LABELLING ALGORITHMS FOR MULTIPLE CRITERIA KNAPSACK PROBLEMS

JOSÉ RUI FIGUEIRA*
CEG-IST, Center for Management Studies - Instituto Superior Técnico
TagusPark, Av. Cavaco Silva, 2780-990 Porto Salvo, Portugal
Phone: + 351 21 423 32 99. Fax: + 351 21 423 35 68. E-mail: figueira@ist.utl.pt

GABRIEL TAVARES
RUTCOR, Rutgers University
640 Bartholomew Road Piscataway, NJ 08854-8003, USA
E-mail: gtavares@rutcor.rutgers.edu

MARGARET M. WIECEK
Department of Mathematical Sciences, Clemson University
Clemson, SC 29634-1907, USA
Phone: +864 656-5245. Fax: +864 656-5230. E-mail: wmalgor@clemson.edu

5 December 2006

ABSTRACT: The paper presents a generic labeling algorithm for finding all nondominated outcomes of the integer multiple criteria knapsack problem (MCKP). The algorithm is based on solving the multiple criteria shortest path problem on an underlying network. Algorithms for constructing four network models, all representing the MCKP, are also presented. Each network is composed of layers and each network algorithm, working forward layer by layer, identifies the set of all permanent nondominated labels for each layer successively. The effectiveness of the algorithms is supported with numerical results obtained for randomly generated problems for up to seven criteria or forty variables. One of the algorithms appears to determine the state of the art in exact algorithms for the MCKP. Extensions of the approach to other classes of problems including binary variables, bounded variables, multiple constraints, and time-dependent objective functions are possible.

Key-words: multiple criteria knapsack problem, labeling algorithms, shortest path problem.

*Part of this work was accomplished when the first author was a visiting Researcher at DIMACS Center, Rutgers University, CoRE Building, 4th Floor, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018, USA. Phone: +1 732 445 0075. Fax: +1 732 445 5932. E-mail: figueira@dimacs.rutgers.edu

1 Introduction

The multiple criteria knapsack problem is a well known combinatorial optimization problem with a wide range of applications. Examples may be found in affordability analysis and capital budgeting where projects have to be chosen with respect to more than a single criterion (see e.g., Bhaskar [5], Vetschera [25], and Kwak et al. [19]), in transportation investment planning (Teng and Tzeng [22]), or in conservation biology to model relocation issues (Kostreva et al. [17]).

In this paper we consider the integer multiple criteria knapsack problem (MCKP) formulated as

$$\begin{array}{lll} \text{vmax} & f(x) = Vx & \\ \text{s.t.} & wx \leq W & \\ & x_j \geq 0, \text{ integer}, & j = 1, \ldots, l \end{array} \tag{1}$$

where V is an $r \times l$ matrix with nonnegative entries v^i_j, $i = 1, \ldots, r$, $j = 1, \ldots, l$. We denote the ith row of V by v^i and the jth column of V by v_j. Thus $f_i(x) = v^i x$, $i = 1, \ldots, r$, represent the r conflicting objective functions. The constraint $wx \leq W$ is interpreted as a *capacity constraint* (*budget constraint*). The set of feasible solutions of (1) is given by $X = \{x \in \mathbb{N}_0^l \,:\, wx \leq W\}$.

Throughout the paper we additionally assume that the weight coefficients w_j, $j = 1, \ldots, l$, and the right-hand-side of the capacity constraint W are positive integers. In order to avoid trivial solutions let $0 < w_j \leq W$, $j = 1, \ldots, l$ and $\sum_{j=1}^{l} w_j > W$.

A special case of the above formulation is the case in which $r = 2$, i.e., the bicriteria case.

Solving (1) is understood as generating its efficient (Pareto) solutions. A feasible solution $\hat{x} \in X$ is said to be an efficient solution of (1) if there is no other feasible solution $x \in X$ such that $f(x) \geq f(\hat{x})$, i.e.:

$$\begin{array}{lll} & \forall i \in \{1, \ldots, r\} & f_i(x) \geq f_i(\hat{x}) \\ \text{and} & \exists i \in \{1, \ldots, r\} \quad s.t. & f_i(x) > f_i(\hat{x}). \end{array} \tag{2}$$

Let $\mathcal{X}_e$ denote the set of efficient solutions of (1) and let $\mathcal{Y}_e$ denote the image of $\mathcal{X}_e$ in the objective space, that is $\mathcal{Y}_e = f(\mathcal{X}_e)$, where $f = [f_1, \ldots, f_r]$. The set $\mathcal{Y}_e$ is referred to as the set of nondominated outcomes of (1).

The MCKP is a difficult problem to solve since the binary single-criterion knapsack problem is already NP-complete. While many authors have proposed algorithms for finding all or some nondominated outcomes of the MCKP, a majority of the algorithms deal

with the bicriteria knapsack problem.

As the MCKP falls into the category of multiple criteria integer programs, algorithms proposed for finding nondominated outcomes of the latter could be also applied to solve the former. In this review we focus on approaches developed specially for the knapsack problem with more than two criteria. In general, these approaches can be classified as exact procedures and metaheuristics. The former aims at finding all nondominated outcomes of the MCKP and includes branch-and-bound procedures, dynamic-programming-based approaches, and labeling algorithms.

A branch-and-bound procedure was proposed by Ulungu and Teghem [23].

Villarreal and Karwan [26] were perhaps the first ones who proposed dynamic programming (DP) approaches to the MCKP. They proposed four approaches: two basic ones, an embedded state approach, and a hybrid approach. Later in [27], they also extended DP recursive equations to the general multiple criteria integer framework and presented them on the binary MCKP with multiple constraints. Klamroth and Wiecek [16] proposed a comprehensive DP methodology able to solve a broad class of knapsack problems including the MCKP and its more complex extensions such as binary variables, multiple constraints, multiple periods, and time-dependent criterion functions. They presented DP recursive equations for five network models representing the MCKP and showed how to apply the equations and networks for the MCKP and its extensions.

An independent research effort was undertaken by Captivo et al. [6] who applied the concept of a labeling algorithm to the MCKP viewed as the multiple criteria shortest path problem on an underlying network. The algorithm turned out to be very effective for some hard instances of the bicriteria binary case of the MCKP.

The combinatorial nature of MCKP motivated the development of meta-heuristic algorithms producing a subset or an approximation of the set of all nondominated outcomes. Arndt and Seelaender [2] outlined an approach based on the concept of ceiling points. Simulated annealing was extensively studied by Czyzak and Jaszkiewicz [8] and Ulungu et al. [24]. Hansen [13] and Gandibleux et al. [9] applied tabu-search principles to construct an approximation of the nondominated set. Combinations of tabu search and a genetic algorithm were developed by Ben Abdelaziz et al. [4]. A comparative study of the effectiveness of genetic algorithms was presented by Zitzler and Thiele [29].

In the current decade, researchers continued efforts on the development of genetic algorithms or hybrid algorithms. Improved performance of genetic algorithms due to the use of approximate dominance was reported by Grosan and Oltsean [11] and Kumar and Banerjee [18]. Barichard and Hao combined a genetic procedure with a tabu search operator [3], Guo et al. proposed a hybrid memetic algorithm [12], and Zhang et al. developed an immune system strength Pareto algorithm based on a clonal selection theory [28]. Some authors also conducted more comparative studies including Laumanns et al. [20], Jaszkiewicz [15], and Colombo and Mumford [7].

Despite the success of metaheuristic algorithms we believe that deterministic algorithms remain competitive for solving the MCKP. In this paper, we therefore continue in the direction of labeling algorithms as they showed to be very promising in constructing an effective algorithm for solving the MCKP [6]. On the other hand, the dynamic-programming framework provided by Klamroth and Wiecek [16] turned out to be a flexible tool for solving a variety of knapsack problems with multiple criteria. We therefore present a new framework featuring the computational effectiveness of labeling algorithms and the modeling flexibility of dynamic programming. This new framework is composed of a family of networks similar to those in [16] for which a generic labeling algorithm is designed. The algorithm with minor adjustments solves the multiple criteria shortest path problem on every network and generates the set of all nondominated outcomes of the MCKP and its extensions.

