CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS ESA 7024

SOLVING MULTIPLE CRITERIA {0, 1}-KNAPSACK PROBLEMS USING A LABELING ALGORITHM

CAHIER N° 181
mai 2001

M. Eugénia Captivo [1]
João Clímaco [2 3]
José Figueira [1 3 4]
Ernesto Martins [5], José Luis Santos [5 6]

received: May 2000.

[1] *DEIO-CIO, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, Bloco C2, 1749-016 Lisboa, Portugal (maria.captivo@fc.ul.pt).*

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

[3] *INESC-Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal (jclimaco@inescc.pt).*

[4] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France (figueira@lamsade.dauphine.fr).*

[5] *Departamento de Matemámatica, Universidade de Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal ({eqvm,zeluis}@mat.uc.pt).*

[6] *CISUC, Departamento de Matemática, Universidade de Coimbra.*

La mort a surpris brutalement notre collègue Ernesto Martins alors que nous terminions le présent travail. Homme de convictions autant en tant que citoyen et comme chercheur, malheureusement pas toujours très bien compris, il a marqué profondément tous ceux qui ont partagé quelques moments avec lui. La mémoire de Monsieur le Professeur Ernesto Martins restera vivante auprès de ses amis.

Contents

LA RÉSOLUTION DES PROBLÈMES DU SAC-À-DOS MULTICRITÈRE PAR UN ALGORITHME DE MARQUAGE

RÉSUMÉ. Dans cet article, on examine les performances d'un nouvel algorithme de marquage conçu pour effectuer une recherche exhaustive de toutes les solutions efficaces (ou vecteurs non-dominés) d'un problème du sac-à-dos multicritère. Cette recherche est effectuée après avoir transformé le problème du sac-à-dos en un problème du plus court chemin dans un réseaux sans cycle. On présente également quelques expériences effectuées avec l'algorithme et les résultats obtenus. L'algorithme s'est révélé très efficace pour les instances du problème bi-critère en variables $\{0,1\}$ considérées dans cet article.

Mots-clés: Problème du sac-à-dos multicritère, Plus court chemin multicritère, Algorithmes de marquage.

SOLVING MULTIPLE CRITERIA $\{0,1\}$-KNAPSACK PROBLEMS USING A LABELING ALGORITHM

ABSTRACT. This paper examines the performances of a new labeling algorithm to find all the efficient solutions (or non-dominated vectors) of the multiple criteria $\{0,1\}$-knapsack problem after converting it into a multiple criteria shortest path problem over acyclic networks. The experiments and results are also presented. The algorithm is very efficient for the hard bi-criteria $\{0,1\}$ knapsack instances considered in the paper.

Keywords: Multiple criteria knapsack problem, Multiple criteria shortest path problem, Labeling algorithms.

1 Introduction

Most of the existing books and papers on knapsack problems deal with a single criterion function (see for example the book by *Martello* and *Toth* [1990]). The classical knapsack problem *seeks to select, from a finite set of items that subset which maximizes a linear function of the items chosen, while a given single inequality constraint must be satisfied.* This model describes real-life applications: capital budgeting problems, cutting stock problems, loading problems and project selection problems, can be found as a sub-problem 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" translated into several criteria. So a multiple criteria model seems to be more realistic. To mitigate the drawbacks of the single criterion model its multiple criteria version is introduced. More precisely the paper introduces a new algorithm for the general multiple criteria $\{0,1\}$-knapsack problem and presents results, in terms of CPU time and total memory used, for a large set of bi-criteria instances. There have been relatively few studies dealing with this new model (*Rosenblatt* and *Sinuany-Stern* [1989], *Kwak et al.* [1996], *Teng* and *Tzeng* [1996]). Knapsack models have also been used in modeling multiple criteria combinatorial optimization problems in the field of conservation biology (see *Kostreva et al.* [1999]).

Recently, several approaches have been proposed to solve multiple criteria knapsack problems: branch-and-bound procedures by *Ulungu* and *Teghem* [1997] and *Visée et al.* [1998], dynamic programming by *Klamroth* and *Wiecek* [2000], tabu search by *Hansen* [1997], simulated annealing by *Ulungu et al.* [1999] and hybrid meta-heuristics (tabu search and genetic algorithms) by *Ben Abdelaziz et al.* [1998].

In this paper a different approach is used which takes advantage of the network structure of the problem, after converting the multiple criteria knapsack model into a multiple criteria shortest path model. The algorithm is inspired on the multiple criteria labeling shortest path algorithms framework for acyclic networks (*Martins* and *Santos* [1999]). But, the new algorithm is much more efficient because the characteristics inherent to the underlying network are taken into account. To our knowledge this algorithm is the first one to be designed and implemented to solve multiple criteria knapsack problems as a multiple criteria shortest path model. However, the computational results presented in this paper concerns only the bi-criteria problem. In the experiments done we had some difficulties to cope with the insufficiency of memory requirements which do not allow us to present results for more than two criteria. We are working on the implementation of an algorithm for more than two criteria and we hope to obtain new results soon.

