Committee Selection with a Weight Constraint Based on a Pairwise Dominance Relation

Charles Delort, Olivier Spanjaard, Paul Weng

Committee Selection with a Weight Constraint Based on a Pairwise Dominance Relation*

Charles Delort, Olivier Spanjaard, and Paul Weng

UPMC, LIP6-CNRS, UMR 7606
4 Place Jussieu, F-75005 Paris, France
{charles.delort,olivier.spanjaard,paul.weng}@lip6.fr

Abstract. This paper is devoted to a knapsack problem with a cardinality constraint when dropping the assumption of additive representability [10]. More precisely, we assume that we only have a classification of the items into ordered classes. We aim at generating the set of preferred subsets of items, according to a pairwise dominance relation between subsets that naturally extends the ordering relation over classes [4, 16]. We first show that the problem reduces to a multiobjective knapsack problem with cardinality constraint. We then propose two polynomial algorithms to solve it, one based on a multiobjective dynamic programming scheme and the other on a multiobjective branch and bound procedure. We conclude by providing numerical tests to compare both approaches.

Key words: Committee selection; Ordinal combinatorial optimization; Multiobjective combinatorial optimization; Knapsack with cardinality constraint; Polynomial algorithms

1 Introduction

Ranking sets of objects based on a ranking relation on objects has been extensively studied in social choice theory within an axiomatic approach [1]. Many *extension rules* have been proposed and axiomatically justified to extend an order relation over a set of objects to an order relation over its power set. This issue is indeed of primary interest in various fields such as choice under uncertainty [12], ranking opportunity sets [3], and of course committee selection [11]. The committee selection problem consists in choosing a subset of inviduals based on an ordering of individuals. Although a lot of works deal with this problem in the economic literature, it has received much less attention from the algorithmic viewpoint. In other words, the computational aspect (i.e., the effective calculability of the preferred committees) is often a secondary issue. This is precisely the issue we study in this paper.

More formally, we investigate the problem of selecting K individuals (or more generally objects) among n with budget B, where the selection of individual i

* This research has been supported by the project ANR-09-BLAN-0361 GUaranteed Efficiency for PAReto optimal solutions Determination (GUEPARD).

requires a cost w^i. The only preferential information is an assignment of each individual i in a preference class $\gamma^i \in \{1, \ldots, C\}$, with $1 \rhd 2 \rhd \ldots \rhd C$, where $\rhd$ means "is strictly preferred to". For illustration, consider the following example. Assume that an English soccer team wishes to recruit $K = 2$ players with budget $B = 6$. The set $\mathcal{N}$ of available players consists of international players (class 1), Premier League players (class 2) and Division 1 players (class 3). This problem can be modeled as a knapsack problem where one seeks a subset $\mathcal{S} \subseteq \mathcal{N}$ such that $\sum_{i \in \mathcal{S}} w^i \leq 6$ and $|\mathcal{S}| = 2$, but where the objective function is not explicited. Consider now the following instance: $\mathcal{N} = \{1, 2, 3, 4\}$, $w^1 = 5$, $w^2 = 2$, $w^3 = 4$, $w^4 = 1$. Player 1 is international, players 2, 3 are from the Premier League, and player 4 is from the Division 1 championship: $\gamma^1 = 1$, $\gamma^2 = \gamma^3 = 2$, $\gamma^4 = 3$.

When the individuals are evaluated in this way (i.e., on an ordinal scale), arbitrarily assigning numerical values to classes (each class can be viewed as a grade in the scale) introduces a bias in the modeling [2]. For instance, if value 8 is assigned to class 1, value 4 is assigned to class 2 and value 1 to class 3, then the ensuing recruitment choice (the one maximizing the sum of the value according to the budget) is $\{1, 4\}$. By valuing class 2 by 5 instead of 4 (which is still compatible with the ordinal classes), the ensuing recruitment choice becomes $\{2, 3\}$. Thus, one observes that slight changes in the numerical values lead to very different choices. This illustrates the need for algorithms specifically dedicated to combinatorial problems with ordinal measurement.

This problem has been studied in a slightly different setting by Klamler *et al.* [14]. They assume to have a preference relation over the set of individuals, expressed as a reflexive, complete and transitive binary relation. Note that, in our setting with C predefined preference classes, it amounts to set $C = n$ (some preference classes may be empty if there are equivalent individuals). The authors provide linear time algorithms to compute optimal committees according to various extension rules, namely variations of max ordering, leximax and leximin. The max (resp. min) ordering relation consists in ranking committees according to the best (resp. worst) individual they include, while the leximax and leximin relations are enrichments of max and min respectively that consist in breaking ties by going down the ranking (e.g., if the best individuals are indifferent, one compares the second bests, and so on...). Though appealing from the algorithmic viewpoint, these extension rules are nevertheless quite simple from the normative and descriptive viewpoints.

In this paper, we investigate an extension rule that encompasses a much larger set of decision behaviors (at the expense of working with preference classes instead of a complete ranking of individuals). Actually, it leads to identify a set of preferred committees, instead of a single one. Provided the ordinal nature of data, it seems indeed relevant to determine a set of acceptable committees, among which the final choice will be made. In order to extend order relation $\rhd$ over the preference classes (1, 2, 3 in the recruitment example, with $1 \rhd 2 \rhd 3$) to a (reflexive and transitive) preference relation $\succsim$ over the committees, the extension rule we study is the following pairwise dominance relation: a committee $\mathcal{S}$ is preferred to another committee $\mathcal{S}'$ if, to each individual i in $\mathcal{S}'$, one can assign

a not-yet-assigned individual i' in $\mathcal{S}$ such that $\gamma^{i'} \trianglerighteq \gamma^i$ (i.e. $\gamma^{i'} \rhd \gamma^i$ or $\gamma^{i'} = \gamma^i$). For instance, in the previous recruiting example, one has $\{1,3\} \succsim \{2,3\}$ since $\gamma^3 = \gamma^2$ and $\gamma^1 \rhd \gamma^3$ (note that $\{1,3\}$ is actually not feasible, due to the budget constraint, but it does not matter for our purposes). To our knowledge, this extension rule was proposed by Bossong and Schweigert [4, 16]. More recent works with ordinal data also use this rule [5–7].