In Section 2 we present algorithms for constructing several network models for the MCKP. Although the models are collected from the literature, the algorithms are new since they aim at minimizing the network size and the related computational effort. The generic labeling algorithm, suitable for customization for the proposed network models, is developed in Section 3 while Section 4 describes a computational experiment. The effectiveness of the algorithms is supported with numerical results obtained for randomly generated problems for up to seven criteria or forty variables. One of the algorithms appears to determine the state of the art in exact algorithms for the MCKP. Section 5 indicates further possibilities for the refinement of the algorithm and concludes the paper. Detailed numerical results and pseudocodes of the algorithms for constructing the networks are given in the Appendix.

2 Algorithms for building network models

In this section we present four network models to be used for solving the MCKP. These models are based on modeling approaches available in the literature and developed within the framework of dynamic programming. The new models are presented in the language of network flow programming [1] and their main feature is reduction of their size. The new networks include a smaller number of vertices and arcs comparing to the dynamic programming based networks.

Every network is defined as a directed and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of vertices with $\mid \mathcal{V} \mid = n$ and $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of arcs with $\mid \mathcal{A} \mid = m$. The arc linking vertices i and j is denoted by (i, j), and the vector $[c^1(i, j), \ldots, c^r(i, j)]$ represents the r criterion values associated with the arc (i, j). In the set $\mathcal{V}$, we distinguish a source vertex, a sink vertex and terminal vertices that are directly connected with the sink vertex. A path p from a source vertex s to a sink vertex 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.

Every network is composed of layers g. A layer is a set of vertices. A layer g is called a successor of a layer g' if there is at least one arc from g' to g.

The networks have several common properties. All networks are acyclic. There may exist arcs from layer g to layers $g+1, g+2, \ldots, G$. Within a layer, there may exist arcs linking vertices in this layer and there may exist terminal vertices. In the topological terms, all the arcs in a network are horizontal, vertical, or diagonal down right. This property allows us to set permanent labels from the top to the bottom of each layer.

Every vertex has a position in a layer. A vertex in position k, for $k = 0, \ldots, K$ in layer g is denoted by g^k. For example, 3^6 denotes a vertex in layer 3 in position 6.

Throughout the paper we use the following didactic example of the bicriteria case of MCKP to illustrate all network models and the generic labeling algorithm:

$$
\begin{aligned}
vmax \quad & f(x) = \begin{pmatrix} 8 & 9 & 3 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \\
s.t. \quad & \begin{pmatrix} 2 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq 6 \\
& x_j \geq 0, \ integer, \ j = 1,2,3
\end{aligned}
$$

In the following subsections we present four different algorithms to construct four network models for MCKP. At the beginning of each subsection we refer the reader to related results from the literature.

2.1 Algorithm I

Algorithm I is based on the recursive equations of Garfinkel and Nemhauser [10] (equations III), Ibaraki [14] (representation 1), Villarreal and Karwan [26] (approach 2), and Klamroth and Wiecek [16] (model I).

Each layer of the network is a singleton.

Let T denote a stack containing vertices already created but not visited. A vertex is said to be visited if the set of all outgoing arcs of this vertex has been created. Otherwise, the vertex is said to be non-visited.

(0) Set $\mathcal{V} \leftarrow \{s\}$, $T \leftarrow \varnothing$.

(1) Connect the vertex s with the vertex $g \leftarrow w_j$, for every $j = 1, \ldots, l$. Add the vertex g to T if it has not been already added and add the arc (s,g) to $\mathcal{A}$. Assign the cost vector $c(s,g) \leftarrow [-v_j^1, \ldots, -v_j^r]$ to this arc.

(2) While T contains non-visited vertices, select the next non-visited vertex u in T. If $g \leftarrow u + w_j \leq W$ for every $j = 1, \ldots, l$, connect u with g. Add the vertex g to T if it has not been already added and add the arc (u, g) to $\mathcal{A}$. Assign the cost vector $c(u, g) \leftarrow [-v_j^1, \ldots, -v_j^r]$ to this arc.

(3) Connect every vertex g in T with the vertex t. Add the vertex t to T and the arcs (g, t) to $\mathcal{A}$. Assign the cost vector $c(g, t) \leftarrow (0, \ldots, 0)$ to each of these arcs.

(4) Let $\mathcal{V} \leftarrow T$.

Output: A network $\mathcal{G} = (\mathcal{V}, \mathcal{A}, c)$.

Figure 1 depicts the network constructed by Algorithm I for the example problem.

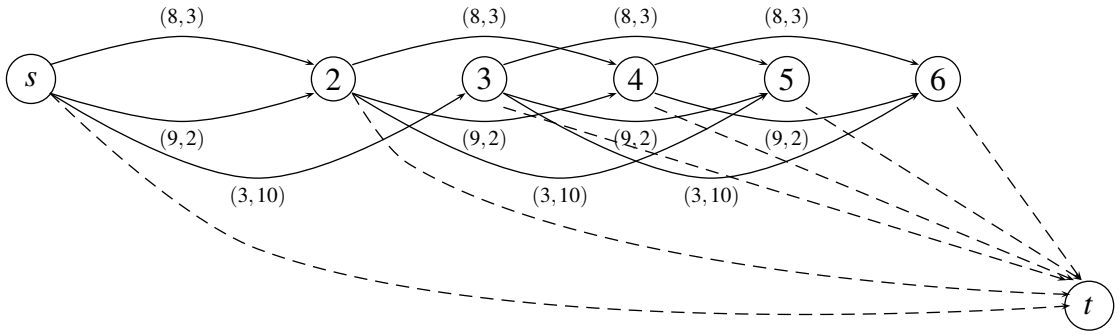

Figure 1: *Network model I (the cost vector of dashed arcs is* $(0,0)$*).*

2.2 Algorithm II

Algorithm II is based on the recursive equations of Ibaraki [14] (representation 2) and used by Klamroth and Wiecek [16] (model II).

Each layer of the network typically contains several vertices (but it may be a singleton). The layers g belong to the set $\{1, \ldots, W\}$ while the positions k in each layer belong to the set $\{1, \ldots, l\}$.

Let T denote a list of all the vertices in the current layer and U denote a list of vertices being successors of the vertices in T.

(0) Set $\mathcal{V} \leftarrow \{s\}, T \leftarrow \varnothing$ and $U \leftarrow \varnothing$.

(1) Connect the vertex s with a vertex g^k such that $g \leftarrow w_j$ and $k \leftarrow j$ for every $j = 1,\ldots,l$ and add the vertex g^k to U and to $\mathcal{V}$. Add the arcs (s,g^k) to $\mathcal{A}$ and assign the cost vector $c(s,g^k) \leftarrow [-v_k^1,\ldots,-v_k^r]$ to each of these arcs.

(2) While U is non-empty, move the vertices g^k of the current layer to T. For all vertices g^k in T, if $g + w_j \leq W$, connect g^k with the vertex $(g+w_j)^j$ and add it to U and to $\mathcal{V}$ if it has not been already added. Add the arcs $(g^k,(g+w_j)^j)$ to $\mathcal{A}$ and assign the cost vector $c(g^k,(g+w_j)^j) \leftarrow [-v_j^1,\ldots,-v_j^r]$ to each of these arcs. Set $T \leftarrow \varnothing$.

(3) Connect every vertex g^k in $\mathcal{V}$ to the vertex t and add t to $\mathcal{V}$. Add the arcs (g^k,t) to $\mathcal{A}$ and assign the cost vector $c(g^k,t) \leftarrow [0,\ldots,0]$ to each of these arcs.