Two more approaches to obtain the set of all non-dominated solutions are known in the literature. The branch-and-bound approach by *Ulungu* and *Teghem* [1997] and *Visée et al.* [1998], which was designed to take into account only two criteria, and the dynamic programming based approach by *Klamroth* and *Wiecek* [2000], 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 it is only a theoretical mathematical work and no results are presented.

The meta-heuristic based approaches by *Hansen* [1997], *Ulungu et al.* [1999] and *Ben Abdelaziz et al.* [1998], 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 it is 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. Finally, in Section 5, some possible improvements are advanced as well as some suggestions for further research.

2 Definitions

Notation concerning the knapsack problem includes:

- l is the number of items;
- r is the number of criteria;
- v_j^k is the k^{th} value (or the k^{th} profit) of the j^{th} item, for $k = 1,\ldots,r$ (v_j will be used when $r = 1$);
- w_j is the weight of the j^{th} 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:

$$\begin{array}{lcl} \max\ f_1(x) & = & \sum_{j=1}^{l} v_j^1 y_j \\ \vdots & & \\ \max\ f_r(x) & = & \sum_{j=1}^{l} v_j^r y_j \\ & & \\ \text{subject to:} & & \sum_{j=1}^{l} w_j y_j \leq W \\ & & y_j \in \{0,1\},\ j = 1,\ldots,l \end{array} \tag{1}$$

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

To better understand the approach some additional notation and definitions from graph theory (*Ahuja et al.* [1993]), linear (Boolean) programming and multiple criteria combinatorial optimization are necessary.

Let $\mathcal{G} = (\mathcal{Q},\mathcal{A})$ be a directed and connected graph, where $\mathcal{Q}$ is the set of vertices with $|\mathcal{Q}| = n$ and $\mathcal{A} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of arcs with $|\mathcal{A}| = m$. It is denoted by (i,j) the arc linking the vertices i and j, and by $c^1(i,j),\ldots,c^r(i,j)$, the values concerning the r criteria associated to the arc (i,j). A path p from a starting point s to an end point t in $\mathcal{G}$ is a sequence of arcs and vertices from s to t, where the tail vertex of a given arc coincides with the head vertex of the next arc in the path. Let $f_k(p)$ denote the value of a path p regarding criterion k, for all $k = 1,\ldots,r$, where k is a minimization criterion. A path p^e is said efficient if and only if, there does not exist any path p in $\mathcal{G}$ such that $f_k(p) \leq f_k(p^e)$, for all $k = 1,\ldots,r$, with at least one strict inequality. The multiple criteria path problem consists of compute the set of all efficient paths from a starting point s to an end point t in $\mathcal{G}$. Let P^e denote the set of all efficient paths from s to t in $\mathcal{G}$.

3 Outline of the method

In order 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, being then 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 *hard* to solve as well as multiple criteria shortest path problems (*Hansen* [1980] and *Martins* and *Santos* [1999]). The major advantage of our approach is to compute non-dominated solutions for multiple criteria knapsack problems faster than other approaches.

3.1 Formulating the knapsack model as a shortest path problem

This sub-section shows how to formulate the single knapsack problem as a shortest path model (see *Ahuja et al.*, 1993). However, in a first stage, it is necessary to convert the knapsack model into a longest path problem. Then, in a second stage, this model is easily transformed into a shortest path problem by setting the arc "costs" equal to the negative of the arc values or profits. 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 $\mathcal{G}$. Both, the path in $\mathcal{G}$ and the solution of the knapsack have the same (but symmetric) value or profit. In a mono-criterion framework, the problem is no more than to search all the optimal solutions. The basic ideas of this technique can be stated as follows:

1. Determine the set of vertices by using a layer technique.

   Each intermediate layer has several vertices. The first layer (*layer 0*) includes the single vertex s. Then, layer j can be directly obtained from layer $j-1$, for all $j = 1,\dots,l$ where each layer, from $j = 1$ to l, has at most $W+1$ vertices, $j^0,\dots,j^a,\dots,j^W$. Finally, the last layer (*layer l+1*) concerns the single vertex t. So, $|Q| \leq (W+1)l+2$.

2. Define the set of arcs and the arc costs.

   The vertex 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 not to include item 1 in the knapsack, and the second arc represents the decision to include it. Concerning the layers from 1 to $l-1$, each vertex j^a, for $a = 0,\dots,W$, has at the most two outgoing arcs:

   - The arc $\left(j^a,(j+1)^a\right)$ with $c\left(j^a,(j+1)^a\right) = 0$, meaning that item $(j+1)$ is not included in the knapsack.
   - The arc $\left(j^a,(j+1)^{a+w_{j+1}}\right)$ with $c\left(j^a,(j+1)^{a+w_{j+1}}\right) = -v_{j+1}$, if $a+w_{j+1} \leq W$, meaning that item $(j+1)$ is included in the knapsack, if and only if it has sufficient capacity to contain this item.

