Active Preference Learning Based on Generalized Gini Functions: Application to the Multiagent Knapsack Problem

Nadjet Bourdache, Patrice Perny
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6, LIP6
F-75005 Paris, France

Abstract

We consider the problem of actively eliciting preferences from a Decision Maker supervising a collective decision process in the context of fair multiagent combinatorial optimization. Individual preferences are supposed to be known and represented by linear utility functions defined on a combinatorial domain and the social utility is defined as a *generalized Gini Social evaluation Function* (GSF) for the sake of fairness. The GSF is a non-linear aggregation function parameterized by weighting coefficients which allow a fine control of the equity requirement in the aggregation of individual utilities. The paper focuses on the elicitation of these weights by active learning in the context of the fair multiagent knapsack problem. We introduce and compare several incremental decision procedures interleaving an adaptive preference elicitation procedure with a combinatorial optimization algorithm to determine a GSF-optimal solution. We establish an upper bound on the number of queries and provide numerical tests to show the efficiency of the proposed approach.

1 Introduction

Fair multiagent optimization problems over combinatorial domains appear in various contexts studied in Artificial Intelligence such as resource allocation and sharing of indivisible items (Bouveret and Lang 2008; Chevaleyre, Endriss, and Maudet 2017; Bouveret et al. 2017), assignment problems (Lesca and Perny 2010; Aziz et al. 2014; Heinen, Nguyen, and Rothe 2015; Gal et al. 2017), multiobjective state-space search (Galand and Spanjaard 2007) and dynamic social choice (Freeman, Zahedi, and Conitzer 2017), but also in other optimization contexts such as facility location (Ogryczak 2009). One common characteristic in all these problems is the focus on fairness: we are looking for a solution allowing a balanced distribution of satisfaction on the different agents.

Defining precisely what is a fair solution in a collective decision context is a matter of preferences and subjectivity. One can put more or less emphasis on the notion of fairness in preference aggregation. This leads to a wide range of attitudes from pure egalitarism (maximization of the minimum of the individual utility functions) to softer notions combining the objective of reducing inequalities and that of preserving a good overall efficiency. One common way of introducing fairness in Social Choice is to require the preference to be *t-monotonic*, i.e., monotonic with respect to mean preserving *transfers* reducing inequalities (a.k.a. Pigou-Dalton transfers). More formally, given a utility vector $x = (x_1, \dots, x_n)$ where x_i is the value of solution x according to agent i, any modification of x leading to a vector of the form $(x_1, \dots, x_i - \varepsilon, \dots, x_j + \varepsilon, \dots, x_n)$ for some i, j, ε such that $x_i - x_j > \varepsilon > 0$ should makes the Decision Maker (DM) better off. Under the Pareto principle (requiring monotonicity in every component) and some other mild requirements such as completeness, requiring t-monotonicity of the social preference leads to define the social utility as a *Generalized Gini social-evaluation Function (GSF)*, i.e., $f_\alpha(x) = \sum_{i=1}^{n} \alpha_i x_{(i)}$, as shown in (Blackorby and Donalson 1978; Weymark 1981), where $x_{(i)}$ is the i^{th} smallest component of x and $\alpha_1 \geq \cdots \geq \alpha_n \geq 0$ (more weight is attached to the least satisfied agents). In other words, $f_\alpha(x)$ is an *ordered weighted average* (Yager 1998) of individual utilities x_i defined using decreasing weights.

The GSF f_α is parameterized by the weighting vector $\alpha = (\alpha_1, \dots, \alpha_n)$ allowing the control of the importance attached to agents according to their rank of satisfaction. The parameterized function f_α includes several well-known social evaluation functions as special cases. For example, if $\alpha_1 = 1$ and $\alpha_i = 0$ for all $i > 1$ then only the least satisfied agent matters and we obtain the egalitarian criterion. On the other hand, if $\alpha_i = 1$ for all i then all components have the same importance and we obtain the utilitarian criterion. Many nuances are possible between these two extremities, leading to various possible tradeoffs.

Eliciting the preferences of the DM on some pairs of solutions will certainly contribute to reveal the implicit tradeoff she makes between the need of overall efficiency and the aim of fairly distribute the satisfaction of individuals. So, assuming that the overall utility of solutions is defined using f_α, we need a preference elicitation procedure allowing to know the α vector sufficiently precisely to be able to determine the optimal solution from individual preferences. The aim of our paper is to introduce interactive decision procedures combining the incremental elicitation of vector α modeling the attitude of the DM towards fairness and the determination of a fair and Pareto-optimal solution (maximizing f_α). For the sake of illustration, we develop this study on the multi-agent

0-1 Knapsack problem (MKP), a multiobjective version of the standard knapsack problem in which objectives are linear utility functions representing the agents' preferences.