Output: A network $\mathcal{G} = (\mathcal{V},\mathcal{A},c)$.

Figure 2 depicts the network constructed by Algorithm II for the example problem.

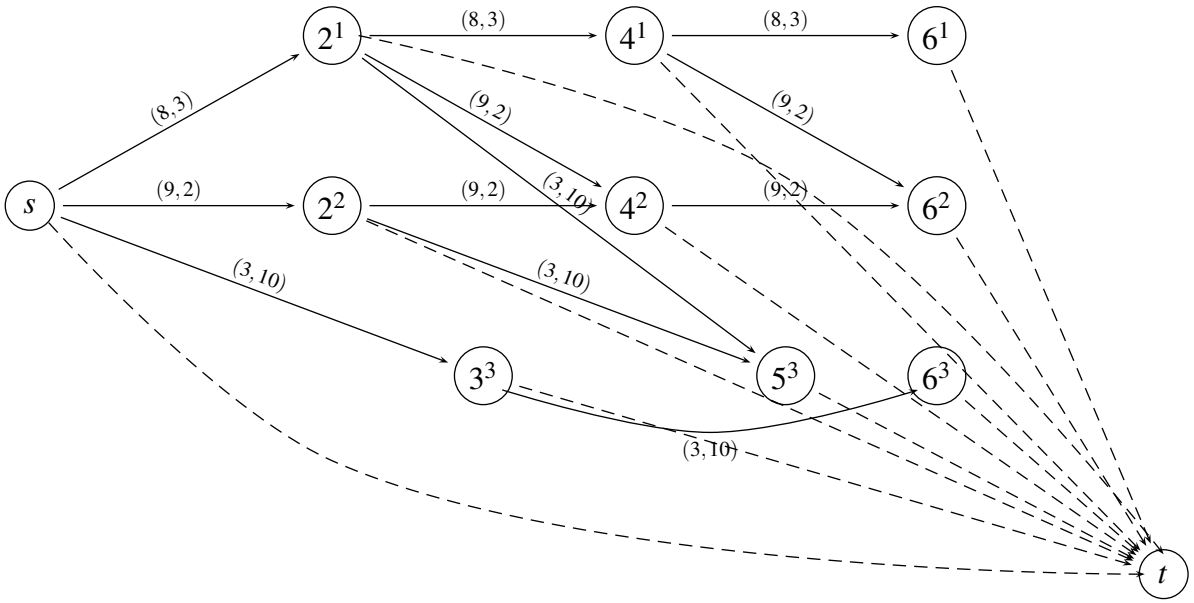

Figure 2: *Network model II (the cost vector of dashed arcs is* $(0,0)$*).*

2.3 Algorithm III

Algorithm III is based on the recursive equations of Garfinkel and Nemhauser [10] (equations I), Ibaraki [14] (representation 3), Villarreal and Karwan [26] (approach 1), and Klamroth and Wiecek [16] (model III). Captivo et al. [6] used this algorithm to construct a network on which a binary bi-criteria knapsack problem was very effectively solved with a labeling algorithm.

Each layer of the network contains several (more than one) vertices. The layers g belong to the set $\{1,\ldots,l\}$, while the positions k in each layer belong to the set $\{0,\ldots,W\}$.

Let T denote a list of positions of vertices in the current layer and U denote an auxiliary list of positions.

(0) Set $\mathcal{V} \leftarrow \{s\}$, $T \leftarrow \varnothing$ and $U \leftarrow \varnothing$.

(1) Connect the vertex s with vertices 1^k in layer 1 for $k \leftarrow \beta w_1$, $\beta = 0,1,\ldots,\lfloor W/w_1 \rfloor$. Add the positions k to U and the vertices 1^k to $\mathcal{V}$. Add the arcs $(s,1^k)$ to $\mathcal{A}$ and assign the cost vector $c(s,1^k) \leftarrow \beta[-v_1^1,\ldots,-v_1^r]$ to these arcs.

(2) While $g < l$, for every g^b with b in T, if $k \leftarrow b + \beta w_j \leq W$ for every $\beta = 0,1,\ldots,(\leq \lfloor W/w_j \rfloor)$:

   a) Connect the vertex g^b with the vertex $(g+1)^k$.

   b) Add the position k to U if it has not been already added, and the vertex $(g+1)^k$ to $\mathcal{V}$.

   c) Add the arc $(g^b,(g+1)^k)$ to $\mathcal{A}$.

   d) If $b = k$ assign the cost vector $c(g^k,(g+1)^k) \leftarrow [0,\ldots,0]$ to each of these arcs; otherwise, $c(g^b,(g+1)^k) \leftarrow \beta[-v_g^1,\ldots,-v_g^r]$.

   e) Set $T \leftarrow \varnothing$ and move the elements of U to T.

(3) Connect the vertex g^k, k in T, with the vertex t. Add the vertex t to $\mathcal{V}$. Add the arcs (g^k,t) to $\mathcal{A}$ and assign the cost vector $c(g^k,t) \leftarrow [0,\ldots,0]$ to each of these arcs.

Output: A network $\mathcal{G} = (\mathcal{V},\mathcal{A},c)$.

Figure 3 depicts the network constructed by Algorithm III for the example problem.

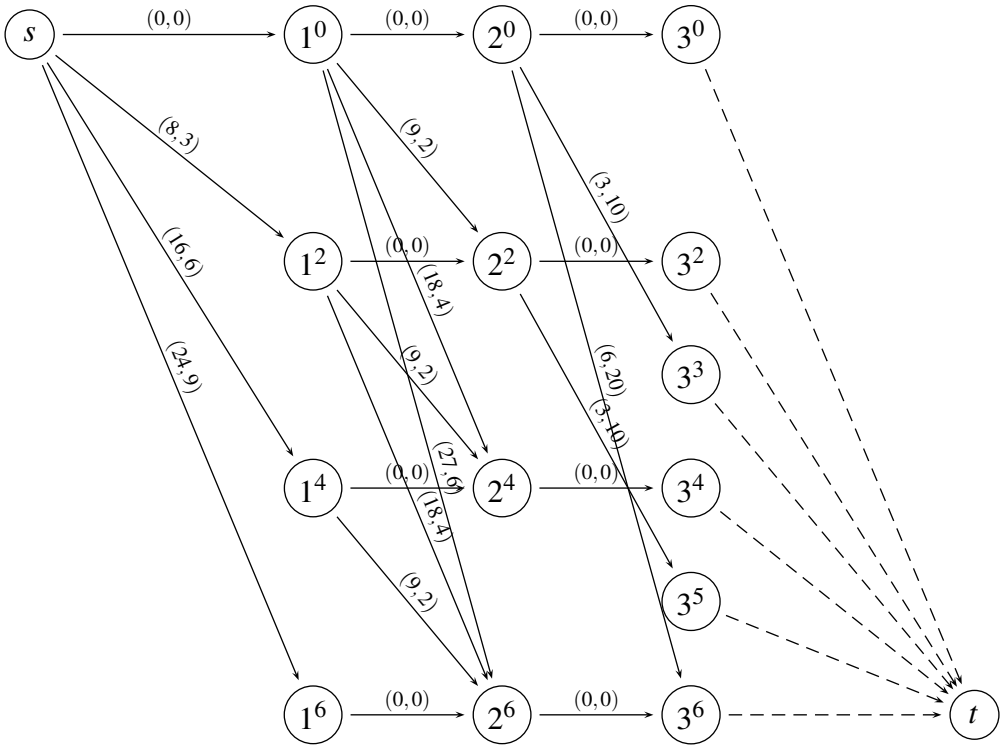

Figure 3: *Network model III (the cost vector of dashed arcs is* $(0,0)$*).*

2.4 Algorithm IV

Algorithm IV is based on the recursive equations of Garfinkel and Nemhauser [10] (equations II) later used by Klamroth and Wiecek [16] (model IV).