Our first contribution in the present paper is to relate ordinal combinatorial optimization to multiobjective combinatorial optimization, by reducing the determination of the non-dominated solutions in an ordinal problem to the determination of the Pareto set in an appropriately defined corresponding multiobjective problem. We then propose two algorithms to determine a set of optimal committees according to the pairwise dominance relation, one based on a multiobjective dynamic programming scheme and the other one on a multiobjective branch and bound procedure. The complexity of both procedures is polynomial *for a fixed number C of preference classes*. Note that in another context, Della Croce et al. [8] also represented an ordinal optimization problem as a multiobjective problem, but their transformation is different from the one presented here.

The paper is organized as follows. Section 2 relates ordinal optimization to multiobjective optimization. Two polynomial (multiobjective) procedures to solve the commitee selection problem are then presented in Sections 3 and 4. Finally, experimental results are provided in Section 5 to compare both approaches.

2 From Ordinal Combinatorial Optimization to Multiobjective Optimization

Formally, an ordinal combinatorial optimization problem can be defined as follows. Consider a set $\mathcal{N}$ of objects (e.g. items in a knapsack problem, edges in a path or tree problem...). A feasible solution is a subset $\mathcal{S} \subseteq \mathcal{N}$ satisfying a given property (for example, satisfying knapsack constraints). As mentioned in the introduction, for each object $i \in \mathcal{N}$, the only preferential information at our disposal is the preference class $\gamma^i \in \{1, \ldots, C\}$ it belongs to, with $1 \rhd 2 \rhd \ldots \rhd C$. Given an extension rule that lifts preference relation $\rhd$ to a preference relation $\succsim$ over subsets of $\mathcal{N}$, a feasible solution $\mathcal{S}$ is said to be *preferred* if there exists no feasible solution $\mathcal{S}'$ such that $\mathcal{S}' \succ \mathcal{S}$, where $\succ$ denotes the asymmetric part of $\succsim$. The aim of an ordinal combinatorial optimization problem is then to find a *complete minimal* set of preferred solutions [13]. A set of solutions is said to be *complete* if for any preferred solution, there is a solution in that set that is indifferent to it. A set of solutions is said to be *minimal* if there does not exist a pair $\mathcal{S}, \mathcal{S}'$ of solutions in this set such that $\mathcal{S} \neq \mathcal{S}'$ and $\mathcal{S} \succsim \mathcal{S}'$.

Let us denote by $\max_{\succsim}$ the operation that consists in determining a complete minimal set of preferred solutions according to $\succsim$. The committee selection problem we consider in this paper can then be simply stated as follows:

$$\max_{\succsim}\{\mathcal{S} \subseteq \mathcal{N} \ : \ |\mathcal{S}| = K \text{ and } \sum_{i \in \mathcal{S}} w^i \leq B\}$$

where K is the size of the committee and w^i the cost of selecting individual i.

In the sequel, we consider the following extension rule:

Definition 1. *The* pairwise dominance relation *between subsets of a set $\mathcal{N}$ is defined, for all $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{N}$, by $\mathcal{S} \succsim \mathcal{S}'$ if there exists an injection $\pi : \mathcal{S}' \to \mathcal{S}$ such that $\forall i \in \mathcal{S}'$, $\gamma^{\pi(i)} \trianglerighteq \gamma^i$.*

Coming back to the example of the introduction, one detects that $\{1,3\} \succsim \{2,3\}$ by setting $\pi(2) = 1$ ($\gamma^1 = 1 \triangleright 2 = \gamma^2$) and $\pi(3) = 3$, or by setting $\pi(2) = 3$ ($\gamma^2 = \gamma^3 = 2$) and $\pi(3) = 1$ ($\gamma^1 = 1 \triangleright 2 = \gamma^3$). Since the opposite relation is not true, one has $\{1,3\} \succ \{2,3\}$.

We are now going to make an original link between ordinal optimization and multiobjective optimization. In this purpose, the following notion will prove useful: for each solution $\mathcal{S}$ and each preference class $c \leq C$, one defines $\mathcal{S}_c = \{i \in \mathcal{S} : \gamma^i \trianglerighteq c\}$. To each solution one associates a *cumulative* vector $(|\mathcal{S}_1|, \ldots, |\mathcal{S}_C|)$. Therefore, one has $|\mathcal{S}_1| \leq |\mathcal{S}_2| \leq \ldots \leq |\mathcal{S}_C|$. Interestingly enough, we now show that comparing solutions according to pairwise dominance amounts to compare those vectors according to weak (Pareto) dominance, which is defined as follows:

Definition 2. *The* weak dominance *relation on C-vectors of $\mathbb{N}^C$ is defined, for all $y, y' \in \mathbb{N}^C$, by $y \succcurlyeq y' \Leftrightarrow [\forall c \in \{1, \ldots, C\}, y_c \geq y'_c)]$. The* dominance *relation $\succ$ is defined as the asymmetric part of $\succcurlyeq$: $y \succ y' \Leftrightarrow [y \succcurlyeq y'$ and $y' \not\succcurlyeq y]$.*