The paper is organized as follows. In Section 2 we recall some background on the MKP. In Section 3 we briefly review some related work. Then, in Section 4 we recall the basics of usual incremental weight elicitation methods based on the minimization of regrets and discuss their application to the elicitation of the weighs defining α. In Section 5 we present an interactive branch and bound algorithm in which preference elicitation is combined with implicit enumeration to determine a necessary optimal knapsack. Then, in Section 6, we propose an alternative approach for regret optimization leading to a faster elicitation procedure. We also propose a variant with bounded query complexity. Finally, in Section 7 we provide numerical tests to show the practical efficiency of the proposed approach.

2 The Fair Multiagent Knapsack Problem

We consider a decision situation involving a set of n agents and p items. Every item k is characterized by a positive weight w_k and a vector of n positive utilities $(u_k^1, ..., u_k^n)$ where $u_k^i \in [0, M]$ is the utility assigned to item k by agent i, and M is a constant representing the top of the utility scale. A solution is then a subset of items represented by a binary vector $z = (z_1, ..., z_p)$, where $z_k = 1$ if item k is selected and $z_k = 0$ otherwise. Feasible subsets are characterized by a capacity constraint of the form $\sum_{k=1}^{p} w_k z_k \leq W$ where W is a positive value bounding from above the total weight of admissible solutions. The utility of any solution z is then characterized by a utility vector $x(z) = (x_1(z), \ldots, x_n(z))$ where $x_i(z) = \sum_{j=1}^{p} u_j^i z_j$ measures the satisfaction of agent i with respect to solution z. The MKP problem is then a multiobjective optimization problem that consists in maximizing utilities $x_i, i = 1, \ldots, n$. This problem has multiple applications in various decision contexts involving multiple agents such as voting (election of a committee of bounded size), resource allocation (distributing resources under a capacity constraint) and transportation (optimal cargo problems with competing demands).

Example 1. *We consider an instance of MKP with* 3 *agents, a maximal weight* $W = 48$ *and* 7 *items whose weights and utilities are given in the following table:*

k	1	2	3	4	5	6	7
w_k	6	5	6	11	13	15	12
u_k^1	5	20	17	16	13	1	4
u_k^2	6	18	3	3	20	12	17
u_k^3	11	0	13	17	4	10	3

If we adopt the utilitarian approach consisting of maximizing the sum $x_1(z) + x_2(z) + x_3(z)$ *under the knapsack constraint, we obtain a standard knapsack problem. The optimal solution is* $z_1^* = (0, 1, 1, 1, 1, 0, 1)$ *leading to utility vector* $x(z_1^*) = (70, 61, 37)$. *Although the average satisfaction is maximal (*56*), we remark that the utility vector is ill-balanced and this solution may be considered as unfair. As far as fairness is concerned, we may prefer maximizing* $\min\{x_1(z), x_2(z), x_3(z)\}$ *which leads to solution* $z_2^* = (1, 0, 1, 1, 1, 0, 1)$ *with utility vector* $x(z_2^*) = (55, 49, 48)$, *a much better balanced utility vector. However the average satisfaction is now lower than* 51. *An intermediary option between these two solutions is* $z_3^* = (1, 1, 1, 1, 1, 0, 0)$ *leading to* $x(z_3^*) = (71, 50, 45)$ *which improves the worse component of* $x(z_1^*)$ *while keeping a near-optimal average* (55).

This example shows that various tradeoffs can be proposed in the MKP depending on the compromise we want to make between fairness and average efficiency. Interestingly, in the instance of Example 1, the f_α-optimal solution for $\alpha = (1, \frac{2}{3}, \frac{1}{3})$ is neither z_1^* (utilitarian optimum) nor z_2^* (egalitarian optimum) but z_3^*. This illustrates how f_α could be used to generate a soft compromise between the utilitarian and egalitarian optima.

Ordered weighted averages such as function f_α are increasingly used in AI to obtain fair solutions in multiobjective optimization problems, see e.g. (Galand and Spanjaard 2007; Golden and Perny 2010; Goldsmith et al. 2014; Heinen, Nguyen, and Rothe 2015). Interestingly, they are also used with different assumptions on weights in multiwinner voting for proportional representation see e.g. (Elkind and Ismaili 2015; Skowron, Faliszewski, and Lang 2016) and in the MKP to define the value of a set of items (Fluschnik et al. 2017). In this paper, we consider the problem of finding a solution to the MKP that maximizes function f_α, α having decreasing components, formally:

$$\max_x f_\alpha(x_1, \ldots, x_n)$$
$$\begin{cases} x_i = \sum_{k=1}^{p} u_k^i z_k & \forall i \in [\![1, n]\!] \\ \sum_{k=1}^{p} w_k z_k \leq W \\ z_k \in \{0, 1\} & \forall k \in [\![1, p]\!] \end{cases} \tag{1}$$