   Finally, all vertices belonging to *layer l* are connected to the vertex t. Thus, there are at most $W+1$ dummy arcs with cost $c(l^a,t) = 0$, for all $a = 0,\dots,W$.

In order to built efficiently the network we proceed as follows:

1. Let T and V be two lists of superscripts related to the vertices in layers j and $j+1$, respectively.
2. In the first layer, define the vertex 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 l*. At each iteration, list T contains all superscripts related to the vertices in *layer j* and successively a new list, V, is built with the superscripts of all vertices in *layer j+1*. The set of the arcs between the vertices with the superscripts in T and the vertices 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 $l-1$.
4. Finally, dummy arcs are added to the network by connecting all the vertices with superscripts belonging to T with the last vertex, t.

Building a network shortest path model.

{ Converting a knapsack model into a shortest path problem. }

```
begin
    Set $\mathcal{Q} \leftarrow \{s\}$, $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{1^0, 1^{w_1}\}$;
    Set $\mathcal{A} \leftarrow \{(s, 1^0), (s, 1^{w_1})\}$;
    $c(s, 1^0) \leftarrow 0$ and $c(s, 1^{w_1}) \leftarrow -v_1$;
    for ($j = 1$) to ($l - 1$) do
    begin
        while ($T \neq \{\}$) do
        begin
            Let $a$ be the first element in $T$ and remove it from $T$;
            if ($a \notin V$) then
            begin
                $V \leftarrow V \cup \{a\}$;
                $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j+1)^a\}$;
            end
            $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{ \left( j^a, (j+1)^a \right) \right\}$;
            $c\left( j^a, (j+1)^a \right) \leftarrow 0$;
            if ($a + w_{j+1} \leq W$) then
            begin
                if ($a + w_{j+1} \notin V$) then
                begin
                    $V \leftarrow V \cup \{a + w_{j+1}\}$;
                    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j+1)^{a+w_{j+1}}\}$;
                end
                $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{ \left( j^a, (j+1)^{a+w_{j+1}} \right) \right\}$;
                $c\left( j^a, (j+1)^{a+w_{j+1}} \right) \leftarrow -v_{j+1}$;
            end
        end
        $T \leftarrow V$ and $V \leftarrow \{\}$;
    end
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{t\}$;
    while ($T \neq \{\}$) do
    begin
        Let $a$ be the first element in $T$ and remove it from $T$;
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{(l^a, t)\}$;
        $c(l^a, t) \leftarrow 0$;
    end
end
```

Figure 1: *Network converting algorithm.*

Some properties of the network generated by the procedure given in Figure 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 $\mathcal{G}$.
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, $\left(c^1(i,j), \ldots, c^r(i,j)\right)$.

To exemplify, consider the following bi-criteria knapsack example:

$$\begin{array}{lcl} \max\ z_1 & = & 8y_1 + 9y_2 + 3y_3 + 7y_4 + 6y_5 \\ \max\ z_2 & = & 3y_1 + 2y_2 + 10y_3 + 6y_4 + 9y_5 \\ \text{subject to:} & & 3y_1 + 2y_2 + 2y_3 + 4y_4 + 3y_5 \leq 9 \end{array}$$

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

The following network model is obtained:

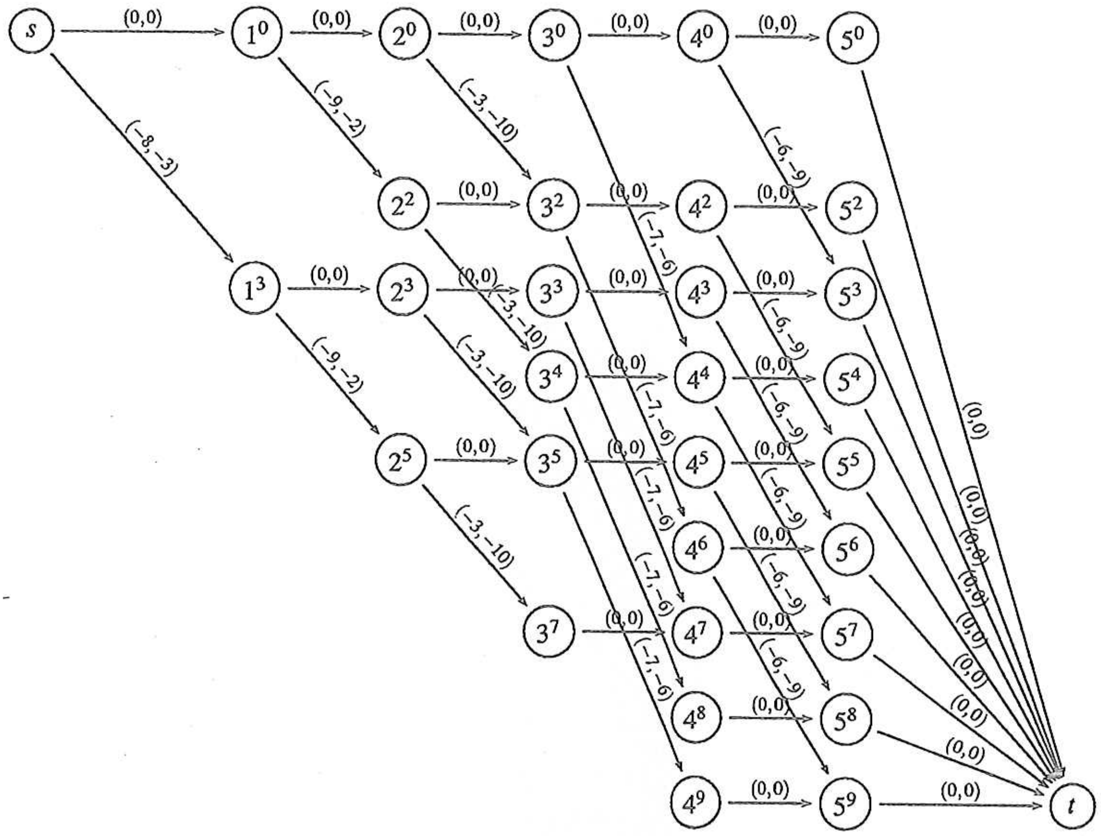

Figure 2: *Network model.*

where $|Q| = 32$ and $|\mathcal{A}| = 49$.

Algorithm of Figure 1 is only pedagogical, because in our computations networks are not explicit.

3.2 A new algorithm for the multiple criteria knapsack problem

The algorithm described in this subsection (see Figure 3) is a particular implementation of the labeling algorithms for multiple criteria shortest path problems on acyclic networks. For more details about labeling algorithms for general networks see *Martins* and *Santos* [1999] (`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; it is only required the knowledge of the vertices belonging to *layer j-1* in order to construct the set of vertices of *layer j* and the arcs connecting these two consecutive layers. Thus, all the other vertices and arcs are not taken into account in this procedure. With this technique a lot of memory space is saved which does not happen when using general networks.

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

In our algorithm, as well as in labeling algorithms for multiple criteria shortest path, all the labels, for each vertex, can be lexicographically ordered, but in the particular case of bi-criteria problems, the following property exists: the values concerning the first criterion are placed by non-decreasing order, while the values of the second one are placed by non-increasing order. So, in order 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 already determined. For more than two criteria the new label must be compared with all the labels already determined. This operation is too much 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 vertices in layers $j-1$ and j, respectively.
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 the whole set of labels of layer 1.
4. Then, for each of the remaining layers j $(j = 2, \ldots, l)$, 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 Figures 4a) and 4b)):
      i) If the arc is $\left((j-1)^a, j^a\right)$, i.e., a belongs to T but $a - w_j$ does not (see Figure 4a)). Then, $S(j^a)$ is equal to $S\left((j-1)^a\right)$ (see the computation of $S(4^2)$ from $S(3^2)$ in Figure 5).
      ii) If the arc is $\left((j-1)^{a-w_j}, j^a\right)$, i.e., a does not belong to T but $a - w_j$ does (see Figure 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 $S\left((j-1)^{a-w_j}\right)$ (see the computation of $S(4^6)$ from $S(3^2)$ in Figure 5).
   b) If j^a has two incoming arcs, i.e., a and $a - w_j$ belong to T (see Figure 4c)). Then, build $S(j^a)$ by choosing the ND labels simultaneously from the labels in $S\left((j-1)^a\right)$, and from the labels obtained by summing the vector $(-v_j^1, \ldots, -v_j^r)$ to each of the labels of set $S\left((j-1)^{a-w_j}\right)$.

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

Computing non-dominated vectors for the multiple criteria shortest path problem.