Each layer of the network contains several (more than one) vertices. The layers g belong to the set $\{1,\ldots,l\}$ while the positions k in each layer belong to the set $\{0,\ldots,W\}$.

Let T denote a list of positions of vertices in the current layer and U denote an auxiliary list of positions.

(0) Set $\mathcal{V} \leftarrow \{s\}$, $T \leftarrow \varnothing$ and $U \leftarrow \varnothing$.

(1) Connect the vertex s with vertices 1^k in layer 1 if $k \leftarrow \beta w_1$ for $\beta = 0,1,\ldots,\lfloor W/w_1 \rfloor$. Add the positions k to T and the vertices 1^k to $\mathcal{V}$. Add the arcs $(s,1^k)$ to $\mathcal{A}$, and assign the cost vector $c(s,1^k) \leftarrow [0,\ldots,0]$ to these arcs.

(2) While $g \leq l$, for every vertex g^b, b in T:

1. Connect g^b with $(g+1)^b$. Add the position b to U, the vertex $(g+1)^b$ to $\mathcal{V}$, and the arc $(g^b,(g+1)^b)$ to $\mathcal{A}$. Assign the cost vector $c(g^b,(g+1)^b) \leftarrow [0,\ldots,0]$ to this arc.
2. If $k \leftarrow b+w_g \leq W$, connect the vertex g^b with the vertex g^k. Add the position k to U if it has not been already added, and the vertex g^k to $\mathcal{V}$. Add the arc (g^b,g^k) to $\mathcal{A}$ and assign the cost vector $c(g^b,g^k) \leftarrow [-v_g^1,\ldots,-v_g^r]$ to this arc.
3. Set $T \leftarrow \varnothing$ and move the elements of U to T.

(3) Connect every vertex g^k in layer l to the vertex t. Add the vertex t to $\mathcal{V}$. Add the arcs (g^k,t) to $\mathcal{A}$ and assign the cost vector $c(g^k,t) = [0,\ldots,0]$ to each of these arcs.

Output: A network $\mathcal{G} = (\mathcal{V},\mathcal{A},c)$.

Figure 4 depicts the network constructed by Algorithm IV for the example problem.

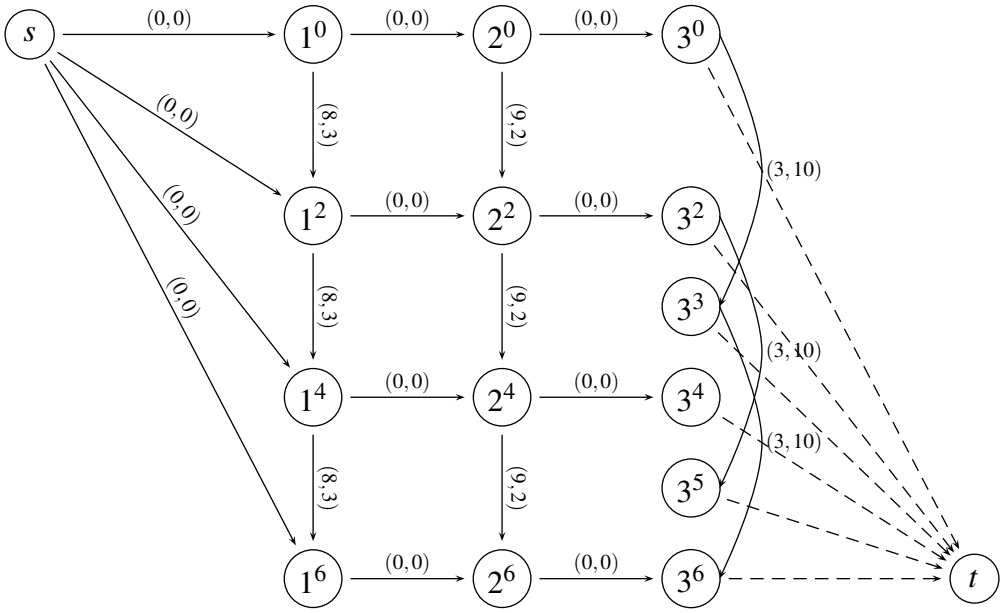

Figure 4: *Network model IV (the cost vector of dashed arcs is* $(0,0)$*).*

2.5 Comparison of the algorithms

Table 1 summarizes some basic properties of the four types of networks constructed by the presented algorithms. In the first four rows of the table the topology of networks is addressed while the other three rows specify the lists (arrays) needed for data storage. The numbers given in the table are based on the worst case data instance.

	Network I	Network II	Network III	Network IV
number of layers	$W+2$	$W+2$	l	l
size of a layer	1	l	$W+1$	$W+1$
connection between layers	various layers	various layers	adjacent layers	adjacent layers
connection within a layer	No	No	No	Yes
lists	T	T,U	T,U	T,U
size of T	$W+2$	l	$W+1$	$W+1$
size of U	-	$l\times(W+1)$	$W+1$	$W+1$

Table 1: Comparison of Networks I-IV

3 Generic labeling algorithm

We now present an algorithm to find all nondominated outcomes of the MCKP. The algorithm is generic in the sense that the MCKP can be modeled by any network from among the four networks presented in the previous section and the solutions of the MCKP are found as solutions of a related multiple criteria shortest path problem. The algorithm is presented in the form of a pseudocode that should be customized for a network of choice.

Let $f_q(p)$ denote the value of a path p with respect to the criterion q, for $q=1,\ldots,r$, i.e., $f_q(p)=\sum_{(i,j)\in p} c_q(i,j)$. A path p^n in $\mathcal{G}$ is said to be nondominated if and only if there does not exist another path p in $\mathcal{G}$ such that $f_q(p)\leq f_q(p^n)$, for all $q=1,\ldots,r$, with at least one strict inequality. The multiple criteria shortest path problem consists in computing the set of all nondominated paths from a source vertex s to a sink vertex t in $\mathcal{G}$. Let P_n denote the set of all nondominated paths in $\mathcal{G}$ from s to t.

The following notation is needed. Let

- $\{1,\ldots,G\}$ denote the set of layers g constructed by each algorithm;
- $K(g)$ denote the set of positions in layer g;
- $L(u)=(\alpha_1,\ldots,\alpha_r)$ denote an $r-$dimensional vector whose every component represents the distance from the vertex s to the vertex u;
- $S(u)$ denote the set of labels of vertex u.

In the initialization of the algorithm, the vertex s is labeled as well as all its successors (including the vertex t if it is a successor of s).

In the main step of the algorithm, the set of labels for all successor vertices of the current vertex $g^k, k \in K(g)$, is calculated. The new set of labels, $S(u)$, for each successor vertex u is composed of the nondominated labels in the union of the set of current nondominated labels of the successor vertex u and the set being the algebraic sum of the set of nondominated labels of the current vertex g^k and the cost vector, $c(g^k,u)$, of the arc from the current vertex g^k to the successor vertex u.

Figure 5 presents a pseudocode of the algorithm. The set of all nondominated paths in $\mathcal{G}$ from s to t computed by the algorithm is equivalent to the set of all nondominated outcomes of the MCKP.

The complexity of the algorithm depends on the complexity of the networks built and examined in each model and the complexity of the main step of the algorithm. The former results for the number of variables and the magnitude of the right-hand-side coefficient W (see Table 1) while the latter is determined by the complexity of the multiple criteria shortest path problem which is known to be NP-complete [21].