The general problem defined in (1) is NP-hard. It indeed includes the standard knapsack problem (which is known to be NP-hard) as special case, when all coefficients α_i are equal and positive. Note that if the f_α function is replaced by a product (Nash welfare), then the problem remains NP-hard (Fluschnik et al. 2017). Moreover, well-known pseudo-polynomial solution methods proposed for the standard knapsack problem (based on dynamic programming) are not valid for ordered weighted averages due to non-linearity. Let us give an example:

Example 2. *Consider an instance of (1) with* $p = 3$ *items* $\{1, 2, 3\}$ *with weights* $w_1 = w_2 = w_3 = 1$ *and* $n = 2$ *agents with utilities* $(u_1^1, u_2^1, u_3^1) = (10, 5, 0)$ *and* $(u_1^2, u_2^2, u_3^2) = (0, 5, 10)$. *Assume that* $W = 2$ *and* $\alpha = (1, \frac{1}{2})$. *If we evaluate the subsets of size 1 we obtain the value* $f_\alpha(10, 0) = 5$ *for* $\{1\}$, $f_\alpha(5, 5) = 7.5$ *for* $\{2\}$ *and* $f_\alpha(0, 10) = 5$ *for* $\{3\}$. *For the subsets of size 2 we obtain the value* $f_\alpha(15, 5) = 12.5$ *for* $\{1, 2\}$, $f_\alpha(10, 10) = 15$ *for* $\{1, 3\}$ *and* $f_\alpha(5, 15) = 12.5$ *for* $\{2, 3\}$. *We can see that no* f_α*-optimal singleton is included in the optimal pair* $\{1, 3\}$.

This example shows that the f_α-optimal knapsack cannot simply be constructed from f_α-optimal subsets. However, for any α with decreasing components, an f_α-optimal knapsack can be obtained by mixed integer programming, due to a smart linearization of function f_α (Ogryczak and Śliwiński 2003) making it possible to solve reasonably large instances in a few seconds. Moreover, there are some tractable cases when we consider integer weights w_k and a constant W (as in a committee election problem of fixed-size; all items have the same weight). In such cases, the number of subsets of size at most W is indeed polynomial in p when W is a constant. Therefore the f_α value of every subset can be computed in polynomial time, yielding the optimal subset.

When α is not known, we must solve the twofold problem of eliciting α and determining an f_α-optimal subset. We propose hereafter an adaptive approach interleaving preference queries with the exploration of feasible subsets.

3 Related Work

A first stream of related work concerns incremental preference elicitation methods for decision making. Incremental elicitation relies on the idea of interleaving preference queries with the exploration of feasible solutions to focus the elicitation burden on the useful part of preference information (active learning). This idea has been succesfully used in AI for decision support. There are indeed various contributions concerning model-based incremental decision-making on explicit sets, see e.g., (White, Sage, and Dozono 1984; Ha and Haddawy 1997; Chajewska, Koller, and Parr 2000; Wang and Boutilier 2003; Braziunas and Boutilier 2007; Benabbou, Perny, and Viappiani 2017), and also some contributions for incremental decision making on implicit sets (combinatorial optimization problems), mainly focused on linear aggregation models, see e.g., (Boutilier et al. 2006; Gelain et al. 2010; Weng and Zanuttini 2013; Benabbou and Perny 2015b; Brero, Lubin, and Seuken 2018). The problem under consideration here is more challenging because on the one hand the set of feasible knapsacks is *implicitly defined* by a capacity constraint and, on the other hand, GSF is a *non-linear* aggregation function which prevents to combine local preference elicitation with a dynamic programming algorithm (because preferences induced by a GSF do not satisfy the Bellman principle). There are a few recent attempts to face these two difficulties simultaneously (non-linearity of the decision criterion and the implicit definition of the solution space) on other problems, e.g., the multiobjective shortest paths and assignment problems (Benabbou and Perny 2015a; Bourdache and Perny 2017) but, to the best of our knowledge, no solution was proposed for the MKP.

Preference elicitation has been recently considered in the MKP, but under a different perspective that consists of eliciting agent's preferences. For instance, in the context of approval voting, an incremental utility elicitation procedure is proposed in (Benabbou and Perny 2016) to reduce the uncertainty about individual utilities until the winning subset can be determined. Another recent study concerns implicit voting and aims to approximate the optimal knapsack by eliciting preference orders induced by individual utilities and applying voting rules (Benade et al. 2017). In both of these works the preference aggregation rule is known and the elicitation concerns individual values. In this paper the setting is different. Individual utilities are assumed to be known and the elicitation part aims to capture the attitude of the DM (supervising the collective decision process) with respect to fairness and to define the social utility. Finally, the fairness issue was recently addressed in the MKP problem in (Fluschnik et al. 2017) where various aggregation rules are studied including the Nash product but the elicitation of the social utility function is not investigate in that work.