The equivalence result writes formally as follows:

Proposition 1. *For any pair $\mathcal{S}, \mathcal{S}'$ of solutions, we have:*
$$\mathcal{S} \succsim \mathcal{S}' \iff (|\mathcal{S}_1|, \ldots, |\mathcal{S}_C|) \succcurlyeq (|\mathcal{S}'_1|, \ldots, |\mathcal{S}'_C|)$$

Proof. We first prove that $\mathcal{S} \succsim \mathcal{S}' \Rightarrow (|\mathcal{S}_1|, \ldots, |\mathcal{S}_C|) \succcurlyeq (|\mathcal{S}'_1|, \ldots, |\mathcal{S}'_C|)$. Assume there exists an injection $\pi : \mathcal{S}' \to \mathcal{S}$. Then $|\mathcal{S}_c| \geq |\pi(\mathcal{S}'_c)| = |\mathcal{S}'_c|$ for all c, since $\gamma^{\pi(i)} \trianglerighteq \gamma^i \trianglerighteq c$ for all $i = 1, \ldots, n$. Therefore $(|\mathcal{S}_1|, \ldots, |\mathcal{S}_C|) \succcurlyeq (|\mathcal{S}'_1|, \ldots, |\mathcal{S}'_C|)$ by definition of $\succcurlyeq$.

Conversely, we now show that $(|\mathcal{S}_1|, \ldots, |\mathcal{S}_C|) \succcurlyeq (|\mathcal{S}'_1|, \ldots, |\mathcal{S}'_C|) \Rightarrow \mathcal{S} \succsim \mathcal{S}'$. Assume that $|\mathcal{S}_c| \geq |\mathcal{S}'_c|$ for all c. Since $|\mathcal{S}_1| \geq |\mathcal{S}'_1|$, there exists an injection $\pi_1 : \mathcal{S}'_1 \to \mathcal{S}_1$. Obviously, $\forall i \in \mathcal{S}'_1, \gamma^{\pi_1(i)} \trianglerighteq \gamma^i$. For any $c > 1$, one can then define by mutual recursion:

- an injection $\pi'_c : \mathcal{S}'_c \backslash \mathcal{S}'_{c-1} \to \mathcal{S}_c \backslash \pi_{c-1}(\mathcal{S}'_{c-1})$
- an injection $\pi_c : \mathcal{S}'_c \to \mathcal{S}_c$ by $\pi_c(i) = \pi_{c-1}(i)$ if $i \in \mathcal{S}'_{c-1}$ and $\pi_c(i) = \pi'_c(i)$ otherwise.

Injection π'_c exists for any $c > 1$ because $|\mathcal{S}_c \backslash \pi_{c-1}(\mathcal{S}'_{c-1})| \geq |\mathcal{S}'_c \backslash \mathcal{S}'_{c-1}|$. We have indeed $|\mathcal{S}_c \backslash \pi_{c-1}(\mathcal{S}'_{c-1})| = |\mathcal{S}_c| - |\pi_{c-1}(\mathcal{S}'_{c-1})|$ since $\pi_{c-1}(\mathcal{S}'_{c-1}) \subseteq \mathcal{S}_c$, $|\mathcal{S}_c| - |\pi_{c-1}(\mathcal{S}'_{c-1})| = |\mathcal{S}_c| - |\mathcal{S}'_{c-1}|$ since π_{c-1} is an injection, $|\mathcal{S}_c| - |\mathcal{S}'_{c-1}| \geq |\mathcal{S}'_c \backslash \mathcal{S}'_{c-1}|$ since $|\mathcal{S}_c| \geq |\mathcal{S}'_c|$. Note that by construction, for any c, $\forall i \in \mathcal{S}'_c$, $\gamma^{\pi_c(i)} \trianglerighteq \gamma^i$. For $c = C$ this is precisely the definition, therefore $\mathcal{S} \succsim \mathcal{S}'$. ■

Coming back again to the example of the introduction, cumulative vector $(1,2,2)$ is associated to $\{1,3\}$, and $(0,2,2)$ to $\{2,3\}$. Note then, that $(1,2,2) \succ (0,2,2)$, consistently with $\{1,3\} \succ \{2,3\}$.

The committee selection problem we consider in this paper can then be formulated as a multiobjective knapsack problem with a cardinality constraint. An instance of this problem consists of a knapsack of integer capacity B, and a set of items $\mathcal{N} = \{1, \ldots, n\}$. Each item i has a weight w^i and a profit $p^i = (p^i_1, \ldots, p^i_C)$, variables w^i, p^i_c ($c \in \{1, \ldots, C\}$) being integers. Without loss of generality, we assume from now on that items in $\mathcal{N}$ are such that $\gamma^1 \trianglerighteq \gamma^2 \trianglerighteq \cdots \trianglerighteq \gamma^n$ and $\forall i, i' \in \mathcal{N}$, $\left(\gamma^i = \gamma^{i'} \text{ and } i \leq i'\right) \Rightarrow w^i \leq w^{i'}$ (i.e. the items of $\mathcal{N}$ are indexed in decreasing order of preference classes and in increasing order of weights in case of ties). Otherwise, one can renumber the items.

