Incremental Approval Voting for Multi-agent Knapsack Problems

Nawal Benabbou, Patrice Perny

Incremental Approval Voting for Multi-agent Knapsack Problems

Nawal Benabbou and Patrice Perny

Sorbonne Universités, UPMC Univ Paris 06
CNRS, LIP6 UMR 7606
4 Place Jussieu, 75005 Paris, France
nawal.benabbou@lip6.fr, patrice.perny@lip6.fr

Abstract

In this paper, we study approval voting for multi-agent knapsack problems under incomplete preference information. The agents consider the same set of feasible knapsacks, implicitly defined by a budget constraint, but they possibly diverge in the utilities they attach to items. Individual utilities being difficult to assess precisely and to compare, we collect approval statements on knapsacks from the agents with the aim of determining the optimal solution by approval voting. We first propose a search procedure based on mixed-integer programming to explore the space of utilities compatible with the known part of preferences in order to determine or approximate the set of possible approval winners. Then, we propose an incremental procedure combining preference elicitation and search in order to determine the set of approval winners without requiring the full elicitation of the agents' preferences.

1 Introduction

Collective decision making on a combinatorial domain appears in various contexts such as investments planning, resource allocation or group configuration. Due to strategic aspects often surrounding group decision-making and the possible divergences in individual values, developing formal methods and tools for modeling preferences and solving multi-agent combinatorial optimization problems is a critical issue. This has motivated a lot of work in the recent years, in the field of computational social choice [Brandt et al., 2016]. We focus here on the multi-agent knapsack problem which consists of determining, given a finite set of items, a subset of maximal utility under a budget constraint. This is a standard example of combinatorial problem with many potential applications such as project selection, portfolio management or committee election, see e.g. [Lu and Boutilier, 2011b, Klamler et al., 2012, Elkind et al., 2014, Oren and Lucier, 2014, Skowron et al., 2015a] for examples of recent contributions.

In combinatorial optimization problems, the agents cannot be expected to provide extensive preference models. Compact representations are needed to handle individual and collective preferences. Usually, in knapsack problems, preference over subsets of items are represented by additive utility functions. More precisely, the utility of a subset of items for an agent is defined as the sum of the utilities of its elements. Individual utilities are difficult to assess, especially on a combinatorial domain. Although the elicitation task is simplified when utilities are decomposable, elicitation methods based on systematic pairwise comparisons are practically unfeasible due to the large amount of feasible subsets and their implicit definition. Hence we are interested in designing incremental elicitation procedures, in which preference queries are selected iteratively, to be as informative as possible at every step, so as to progressively reduce the set of admissible utility profiles until the set of optimal knapsacks can be determined. This approach has been successfully used in AI for additive utility elicitation on explicit sets [Chajewska et al., 2000, Wang and Boutilier, 2003, Boutilier et al., 2006], but also on combinatorial solution spaces [Gelain et al., 2010, Benabbou and Perny, 2015].

We remark that, even if numerical representations of individual preferences are accessible under

the form of utility functions, they are generally constructed independently for each agent. Hence, it is unlikely that such representations allow the welfare of individuals to be compared. In this context, the definition of a social utility as the sum of individual utilities (utilitarism), for example, would be meaningless. Assuming that utilities of items are expressed on the same scale or that utilities are normalized would not be sufficient to overcome the problem, as shown by the following example:

Example 1. *Consider a multi-agent knapsack problem involving 3 items and 2 agents with utilities:* $u^1 = u^1_1x_1 + u^1_2x_2 + u^1_3x_3$ *and* $u^2 = u^2_1x_1 + u^2_2x_2 + u^2_3x_3$ *to be maximized under the constraint* $x_1 + x_2 + x_3 \leq 2$*, where* $x_i \in \{0, 1\}, i = 1, 2, 3$*, are the decision variables and* u^i_j *represents the utility of item* j *for agent* i*. This problem could appear to elect a committee of size 2, given 3 candidates and 2 voters. Assume that individual preference orders over committees have been elicited, and are equal to* $\{2, 3\} \succ_1 \{1, 3\} \succ_1 \{1, 2\}$ *and* $\{1, 2\} \succ_2 \{1, 3\} \succ_2 \{2, 3\}$ *for agents* 1 *and* 2 *respectively. One possible numerical representation of these preferences (using the same utility scale for the two agents) is given by:* $(u^1_1, u^1_2, u^1_3) = (1, 2, 4)$ *and* $(u^2_1, u^2_2, u^2_3) = (4, 2, 1)$ *which leads to the following utilities for solutions of size 2:*

	$\{1,2\}$	$\{2,3\}$	$\{1,3\}$
u^1	3	6	5
u^2	6	3	5

Note that these individual values are consistent with preference orders $\succ_1$ *and* $\succ_2$*. Now, if we are utilitarian, we could be tempted to deduce that* $\{1, 3\}$ *is the optimal knapsack because it maximizes the total utility (*$5 + 5 = 10$*). However, such a conclusion would be meaningless, it is only due to the particular numerical representation chosen for individual utilities. Let us change the initial numerical scale by replacing numbers* $(1, 2, 4)$ *by* $(0, 3, 4)$*. In this case we obtain two new utility functions characterized by* $(u^1_1, u^1_2, u^1_3) = (0, 3, 4)$ *and* $(u^2_1, u^2_2, u^2_3) = (4, 3, 0)$ *which leads to the following utilities for solutions of size 2:*

	$\{1,2\}$	$\{2,3\}$	$\{1,3\}$
u^1	3	7	4
u^2	7	3	4

Note that these new individual values are still consistent with preference orders $\succ_1$ *and* $\succ_2$*. Yet, with the same utilitarian principle, we should admit now that* $\{1, 3\}$ *is the least preferred knapsack. Therefore, choosing the solution maximizing the sum of individual utilities would not be a good procedure. It would merely be a consequence of arbitrary choices of numerical representations of preferences rather than a robust conclusion derived from the observed preference profile. Note that the problem persists if we normalize* u^1 *and* u^2 *utilities to obtain value* 1 *for set* $\{1, 2, 3\}$*. Other examples could be found for other aggregators than the sum (e.g. the minimum for an egalitarian aggregation).*