4 Incremental Elicitation of GSF Weights

We first explain how standard elicitation schemes based on regret minimization (Wang and Boutilier 2003; Boutilier et al. 2006) can be implemented to elicit the α weighting vector in order to determine an f_α-optimal element on an explicit set of solutions. We will then underline the main issue to overcome to extend the approach to implicit sets defined by an instance of the knapsack problem.

At the beginning of the elicitation process, we assume that no information about α is available except that $\alpha \in \mathbb{R}^n$ and $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n \geq 0$ (to enforce monotonicity and fairness, as previously explained). To bound the set of possible α, we can also add a (non-restrictive) normalization condition of type $\max_i \alpha_i = 1$ or $\sum_i \alpha_i = 1$. These constraints define the initial *uncertainty set* A containing all admissible α vectors. The set A is then reduced using the constraints derived from preference statements collected during the elicitation process. Any new statement of type "x is preferred to y" translates into the constraint $f_\alpha(x) \geq f_\alpha(y)$. It is important to remark that, although f_α is not linear in x, it is linear in α for a fixed x. Hence, preferring x to y leads to the linear inequality $\sum_{i=1}^n \alpha_i(x_{(i)} - y_{(i)}) \geq 0$ defining an half-space in $\mathbb{R}^n$. Once several preference statements have been collected, A is reduced to the bounded convex polyhedron defined as the intersection of all the half-spaces obtained from preference statements.

In order to measure the impact of uncertainty A on the evaluation of solutions, one can define the max-regret (MR) of choosing solution x in X as the maximum utility loss against all possible alternative choices. Under the GSF model assumption, it is formally defined for all $x \in X$ by:

$$\mathrm{MR}(x, X, A) = \max_{y \in X} \max_{\alpha \in A} \{f_\alpha(y) - f_\alpha(x)\} \qquad (2)$$

Remark that $\mathrm{MR}(x, X, A) \geq 0$ since $y = x$ is allowed. Given the uncertainty set A, a robust recommendation would be to select in X the solution x^* minimizing $\mathrm{MR}(x, X, A)$ over X. This solution is named the *minimax-regret solution* and the associated MR value (namely the *minimax regret*) is defined by $\mathrm{MMR} = \min_{x \in X} \mathrm{MR}(x, X, A)$. Note that whenever MMR = 0 then x^* is a necessary optimal solution because, by definition, $f_\alpha(y) - f_\alpha(x^*) \leq 0$ for all $y \in X$ and all $\alpha \in A$. When MMR is strictly positive one can collect new preference statements in order to further reducing the uncertainty set A and therefore the MR and MMR values. In order to generate a new preference query, one may use the Current Solution Strategy (CSS) proposed in (Boutilier et al. 2006) that consists in asking the decision maker to compare the current MMR solution x^* to its best challenger y^* obtained by maximizing $f_\alpha(y) - f_\alpha(x^*)$ over all $y \in X$ and all $\alpha \in A$. Once x^* is known, y^* can easily be determined using linear programming, provided that X is explicitly defined. The generation of preference queries is iterated until the MMR value is sufficiently small.

We have seen that MR computations are in the core of the elicitation procedure, both for measuring the impact of uncertainty on the quality of the MMR solution, and for selecting the next preference query. However, when the set of feasible utility vectors X is implicitly defined, x is a vector of decision variables x_i defined in (1) and $f_\alpha(x)$ includes quadratic terms of type $\alpha_i x_{(i)}$. Hence the computation of MR values requires non-linear optimization and the computation of MMR requires min-max quadratic optimization. This is critical because MR and MMR values must be recomputed at every step of the elicitation process. Let us present two different ways to overcome this problem.

5 Interactive Branch and Bound

In the MKP introduced in (1), the set of feasible utility vectors is defined by: $X = \{x(z), z \in \{0,1\}^p : \sum_{k=1}^{p} w_k z_k \leq W\}$ and possibly includes a huge number of elements. In order to implement the CSS strategy on such a combinatorial domain, we first introduce an interactive branch and bound algorithm with a depth first search strategy. It consists in the implicit enumeration of feasible knapsacks by exploration of a binary search tree. Each node of the tree is a subproblem of its parent node, it represents a subset of feasible knapsacks defined by some constraints on variables z_i (see (1)). More precisely, the algorithm consists of two interleaved phases executed iteratively: a branching phase that progressively develops the nodes to construct the exploration tree, and a bounding phase enabling to decide whether or not a given node is worth developing. We present now these two phases.