{ A new labeling algorithm to determine non-dominated vectors. }

begin
  Set $S(1^0) \leftarrow \{(0,\ldots,0)\}$ and $S(1^{w_1}) \leftarrow \left\{(-v_1^1,\ldots,-v_1^r)\right\}$;
  Set $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
  for $(j = 2)$ **to** l **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** { two incoming arcs are defined }
          $S(j^a) \leftarrow$ Set of ND labels of $\left(S\left((j-1)^a\right) \cup\right.$
          $\left.\{(-v_j^1,\ldots,-v_j^r)\} \oplus S\left((j-1)^{a-w_j}\right)\right)$;
        else { only the arc $\left((j-1)^a, j^a\right)$ is defined }
          $S(j^a) \leftarrow S\left((j-1)^a\right)$;
      end
      else
      begin
        if $(a - w_j \in T)$ **then** { only the arc $\left((j-1)^{a-w_j}, j^a\right)$ is
        defined }
        begin
          $S(j^a) \leftarrow \left\{(-v_j^1,\ldots,-v_j^r)\right\} \oplus S\left((j-1)^{a-w_j}\right)$;
          $V \leftarrow V \cup \{a\}$;
        end
        { **else** The vertex j^a does not exist }
      end
    end
    $T \leftarrow V$ and $V \leftarrow \{\}$;
  end
  $S(t) \leftarrow$ Set of ND labels of $\bigcup_{a=0}^{W} S(l^a)$;
end

Figure 3: *The new algorithm.*

In the above filtering algorithm:

$$\left\{(-v_j^1,\ldots,-v_j^r)\right\} \oplus S\left((j-1)^{a-w_j}\right) = \left\{(-v_j^1,\ldots,-v_j^r) + (\alpha^1,\ldots,\alpha^r) \mid (\alpha^1,\ldots,\alpha^r) \in S\left((j-1)^{a-w_j}\right)\right\}.$$

Now, the case 4*b*) is clarified by using the bi-criteria example (see the computation of $S(5^5)$ from $S(4^2)$ and $S(4^5)$ in Figure 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 by 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:

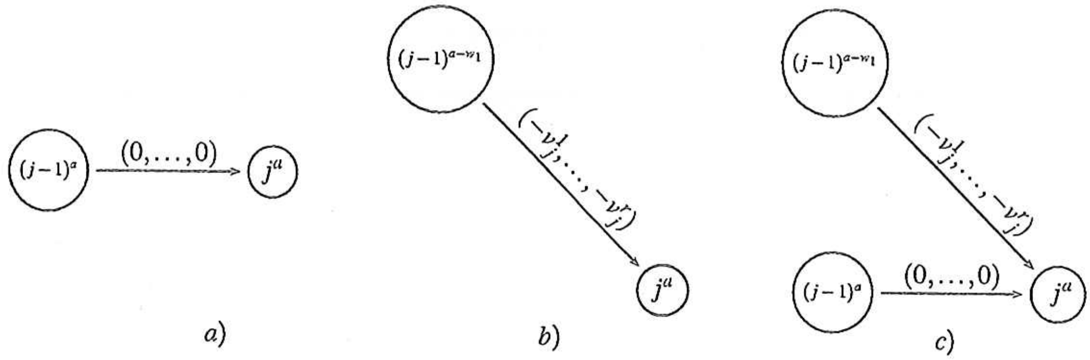

Figure 4: *Different cases of incoming arcs.*

- First of all, compare the first label of $S(4^5)$ with the label obtained from the first element of $S(4^2)$ plus $\left(c^1(4^2,5^5),c^2(4^2,5^5)\right)$, i.e., $(-17,-5)$ 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 $(-11,-13)$.
- Secondly, 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 $(-3,-10)$.
- Thirdly, compare the first label of $S(4^5)$ with the first label of $S(4^2)$ plus $\left(c^1(4^2,5^5),c^2(4^2,5^5)\right)$, 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 7 non-dominated solutions: $S(t)=\{(-24,-11),(-23,-14),(-22,-17),(-19,-18),(-18,-21),(-17,-22),(-16,-25)\}$. The solutions for the bi-criteria knapsack example can be easily rebuilt by using a backtracking technique.

Since the network is built layer by layer, during the optimization process, ranking algorithms (*Clímaco* and *Martins* [1982], *Martins* [1984] and *Martins* and *Santos* [2000]) cannot be used because they start by computing the shortest tree using the entire network. A previous study had used *Martins* and *Santos* [2000] ranking shortest paths algorithm. The algorithm appears inefficient because it quickly reaches the maximum memory capacity available. So, it was not possible to solve problems with more that 20 items.