In order to be able to compare the solutions of a knapsack problem when individual utility scales are not commensurate and/or not rich enough to allow the construction of a social utility, it seems natural to resort to a voting rule. The main advantage of a voting rule is indeed to perform an ordinal aggregation procedure. There is no need to know how the welfare of individuals should be compared, we only need to elicit individual preference orders. Preference elicitation can be performed incrementally so as to determine the winner with a reduced amount of preference queries. Various incremental elicitation procedures have been proposed and studied in the context of single-winner elections with incomplete preferences [Kalech et al., 2010, Lu and Boutilier, 2011a, Dery et al., 2014]. In this setting, several contributions study the determination of possible and necessary winners from a partial preference profile, e.g., [Konczak and Lang, 2005, Xia and Conitzer, 2011, Lang et al., 2012, Ding and Lin, 2013], when the set of candidates is defined explicitly.

In knapsack problems however, solutions are numerous and defined implicitly, which is an additional challenge for the winner determination. This explains the current interest for incremental

voting procedures on combinatorial domains and the purpose of this paper. Our aim here is to propose an incremental voting rule in which individual preferences are progressively revealed until a collective decision can be made, and to apply this procedure on the multi-agent knapsack problem, taking advantage of the fact that individual preferences are representable by additive utilities.

It is important to note that implementing a voting rule is not in contradiction with the representation of individual values by utilities. Individual utility functions are indeed seen as convenient representations of individual preference orders, and their use will significantly contribute to relieve the preference elicitation burden. In the standard knapsack problem, due to the linearity of preferences, the set of all preference orders compatible with a given partial order can be characterized by a convex polyhedron in the utility space. This makes it possible to resort to mathematical programming to explore all possible completions of any partially known preference profile, but also to look for possible winners, and to develop an efficient incremental elicitation procedure for winner determination, as it will be seen later in the paper.

Implementing a voting rule for the knapsack problem with a partially specified preference profile could be related to multi-winner voting rules studied in the field of computational social choice. Most approaches recently proposed for multi-winner elections assume that individual preferences over items are sufficient to explain preference over subsets because they derive satisfaction from their most preferred candidate (see e.g., [Chamberlin and Courant, 1983, Monroe, 1995, Potthoff and Brams, 1998, Procaccia et al., 2008, Meir et al., 2008, Lu and Boutilier, 2011b, Betzler et al., 2013, Elkind et al., 2014, Skowron et al., 2015b], and see [Lu and Boutilier, 2013] for incremental elicitation of voter preferences). This assumption is well-suited to the election of representatives. However, for any agent, it may happen that the selection of the most preferred candidate is not sufficient to counterbalance the presence of multiple least preferred candidates in the elected committee. This limitation also applies to the selection of items in the multi-agent knapsack problem.

In this paper, we focus on approval voting because this is a simple rule that can be decisive even if only a part of the preference profile is known. In approval voting, we only need to learn, for every agent, which are the approved or disapproved subsets, so as to elect a solution receiving the maximal support. We therefore investigate incremental procedures for approval voting and their application to the knapsack problem. This work differs from multi-winner approval voting [Kilgour, 2010] which only collects approval statements over items instead of feasible subsets of items.

The paper is organized as follows: after introducing our framework for preference elicitation in multi-agent knapsack problems in Section 2, we study the determination of possible winners for approval voting in Section 3. Then, an incremental voting procedure to determine the set of winners are proposed in Section 4 and experimented in Section 5.

2 The General framework

We consider a collective decision problem where a set of agents $N = \{1, \ldots, n\}$ has to jointly select a set of items (e.g., candidates, projects, objects) in a set $P = \{1, \ldots, p\}$. Any subset of items can be represented by a solution vector $x = (x_1, \ldots, x_p) \in \{0, 1\}^p$ where $x_j = 1$ if item j is in the subset and $x_j = 0$ otherwise. Some linear constraints on variables $x_j, j \in P$, are imposed to define the admissible solution vectors. For instance, one may want to impose cardinality constraints to control the size of the set and/or to ensure gender parity in the elected committee; there may also exist budget constraints (e.g., when the decision is subject to a maximum total cost) or capacity constraints as in knapsack problems, making some subsets of items unfeasible. For the simplicity of the presentation, we will only consider the standard knapsack constraint of the form $\sum_{j \in P} w_j x_j \leq W$ where w_j is the (positive) weight of item j and W is a positive value representing the maximum total weight; the set of feasible solutions will be denoted by $\mathcal{X}$ in the sequel. Our purpose and the algorithms proposed in the paper also apply when additional (linear) feasibility constraints are considered.

We assume that the preference of agent i can be represented by a function $u^i : \{0,1\}^p \to \mathbb{R}$ measuring the overall utility of any solution. Hence, given two solutions x, y representing two subsets of items, x is at least as good as y for agent $i \in N$ whenever $u^i(x) \geq u^i(y)$. Here $u^i(x) = \sum_{j \in P} u^i_j x_j$, where $u^i_j \in \mathbb{R}$ represents the utility of item j for agent i. The profile $(u^1, \ldots, u^n)$ of utility functions will be denoted by u. Note also that u^i, as a numerical representation of a preference order, is generally not unique and any transform preserving inequalities of type $u^i(x) \geq u^i(y)$ for all solutions x, y could be considered as well.