GENERIC ALGORITHM
{ A generic labeling algorithm for MCKP models. }
INITIALIZATION
  $L(s) \leftarrow (0,\ldots,0)$;
  if (s is a terminal vertex) **then**
    $S(t) \leftarrow \{(0,\ldots,0)\}$;
  else
    $S(t) \leftarrow \{(\infty,\ldots,\infty)\}$;
  for (all $(s,u) \in \mathcal{A}$, $u \neq t$) **do**
    $S(u) \leftarrow \{c(s,u)\}$;
MAIN STEP
  for (each layer $g \in \{1,\ldots,G\}$) **do**
    for (each $k \in K(g)$) **do**
      for (each u such that $(g^k,u) \in \mathcal{A}$) **do**
      begin
        $S(u) \leftarrow vmin\Big\{S(u) \cup \{S(g^k) \oplus c(g^k,u)\}\Big\}$;
        if (g^k is a terminal vertex) **then**
          $S(t) \leftarrow vmin\Big\{S(t) \cup S(g^k)\Big\}$;
      end
END

Output: The set $P_n \in \mathcal{G}$ from s to t.

Figure 5: *The generic labeling algorithm.*

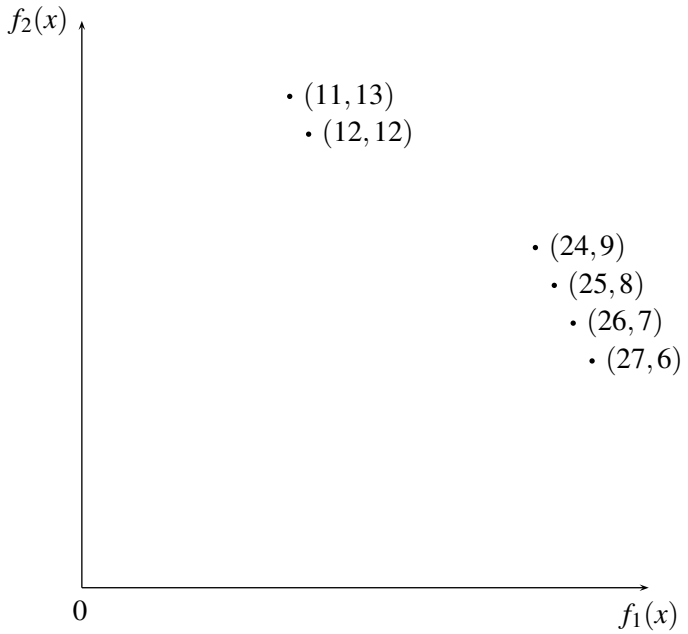

Figure 6: *The set of nondominated paths for the example problem.*

4 Numerical results

In this section we report on our computational experiment. The four models and algorithms were implemented in C++, compiled using the Microsoft Windows 32-bit C/C++ Optimizing Compiler (version 12) for 80x86, and linked with the Microsoft Incremental Linker (version 6). All the experiments were carried out on an 1133 MHz Pentium III with 256 MB of RAM and Windows XP.

Instances of the MCKP were randomly generated for a given number of criteria and a given number of variables. Thirty instances of the problem were typically solved when the solution time for every instance did not exceed one hour. Occasionally, the calculations were stopped and a smaller number of instances is reported. For each instance, the objective and constraint coefficients were randomly generated within a certain range.

The numerical results are reported in the Appendix in Tables 2-6 in which the following notation is used:

- UB is the upper bound for the coefficients w_j and v_j^k that are randomly generated within the range $[1, UB]$;
- l is the number of variables;

- N is the number of instances of the MCKP solved;
- $\bar{W}$ is the average capacity of the knapsack calculated as the average of the capacities of all the instances solved;
- $|\bar{X}^e|$ is the average number of efficient solutions calculated as the average of the number of efficient solutions of all the instances solved;
- $|\bar{Y}^e|$ is the average number of nondominated outcomes calculated as the average of the number of nondominated outcomes of all the instances solved;
- $St.Dev.\bar{\sigma}$ is the standard deviation of the number of nondominated outcomes obtained for all the instances solved;
- $\bar{n}$ is the average number of vertices of the graph obtained when converting the knapsack problem into a multicriteria shortest path problem calculated with respect to all the instances solved;
- $\bar{m}$ is the average number of arcs of the graph obtained when converting the knapsack problem into a multicriteria shortest path problem calculated with respect to all the instances solved;
- Lab. is the average number of labels generated to compute the efficient solutions calculated with respect to all the instances solved;
- CPU is the average CPU time of all the instances solved;
- Mem. Mb. is the average memory requirements in mega bytes of all the instances solved;
- Roman numerals I through IV denote the four network models of Section 2.

Tables 2 and 3 present results for MCKPs with three criteria and five to thirty five variables. Among the four models implemented, Model IV has almost always the best CPU times while Model I has almost always the second smallest CPU times and always the smallest memory requirements. Model II seems to be the least attractive with respect to the CPU time and required memory.

Model IV was applied to 3-criteria problems with 10, 20, and 40 variables and the results are shown in Table 4. Note the exponential increase of the CPU time and memory requirements due to the increase of the number of variables.

Tables 5 and 6 report results for MCKPs with four, five, six, and seven criteria and five to twenty variables. Again, model IV (almost always) outperforms the other models

as far as the CPU time while model I always uses the smallest amount of memory. When using model IV as the number of criteria increases, the savings in CPU time seem to be relatively bigger than the losses due to the required memory.

The presented results reveal differences between the four network models and also give the evidence that this algorithmic approach is quite effective. Since our experiment covers instances with up to 7 criteria and twenty variables, we believe that these are the best numerical results for integer MCKPs ever solved with an exact algorithm. Based on the analysis above, we make the conjecture that Model IV is currently the **best** exact approach to solving MCKPs.

As we recognized in Section 1, many computational results with metaheuristic algorithms have been reported in the literature. However, binary rather than integer instances of the MCKP have been researched [3, 11, 15, 12, 18, 28]. Furthermore, since metaheuristics produce approximate rather than true efficient solutions, the focus has been on the effectiveness of those algorithms which is reflected in the quality of the obtained solutions. Also, for example in [15], binary MCKPs with no more than five criteria were tested.

5 Conclusions

In this paper, we developed a comprehensive algorithmic framework for solving the integer MCKP. Based on the dynamic programming models available in the litertaure, we proposed four algorithms producing network models of the MCKP and a generic labeling algorithm that computes all nondominated outcomes of this problem.

The algorithms were implemented and tested on randomly generated problems with up to forty variables or seven criteria. Overall, two different models (I and IV) have the smallest memory requirements and the fastest CPU times. In particular, Model IV offers relatively bigger savings of CPU times in comparison to its increasing memory requirements. We are not aware of another computational study with exact algorithms applied to more complex instances of the integer MCKP with better numerical results. We therefore claim that our results currently determine the state of the art in exact algorithms for the MCKP.

We see two directions of further research. First, as shown by Klamroth and Wiecek [16], the dynamic programming based models of the integer MCKP can handle problems with binary variables, bounded variables, multiple constraints, multiple periods, and time-dependent objective functions. The networks presented in this paper can also be extended to accommodate all these cases while the generic labeling algorithm remains applicable. The proposed framework therefore covers a family of multiple criteria knapsack problems suitable to model a variety of real-life decision making situations.

Second, a joint study of deterministic and metaheuristic algorithms for MCKPs should be performed to evaluate both approaches on a common platform and to better understand pros and cons of each of them.

Acknowledgements This work has benefited from the FLAD (Fundação Luso-Americana para o Desenvolvimento) grant 640/2000. The first author was also supported by the grant SFRH/BDP/6800/2001 (Fundação para a Ciência e Tecnologia, Portugal) and MONET research project POCTI/GES/37707 /2001.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] T. Arndt and J. Seelaender. A method for solving a generalized knapsack problem with multiple objectives. In Seelaender J. Goepfert, A. and Ch. Tammer, editors, *Methods of Multicriteria Decision Theory*, pages 163–173. Haensel-Hohenhausen, Frankfurt, 1997.