Consequently, the profit vector of item i is defined by $p^i_c = 0$ for $c < \gamma^i$, and $p^i_c = 1$ for $c \geq \gamma^i$. This way, summing up the profit vectors of the items in a solution $\mathcal{S}$ yields the cumulative vector of $\mathcal{S}$. A solution $\mathcal{S}$ is characterized by a binary n-vector x, where $x^i = 1$ iff $i \in \mathcal{S}$. A solution is feasible if binary vector x satisfies the constraints $\sum_{i=1}^n w^i x^i \leq B$ and $\sum_{i=1}^n x^i = K$. The goal of the problem is to find a complete minimal set of feasible solutions (i.e. one feasible solution by non-dominated cumulative vector), which can be formally stated as follows:

$$\begin{aligned}
&\text{maximize} && \sum_{i=1}^{n} p^i_c x^i && c \in \{1, \ldots, C\} \\
&\text{subject to} && \sum_{i=1}^{n} w^i x^i \leq B && \\
& && \textstyle\sum_{i=1}^{n} x^i = K && \\
& && x^i \in \{0, 1\} && i \in \{1, \ldots, n\}
\end{aligned}$$

Note that, since vectors p^i are non-decreasing (i.e. $p^i_1 \leq \ldots \leq p^i_C$), the image of all feasible solutions is a subset of $[\![0, K]\!]^C_{\uparrow}$, which denotes the set of non-decreasing vectors in $[\![0, K]\!]^C = \{0, \ldots, K\}^C$. Furthermore, one has $|\mathcal{S}_C| = K$ for any feasible solution $\mathcal{S}$.

Example 1 *The example of the introduction is formalized as follows:*

$$\begin{aligned}
&\text{maximize} && x_1 \\
&\text{maximize} && x_1 + x_2 + x_3 \\
&\text{maximize} && x_1 + x_2 + x_3 + x_4 \\
&\text{subject to} && 5x_1 + 2x_2 + 4x_3 + x_4 \leq 6 \\
& && x_1 + x_2 + x_3 + x_4 = 2 \\
& && x^i \in \{0, 1\} \quad i \in \{1, \ldots, 4\}
\end{aligned}$$

3 A Multiobjective Dynamic Programming Algorithm

Multiobjective dynamic programming is a well-known approach to solve multiobjective knapsack problems [15]. In this section, we will present an algorithm proposed by Erlebach et al. [9], and apply it to our committee selection problem. The method is a generalization of the dynamic programming approach for the single objective knapsack problem using the following recursion:

$$W[p + p^i, i] = \min\{W[p + p^i, i-1], W[p, i-1] + w^i\} \text{ for } i = 1, \ldots, n$$

where $W[p, i]$ is the minimal weight for a subset of items in $\{1, \ldots, i\}$ with profit p. The recursion is initialized by setting $W[0, 0] = 0$ and $W[p, 0] = B + 1$ for all $p \geq 1$. The formula can be explained as follows. To compute $W[p + p^i, i]$, one compares the minimal weight for a subset of $\{1, \ldots, i\}$ with profit $p + p^i$ that *does not* include item i, and the minimal weight for a subset of $\{1, \ldots, i\}$ with profit $p + p^i$ that *does* include item i.

In a multiobjective setting, the difference lies in the profits, which are now vectors instead of scalars. Nevertheless, the dynamic programming procedure works in a similar way, by using the following recursion:

$$W[(p_1 + p_1^i, \ldots, p_C + p_C^i), i] = \min \begin{cases} W[(p_1 + p_1^i, \ldots, p_C + p_C^i), i-1] \\ W[(p_1, \ldots, p_C), i-1] + w^i \end{cases}$$

for $i = 1, \ldots, n$. The recursion is initialized by setting $W[(0, \ldots, 0), 0] = 0$ and $W[p, 0] = B + 1$ for all $p \neq (0, \ldots, 0)$. Once column $W[\cdot, n]$ is computed, the preferred items can then be identified in two steps:

1. one identifies profit vectors p for which $W[p, n] \leq B$;
2. one extracts the non-dominated elements among them.

The corresponding preferred solutions can then be retrieved by using standard bookkeeping techniques.

We adapt this method as follows to fit the committee selection problem, where one has to take into account cardinality constraint $\sum_{i=1}^{n} x^i = K$ and where $(p_1, \ldots, p_C) \in [\![0, K]\!]_{\uparrow}^{C}$. In step 1 above, one identifies profit vectors p for which $W[p, n] \leq B$ and $p_C = K$. This latter condition amounts to check that the cardinality of the corresponding solution is K: all items are indeed of preference class at least C (in other words, $p_C^i = 1$ for $i \in \{1, \ldots, n\}$).

Example 2 *For the instance of Example 1, the dynamic programming procedure can be seen as filling the cells of Table 1.*

Table 1. Dynamic programming table for Example 1. Each cell is computed by using the recursion $W[p + p^i, i] = \min\{W[p + p^i, i-1], W[p, i-1] + w^i\}$. For instance, the dark gray cell is computed from the light gray cells.

p \ i	1	2	3	4
$(0,0,0)$	0	0	0	0
$(0,0,1)$	7	7	7	$\min(7, 0+1) = 1$
$(0,0,2)$	7	7	7	7
$(0,1,1)$	7	$\min(7, 0+2) = 2$	2	2
$(0,1,2)$	7	7	7	3
$(0,2,2)$	7	7	$\min(7, 2+4) = 6$	6
$(1,1,1)$	$\min(7, 0+5) = 5$	5	5	5
$(1,1,2)$	7	7	7	$\min(7, 5+1) = 6$
$(1,2,2)$	7	7	7	7
$(2,2,2)$	7	7	7	7

In order to determine the complexity of this procedure, we assume that the number C of preference classes is fixed. At each step of the recursion, the computations required to compute one cell of the dynamic programming table are performed in constant time since it simply consists in a min operation. Furthermore, the number of steps is also polynomial since the number of rows (resp. columns) is within $\Theta(K^C)$ (resp. n). There are indeed as many rows as the number of vectors in $[\![0, K]\!]^C_\uparrow$. The cardinality of $[\![0, K]\!]^C_\uparrow$ is upper bounded by K^C (the cardinality of $[\![0, K]\!]^C$) and lower bounded by $K^C/C!$ (since there are at most $C!$ distinct vectors in $[\![0, K]\!]^C$ which are permutations of a same vector in $[\![0, K]\!]^C_\uparrow$), and therefore the number of rows is within $\Theta(K^C)$. Finally, the identification of preferred items can of course also be done in polynomial time as the number of cells in column $W[\cdot, n]$ is within $\Theta(K^C)$.