Nevertheless, numerical representations of individual preferences by utility functions $u^i, i \in N$, can be used in approval voting. Recall that approval voting is a single-winner voting method which allows each agent to approve of (vote for) as many candidates as she wishes. The candidate with the most approval votes is the winner of the election [Brams and Fishburn, 1978]. Under the assumption that individual preferences are represented by utility functions $u^i, i \in N$, a solution x is approved by agent i if and only if $u^i(x) \geq \delta^i$ where $\delta^i \in \mathbb{R}$ is an approval (or utility) threshold that separates approved and non-approved solutions. The profile of thresholds $(\delta^1, \ldots, \delta^n)$ will be denoted by δ in the sequel. The pair (u, δ) characterizes approved and non-approved solutions for all agents and enables the computation of approval scores for any feasible solution x. This score is given by $f(x, u, \delta) = |\{i \in N, u^i(x) \geq \delta^i\}|$, and the winner of the election is a feasible solution maximizing this score; various tie-breaking rules can be considered. The approval multi-agent knapsack problem can be defined as follows:

APPROVAL MULTI-AGENT KNAPSACK PROBLEM (AMKP)
Input: Finite set P of items, for each $j \in P$, a weight w_j and a positive integer W; a finite set N of agents, for each $i \in N$, an approval threshold δ^i, for each $j \in P$, a utility value u^i_j and a positive integer K.
Question: Is there a subset $X \subseteq P$ such that $\sum_{j \in X} w_j \leq W$ and $|\{i \in N, \sum_{j \in X} u^i_j \geq \delta^i\}| \geq K$?

This problem is NP-complete, due to a simple reduction from the knapsack decision problem. Testing the existence of an admissible knapsack having a utility greater or equal to a given value K' is indeed equivalent to solving an instance of the AMKP problem involving a single agent with the same utilities over items, an approval threshold equal to K' and with $K = 1$. Therefore, finding the knapsack maximizing the approval score is NP-hard.

Moreover, well-known pseudo-polynomial solution methods for the knapsack problem based on dynamic programming are no longer valid for the approval winner determination problem, as shown by the following example:

Example 2. *Consider a collective decision problem where 2 agents have to choose 2 representatives from a pool of 3 candidates, i.e.* $N = \{1, 2\}$ *and* $P = \{1, 2, 3\}$. *Assume that utilities and approval thresholds are the following:*

u^1_1	u^1_2	u^1_3	δ^1	u^2_1	u^2_2	u^2_3	δ^2
0.5	0.3	0.2	0.5	0.1	0.5	0.3	0.7

In this case, we have $f(\{1\}, u, \delta) = 1 > 0 = f(\{2\}, u, \delta)$ *but* $f(\{1, 3\}, u, \delta) = 1 < 2 = f(\{2, 3\}, u, \delta)$. *We observe a preference reversal because* $\{1\}$ *is preferred to* $\{2\}$ *whereas* $\{2, 3\}$ *is preferred to* $\{1, 3\}$. *Thus, preferences induced by the approval score are not additive with respect to union with disjoint items. This precludes to construct the optimal knapsack from optimal subsets of items.*

Nevertheless, the winner can be obtained by solving the following mixed integer program:

$$\begin{aligned}
\max \quad & \sum_{i \in N} a^i \\
\text{s.t.} \quad & \sum_{j \in P} u^i_j x_j - \delta^i \geq M(a^i - 1),\ \forall i \in N \\
& \sum_{j \in P} w_j x_j \leq W \\
& x_j \in \{0,1\}, a^i \in \{0,1\},\ \forall i \in N, \forall j \in P
\end{aligned}$$

The first constraint allows the introduction of a boolean variable a^i which is equal to 1 if and only if agent i approves solution x. The second constraint is the knapsack constraint. Finally, M is a constant greater than $\max\{\delta^i - u^i(x), i \in N\}$.

However, in practice, the full elicitation of individual utilities and approval thresholds is too expensive. Usually, we can observe simple preference statements of type "I prefer solution x to solution y", and in the case of approval voting, "I approve solution x", or "I don't approve solution y". These preference statements enable to restrict the sets of possible utility functions but generally do not allow to derive a precise utility function for each agent (see Example 1). Instead, uncertainty sets representing all possible utility functions compatible with the preference information obtained so far must be considered. The same observation applies to approval thresholds. Under such utility uncertainty, we investigate now the determination of possible winners.

3 Determination of Possible Approval Winners

In this section, we propose an algorithm that enables to compute the set of possible approval winners given some partial knowledge of the agents' preferences. More precisely, the input of the algorithm consists of three sets of preference information for each agent $i \in N$: a set A^i (resp. $\bar{A}^i$) of solutions that are known to be approved (resp. not approved) by agent i, and a set $\mathcal{P}^i$ of pairs (y, z) such that solution y is known to be preferred to solution z by agent i. The elements of $\mathcal{P}^i$ are not explicit approval statements, but can be used to derive new positive or negative approval statements from those included in A^i and $\bar{A}^i$.

Let U^i (resp. Δ^i) denote the set of utility functions (resp. approval thresholds) compatible with the available preference statements A^i, $\bar{A}^i$ and $\mathcal{P}^i$. Formally, (U^i, Δ^i) is the set of all pairs (u^i, δ^i) such that: $\forall x \in A^i, u^i(x) \geq \delta^i$; $\forall x \in \bar{A}^i, u^i(x) < \delta^i$ and $\forall (y,z) \in \mathcal{P}^i, u^i(y) \geq u^i(z)$, where u^i is a function of the form $u^i(x) = \sum_{j \in P} u^i_j x_j$ and $\delta^i \in \mathbb{R}$. Let U (resp. Δ) be the cartesian product $U^1 \times \ldots \times U^n$ (resp. $\Delta^1 \times \ldots \times \Delta^n$). Given such uncertainty sets, the set $\text{PW}(\mathcal{X}, U, \Delta)$ of possible approval winners is defined as follows:

Definition 1.

$$PW(\mathcal{X}, U, \Delta) = \bigcup_{u \in U, \delta \in \Delta} \arg\max_{x \in \mathcal{X}} f(x, u, \delta)$$

In other words, the set of possible approval winners is the set of all solutions $x \in \mathcal{X}$ that maximize the approval score $f(x, u, \delta)$ for some utility profile $u = (u^1, \ldots, u^n) \in U$ and some approval threshold vector $\delta = (\delta^1, \ldots, \delta^n) \in \Delta$. Recall that Example 2 shows that standard dynamic programming procedures cannot be used to determine the approval winners when utilities and approval thresholds are known. This difficulty remains when utilities and/or approval thresholds are partially known.