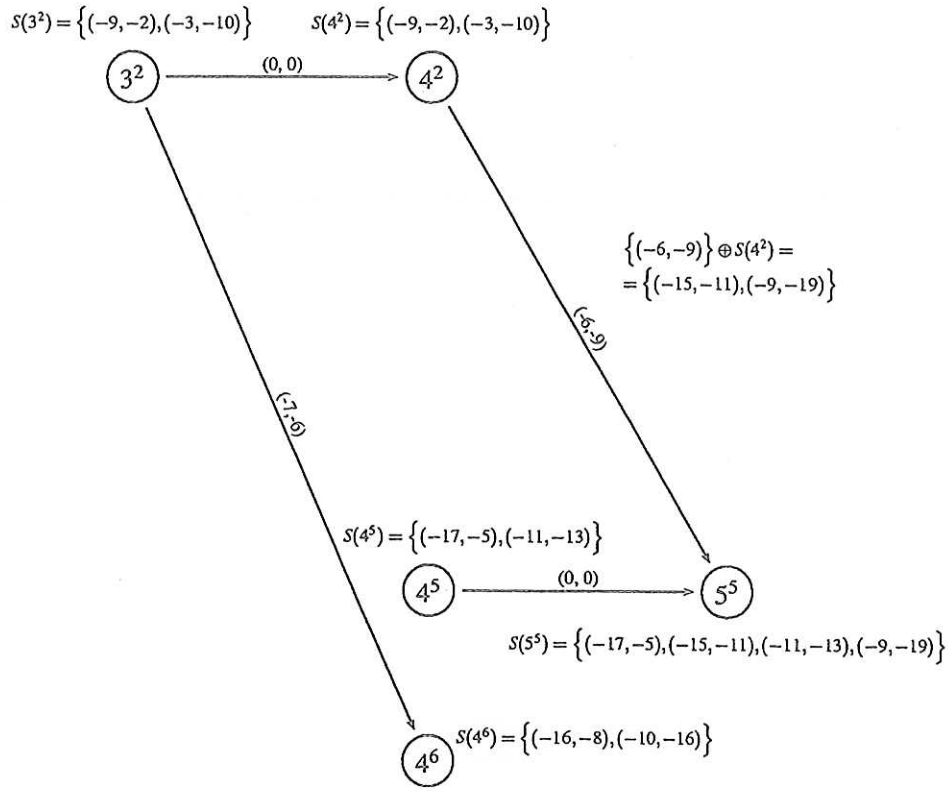

Figure 5: *An illustrative example.*

4 Experiments and Results

This section deals with the computational behaviour of the algorithm on certain sets of randomly generated instances. In this study we considered only bi-criteria $\{0,1\}$ knapsack instances because multiple criteria instances are much more demanding memory requirements. The design of the computational experiments was inspired by the framework proposed in *Martello* and *Toth*, [1990], for the classical single criterion $\{0,1\}$ knapsack problem. 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 $\frac{P}{100}\sum_{j=1}^{l} w_j$ (where P is a percentage of $\sum_{j=1}^{l} w_j$).

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, $P = 50$.

The numerical experiments have different aims:

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 know 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.

In this study, 20 different instances were generated for each problem. Every instance was built with the help the C version procedure `random.h` of NETGEN (*Klingman et al.* [1974]). 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[1] was executed on a Pentium II (two processors, 256 Mhz over Linux, 64 MB of RAM) at the CISUC (University of Coimbra, Portugal).

The results shown in Tables 1 to 8 (see Appendix) concern the average obtained from a set of 20 different instances. The memory space available is around 2 Gbytes.

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 P^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 is very low, the number of solutions is small. Consequently, the efficient solutions are also few. 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 high, but the efficient solutions are scarce. There are a few number of efficient solutions which dominate all the other solutions.

- The storage memory capacity limits the computational experimentation. So, uncorrelated bi-criteria 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 (hard) instances, we observe that, for a given number of items, the number of efficient solutions is similar when 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.

[1] 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`).

- 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.* Table 7 and 8 with Tables 2 to 6).

- 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 one. It is too much time and space memory demanding. The number of labels estimated at the last iterations 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).

- In general the parameters studied grow with the number of the items. However, the number of vertices, the number of arcs and the number of efficient solutions grow more slowly than the total number of labels created, the memory space and the CPU time. It should be noted that the memory capacity used depends on the amount of labels in two consecutive layers. This fact explains why problems of larger dimension cannot be solved.

- However, the code is very fast for the instances being considered here even when huge networks are considered. See for example Table 7 for $l = 900$. In such a case, the number of vertices is greater than $149,000,000$ and the number of arcs is greater than $297,000,000$. The CPU time is around 70 seconds.

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 executed. In particular,

1. To perform some experiments for more than two criteria for instances with a smaller number of items.

2. To compare the algorithm with the available approaches applied to the same kind of knapsack instances. For the moment, it was not possible to compare the proposed approach with other exact approaches because until now, it was not possible to obtain the data used by other approaches or their codes. It should be noted that in the above results, the size of instances which can be solved strongly depends on the way the instances are generated. In this study, instances of greater scope can be solved by choosing the "right" seed-numbers (see, for example, what happens with the results presented in Tables 7 and 8). So, any comparison with other codes requires the use of exactly the same set of instances. The comparison of the proposed approach with meta-heuristics techniques is not adequate.

3. To analyze the CPU time with a more advanced version of ranking algorithms (*Martins et al.* [1999a,b]) and/or other ranking paths algorithms.

4. To apply this approach to multiple criteria interactive procedure contexts, where a specific type of local search is needed.