Bounds. At every node η of the search tree, two bounds are computed, an upper bound UB(η) and a lower bound LB(η) on feasible f_α values attached to node η. The definition of UB(η) is based on the following well-known result:

Proposition 1. *For any weighting vector $\alpha \in \mathbb{R}^n_+$ with decreasing weights and such that $\sum_{i=1}^{n} \alpha_i = 1$, we have for all $x \in \mathbb{R}^n, f_\alpha(x) \leq \frac{1}{n}\sum_{i=1}^{n} x_i$.*

Thus an upper bound can be obtained by maximizing $\frac{1}{n}\sum_{i=1}^{n} x_i$ over the set of feasible knapsacks associated to η. Note that this upper bound is independent of α which is practical here since α is unknown. This bound is the optimal value of a standard knapsack problem at node η denoted KP(η). This problem can be solved by dynamic programming. Actually, to reduce computation times, we did not use this bound but a relaxed upper bound UB(η) obtained by solving the continuous relaxation of KP(η). This can be done very efficiently using a well-known greedy algorithm that consists in repeatedly selecting items in decreasing order of values $\overline{u}_k/w_k$ (where $\overline{u}_k = \frac{1}{n}\sum_{i=1}^{n} u_k^i$ is the average utility of item k) until there is no space left for the next item. This last item is then truncated to reach the maximum weight W. This leads to a solution z^* having at most one fractional component which is actually the optimal solution of the relaxed knapsack problem.

The lower bound LB(η) is defined from the minimal f_α value that can be reached by some feasible solution present at node η. In order to obtain a non-trivial bound, we use solution z_* obtained by considering the integer solution corresponding to z^* defined above after removing the fractional item. We then compute the smallest f_α value for $x(z_*)$ by solving the linear program $\min_{\alpha \in A} f_\alpha(x(z_*))$.

Branching Scheme. Every node η implicitly represents a set of feasible solutions of the knapsack problem, characterized by elementary decisions ($z_k = 1$ or $z_k = 0$) made on some items $k \in [\![1, p]\!]$. When η is developed (following the depth first search strategy), two children η' and η'' are generated, partitioning the set of feasible solutions attached to η. To create these nodes, we consider solution z^* introduced above for computing UB and we branch on the unique fractional variable z_r in z^* by setting either $z_r = 1$ or $z_r = 0$.

Pruning Rules: Let O denote the set of nodes that have been generated but not yet developed at a given step of the algorithm. Our first pruning rule (detnoted PR1) is quite standard and relies on bounds UB(η) and LB(η), for all $\eta \in O$.
PR1: Let η^* be the node with the higher LB in O. We prune all nodes η such that UB(η) $\leq$ LB(η^*) because no solution in the search tree rooted in η can improve the value $f_\alpha(x(z_*(\eta^*)))$ obtained at node η^*.

At any step of the algorithm, the computations of bounds LB(η) for all $\eta \in O$ yields a set of feasible knapsacks defined by $\{z_*(\eta)), \eta \in O\}$. The set $S = \{x(z_*(\eta)), \eta \in O\}$ of utility vectors attached to these solutions provides a sample of feasible utility vectors. Our second pruning rule (denoted PR2) consists in filtering this set S to keep only the potentially optimal solutions. A solution $x \in S$ is said to be *potentially optimal* in S if there exists $\alpha \in A$ such that, $\forall y \in S$, $f_\alpha(x) \geq f_\alpha(y)$. Hence the second pruning rule is:

PR2: Prune any solution $x \in S$ that is not potentially optimal in S. This amounts to testing whether $\min_{\alpha \in A} \max_{y \in S} f_\alpha(y) - f_\alpha(x) > 0$ which can easily be done by linear programming since S is finite.

Incremental Elicitation. The branch and bound procedure presented above progressively collects in S all possibly optimal solutions. To reduce the cardinality of this set periodically during the process, we use the CSS strategy recalled at the end of Section 3 to generate preference queries as soon as $|S|$ exceeds a given threshold (e.g., 5 or 10 elements), and once more at the final step. Thus, the computational cost of the CSS strategy is reduced because it only applies to set S whose size is bounded by a constant rather than to the entire set X. The new preference statements obtained allow further reductions of A and therefore of MR and MMR values on the elements of S. Preference queries are repeatedly generated until MMR = 0. Thus, all solutions $x \in S$ such that MR(x, S, A) > 0 are removed from S. We combine this adaptive elicitation process with the branch and bound exploration to progressively reduce the set A and therefore the set of possibly optimal solutions. When this process terminates, S provides at least one necessary optimal solution:

Proposition 2. *Let A_0 be the initial uncertainty set. The interactive branch and bound presented above terminates with an uncertainty set $A \subseteq A_0$ and a solution set S such that: $\forall s \in S, \forall \alpha \in A, f_\alpha(s) = \max_{x \in X} f_\alpha(x)$.*