The branch and bound approach is the most commonly used tool for solving NP-hard optimization problems. We propose here a branch and bound procedure to compute the set PW$(\mathcal{X}, U, \Delta)$, where nodes of the search tree represent partial instances of the decision variable vector $x = (x_1, \ldots, x_p)$. More precisely, each node η of the tree is characterized by a pair (P^0_η, P^1_η) where $P^k_\eta = \{j \in P, x_j = k\}, k = 0, 1$. Let $P_\eta = P \setminus (P^0_\eta \cup P^1_\eta)$ denote the set of all undecided variables at node η. Thus, each node η is associated with a region of the solution space as follows: solution $x = (x_1, \ldots, x_p)$ is attached to node η if and only if $x_j = k$ for all $j \in P^k_\eta, k = 0, 1$. The set of feasible solutions attached to node η is denoted by S_η hereafter. The main features of the search procedure are the following:

Initialization. Using a heuristic, a branch and bound procedure determines some feasible solutions before performing the search so as to define an initial bound on candidate solutions.

In order to obtain such a bounding set for the knapsack problem, denoted by S_0 hereafter, we propose to initially ask the agents to rank all the items by preference order. Let $r_i(j)$ denote the rank of item j in the preference order provided by agent i, we can define the score $\alpha^i(j) = \frac{p - r_i(j)}{w_j}$ for each agent $i \in N$ and each item $j \in P$ representing the tradeoff achieved between preference and weight. Hence, for each agent i, a "good" solution to the knapsack problem can be obtained by a greedy algorithm selecting items one by one, by decreasing order with respect to scoring function α^i, skipping elements whose weight is greater than the residual weight capacity. The resulting solution is inserted in S_0 for initialization because it represents a good solution from the point of view of agent i. This process is repeated for all agents $i \in N$.

Moreover, a similar procedure is used with the average scoring function defined by $\alpha(j) = 1/n \sum_{i=1}^{n} \alpha^i(j)$, to complete S_0 with a solution which is likely to be more consensual. This solution will be denoted by $\bar{s}$ in the sequel.

Evaluation and pruning. Our pruning rule is based on the notion of setwise regret defined as follows: the setwise max regret $R(A, B, U, \Delta)$ of a set $A \subseteq \mathcal{X}$ with respect to a set $B \subseteq \mathcal{X}$ is the maximal feasible approval score difference between the best solution in B and the best solution in A. More formally:

$$R(A, B, U, \Delta) = \max_{u \in U, \delta \in \Delta} \left\{ \max_{b \in B} f(b, u, \delta) - \max_{a \in A} f(a, u, \delta) \right\}$$

If $R(A, B, U, \Delta) < 0$, then we know that B does not contain any possible approval winner; it indeed induces that, for all solutions $b \in B$, for all $u \in U$ and for all $\delta \in \Delta$, there exists $a \in A$ such that $f(b, u, \delta) < f(a, u, \delta)$.

Let S be the set of solutions found so far (initially $S = S_0$) and O be the current set of nodes to be explored. We propose to prune a node $\eta \in O$ if the setwise max regret $R(S, S_\eta, U, \Delta)$ of set S with respect to set S_η is strictly negative. Note that $R(S, S_\eta, U, \Delta) = \max_{x \in S_\eta} \max_{u \in U, \delta \in \Delta} \min_{s \in S} \{f(x, u, \delta) - f(s, u, \delta)\}$. This alternative formulation enables to compute $R(S, S_\eta, U, \Delta)$ as the optimal value of the mixed-integer quadratic program (denoted MIQP$_\eta$) given in Figure 1.

In this program, $\xi > 0$ is an arbitrary small value allowing us to model strict inequalities. Equations (5-7) enable to restrict utility functions and approval thresholds to those compatible with the available preference information. Since preferences of agent i, for any $i \in N$, are invariant by positive affine transformations jointly applied to function u^i and threshold δ^i, we can assume without loss of generality that utilities are positive and bounded above by a constant $M > 0$ (cf. Equation (4)). Moreover, a^i_s is a boolean variable equal to 1 iff agent i approves solution $s, s \in S$, and a^i is a boolean variable equal to 1 if agent i approves solution x. Finally, Equation (1) introduces variable $t \in \mathbb{R}$ representing the smallest approval score difference between solution x and solution $s, s \in S$.

$$\max\ t$$

$$\text{s.t. } t \leq \sum_{i \in N} a^i - \sum_{i \in N} a^i_s,\ \forall s \in S \tag{1}$$

$$\sum_{j \in P^1_\eta} u^i_j + \sum_{j \in P_\eta} u^i_j x_j - \delta^i \geq M(a^i - 1),\ \forall i \in N \tag{2}$$

$$\sum_{j \in P} u^i_j s_j - \delta^i + \xi \leq M a^i_s,\ \forall s \in S,\ \forall i \in N \tag{3}$$

$$\sum_{j \in P_\eta} w_j x_j + \sum_{j \in P^1_\eta} w_j \leq W$$

$$\sum_{j \in P} u^i_j = M,\ \forall i \in N \tag{4}$$

$$\sum_{j \in P} u^i_j y_j \geq \delta^i,\ \forall i \in N, \forall y \in A^i \tag{5}$$

$$\sum_{j \in P} u^i_j y_j \leq \delta^i - \xi,\ \forall i \in N, \forall y \in \bar{A}^i \tag{6}$$

$$\sum_{j \in P} u^i_j y_j \geq \sum_{j \in P} u^i_j z_j,\ \forall i \in N, \forall (y,z) \in \mathcal{P}^i \tag{7}$$