To summarize, the time complexity of the procedure is polynomial for a fixed number C of preference classes, and the dynamic programming table has $\Theta(nK^C)$ cells. Regarding the spatial complexity, note that one only needs to keep one column at each step to perform the recursion, and therefore it is in $\Theta(K^C)$.

4 A Multiobjective Branch and Bound Algorithm

4.1 Principle

A classical branch and bound algorithm (BB) explores an enumeration tree whose leaves represent a set of *possibly optimal* solutions (i.e., it is not required that the leaves represent the set of *all feasible* solutions, provided it is guaranteed that at least one optimal solution is present). One can distinguish two main parts in this type of procedure: the *branching part* describing how the set of solutions associated with a node of the tree is separated into subsets, and the *bounding part* describing how the quality of the current subset of solutions is optimistically evaluated. The complete enumeration of the children of a node can be avoided when its optimistic evaluation is worse than the best solution found so far.

A multiobjective BB (MOBB) is an extension of the classical BB. The branching scheme now must be able to enumerate a complete set of feasible solutions in an enumeration tree (i.e., at least one solution must be present in the leaves for each Pareto point in the objective space). In the bounding part, the optimistic evaluation is a vector and the enumeration is stopped when the optimistic evaluation of a node is dominated by an already found non-dominated solution.

4.2 Branching Part

Let us introduce a new notation. For any pair of classes c, c', let $\mathcal{N}_{c,c'} = \{i \in \mathcal{N} : c \trianglerighteq \gamma^i \trianglerighteq c'\}$ be the set of items whose classes are between classes c and c'. Set $\mathcal{N}_{1,c}$ will be denoted by $\mathcal{N}_c$.

Our multiobjective branch and bound approach for the committee selection problem relies on the following property:

Proposition 2. *For any feasible profit vector* $p = (p_1, \ldots, p_C)$*, solution* $x = (x^1, \ldots, x^n)$ *defined by*

- $x^i = 1$, $\forall i = 1, \ldots, p_1$
- $x^i = 0$, $\forall i = p_1 + 1, \ldots, |\mathcal{N}_1|$
- $x^i = 1$, $\forall i = |\mathcal{N}_{c-1}| + 1, \ldots, |\mathcal{N}_{c-1}| + p_c - p_{c-1} \quad \forall c = 2, \ldots, C$
- $x^i = 0$, $\forall i = |\mathcal{N}_{c-1}| + p_c - p_{c-1} + 1, \ldots, |\mathcal{N}_c| \quad \forall c = 2, \ldots, C$

is a minimal weight feasible solution for this profit vector.

Proof. We recall that the lower the index, the better the class, and within each class, the lower the index, the lighter the item. We first show by induction that solution x yields profit vector p. Clearly, solution x admits p_1 items of class 1 and therefore its value on component 1 is p_1. Assume now that the value of x on component $c-1$ is p_{c-1} (induction hypothesis). Then its value on component c is by construction $p_c = p_c - p_{c-1} + p_{c-1}$ since $p_c - p_{c-1}$ items of class c are selected. Solution x yields therefore profit vector $(p_1, \ldots, p_C)$.

By noting that at each step one selects the lightest items of each class, one concludes that x is a minimal weight feasible solution for profit vector p. ■

This observation justifies that we focus on feasible solutions of this type. Now, the branching scheme can be simply explained. Let $P(k, c, p, b)$ denote the subproblem where one wants to select k items whose total weight is less than budget b, where the remaining items are classified in classes $(c, \ldots, C)$ and the profit vector of the already selected items is $p \in [\![0, K]\!]^C_{\uparrow}$. The initial problem is then denoted by $P(K, 1, (0, \ldots, 0), B)$. A node in the enumeration tree represents a problem $P(k, c, p, b)$ where $p = (p_1, \ldots, p_C)$ accounts for the items selected in the previous steps. Such a problem can be subdivided into at most $k + 1$ subproblems $P(k', c + 1, p', b')$ for $k' = 0, \ldots, \min\{k, |\mathcal{N}_{c,c}|\}$, where branching consists in deciding to select exactly $k - k'$ items in class c (the ones with the lowest weights in class c), and p', b' are the updated profit vector and budget to take into account these newly selected items.

Note that in some cases, some subproblems have an empty set of feasible solutions due to the budget constraint, and are therefore discarded. For illustration, the enumeration tree for Example 1 is provided in Figure 1. The vector in a node is the current value of p, and each branch is labelled by the selected items at this step. The dashed node (on the right) is discarded due to the budget constraint, and the gray nodes correspond to non-dominated solutions.