[3] V. Barichard and J.-K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Tsinghua Science and Technology*, 8(1):8–13, 2003.

[4] F. Ben Abdelaziz, S. Krichen, and J. Chaouadi. A hybrid heuristic for multiobjective knapsack problems. In I. Osman S. Voss, S. Martello and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 205–212. Kluwer Academic Publishers, Dordrecht, 1999.

[5] K. Bhaskar. A multiple objective approach to capital budgeting. *Accounting and Business Research*, 9:25–46, 1979.

[6] M. E. Captivo, J. Clímaco, J. Figueira, E. Martins, and J.L. Santos. Solving bicriteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30:1865–1886, 2003.

[7] G. Colombo and Ch. Mumford. Comparing algorithms, representations and operators for the multi-objective knapsack problem. In *Proceedings of 2005 IEEE Congress on Evolutionary Computation*, pages 1268–1275, 2005.

[8] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

[9] X. Gandibleux, N. Mezdaoui, and A. Freville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In Ruiz F. Caballero, R. and Steuer E., editors, *Advances in Multiple Objective and Goal Programming*, pages 292–300. Springer-Verlag, 1997.

[10] R. S. Garfinkel and G. L. Nemhauser. *Integer Pogramming*. John Wiley & Sons, New York, 1972.

[11] C. Grosan and M. Oltean. Improving the performance of evolutionary algorithms for the multiobjective 0/1 knapsack problem using ε-dominance. In *Lecture Notes in Computer Science; Computational Science - ICCS 2004*, volume 3037, pages 674–677. Springer, Berlin, 2004.

[12] X.-P. Guo, Z.-M. Wu, and G.-K. Yang. A hybrid adaptive multi-objective memetic algorithm for0/1 knapsack problem. In *Lecture Notes in Computer Science; AI 2005: Advances in Artificial Intelligence - 18th Australian Joint Conference on Artificial Intelligence, Proceedings*, volume 3809, pages 176–185. Springer-Verlag, 2005.

[13] M Hansen. Solving multiobjective knapsack problems using MOTS. Conference Paper presented at MIC'97, Sophia, Antipolis, France, July, 1997.

[14] T. Ibaraki. Enumerative approaches to combinatorial optimization, part ii. *Annals of Operations Research*, 11:343–602, 1987.

[15] A. Jaszkiewicz. On the computational efficiency of multiple objective metaheuristics. The knapsack problem case study. *European Journal of Operational Research*, 158:418–433, 2004.

[16] K. Klamroth and M. M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Reserach Logistics*, pages 57–76, 2000.

[17] M. M. Kostreva, W. Ogryczak, and D. W. Tonkyn. Relocation problems arising in conservation biology. *Computers and Mathematics with Applications*, 37:135–150, 1999.

[18] R. Kumar and N. Banerjee. Analysis of a multiobjective evolutionary algorithm on the 0/1 knapsack problem. *Theoretical Computer Science*, 358(1):104–120, 2006.

[19] W. Kwak, Y. Shi, H. Lee, and C. F. Lee. Capital budgeting with multiple criteria and multiple decision makers. *Review of Quantitative Finance and Accounting*, 7:97–112, 1996.

[20] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing*, 3(1):37–51, 2004.

[21] P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, pages 222–232. Springer-Verlag, Berlin, 1986.

[22] J.-Y. Teng and G.-H. Tzeng. A multiobjective programming approach for selecting non-independent transportation investment alternatives. *Transportation Research - B*, 30:291–307, 1996.

[23] E.L. Ulungu and J. Teghem. Solving multi-objective knapsack problems by a branch-and-bound procedure. In J.N. Climaco, editor, *Multicriteria Analysis*, pages 269–278. Springer-Verlag, 1997.

[24] Teghem J. Fortemps P. H. Ulungu, E. L. and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8:221–236, 1999.

[25] R. Vetschera. Time preferences in capital budgeting - an application of interactive multiobjective optimization. *Methods of Operations Research*, 50:649–660, 1985.

[26] B. Villarreal and M. H. Karwan. Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. *Mathematical Programming*, 21:204–223, 1981.

[27] B. Villarreal and M. H. Karwan. Multicriteria dynamic programming with an application to the integer case. *Journal of Mathematical Analysis and Applications*, 38:43–69, 1982.

[28] B. Zhang, X.-H. Zhang, M.-G. Gong, and B. Lu. Immune system strength Pareto algorithm for multiobjective 0/1 knapsack problems. *Journal of Harbin Engineering University*, 27:214–218, 2006.

[29] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Appendix

Table 2: Average results for uncorrelated instances with $r = 3$ and $w_j, v_j^k \in [1, UB]$ and $W = \lfloor \frac{50}{100} \sum_{j=1}^{l} w_j \rfloor$.

UB	l	N	$\bar{W}$	$\lvert\bar{X}^e\rvert$	$\lvert\bar{Y}^e\rvert$	St. Dev. $\bar{\sigma}$	$\bar{n}$ I	$\bar{n}$ II	$\bar{n}$ III	$\bar{n}$ IV	$\bar{m}$ I	$\bar{m}$ II	$\bar{m}$ III	$\bar{m}$ IV
	5	30	1,262.13	6.27	6.27	8.46	181.53	274.80	704.97	704.97	606.97	570.70	1,037.87	1,005.93
1000	10	30	2,514.20	30.83	30.83	31.84	1,667.53	8,257.17	12,603.40	12,603.40	13,378.27	29,708.87	30,153.33	20,951.67
	15	29	3,741.83	157.86	157.86	537.61	3,276.55	35,862.31	43,223.72	43,223.72	44,925.41	238,675.90	167,882.86	79,249.69
	20	28	5,055.00	125.29	125.29	99.94	4,692.32	75,680.61	85,687.18	85,687.18	88,407.64	709,120.14	498,057.69	161,592.57
	5	30	128.80	4.60	4.60	6.31	49.87	81.10	191.77	191.77	159.67	179.67	324.23	288.37
	10	30	254.20	14.93	14.93	25.67	221.90	1,423.40	1,912.63	1,912.63	1,921.27	6,122.90	5,455.90	3,399.93
	15	30	386.30	43.33	43.33	67.18	365.87	4,231.53	4,989.57	4,989.57	5,065.50	29,963.13	24,908.93	9,331.93
100	20	30	514.87	137.10	125.00	243.85	498.63	8,246.70	9,261.83	9,261.83	9,420.53	79,613.63	66,373.70	17,670.77
	25	30	637.73	339.17	301.90	618.83	627.03	13,639.67	14,903.87	14,903.87	15,001.50	167,675.87	147,509.77	28,766.17
	30	27	755.25	369.68	360.68	705.79	747.57	19,998.93	21,498.68	21,498.71	21,633.14	296,412.11	270,826.43	41,825.36
	35	26	887.93	242.89	232.18	262.72	883.82	28,098.68	29,858.36	29,858.36	30,005.86	487,506.68	462,923.64	58,412.68

Table 3: Average results for uncorrelated instances with $r = 3$ and $w_j, v_j^k \in [1, UB]$ and $W = \lfloor \frac{50}{100} \sum_{j=1}^{l} w_j \rfloor$.