$$x_j \in \{0,1\},\ \forall i \in N, \forall j \in P_\eta$$

$$a^i \in \{0,1\}, a^i_s \in \{0,1\},\ \forall i \in N, \forall s \in S$$

$$u^i_j \geq 0, \delta^i \geq 0,\ \forall i \in N, \forall j \in P$$

Figure 1: Problem MIP$_\eta$

Note that constraints given in Equation (2) include quadratic terms of type $u^i_j x_j, j \in P_\eta$, since u^i_j are also variables of the optimization problem. In order to linearize theses constraints, we introduce positive variables $v^i_j, i \in N, j \in P_\eta$, representing the product $u^i_j x_j$ and Equation (2) is replaced by the following constraints:

$$\sum_{j \in P^1_\eta} u^i_j + \sum_{j \in P_\eta} v^i_j - \delta^i \geq M(a^i - 1),\ \forall i \in N$$

$$v^i_j \leq u^i_j,\ \forall i \in N, \forall j \in P_\eta$$

$$v^i_j \leq M x_j,\ \forall i \in N, \forall j \in P_\eta$$

$$v^i_j - u^i_j \geq M(x_j - 1),\ \forall i \in N, \forall j \in P_\eta$$

The resulting mixed-integer linear program is denoted by MIP$_\eta$ in the sequel.

Branching. Values $R(S, S_\eta, U, \Delta)$ available for all nodes $\eta \in O$ is also used to select the next node to be explored. We select here a node $\eta \in O$ which maximizes $R(S, S_\eta, U, \Delta)$. This selection strategy aims at maximally improving the current solution set S. The optimal solution of MIP$_\eta$ indeed maximizes the gap $f(x,u,\delta) - \max_{s \in S} f(s,u,\delta)$ over all $u \in U$, all $\delta \in \Delta$ and all $x \in \bigcup_{\eta' \in O} S_{\eta'}$. Then, set S_η is split in two by considering possible instantiations of a new variable $x_j, j \in P_\eta$. This variable is chosen among those equal to 1 in the optimal solution of MIP$_\eta$.

Filtering. As we will see in Proposition 1, the proposed Branch and Bound outputs, in general, a superset of the set of possible approval winners. To remove undesirable elements, we use a final filtering process which deletes solution $s' \in S$ such that $R(S \backslash \{s'\}, \{s'\}, U, \Delta) < 0$, using a simplified version of MIP$_\eta$.

The algorithm implementing these principles is referred to AS (Approval-based Search) in the sequel and it is summarized by Algorithm 1.

Algorithm 1: Possible Approval Winners Determination

Input: S_0: initial solutions
Output: $\text{PW}(\mathcal{X}, U, \Delta)$

$S \leftarrow S_0$
$\eta \leftarrow [\emptyset, \emptyset]$
$O \leftarrow \{\eta\}$
while $O \neq \emptyset$ **do**
  Select a node $\eta \in O$
  if $P_\eta = \emptyset$ **then**
    $S \leftarrow S \cup S_\eta$
  else
    Select an item $j \in P_\eta$
    Generate the nodes $\eta' = [P_\eta^0 \cup \{j\}, P_\eta^1]$ and $\eta'' = [P_\eta^0, P_\eta^1 \cup \{j\}]$
    if $S_{\eta'} \neq \emptyset$ and $R(S, S_{\eta'}, U, \Delta) \geq 0$ **then**
      $O \leftarrow O \cup \{\eta'\}$
    end
    if $S_{\eta''} \neq \emptyset$ and $R(S, S_{\eta''}, U, \Delta) \geq 0$ **then**
      $O \leftarrow O \cup \{\eta''\}$
    end
  end
  $O \leftarrow O \setminus \{\eta\}$
end
Filter the solution set S
return S

This algorithm is justified by the following proposition:

Proposition 1. *AS returns the set PW*$(\mathcal{X}, U, \Delta)$.

Proof. Since the pruning rule only discards nodes including no possible approval winner, we know that S is a superset of $\text{PW}(\mathcal{X}, U, \Delta)$ at the end of the while loop. Let x be an element inserted in S such that $x \notin \text{PW}(\mathcal{X}, U, \Delta)$, if it exists. Since x is not a possible winner, we know that, for all $u \in U$ and for all $\delta \in \delta$, there exists $x' \in \text{PW}(\mathcal{X}, U, \Delta) \subseteq S$ such that $f(x, u, \delta) < f(x', u, \delta)$. Therefore, x is necessarily removed from S during the filtering by definition of the filtering process. □

Depending on the uncertainty sets (U and Δ) implicitly defined by the available preference information, possible approval winners might be too numerous to be enumerated efficiently. In order to save time, one may be interested in approximating the set of possible approval winners with performance guarantee. We propose below a variant of Algorithm AS for approximating possible winners with some guarantee on the quality of the output.

Approximation. Given a constant $\varepsilon > 0$, a set $X \subseteq \mathcal{X}$ is an $(1 + \varepsilon)$-approximation of the set of possible winners if, for all $x \in \mathcal{X}$, all $u \in U$ and all $\delta \in \Delta$, there exists $x' \in X$ such that $f(x, u, \delta) \leq (1 + \varepsilon) f(x', u, \delta)$. In order to compute an $(1 + \varepsilon)$-approximation of the set of possible winners, we introduce an approximate version of the setwise max regret. The setwise max ε-regret $R_\varepsilon(A, B, U, \Delta)$ of a set $A \subseteq \mathcal{X}$ with respect to a set $B \subseteq \mathcal{X}$ is defined as follows:

$$R_\varepsilon(A, B, U, \Delta) = \max_{u \in U, \delta \in \Delta} \left\{ \max_{b \in B} f(b, u, \delta) - (1 + \varepsilon) \max_{a \in A} f(a, u, \delta) \right\}$$

If $R_\varepsilon(A, B, U, \Delta) \leq 0$, then we know that, for all $b \in B$, all $u \in U$ and all $\delta \in \Delta$, there exists $a \in A$ such that $f(b, u, \delta) \leq (1+\varepsilon) f(a, u, \delta)$. Therefore, we propose a variant of AS where a node $\eta \in O$ is discarded if $R_\varepsilon(S, S_\eta, U, \Delta) \leq 0$. This pruning rule is sharper than the previous one and is more likely to discard nodes in the search tree. The implementation is simple within Algorithm AS: computations of R values must simply be replaced by computations of R_ε values. Moreover, the value R_ε is obtained using MIP$_\eta$ in which Equation (1) is replaced by:

$$t \leq \sum_{i \in N} a^i - (1+\varepsilon) \sum_{i \in N} a^i_s, \ \ \forall s \in S$$

The resulting algorithm is denoted by AS$_\varepsilon$ in the sequel. Then, the following proposition holds.