4.3 Bounding Part

For a problem $P(k, c, p, b)$ and a preference class c' such that $c \trianglerighteq c'$, the optimistic evaluation UB of the corresponding node in the enumeration tree is defined by:

$$UB = (m_1, \ldots, m_C) \text{ where } \begin{cases} m_{c'} = p_{c'} & \forall c' = 1, \ldots, c-1 \\ m_{c'} = m_{c,c'} & \forall c' = c, \ldots, C \end{cases} \quad (1)$$

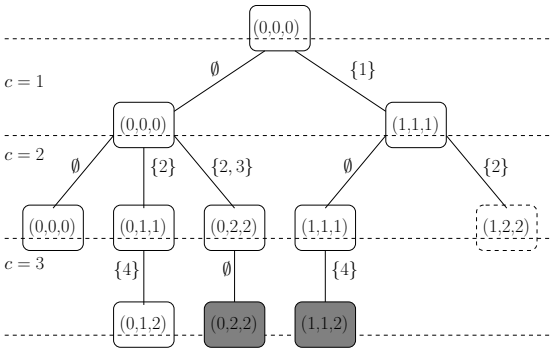

Fig. 1. Enumeration tree for Example 1.

and where $m_{c,c'}$ is defined by:

$$m_{c,c'} = \begin{cases} \max \displaystyle\sum_{i \in \mathcal{N}_{c,c'}} x_i \\ \text{s.t.} \displaystyle\sum_{i \in \mathcal{N}_{c,c'}} w_i x_i \leq b \qquad \displaystyle\sum_{i \in \mathcal{N}_{c,c'}} x_i \leq k \\ x_i \in \{0,1\} \quad \forall i \in \mathcal{N}_{c,c'} \end{cases}$$

Note that the above program can be very simply solved by a greedy algorithm.

The following proposition states that UB is indeed an optimistic evaluation:

Proposition 3. *For any $k = 0, \ldots, K$, any $c = 1, \ldots, C$, any vector p of $[\![0, K]\!]^C_{\uparrow}$ and any $b = 0, \ldots, B$, the profit vector of any feasible solution in $P(k, c, p, b)$ is weakly dominated by UB.*

Proof. Let p' be the profit vector of a feasible solution in $P(k, c, p, b)$. Let $UB = (m_1, \ldots, m_C)$ be computed as in Eq. 1. For $c' = 1, \ldots, c-1$, by definition, $m_{c'} \geq p'_{c'}$. For $c' = c, \ldots, C$, by definition, $m_{c,c'}$ is the greatest number of items one can pick in $\mathcal{N}_{c,c'}$. Therefore $m_{c'} \geq p'_{c'}$. ■

Example 3 *At the root of the enumeration tree for Example 1, one has $UB = (1, 2, 2)$. For instance, when considering class 1 and 2, the greatest number of items that can be selected under the constraints is 2 (individuals 2 and 3, with $w^2 + w^3 = 6$), and therefore the second component of UB equals 2.*

4.4 Complexity

The number of nodes in the enumeration tree is clearly upper bounded by $(K+1)^C$, since the tree is of depth C and the number of children of a node is upper bounded by $K+1$. Furthermore, note that each node representing a problem $P(k, C-1, \cdot, \cdot)$ with $k \leq K$ has at most one child: the only decision that can be made is indeed to select $K-k$ items of class C, so that the cardinality constraint holds. The number of nodes in the enumeration tree is therefore in $\mathcal{O}(K^{C-1})$.

As the computation time required for the bounding procedure (at each node) is polynomial provided C is a constant, the complexity of the whole branch and bound algorithm is also polynomial. By comparing the number of cells in the dynamic programming table ($\Theta(nK^C)$) and the number of nodes in the enumeration tree ($\mathcal{O}(K^{C-1})$), it appears that the branch and bound algorithm should perform better. This observation is confirmed experimentally for all problems we tested.

Besides, the spatial complexity of the branch and bound algorithm in the worst case is in $\mathcal{O}(K^{C-1})$. Therefore it is also better than the dynamic programming algorithm from this point of view.

5 Experimental Results

We present here numerical results concerning the multiobjective dynamic programming method, and the branch and bound method. The computer used is an Intel Core 2 duo @3GHz, with 3GB RAM, and the algorithms were coded in C++. We first test our methods on randomly generated instances, and then on a real-world data set (IMDb dataset).

5.1 Randomly Generated Instances

We chose to run our tests on two different types of instances:

- uncorrelated instances (Un): for each item i, γ^i is randomly drawn in $\{1, \ldots, C\}$, and w^i is randomly drawn in $\{1, \ldots, 1000\}$.
- correlated instances (Co): for each item i, γ^i is randomly drawn in $\{1, \ldots, C\}$, and w^i is randomly drawn in $\{1+1000*(C-\gamma^i)/C, \ldots, 1000*(C-\gamma^i+1)/C\}$. In other words, the better the class, the higher the weight; for instance, if $\gamma^i = 3$ (resp. $\gamma^i = 2$) and $C = 5$, then w^i is randomly drawn in $\{401, \ldots, 600\}$ (resp. $\{601, \ldots, 800\}$).

For all instances, we chose to set B so that the following properties hold:

- $B \geq \sum_{i=1}^{K} w^{(i)}$, where item (i) is the item with the i-th smallest weight: this inequality ensures that there is at least one feasible solution;
- $B < \sum_{i=1}^{K} w^i$: this inequality ensures that the solution consisting of the K best items is not feasible (we recall that items are first ordered decreasingly with respect to their classes, and increasingly ordered with respect to their weights within each class).

By setting $B = 0.5 \sum_{i=1}^{K} w^{(i)} + 0.5 \sum_{i=1}^{K} w^i$ in the tests, both properties hold (unless $\sum_{i=1}^{K} w^{(i)} = \sum_{i=1}^{K} w^i$, in which case the only non-dominated solution consists in selecting the K first items).