Proof. First, let us remark that without pruning and filtering, our algorithm would output a set S containing all feasible knapsacks (explicit enumeration). Let η be a node pruned by PR1 at some step of the procedure and let A_η denote the uncertainty set at this step. Clearly, $A \subseteq A_\eta \subseteq A_0$. Moreover, for all $y \in \eta$ and all $\alpha \in A_\eta$ we have $f_\alpha(y) \leq \text{UB}(\eta) \leq \text{LB}(\eta^*) \leq f_\alpha(y')$ for some $y' \in \eta^*$. The inequality $f_\alpha(y) \leq f_\alpha(y')$ holds in particular for all $\alpha \in A$ which shows that y' is as least as good as y, thus justifying the pruning of y. Note that removing y does not weaken our pruning possibilities for the next iterations due to the presence of y' and the transitivity of preferences. The same arguments holds if y' is later removed due to another solution y'' and so on. Similarly, for any x removed in S using PR2 and an uncertainty set A_x we know that for all $\alpha \in A_x$ there exists x' such that $f_\alpha(x) \leq f_\alpha(x')$. This remains true whenever A_x is reduced to A. Here also, removing x does not weaken our pruning possibilities for the next iterations due to the presence of x' and transitivity. Thus, all pruning operations made at any step using PR1 or PR2 remain valid for the final set A. Moreover, when the procedure stops, the MMR value is equal to 0. Hence, all remaining solutions in S are f_α-optimal, for all $\alpha \in A$ which completes the proof. □

6 Extreme Points Exploration

The second approach we introduce is based on the direct optimization of regrets on X. To overcome the problem on nonlinearity of MR functions (discussed at the end of Section 4) we propose a method based on the exploration of the extreme points of the uncertainty polyhedron A. Due to the convexity of A and the linearity of $f_\alpha(x)$ in parameter α, optimal MR values are necessarily obtained on one of the extreme points of A. Thus, in the MR and MMR computations, we can restrict the possible instances of α to this finite set of points. This enables to change the quadratic formulation of regrets into a linear one. More precisely, we have:

Proposition 3. *Let A^{ext} be the set of all extreme points of A and $m = |A^{ext}|$, then the MMR value on X can be obtained by solving only m+1 mixed integer linear programs.*

Proof.
$$\begin{aligned}\text{MMR} &= \min_{x\in X}\max_{y\in X}\max_{\alpha\in A}\{f_\alpha(y) - f_\alpha(x)\}\\ &= \min_{x\in X}\max_{y\in X}\max_{\alpha\in A^{\text{ext}}}\{f_\alpha(y) - f_\alpha(x)\}\\ &= \min_{x\in X}\max_{\alpha\in A^{\text{ext}}}\max_{y\in X}\{f_\alpha(y) - f_\alpha(x)\}\\ &= \min_{x\in X}\max_{\alpha\in A^{\text{ext}}}\{f^*_\alpha - f_\alpha(x)\}\end{aligned}$$
where $f^*_\alpha = \max_{y\in X} f_\alpha(y)$ is obtained for any $\alpha \in A^{\text{ext}}$ by solving a mixed integer linear program using a linearization of f_α proposed in (Ogryczak and Śliwiński 2003). Thus $|A^{\text{ext}}| = m$ such optimizations must be peformed. Then, the MMR value is obtained by solving: $\min t$ subject to the constraints $\{t \geq f^*_\alpha - f_\alpha(x),\ \forall\ \alpha \in A^{\text{ext}}, x \in X\}$. Thus, we get the MMR value after $(m + 1)$ optimizations. □

Algorithm. The idea of our algorithm is to interleave preference queries with MMR computations until the MMR value drops to zero (see Algorithm 1). Note that with the initial definition of A given in line 1 the set A^{ext} contains only n elements (all vectors whose i first components are equal to 1, the others being equal to zero). The answer to the preference query generated in line 4 allows a reduction of the MMR

Algorithm 1:

```
1 $A \leftarrow \{\alpha \in \mathbb{R}^n_+ : \alpha_1 = 1\ ,\ \alpha_i \geq \alpha_{i+1} \forall i\}$
2 Compute $A^{\text{ext}}$
3 repeat
4     Ask a query to the DM
5     Restrict $A$ according to the answer
6     Update $A^{\text{ext}}$
7 until $MMR \leq \Delta$;
8 return $x^*$ = arg $\min_{x\in X}$ MR$(x, X, A^{\text{ext}})$
```

value, and this can be iterated until MMR $\leq \Delta$, where Δ is a positive acceptance threshold. If $\Delta = 0$ then the algorithm yields an optimal solution, otherwise we only obtain a near-optimal solution. We propose below two different strategies for generating preference queries:

Strategy S1: The first strategy is focused on the fast reduction of MMR values. It uses the *CSS* strategy recalled at the end of Section 4. It makes it possible to determine a necessary winner or almost necessary winner while parameter α remains largely imprecise, thus saving a lot of queries. This strategy is illustrated below on Example 1.