UB	l	Lab. I	Lab. II	Lab. III	Lab. IV	CPU I	CPU II	CPU III	CPU IV	Mem. Mb. I	Mem. Mb. II	Mem. Mb. III	Mem. Mb. IV
	5	104.17	127.60	257.47	213.70	0.01	0.01	0.01	0.01	0.0100	0.0300	0.0200	0.0200
1000	10	3,312.30	8,969.23	7,240.00	5,378.27	0.33	0.45	0.77	0.51	0.1500	0.4900	0.3300	0.2500
	15	22,566.79	115,366.24	61,491.93	44,017.17	13.26	28.04	19.75	11.13	0.8900	4.7200	2.4500	1.7600
	20	45,872.57	350,944.48	169,678.61	116,882.36	39.70	340.79	183.78	43.61	1.7800	13.9200	6.6200	4.5700
	5	50,70	64.03	128.57	104.97	0.00	0.00	0.01	0.01	0.0029	0.0055	0.0068	0.0057
	10	557,07	1,915.80	1,508.60	1,067.93	0.09	0.12	0.21	0.10	0.0236	0.0900	0.0650	0.0461
100	15	2,343.27	13,030.47	7,497.93	5,162.57	1.29	2.39	2.95	1.07	0.0923	0.5308	0.2984	0.2055
	20	6,170.97	43,249.80	20,378.10	14,025.03	7.77	18.74	18.04	5.11	0.2388	1.7048	0.7940	0.5466
	25	19,454.70	186,683.83	76,318.53	52,688.67	129.66	300.40	236.72	42.42	0.7461	7.2017	2.9319	2.0244
	30	26,913.79	314,847.14	126,493.79	87,307.43	338.85	1,386.12	1,150.57	131.52	1.0311	12.1201	4.8499	3.3477
	35	23,874.54	338,630.32	148,432.36	100,663.36	236.09	2,386.12	1,341.43	183.16	0.9156	13.0636	5.6916	3.8603

Table 4: Average results for uncorrelated instances with $r = 3$ and $w_j, v_j^k \in [1, 100]$ and $W = \lfloor \frac{50}{100} \sum_{j=1}^{l} w_j \rfloor$. Solutions for Model IV

l	$N.$	$\bar{W}$	$\bar{n}$	$\bar{m}$	$\lvert\bar{Y}^e\rvert$	St. Dev. $\lvert\bar{\sigma}\rvert$	$\lvert\bar{X}^e\rvert$	Lab.	CPU	Mem. Mb.
10	30	254.20	1,912.63	3,399.93	14.93	25.67	14.93	1,067.93	0.10	0.05
20	30	514.87	9,261.83	17,670.77	125.00	243.85	137.10	14,025.03	5.05	0.55
30	30	759.93	21,628.70	42,062.07	714.40	1,556.46	722.80	147,936.20	737.90	5.66
40	29	1,015.62	39,476.83	77,477.76	836.24	2,573.68	848.03	240,584.93	2,721.32	9.20

Table 5: Average results for uncorrelated instances with $UB = 100$ and $w_j, v_j^k \in [1, UB]$ and $W = \lfloor \frac{50}{100} \sum_{j=1}^{l} w_j \rfloor$.

r	l	N.	$\bar{W}$	$\lvert\bar{X}^e\rvert$	$\lvert\bar{Y}^e\rvert$	St. Dev. $\bar{\sigma}$	$\bar{n}$ I	II	III	IV	$\bar{m}$ I	II	III	IV
	5	30	128.80	6.50	6.50	6.50	49.87	81.10	191.77	191.77	159.67	179.67	324.23	288.37
4	10	30	254.20	30.60	30.60	44.45	221.90	1,423.40	1,912.63	1,912.63	1,921.27	6,122.90	5,455.90	3,399.93
	15	30	386.30	236.27	236.27	557.68	365.87	4,231.53	4,989.57	4,989.57	5,065.50	29,963.13	24,908.93	9,331.93
	20	28	513.46	767.57	560.00	1,526.67	496.54	8,206.46	9,218.29	9,218.29	9,379.29	79,192.36	65,593.82	17,591.50
	5	30	128.80	7.27	7.27	6.69	49.87	81.10	191.77	191.77	159.67	179.67	324.23	288.37
5	10	30	254.20	47.13	47.13	61.03	221.90	1,423.40	1,912.63	1,912.63	1,921.27	6,122.90	5,455.90	3,399.93
	15	30	386.30	331.67	331.67	663.74	365.87	4,231.53	4,989.57	4,989.57	5,065.50	29,963.13	24,908.93	9,331.93
	20	28	513.46	973.18	814.36	1,884.57	496.54	8,206.46	9,218.29	9,218.29	9,379.29	79,192.36	65,593.82	17,591.50
	5	30	128.80	7.73	7.73	6.99	49.87	81.10	191.77	191.77	159.67	179.67	324.23	288.37
6	10	30	254.20	59.50	59.50	78.02	221.90	1,423.40	1,912.63	1,912.63	1,921.27	6,122.90	5,455.90	3,399.93
	15	29	386.30	472.41	472.41	1,081.48	365.28	4,230.21	4,984.97	4,984.97	5,059.41	30,007.31	25,164.97	9,329.10
	20	26	513.46	794.08	794.08	1,693.10	497.50	8,225.81	9,237.65	9,237.65	9,399.23	79,446.08	66,204.00	17,638.08
	5	30	128.80	7.83	7.83	7.00	49.87	81.10	191.77	191.77	159.67	179.67	324.23	288.37
7	10	30	254.20	62.80	62.80	77.48	221.90	1,423.40	1,912.63	1,912.63	1,921.27	6,122.90	5,455.90	3,399.93
	15	29	390.97	614.07	614.07	1,448.29	369.97	4,279.86	5,046.93	5,046.93	5,121.76	30,311.14	24,996.31	9,439.41
	20	25	514.08	1,016.48	1,016.48	2,338.46	497.88	8,255.48	9,268.52	9,268.52	9,406.20	79,899.24	66,479.68	17,699.04

Table 6: Average results for uncorrelated instances with $UB = 100$ and $w_j, v_j^k \in [1, UB]$ and $W = \lfloor \frac{50}{100} \sum_{j=1}^{l} w_j \rfloor$.

r	l	Lab.				CPU				Mem. Mb.			
		I	II	III	IV	I	II	III	IV	I	II	III	IV
	5	54.07	66.93	136.20	111.50	0.00	0.00	0.01	0.00	0.0032	0.0059	0.0076	0.0064
4	10	890.60	2,814.63	2,220.90	1,601.57	0.16	0.17	0.31	0.16	0.0397	0.1349	0.1004	0.0725
	15	7,130.70	34,238.33	18,908.00	13,456.27	13.31	13.92	16.14	7.05	0.3021	1.4703	0.8053	0.5732
	20	16,751.00	110,814.00	47,762.68	34,304.07	76.45	82.38	76.82	37.78	0.7063	4.7046	2.0204	1.4510
	5	54.67	67.13	137.27	111.93	0.00	0.00	0.01	0.01	0.0034	0.0061	0.0082	0.0069
5	10	1,140.47	3,533.80	2,805.90	2,078.73	0.22	0.21	0.37	0.20	0.0545	0.1785	0.1356	0.1005
	15	9,512.33	43,045.80	23,920.40	17,122.57	17.83	16.71	18.03	8.37	0.4383	2.0041	1.1068	0.7923
	20	23,718.36	152,085.71	65,747.82	47.589.21	139.69	139.94	112.83	59.52	1.0892	7.0166	3.0256	2.1900
	5	54.93	67.37	137.73	111.97	0.01	0.00	0.01	0.01	0.0037	0.0064	0.0087	0.0073
6	10	1,376.03	4,075.33	3,254.70	2,429.27	0.29	0.25	0.45	0.25	0.0705	0.2188	0.1685	0.1258
	15	15,123.00	63,531.76	35,885.07	25,551.83	91.56	60.30	70.93	29.73	0.7529	3.1840	1.7913	1.2737
	20	33,022.88	225,779.77	95,856.96	69,232.35	403.59	573.98	548.65	145.38	1.6411	11.2513	4.7695	3.4448
	5	54.93	67.37	137.83	112.03	0.00	0.00	0.01	0.01	0.0039	0.0167	0.0092	0.0077
7	10	1,536.67	4,322.23	3,422.47	2,555.83	0.30	0.27	0.45	0.27	0.0843	0.2475	0.1898	0.1418
	15	16,698.10	73,705.31	40,793.10	29,665.66	74.78	42.00	58.43	23.69	0.8947	3.9702	2.1903	1.5930
	20	33,708.64	211,054.64	90,646.56	65,234.92	260.43	214.85	245.85	79.30	1.8037	11.3262	4.8567	3.4955