Proposition 2. *AS$_\varepsilon$ returns an $(1+\varepsilon)$-approximation of the set PW$(\mathcal{X}, U, \Delta)$.*

Proof. Let x be a solution that does not belong to S at the end of the search procedure. Let $u \in U$ and $\delta \in \Delta$. We want to prove that $f(x, u, \delta) \leq (1+\varepsilon) f(s, u, \delta)$ for some $s \in S$.

If x was inserted in S at some step, then x has necessarily been deleted during the filtering of S. In that case, by definition of the filtering, we know that there exists $s \in S$ such that $f(x, u, \delta) \leq f(s, u, \delta)$; hence, $f(x, u, \delta) \leq (1+\varepsilon) f(s, u, \delta)$.

Assume now that x was discarded at some step without ever belonging to S. In that case, we know that, there exists at this step $s \in S$ such that $f(x, u, \delta) \leq (1+\varepsilon) f(s, u, \delta)$. If solution s belongs to S at the end of the procedure, then we can directly infer the result. If solution s is deleted during the filtering, then we know that there exists $s' \in S$ such that $f(s, u, \delta) \leq f(s', u, \delta)$. Therefore, $f(x, u, \delta) \leq (1+\varepsilon) f(s, u, \delta) \leq (1+\varepsilon) f(s', u, \delta)$ which completes the proof (note that there is no chaining of errors). □

4 Elicitation for Winners Determination

In Section 3, we have introduced a procedure to determine the set of possible approval winners. It can be used in situations where a significant part of approval judgements is available. However, with incomplete preference information, the number of possible winners often remains very large, due to the combinatorial nature of the knapsack problem. Actually, the notion of possible winner (even its approximate version) is not sufficiently discriminant to support efficiently a collective decision making process. On the other hand, the full elicitation of approval statements in a multi-agent knapsack problem requires $O(n2^p)$ queries, for n agents and p items. This prevents to perform a full elicitation of approval statements prior to the aggregation of preferences.

In order to overcome these problems, we propose now an incremental approach that combines preference elicitation and search to determine the approval winner(s). The basic principle of our approach is to collect approval statements from individuals so as to progressively reduce the uncertainty attached to approval scores until the actual winners can be determined.

The number of possible winners reduces with the uncertainty about the agents' preferences. More precisely, for any $U' \subseteq U$ and any $\Delta' \subseteq \Delta$, we have PW$(\mathcal{X}, U', \Delta') \subseteq$ PW$(\mathcal{X}, U, \Delta)$. If U reduces to a single utility profile and Δ to a single approval threshold vector, then the possible winners become the actual winners of the election, i.e. the solutions maximizing the approval score, among which the final winner is selected using a tie-breaking rule. If we consider an incremental elicitation procedure which progressively collects preference information so as to reduce sets U and Δ, we will reach a point where all possible winners are necessary winners. This will generally happen long before reducing U and Δ to singletons.

We propose now an incremental elicitation procedure progressively reducing uncertainty sets to U' and Δ' such PW$(\mathcal{X}, U', \Delta') =$ NW$(\mathcal{X}, U', \Delta')$ where NW$(\mathcal{X}, U', \Delta')$ represents the set of necessary winners, i.e., the set of elements in $\mathcal{X}$ which are optimal for all $u \in U'$ and all $\delta \in \Delta'$. Our incremental elicitation algorithm consists in inserting preference queries in Algorithm AS (see

the previous section) so as to discriminate between the current best solutions (those that were stored in S). In practice, S is now restricted to the most approved solutions found so far. Within this set we can arbitrarily select a representative, named the incumbent. Initially, the incumbent may be any feasible solution to the knapsack problem (e.g., solution $\bar{s}$, see the initialization in Section 3). Each time a new solution is found (a challenger), it is compared to the incumbent w.r.t. approval scores.

Note that, given the current uncertainty sets U and Δ, a solution $x \in \mathcal{X}$ is necessarily approved by an agent $i, i \in N$, if and only if $u^i(x_j) \geq \delta^i$ holds for all $u \in U$ and all $\delta \in \Delta$. This can be checked by solving the following linear program: $\min \sum_{j \in P} u^i_j x_j - \delta^i$ subject to Equations (4-7) and $u^i_j \geq 0, \delta^i \geq 0, \ \forall i \in N, \forall j \in P$, and testing whether the optimum is positive. Similarly, testing whether a solution $x \in \mathcal{X}$ is necessarily disapproved by an agent can be performed using linear programming. Thus, each time a new solution is found, we can efficiently determine the agents - if they exist - that necessarily approve/disapprove it. Then, a natural query generation strategy, denoted by σ_0 hereafter, consists in questioning all the other agents to know whether they approve this solution. The challenger will be inserted in S if it has the same approval score as the incumbent. If the challenger is strictly better than the incumbent, then S is reinitialized to include only the challenger. This strategy provides an incremental search algorithm to determine the set of approval winners, named Algorithm IAS (Incremental Approval-based Search) in the sequel.