5. To test the model in real-world applications.

The main conclusion of this article consists of showing the efficiency of the labeling algorithm for both uncorrelated and (weakly and strongly) correlated bi-criteria $\{0,1\}$ knapsack instances considered in this work.

6. References

AHUJA R.K., MAGNANTI T.M. and ORLIN J.B. (1993): *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

BEN ABDELAZIZ F., SAOUSSEN K. and JOUHAINA, C. (1998): "A hybrid heuristic for multiobjective knapsack problems", in S. Voss, S. Martello, I. Osman and C. Roucairol (eds.), *Meta-Heuristics: Advances and trends in local search paradigms for optimization*, Kluwer Academic Publishers, Dordrecht.

CLÍMACO J. and MARTINS E. (1982): "A bicriterion shortest path algorithm", *European Journal of Operational Research*, 11, 399-404.

HANSEN P. (1980): "Bicriterion path problems", in G. Fandel and T. Gal (Eds.), *Multiple Criteria Decision Making: Theory and Applications*, Lecture Notes in Economics and Mathematical Systems, Vol. 177, Springer-Verlag, Heidelberg, 109-127.

HANSEN M. (1997): "Solving multiobjective knapsack problems using MOTS", *Conference Paper*, presented at MIC'97, Sophia Antipolis, France, July, 21-24, 9p.

KLAMROTH K. and WIECEK M. (2000): "Dynamic programming approaches to the multiple criteria knapsack problems", *Naval Research Logistics Quarterly*, 47(1), 57-76.

KLINGMAN D., NAPIER A. and STUTZ, J. (1974): "NETGEN: A program for generating large scale capacity assignment, transportation, and minimum cost flow problems", *Management Science*, 20(5), 814-821.

KOSTREVA M., OGRYCZAK W. and TONKYN D.W. (1999): "Relocation problems arising in conservation biology", *Computers and Mathematics with Applications*, 37, 135-150.

KWAK W., SHI Y., LEE H. and LEE C.F. (1996): "Capital budgeting with multiple criteria and multiple decision makers", *Review of Finance and Accounting*, 7, 97-112.

MARTELLO S. and TOTH P. (1990): *Knapsack Problems: Algorithms and Computer Implementations*, Wiley and Sons, Chichester.

MARTINS E. (1984): "On a multicriteria shortest path problem", *European Journal of Operational Research*, 16, 236-245.

MARTINS E. and SANTOS J.L. (1999): "The labelling algorithm for the multiobjective shortest path problem", *Technical Report 99/005* CISUC, Departamento de Matemática, Universidade de Coimbra, Portugal, 24p (`http://www.mat.uc.pt/~ eqvm/eqvm.html`).

MARTINS E., PASCOAL M. and SANTOS J.L. (1999a): "Deviation algorithms for ranking shortest paths", *International Journal of Fundations of Computer Science*, 10(3), 247-261.

MARTINS E., PASCOAL M. and SANTOS J.L. (1999b): "A new improvement for a *K* Shortest paths algorithm" To appear in *Investigação Operacional.*

MARTINS E. and SANTOS J.L. (2000): "A new shortest path ranking algorithm", *Investigação Operacional*, 20, 47-61.

ROSENBLATT M.J. and SINUANY-STERN Z. (1989): "Generating the discrete efficient frontier to the capital budgeting problem", *Operations Research*, 37(3), 384-394.

TENG J.-Y. and TZENG G.-H. (1996): "A multiobjective programming approach for selecting non-independent transportation investment alternatives", *Transportation Research-B*, 30, 291-307.

ULUNGU E.L. and TEGHEM J. (1997): "Solving multi-objective knapsack problems by a branch-and-bound procedure", in J. Clímaco (ed.), *Multicriteria Analysis*, Proceedings of the XIth International Conference on MCDM, 1-6 August, Coimbra, Portugal, Springer-Verlag, Berlin, 269-278.

ULUNGU E.L., TEGHEM J., FORTEMPS, Ph. and TUYTTENS D. (1999): "MOSA Method: A tool for solving multi-objective combinatorial optimization problems", *Journal of Multi-Criteria Decision Analysis*, 8, 221-236.

VISÉE M., TEGHEM J., PIRLOT, M. and ULUNGU E.L. (1998): "Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem", *Journal of the Global Optimization*, 12, 139-155.

