6.21	9.74	15.95	6.54	0.18
2	10	2.90	0.85	3.75	0.01	0.00
	20	3.65	1.85	5.50	0.03	0.01
	30	3.90	3.75	7.65	0.15	0.01
	40	5.30	6.20	11.50	0.63	0.02
	50	5.45	7.40	12.85	3.67	0.05
	60*	7.21	10.16	17.37	13.19	0.07

The main contribution of this article has been to show the efficiency of the labeling algorithm for both uncorrelated and (weakly and strongly) correlated bicriteria 0–1 knapsack instances considered in this work.

Acknowledgements

The authors are thankful to Jacques Teghem and Daniel Tuyttens for providing a copy of their branch-and-bound software, and to an anonymous reviewer who carefully read early versions of this paper and made many corrections, suggestions and recommendations. This research was partially supported by the MONET research grant POCTI/GES/37707/2001 and CISUC "Centro de Informática e Sistemas da Universidade de Coimbra".

References

[1] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. Chichester: Wiley, 1990.
[2] Rosenblatt MJ, Sinuany-Stern Z. Generating the discrete efficient frontier to the capital budgeting problem. Operations Research 1989;37(3):384–94.
[3] Kwak W, Shi Y, Lee H, Lee CF. Capital budgeting with multiple criteria and multiple decision makers. Review of Finance and Accounting 1996;7:97–112.
[4] Teng J-Y, Tzeng G-H. A multiobjective programming approach for selecting non-independent transportation investment alternatives. Transportation Research-B 1996;30:291–307.
[5] Kostreva M, Ogryczak W, Tonkyn DW. Relocation problems arising in conservation biology. Computers and Mathematics with Applications 1999;37:135–50.
[6] Visée M, Teghem J, Pirlot M, Ulungu EL. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. Journal of the Global Optimization 1998;12:139–55.
[7] Klamroth K, Wiecek M. Dynamic programming approaches to the multiple criteria knapsack problems. Naval Research Logistics Quarterly 2000;47(1):57–76.
[8] Hansen M. Solving multiobjective knapsack problems using MOTS. Conference Paper presented at MIC'97, Sophia Antipolis, France, July, 21–4, 1997. 9pp.
[9] Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D. Mosa method: a tool for solving multi-objective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 1999;8:221–36.
[10] Ben Abdelaziz F, Krichen S, Chaouadi J. A hybrid heuristic for multiobjective knapsack problems. In: Voss S, Martello S, Osman I, Roucairol C, editors. Meta-heuristics: advances and trends in local search paradigms for optimization. Dordrecht: Kluwer Academic Publishers, 1999. p. 205–12.
[11] Martins E, Santos JL. The labelling algorithm for the multiobjective shortest path problem. Technical Report 99/005 CISUC, Departamento de Matemática, Universidade de Coimbra, Portugal, 1999. 24pp. (http://www.mat.uc.pt/∼eqvm/eqvm.html).
[12] Ahuja RK, Magnanti TM, Orlin JB. Network flows: theory, algorithms, and applications. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[13] Hansen P. Bicriterion path problems. In: Fandel G, Gal T, editors. Multiple criteria decision making: theory and applications. Lecture Notes in Economics and Mathematical Systems, vol. 177. Heidelberg: Springer, 1980. p. 109–27.
[14] Clímaco J, Martins E. A bicriterion shortest path algorithm. European Journal of Operational Research 1982;11: 399–404.
[15] Martins E. On a multicriteria shortest path problem. European Journal of Operational Research 1984;16:236–45.
[16] Martins E, Santos JL. A new shortest path ranking algorithm. Investigação Operacional 2000;20:47–61.

[17] Klingman D, Napier A, Stutz J. Netgen: a program for generating large scale capacity assignment, transportation, and minimum cost flow problems. Management Science 1974;20(5):814–21.
[18] Martins E, Pascoal M, Santos JL. Deviation algorithms for ranking shortest paths. International Journal of Fundations of Computer Science 1999;10(3):247–61.
[19] Martins E, Pascoal M, Santos JL. A new improvement for a *K* Shortest paths algorithm. Investigação Operacional 2001;21(1):47–60.