Proposition 3. *For any initial uncertainty sets U and Δ, IAS terminates with uncertainty sets $U' \subseteq U$ and $\Delta' \subseteq \Delta$ such that PW$(\mathcal{X}, U', \Delta')$ = NW$(\mathcal{X}, U', \Delta')$.*

Proof. Let $s \in S$ be any solution returned by IAS. For all $x \in \mathcal{X}$, we want to prove that, at the end of the search procedure, we have $f(s, u, \delta) \geq f(x, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$. First, whenever a node η is pruned at some step k of IAS using the current incumbent s_k, we know that, for all $x \in S_\eta$, $f(x, u, \delta) \leq f(s_k, u, \delta)$ for all $u \in U_k$ and all $\delta \in \Delta_k$ where U_k and Δ_k are the current uncertainty sets at step k. Since $U' \subseteq U_k$ and $\Delta' \subseteq \Delta_k$, we have $f(x, u, \delta) \leq f(s_k, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$. Then, according to strategy σ_0, we know that s_k is such that $f(s_k, u, \delta) \leq f(s, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$. The result is obtained by transitivity. □

5 Numerical Tests

We report here numerical experiments where mixed-integer and linear programs are solved using the Gurobi Optimizer.

The first series of tests aims at evaluating the computation times (given in seconds) of possible winners calculation using AS and AS$_\varepsilon$. In these experiments, instances of the multi-agent knapsack problem with $p = 12$ are generated as follows: weights $w_j, j \in P$ are uniformly drawn in $\{1, \ldots, 100\}$. Then, capacity W is set to $d \times \sum_{j \in P} w_j$ where $d = 0.3, 0.4$ and 0.5 so as to vary the number of solutions: for $p = 12$, the number of feasible solutions is then approximatively equal to $500, 1000$ and 2000 respectively. Moreover, to evaluate the impact of the uncertainty sets, we randomly generate $q = 10, 20$ preference statements per agent before running the algorithms. The computation times obtained by averaging over 50 runs for instances with $n = 20$ are reported in Table 1. We also report the average number of possible winners denoted by $\#S$. As expected, we can see that computation times increase with the size of the problem, and decrease as the number of available preference statements increase. Moreover, we observe that AS$_{0.1}$ is drastically faster than AS. More precisely, AS$_{0.1}$ determines an 1.1-approximation of possible winners in a few seconds while AS needs a few minutes on average.

The second series of experiments aims at evaluating the performance of IAS in terms of computation times (given in seconds) and average number of queries per agent (denoted by $\#Q$ hereafter). We also report the average number of actual winners (denoted by $\#S$ hereafter). Initially, only the preference ranking over single items is available for each agent. Then, answers to approval

		$d = 0.3$		$d = 0.4$		$d = 0.5$	
method	q	time	$\#S$	time	$\#S$	time	$\#S$
AS	10	26.6	19.8	71.8	29.8	614.4	111.4
AS	20	25.2	17.4	55.7	23.9	442.3	90.3
$AS_{0.1}$	10	1.1	1.9	2.7	3.8	13.9	8.3
$AS_{0.1}$	20	1.0	1.6	2.3	2.7	9.4	5.0

Table 1: Computations of possible winners (in seconds).

queries are simulated using approval thresholds and utility functions randomly generated with negative correlations so as to obtain difficult instances. Due to the possibility of collecting preference information during the search, algorithm IAS is significantly faster than AS and can solve much larger instances. For example, we report the results obtained for $p = 15$ and $p = 18$ (with $d = 0.5$), which approximatively represents 2000, 16000 and 130000 feasible solutions respectively. Results obtained by averaging over 50 runs are reported in Table 2 for instances involving 10 and 30 agents. We can see that IAS is very efficient both in terms of number of queries per agent and computation times; for instance, for problems with 10 agents and 18 items (130000 feasible solutions), it enables to determine the set of optimal knapsacks for approval voting in about 70 seconds with less than 16 queries per agent on average, whereas the full elicitation of approval statements would require 130000 queries per agent.

	$p = 15$			$p = 18$		
n	time	$\#S$	$\#Q$	time	$\#S$	$\#Q$
10	24.3	26.5	10.3	69.8	71.9	15.5
30	90.3	11.4	9.3	314.5	16.8	13.9

Table 2: Computations of necessary winners (in seconds).

We focus now on the evaluation of the branching strategy which is of crucial importance for the efficiency of the query generation strategy proposed in Section 4. This elicitation strategy indeed consists in asking agents whether they approve or not some solutions found during the search; these solutions are obviously dependent on the branching strategy. For comparison, we also consider the interactive branch and bound procedure (named Random hereafter) that differs from IAS only on the branching strategy as follows: the next node to be explored and the next variable to be instantiated are both selected at random. For both branching strategies, each time an agent answers an approval query during the resolution, we compute the maximum regret attached to the incumbent s, which is equal to $\max_{x \in \mathcal{X}} \max_{u \in U, \delta \in \Delta} \Big\{ f(x, u, \delta) - f(s, u, \delta) \Big\}$. This quantity is an upper bound on the actual regret when choosing the incumbent instead of any other solution in terms of approval scores. Whenever this value equals zero, the incumbent is necessarily an approval winner. Regrets are here expressed on a normalized scale assigning value 1 to the initial maximum regret (computed before collecting any preference information) and value 0 when the maximum regret is zero. Figure 2 shows the results obtained by averaging over 30 runs for decision problems involving 30 agents. We can see that the maximum regret reduces much more quickly with IAS than with Random. For instance, after 240 queries (i.e., 8 queries per agent) on average, the regret of choosing the incumbent is under 10% of the initial regret, whereas it remains above 65% with the random branching strategy. Hence, our elicitation strategy is much more informative when using our regret-based branching strategy rather than the random branching strategy.