Appendix

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

P	W	n	m	Total number of labels created	$\lvert P^e \rvert$	CPU time (seconds)	Total memory used (Mbytes)
10	2,435.15	78,143.70	135,108.80	235,987.25	13.15	0.03	1.55
20	4,870.10	173,174.05	323,549.69	846,833.69	24.35	0.08	5.11
30	7,305.20	260,064.84	496,824.84	1,702,653.25	35.75	0.15	10.54
40	9,740.35	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

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

U	l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes
	10	2,363.40	1,126.50	1,611.35	1,175.00	4.00	0.00	0.3[illegible]
	20	4,908.25	37,293.70	66,893.85	77,756.95	9.75	0.04	2.2[illegible]
	30	7,445.60	121,021.60	228,378.95	466,598.75	19.65	0.11	6.4[illegible]
	40	9,769.60	241,556.95	464,428.50	1,461,549.62	30.45	0.19	13.5[illegible]
	50	12,175.15	401,911.31	780,121.12	3,537,664.50	43.60	0.21	25.3[illegible]
	60	14,629.80	601,491.38	1,174,242.25	7,195,329.00	60.40	0.34	42.7[illegible]
	70	17,233.90	844,104.38	1,654,160.38	13,434,125.00	83.60	0.52	68.6[illegible]
1000	80	19,881.10	1,126,660.25	2,213,888.75	22,916,352.00	101.95	0.90	99.6[illegible]
	90	22,433.25	1,443,420.38	2,842,250.00	36,360,640.00	126.30	1.05	138.4[illegible]
	100	24,932.25	1,794,925.25	3,540,228.00	54,559,704.00	147.90	1.50	186.4[illegible]
	120	30,059.85	2,620,489.50	5,181,052.50	109,194,928.00	200.70	2.57	306.5[illegible]
	140	34,861.30	3,577,586.00	7,085,609.50	193,496,672.00	246.60	4.11	458.1[illegible]
	160	39,764.45	4,689,653.00	9,299,918.00	320,856,384.00	317.20	6.38	680.4[illegible]
	180	44,767.10	5,959,491.00	11,829,573.00	504,200,544.00	403.85	9.44	957.1[illegible]
	190	47,272.25	6,651,830.00	13,209,230.00	620,352,640.00	449.75	13.57	1,109.8[illegible]
	200	49,674.45	7,371,979.00	14,644,720.00	753,986,624.00	477.70	13.02	1,274.1[illegible]
	210	52,083.60	8,129,389.00	16,154,717.00	907,709,440.00	527.50	24.02	1,465.9[illegible]
	10	755.45	921.10	1,368.15	994.55	3.80	0.00	0.1[illegible]
	20	1,473.95	14,020.55	25,517.10	33,629.90	10.05	0.01	0.7[illegible]
	30	2,260.70	40,920.70	77,562.30	181,812.25	20.35	0.04	2.3[illegible]
	40	3,032.55	79,682.75	153,465.16	559,105.12	31.10	0.07	4.9[illegible]
	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.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,836.50	1,116.45	1,605.15	1,183.20	5.30	0.00	0.41
20	5,558.15	39,449.15	70,511.15	117,016.00	16.70	0.05	3.42
30	8,308.25	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.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,836.50	1,116.45	1,605.15	1,162.15	5.65	0.00	0.40
20	5,558.15	39,449.15	70,511.15	109,515.80	14.40	0.03	3.20
30	8,308.25	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

Table 5: Average results for weakly correlated instances of Type 3.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,400.45	1,129.80	1,616.85	1,131.95	1.15	0.00	0.35
20	4,856.45	36,933.95	66,179.20	41,247.00	1.40	0.03	1.21
30	7,445.70	121,022.95	228,381.66	148,925.09	1.95	0.05	2.11
40	9,770.05	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

Table 6: Average results for weakly correlated instances of Type 4.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,400.45	1,129.80	1,616.85	1,132.70	1.15	0.00	0.35
20	4,856.45	36,933.95	66,179.20	40,804.40	1.30	0.03	1.20
30	7,445.70	121,022.95	228,381.66	147,886.16	2.25	0.06	2.09
40	9,770.05	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	1,028.89
800	195,507.91	117,724,592.00	235,058,208.00	>2,147,483,647.00	100.35	44.92	1,496.91

Table 7: Average results for strongly correlated instances of Type 5.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,400.45	1,129.80	1,616.85	1,149.25	3.35	0.00	0.35
20	4,856.45	36,933.95	66,179.20	48,615.20	5.45	0.03	1.40
30	7,445.70	121,022.95	228,381.66	205,729.30	9.45	0.08	2.80
40	9,770.05	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	1,260.56
900	220,187.80	149,068,448.00	297,696,576.00	>2,147,483,647.00	159.85	71.53	1,610.71

Table 8: Average results for strongly correlated instances of Type 6.

l	W	n	m	Total number of labels created	$\|P^e\|$	CPU time (seconds)	Total memory used (Mbytes)
10	2,400.45	1,129.80	1,616.85	1,146.45	3.75	0.00	0.35
20	4,856.45	36,933.95	66,179.20	47,687.00	5.50	0.01	1.37
30	7,445.70	121,022.95	228,381.66	198,618.16	7.65	0.01	2.71
40	9,770.05	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	147,143.95	66,552,872.00	132,810,600.00	1,609,998,720.00	120.74	25.26	746.21

The asterisk (*) means that only 19 instances were solved.