Table 2 shows the average computation times in seconds for both methods (DP: dynamic programming, BB: branch and bound), and the average number of non-dominated profit vectors (in other words, the size of a complete minimal set of preferred solutions) over 30 random instances for each type and size.

Symbol "-" means that no instance could be solved due to memory constraints, i.e. more than 3GB RAM were required. All generated instances have $n = 1000$ items. Notation "Un-x-y" (resp. "Co-x-y") means Uncorrelated (resp. Correlated) instances with $C = x$ and $K = y$. Since there is very little variance in the computation times for a given type and size, only the average computation times are reported.

Table 2. Average computation times of both methods, and average number of non-dominated profit vectors (ND), for uncorrelated and correlated instances of size $n = 1000$.

Type	DP (sec.)	BB(sec.)	ND	Type	DP(sec.)	BB(sec.)	ND
Un-3-100	3.9	0.005	3	Co-3-100	3.9	0.004	44
Un-3-200	32.1	0.06	4	Co-3-200	32.0	0.06	75
Un-3-500	506	0.45	38	Co-3-500	505	0.5	108
Un-4-100	132	0.007	5	Co-4-100	132	0.08	1101
Un-4-150	656	0.03	12	Co-4-150	654	0.4	2166
Un-4-200	-	0.07	16	Co-4-200	-	0.8	3346
Un-5-50	117	0.003	2	Co-5-50	121	0.5	3657
Un-5-80	1114	0.004	7	Co-5-80	1263	7.0	13526
Un-5-100	-	0.018	15	Co-5-100	-	23.2	24800

First note that, for all instances, the branch and bound approach is faster than the dynamic programming one. As expected, more classes make the problem harder, and the same goes for size K of the committee. The number of non-dominated profit vectors is small for uncorrelated instances, because there are low weighted items in good classes. This number is much larger for correlated instances, because this property does not hold anymore. Comparing the results obtained for uncorrelated and correlated instances shows that the correlation has no impact on the computation times of the dynamic programming procedure. However, its impact is noticeable for the branch and bound method, since the number of nodes expanded in the enumeration tree grows with the number of non-dominated profit vectors, and this number is very high for correlated instances. The impact of the correlation on the number of non-dominated profit vectors is consistent with what can be observed in multiobjective combinatorial optimization. We will come back to the question of the size of the non-dominated set in the next subsection.

Since the branch and bound procedure is very fast, and does not have high memory requirements, we tested it on larger instances. We set $n = 10000$ and $K = 100$ for all these instances. Table 3 shows the results of those experiments for $C \in \{3, 4, 5, 10, 20, 50\}$. Resolution times are in seconds, and symbol "-" means that it exceeds 600 seconds. Most of the resolution time is now spent in the bounding part, more precisely for the comparison between the optimistic evaluation of a node and the non-dominated profit vectors. For uncorrelated instances with $3, 4, 5$ classes, the resolution times are nevertheless particularly small because the bounds enable to discard a huge amount of nodes, since there

are few good feasible profit vectors (around 70% of selected items in these solutions belong to class 1). This is no longer true for correlated instances, which results in much greater resolution times.

Furthermore, as is well-known in multiobjective optimization, the number of objectives (here, the number C of classes) is a crucial parameter for the efficiency of the solution methods. For this reason, when $C = 10$, 20 or 50, the resolution is of course computationally more demanding, as can be observed in the table (for instance, for $C = 20$ and $K = 100$, the resolution time is on average 2.21 seconds for uncorrelated instances). The method seems nevertheless to scale well, though the variance in the resolution times is much higher.

Table 3. Average computation times of the BB method, and average number of non-dominated profit vectors (ND), for uncorrelated and correlated instances of size $n = 10000$ with $K = 100$ and $C \in \{3 \cdots 50\}$.

Type	BB(sec.)			ND	Type	BB(sec.)			ND
	min.	avg.	max.			min.	avg.	max.	
Un-3-100	0.01	0.02	0.02	3	Co-3-100	0.03	0.05	0.06	50
Un-4-100	0.02	0.02	0.03	6	Co-4-100	1.27	1.31	1.37	4960
Un-5-100	0.02	0.03	0.04	10	Co-5-100	27.3	28.0	29.0	29418
Un-10-100	0.10	0.12	0.15	264	Co-10-100	-	-	-	-
Un-20-100	0.37	2.21	14.24	467	Co-20-100	-	-	-	-
Un-50-100	2.09	21.1*	101*	968*	Co-50-100	-	-	-	-

* Note that one instance largely exceeded the time limit, and the values indicated do not take this instance into account.

Table 4(A) (resp. 4(B)) is there to give an idea of the order of magnitude of K with respect to C in order to get tractable uncorrelated (resp. correlated) instances. For each C, the order of magnitude of parameter K in the table is the one beyond which the resolution becomes cumbersome.

Table 4. Average computation times of the BB method, and average number of non-dominated profit vectors (ND), for uncorrelated and correlated instances of size $n = 10000$ with $C \in \{3 \cdots 50\}$, for different values of K.