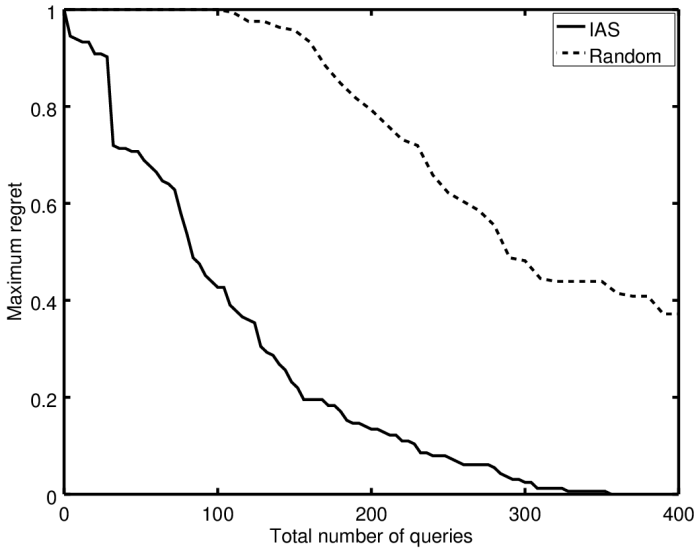

Figure 2: Maximum regret attached to the incumbent with respect to the total number of queries ($p = 12$, $d = 0.4$).

6 Conclusion

We have presented a new approach for incremental approval voting on combinatorial domains, illustrated on the knapsack problem. The first specificity of this approach is to exploit compact numerical representations of individual preferences (additive utilities) to propose more efficient elicitation sequences. Thus, learning that agent i approves or not solution x is no longer an isolated preference information; it induces a constraint on the utility space possibly reducing the set of weak-orders consistent with the observed preferences. This contributes to derive implicitly other approval judgements on other knapsacks, which saves many preference queries.

The second specificity of our approach is to interleave preference elicitation and search. This makes it possible to elicit preferences on a combinatorial set implicitly defined with a twofold benefit in view of winner determination: on the one hand, working on partial instances of feasible solutions in the search tree facilitates the identification of relevant preference queries and relieves the elicitation burden. On the other hand, the search is earlier focused on the relevant part of the solution space due to the integration of new preference information at any decisive step of the search algorithm, which saves a significant part of the computational effort. This enables to solve large instances for which the systematic elicitation of all approval statements is not feasible.

We see several directions to extend this work. The first one is to relax the additivity of individual utilities so as to be able to model interactions between items. In this line, it seems natural to use generalized additive utilities functions (GAI) that could also be elicited incrementally [Braziunas and Boutilier, 2007]. Another direction would be to consider alternative voting rules using possibly more information than approval statements. For example, positional scoring rules may be worth investigating under the assumption that individual preferences are representable by additive utilities. This is a challenging issue because, when preferences are only partially known, the ranges of possible ranks of solutions in individual rankings is generally too large to be decisive. Similar difficulties may also occur with other standard voting rules (e.g., single transferable voting, Copeland's rule) when applied to a combinatorial problem.

References

N. Benabbou and P. Perny. Incremental Weight Elicitation for Multiobjective State Space Search. In *Proceeding of AAAI'15*, pages 1093–1099, 2015.

N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, pages 475–519, 2013.

C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

S. J. Brams and P. C. Fishburn. Approval voting. *The American Political Science Review*, 72(3): 831–847, 1978.

F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

D. Braziunas and C. Boutilier. Minimax regret based elicitation of generalized additive utilities. In *Proceedings of UAI'07*, pages 25–32, 2007.

U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of AAAI'00*, pages 363–369, 2000.

J. R. Chamberlin and P. N. Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review*, 77(03):718–733, 1983.

L. N. Dery, M. Kalech, L. Rokach, and B. Shapira. Reaching a joint decision with minimal elicitation of voter preferences. *Information Sciences*, 278:466–487, 2014.

N. Ding and F. Lin. Voting with partial information: what questions to ask? In *Proceedings of AAMAS'13*, pages 1237–1238, 2013.

E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. In *Proceedings of AAMAS'14*, pages 53–60, 2014.

M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence Journal*, 174(3-4):270–294, 2010.

M. Kalech, S. Kraus, and G. A. Kaminka. Practical voting rules with partial information. *Autonomous Agents and Multi-Agent Systems*, 22(1):151–182, 2010.

D. M. Kilgour. Approval balloting for multi-winner elections. In *Handbook on approval voting*, pages 105–124. Springer, 2010.

C. Klamler, U. Pferschy, and S. Ruzika. Committee selection under weight constraints. *Mathematical Social Sciences*, 64(1):48–56, 2012.

K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, 2005.

J. Lang, M. S. Pini, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh. Winner determination in voting trees with incomplete preferences and weighted votes. *Autonomous Agents and Multi-Agent Systems*, 25(1):130–157, 2012.

T. Lu and C. Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of IJCAI'11*, pages 287–293, 2011a.

T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of IJCAI'11*, volume 11, pages 280–286, 2011b.

T. Lu and C. Boutilier. Multi-winner social choice with incomplete preferences. In *Proceedings of IJCAI'13*, pages 263–270, 2013.

R. Meir, A. D. Procaccia, J. S. Rosenschein, and A. Zohar. Complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research (JAIR)*, 33:149–178, 2008.

B. L. Monroe. Fully proportional representation. *American Political Science Review*, 89(04):925–940, 1995.

J. Oren and B. Lucier. Online (budgeted) social choice. *Proceedings of AAAI'14*, pages 1456–1462, 2014.

R. F. Potthoff and S. J. Brams. Proportional representation broadening the options. *Journal of Theoretical Politics*, 10(2):147–178, 1998.

A. D. Procaccia, J. S. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. In *Proceedings of AAAI'15*, pages 2131–2137, 2015a.

P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. The complexity of fully proportional representation for single-crossing electorates. *Theoretical Computer Science*, 569:43–57, 2015b.

T. Wang and C. Boutilier. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. In *Proceedings of IJCAI-03*, pages 309–316, 2003.

L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research (JAIR)*, 41:25–67, 2011.