Example 2 (continued). *If we apply Algorithm 1 to Example 2 with strategy S1, the initial uncertainty set A is pictured on Figure 1.a. To simulate the DM's answers we use the weighting vector $(1, 2/3, 1/3)$ (the dark point in the figures). The MMR value is computed a first time and we obtain MMR $= 3 > 0$. So, the DM is asked to compare $x^* = \arg\min_{x\in X}$ MR(x, A, X) $= (71, 50, 45)$ obtained from $z_1 = (1, 1, 1, 1, 1, 0, 0)$ and y^* = $\arg\max_{y\in X}\max_{\alpha\in A} f_\alpha(y) - f_\alpha(x^*) = (55, 49, 48)$ obtained from $z_2 = (1, 0, 1, 1, 1, 0, 1)$. Assume the DM prefers x^*, then we add $f_\alpha(x^*) \geq f_\alpha(y^*)$ to the definition of A, leading to a smaller uncertainty set pictured in Figure 1.b. The MMR value is computed again and we find MMR $= 2 > 0$, the DM compare $x^* = (71, 50, 45)$ and $y^* = (70, 61, 37)$ obtained from z_1 and $z_3 = (0, 1, 1, 1, 1, 0, 1)$ respectively. Assume the DM still prefers x^*, then the constraint $f_\alpha(x^*) \geq f_\alpha(y^*)$ is added to the definition of A. The final set A is pictured in Figure 1.c. The MMR is updated and we find MMR $= 0$ so the algorithm stops. By construction, solution x^* remains f_α-optimal whatever $\alpha \in A$.*

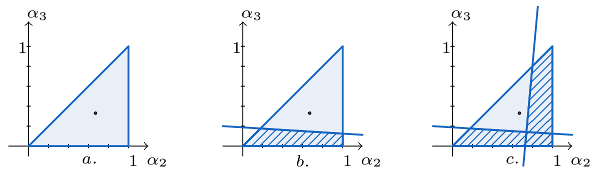

Figure 1: Evolution of the uncertainty set A

Strategy S2: This strategy is focused on a methodic division of the uncertainty set, which indirectly entails a reduction of MMR values. Every component $\alpha_i, i \in [\![1, n]\!]$ is bounded by an interval $I_i = [l_i, u_i]$ of possible values (initially $[0, 1]$). At every step, we select the largest interval I_i and we generate a preference query enabling to remove the half of the interval. More precisely, we ask the DM to compare the two utility vectors x and y defined as follows:

$$x = (0, \underbrace{\frac{\lambda c}{1+\lambda}, \dots, \frac{\lambda c}{1+\lambda}}_{\times(i-2)}, \underbrace{c, \dots, c}_{\times(n-i+1)}), \;\; y = (\underbrace{\frac{\lambda c}{1+\lambda}, \dots, \frac{\lambda c}{1+\lambda}}_{\times i}, \underbrace{c, \dots, c}_{\times(n-i)})$$

and $\lambda = (l_i + u_i)/2$ is the middle of I_i and $c \in (0, M]$. Suppose that x is preferred to y, we deduce $f_\alpha(x) \geq f_\alpha(y)$ and then:

$$\frac{\lambda c}{1+\lambda}(\sum_{j=2}^{i-1} \alpha_j) + c \sum_{j=i}^{n} \alpha_j \geq \frac{\lambda c}{1+\lambda}(\sum_{j=1}^{i} \alpha_j) + c \sum_{j=i+1}^{n} \alpha_j$$
$$\Rightarrow \; \alpha_i \geq \frac{\lambda}{1+\lambda}\alpha_1 + \frac{\lambda}{1+\lambda}\alpha_i \; \Rightarrow \; (1+\lambda)\alpha_i \geq \lambda\alpha_1 + \lambda\alpha_i$$
$$\Rightarrow \; \alpha_i \geq \lambda\alpha_1$$

Under the non-restrictive assumption that $\alpha_1 = 1$ we obtain $\alpha_i \geq \lambda = (l_i + u_i)/2$. On the other hand, whenever y is preferred to x, a similar reasoning leads to $\alpha_i \leq (l_i + u_i)/2$. In both cases we can update one of the bounds of I_i.