Type	BB(sec.)			ND
	min.	avg.	max.	
Un-3-5000	375	394	425	368
Un-4-3000	208	237	266	7203
Un-5-2000	185	292	428	15812
Un-10-250	1.86	10.5	55.4	2646
Un-20-150	0.69	91.5	562	2603
Un-50-80	1.98	24.6	208	1052

(A)

Type	BB(sec.)			ND
	min.	avg.	max.	
Co-3-5000	415	419	424	1086
Co-4-1000	666	706	767	105976
Co-5-100	27.3	28.0	29.0	29418
Co-10-15	95.2	97.4	103	30441
Co-20-7	20.0	20.2	20.6	14800
Co-50-5	521	526	534	36471

(B)

5.2 IMDb Dataset

Let us now evaluate the operationality of the BB method on a real data set, namely the Internet Movie Database (www.imdb.com). On this web site, one can indeed find a top 250 movies as voted by the users. Assume that a film festival organizer wants to project K top movies within a given time limit. If the organizer refers to the IMDb Top 250 to make his/her choice (i.e., the preference classes are directly inferred from the Top 250), it amounts to a committee selection problem where the weights are the durations of the movies. The numerical tests carried out are the following:

- size K of the committee varies from 5 to 50;
- number C of classes varies from 10 to 250 (in this latter case, the setting is the same as in Klamler et al. [14], i.e. there is a linear order on the elements);
- the time limit follows the formula used for the budget constraint in the previous tests, so that both constraints (cardinality and weight) are taken into account in the choice.

Table 5 shows the computation times in seconds for the BB method, as well as the number ND of non-dominated committees (i.e., non-dominated subsets of movies). Symbol "-" means that the computation time exceeds 600 sec. Interestingly, one observes that the method remains operational even when the number of preference classes is high. The size of the non-dominated set of course increases, but this is not a real drawback if one sees the pairwise dominance relation as a first filter before an interactive exploration of the non-dominated set (by interactively adding constraints for instance, so as to reduce the set of potential selections).

Table 5. Computation times of the BB method for the IMDb data set.

	$C = 10$		$C = 25$		$C = 50$		$C = 250$	
	Time	ND	Time	ND	Time	ND	Time	ND
$K = 5$	0.01	5	0.03	9	0.15	7	2.7	11
$K = 10$	0.01	8	0.08	24	0.6	108	131.6	323
$K = 15$	0.01	12	0.6	156	11.5	469	-	-
$K = 20$	0.01	16	5.17	222	295	1310	-	-
$K = 25$	0.01	14	131.3	883	-	-	-	-
$K = 50$	3.0	749	-	-	-	-	-	-

6 Conclusion

We studied the committee selection problem with a cardinality constraint, where the items are classified into ordered classes. By reducing the problem to a multiobjective knapsack problem with a cardinality constraint, we proposed two polynomial time solution algorithms: a dynamic programming scheme and a

branch and bound procedure. The theoretical complexities and numerical tests tend to prove that the latter one is better, both in time and space requirements.

Note that all the results presented here naturally extends when the preference classes are only partially ordered. The only difference is that the profit vectors are then not necessarily non-decreasing. For instance, consider three partially ordered preference classes 1, 2 and 3 with: $1 \rhd 2$ and $1 \rhd 3$ (2 and 3 are not comparable). The profit vector for an item of class 2 is then $(0, 1, 0)$.

Finally, it would be interesting to study more expressive settings for ranking sets of objects. For instance, when the order relation is directly defined on the items, Fishburn [11] proposed a setting where preferences for the inclusion (resp. exclusion) of items in (resp. from) a subset can be expressed.

Acknowledgments. We would like to thank the reviewers for their helpful comments and suggestions.

References

1. S. Barberà, W. Bossert, and P.K. Pattanaik. Ranking sets of objects. In S. Barberà, P.J. Hammond, and C. Seidl, editors, *Handbook of Utility Theory, vol. 2*. Kluwer Academic Publishers, 2004.
2. E.M. Bartee. Problem solving with ordinal measurement. *Management Science*, 17(10):622–633, 1971.
3. W. Bossert, P.K. Pattanaik, and Y. Xu. Ranking opportunity sets: An axiomatic approach. *Journal of Economic Theory*, 63(2):326–345, 1994.
4. U. Bossong and D. Schweigert. Minimal paths on ordered graphs. Technical Report 24, Report in Wirtschaftsmathematik, Universität Kaiserslautern, 1996.
5. S. Bouveret, U. Endriss, and J. Lang. Fair division under ordinal preferences: Computing envy-free allocations of indivisible goods. In *European Conference on Artificial Intelligence (ECAI 2010)*, pages 387–392. IOS Press, 2010.
6. S. Brams, P. Edelman, and P. Fishburn. Fair division of indivisible items. *Theory and Decision*, 5(2):147–180, 2004.
7. S. Brams and D. King. Efficient fair division – help the worst off or avoid envy? *Rationality and Society*, 17(4):387–421, 2005.
8. F. Della Croce, V. Th. Paschos, and A. Tsoukias. An improved general procedure for lexicographic bottleneck problems. *Op. Res. Letters*, 24:187–194, 1999.
9. T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multi-objective knapsack problems. In F. Dehne, J.-R. Sack, and R. Tamassia, editors, *Algorithms and Data Structures*, volume 2125 of *LNCS 2125*, pages 210–221. Springer Berlin / Heidelberg, 2001.
10. P.C. Fishburn. *Utility Theory for Decision Making*. Wiley, New York, 1970.
11. P.C. Fishburn. Signed orders and power set extensions. *Journal of Economic Theory*, 56:1–19, 1992.
12. J.Y. Halpern. Defining relative likelihood in partially-ordered preferential structures. *Journal of Artificial Intelligence Research*, 7:1–24, 1997.
13. P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multicriteria Decision Making*, 1980.

14. C. Klamler, U. Pferschy, and S. Ruzika. Committee selection with a weight constraint based on lexicographic rankings of individuals. In *First International Conference on Algorithmic Decision Theory (ADT 2009)*, pages 50–61. Springer, 2009.
15. K. Klamroth and M.M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 47:57–76, 2000.
16. D. Schweigert. Ordered graphs and minimal spanning trees. *Foundations of Computing and Decision Sciences*, 24(4):219–229, 1999.