Algorithms for constructing network models I-IV

MODEL−I
Input: w, W, l and v^k, for $k = 1, \ldots, r$;
Output: a network $G = (V, A, C)$;
(1) **begin**
(2) $V \leftarrow \{s\}$ and $A \leftarrow \emptyset$;
(3) $u \leftarrow s$; { s is the first vertex to be visited }
(4) **for** $(1 \leq j \leq l)$ **do**
(5) **begin**
(6) $g \leftarrow w_j$;
(7) **if** $(g \notin V)$ **then** $V \leftarrow V \cup \{g\}$;
(8) $A \leftarrow A \cup \{(s,g)\}$;
(9) $c(s,g) \leftarrow [-v_j^1, \ldots, -v_j^r]$;
(10) **end**
(11) **while** (V contains non-visited vertices) **do**
(12) **begin**
(13) Let u denote the next vertex of V to be visited;
(14) **for** $(1 \leq j \leq l)$ **do**
(15) **begin**
(16) $g \leftarrow u + w_j$;
(17) **if** $(g \leq W)$ **then**
(18) **begin**
(19) **if** $(g \notin V)$ **then** $V \leftarrow V \cup \{g\}$;
(20) $A \leftarrow A \cup \{(u,g)\}$;
(21) $c(u,g) \leftarrow [-v_j^1, \ldots, -v_j^r]$;
(22) **end**
(23) **end**
(24) **end**
(25) **for** (all $g \in V$) **do**
(26) **begin**
(27) $A \leftarrow A \cup \{(g,t)\}$;
(28) $c(g,t) \leftarrow [0, \ldots, 0]$;
(29) **end**
(30) $V \leftarrow V \cup \{t\}$;
(31)**end**

Figure 7: *Algorithm for Model I*

MODEL−II
Input: w, W, l and v^k, for $k = 1, \ldots, r$;
Output: a network $G = (V, A, C)$;
(1) **begin**
(2) $V \leftarrow \{s\}$ and $A \leftarrow \emptyset$;
(3) **for** $(1 \leq j \leq l)$ **do**
(4) **begin**
(5) $g \leftarrow w_j$;
(6) $V \leftarrow V \cup \{g^j\}$;
(7) $A \leftarrow A \cup \{(s, g^j)\}$;
(8) $c(s, g^j) \leftarrow [-v_j^1, \ldots, -v_j^r]$;
(9) **end**
(10) **for** $(\min_j\{w_j\} \leq g \leq W)$ **do**
(11) **begin**
(12) **for** $(0 \leq k \leq l$ such that $g^k \in V)$ **do**
(13) **for** $(1 \leq j \leq l)$ **do**
(14) **if** $(g + w_j \leq W)$ **then**
(15) **begin**
(16) **if** $((g + w_j)^j \notin V)$ **then** $V \leftarrow V \cup \{(g + w_j)^j\}$;
(17) $A \leftarrow A \cup \{(g^k, (g + w_j)^j)\}$;
(18) $c(g^k, (g + w_j)^j) \leftarrow [-v_j^1, \ldots, -v_j^r]$;
(19) **end**
(20) **end**
(21) **for** (all $g^k \in V$) **do**
(22) **begin**
(23) $A \leftarrow A \cup \{(g^k, t)\}$;
(24) $c(g^k, t) \leftarrow [0, \ldots, 0]$;
(25) **end**
(26) $V \leftarrow V \cup \{t\}$;
(27)**end**

Figure 8: *Algorithm for Model II*

MODEL−III
Input: w, W, l and v^k, for $k = 1,\ldots,r$;
Output: a network $G = (V,A,C)$;
(1) **begin**
(2) $V \leftarrow \{s\}$ and $A \leftarrow \emptyset$;
(3) **for** $(0 \leq \beta \leq \lfloor W \setminus w_1 \rfloor)$ **do**
(4) **begin**
(5) $k \leftarrow \beta w_1$;
(6) $V \leftarrow V \cup \{1^k\}$;
(7) $A \leftarrow A \cup \{(s,1^k)\}$;
(8) $c(s,1^k) \leftarrow \beta[-v_1^1,\ldots,-v_1^r]$;
(9) **end**
(10) **for** $(1 \leq g < l)$ **do**
(11) **for** ($0 \leq b \leq W$ such that $g^b \in V$) **do**
(12) **begin**
(13) $\beta \leftarrow 0$, $k \leftarrow b$ and $j \leftarrow 1$;
(14) **while** $(\beta \leq \lfloor W \setminus w_j \rfloor)$ **and** $(k \leq W)$ **do**
(15) **begin**
(16) **if** $((g+1)^k \notin V)$ **then** $V \leftarrow V \cup \{(g+1)^k\}$;
(17) $A \leftarrow A \cup \{(g^b,(g+1)^k)\}$;
(18) **if** $(k = b)$ **then** $c(g^k,(g+1)^k) \leftarrow [0,\ldots,0]$;
(19) **else** $c(g^k,(g+1)^k) \leftarrow \beta[-v_j^1,\ldots,-v_j^r]$;
(20) $\beta \leftarrow \beta + 1$ and $j \leftarrow j+1$;
(21) $k \leftarrow b + \beta w_j$;
(22) **end**
(23) **end**
(24) **for** $(0 \leq k \leq W$ such that $l^k \in V)$ **do**
(25) **begin**
(26) $A \leftarrow A \cup \{(l^k,t)\}$;
(27) $c(l^k,t) \leftarrow [0,\ldots,0]$;
(28) **end**
(29) $V \leftarrow V \cup \{t\}$;
(30)**end**

Figure 9: *Algorithm for Model III*

MODEL−IV
Input: w, W, l and v^k, for $k = 1,\ldots,r$;
Output: a network $G = (V,A,C)$;
(1) **begin**
(2) $V \leftarrow \{s\}$ and $A \leftarrow \emptyset$;
(3) **for** $(0 \leq \beta \leq \lfloor W \setminus w_1 \rfloor)$ **do**
(4) **begin**
(5) $k \leftarrow \beta w_1$;
(6) $V \leftarrow V \cup \{1^k\}$;
(7) $A \leftarrow A \cup \{(s,1^k)\}$;
(8) $c(s,1^k) \leftarrow \beta[-v_1^1,\ldots,-v_1^r]$;
(9) **end**
(10) **for** $(1 \leq g \leq l)$ **do**
(11) **for** ($0 \leq b \leq W$ such that $g^b \in V$) **do**
(12) **begin**
(13) **if** $((g+1)^b \notin V)$ **then** $V \leftarrow V \cup \{(g+1)^b\}$;
(14) $A \leftarrow A \cup \{(g^b,(g+1)^b)\}$;
(15) $c(g^k,(g+1)^b) \leftarrow [0,\ldots,0]$;
(16) $k \leftarrow b + w_g$;
(17) **if** $(k \leq W)$ **then**
(18) **begin**
(19) **if** $(g^k \notin V)$ **then** $V \leftarrow V \cup \{g^k\}$;
(20) $c(g^b,g^k) \leftarrow \beta[-v_j^1,\ldots,-v_j^r]$;
(21) $A \leftarrow A \cup \{(g^b,g^k)\}$;
(22) **end**
(23) **end**
(24) **for** $(0 \leq k \leq W$ such that $l^k \in V)$ **do**
(25) **begin**
(26) $A \leftarrow A \cup \{(l^k,t)\}$;
(27) $c(l^k,t) \leftarrow [0,\ldots,0]$;
(28) **end**
(29) $V \leftarrow V \cup \{t\}$;
(30)**end**

Figure 10: *Algorithm for Model IV*