Example 2 (continued). *Let us now apply Algorithm 1 with strategy S2, the weighting vector used to simulate the DM's answers being still* $(1, 2/3, 1/3)$*. The initial uncertainty set A is the same as in Figure 1.a. and the first MMR value is* $3 > 0$*. We have initially* $I_2 = I_3 = [0, 1]$*, we first chose to reduce* I_2 *(arbitrarily). For this, we construct two vectors x and y as explained above (with* $i = 2$*) to generate the first preference query. The DM's answer allows to add the constraint* $w_2 \geq 0.5$ *as pictured in Figure 2.a. The MMR value is recomputed and we still find MMR* $= 3 > 0$ *so we ask a second question to the DM aiming to reduce* I_3 *since it is now bigger than* I_2*. The answer leads to the constraint* $w_3 \leq 0.5$ *which is added to the definition of A (Figure 2.b). The MMR is still positive. The process continue and two more questions will be necessary to obtain MMR=0 (the associated constraints are pictured in Figures 2.c and 2.d.).*

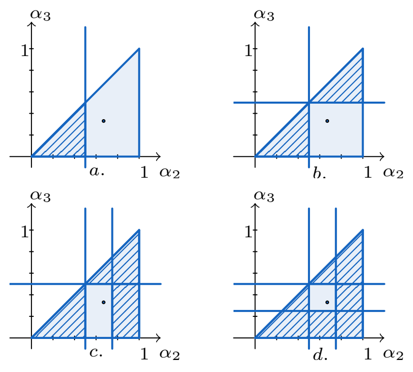

Figure 2: Evolution of the uncertainty set A

This dichotomous reduction of intervals makes it possible to bound from above the number of queries necessary to obtain a MMR smaller than Δ. Since the utilities of the p items for the n agents belong to $[0, M]$, we indeed have:

Proposition 4. *Algorithm 1 used with S2 and* $\Delta = \delta npM$ *requires at most* $n\lceil \log_2(1/\delta) \rceil$ *queries for any* $\delta \in (0, 1]$.

Proof. Let us first prove that if $u_i - l_i \leq \delta$ for all $i \in [\![1, n]\!]$ then MMR $\leq \Delta$. Let α' be a vector arbitrary chosen in A at the current step and x' the solution maximizing $f_{\alpha'}$ over X. For any solution $y \in X$ we have:

$$\max_{\alpha \in A}\{f_\alpha(y) - f_\alpha(x')\} + f_{\alpha'}(x') - f_{\alpha'}(y)$$
$$= \max_{\alpha \in A}\{(f_\alpha(y) - f_{\alpha'}(y)) - (f_\alpha(x') - f_{\alpha'}(x'))\}$$
$$= \max_{\alpha \in A}\{\textstyle\sum_{i=1}^{n} \underbrace{(\alpha_i - \alpha'_i)}_{\leq \delta} y_{(i)} - \sum_{i=1}^{n} \underbrace{(\alpha_i - \alpha'_i)}_{\leq \delta} x'_{(i)}\}$$
$$\leq \delta \textstyle\sum_{i:\alpha_i > \alpha'_i} y_{(i)} + \delta \sum_{i:\alpha_i < \alpha'_i} x'_{(i)} \;\leq\; \delta npM = \Delta$$

The term $f_{\alpha'}(x') - f_{\alpha'}(y)$ being positive (x' being optimal for α'), then $\max_{\alpha \in A}\{f_\alpha(y) - f_\alpha(x')\} \leq \Delta$. This reasoning being true for any solution y, then MR$(x', X, A) \leq \Delta$. The MMR value being obtained by minimizing MR(x, X, A) over all $x \in X$, we obtain MMR $\leq \Delta$.

Moreover, note that we initially have $I_i = [0, 1]$, for all $i \in [\![1, n]\!]$ due to line 1 of Algorithm 1. Hence, the iterative dichotomous reduction of an interval I_i requires at most $\lceil \log_2(1/\delta) \rceil$ queries to obtain $u_i - l_i \leq \delta$ for any $i \in [\![1, n]\!]$ which gives $n\lceil \log_2(1/\delta) \rceil$ overall. □

Note that this bound is based on the worst case scenario. In practice it frequently happens that the MMR drops to 0 before reducing the size of all intervals to δ which saves multiple preference queries.

Example 3. *If* $n = 10, p = 50$ *and* $M = 20$ *then* $npM = 10000$ *represents the range of possible* f_α *values on subsets and therefore the range of possible MMR values. Parameter* δ *represents the maximal percent error allowed on* f_α *values. If we accept a 5% error (*$\delta = 0.05$*) it gives* $\Delta = 500$ *and* $10\lceil \log_2(1/0.05) \rceil = 50$*, therefore we need at most 50 preference queries. We will see in the next section that only 17 preference queries are used on average.*

7 Experimental Results

We have tested the interactive branch and bound (IBB) introduced in Section 5 and Algorithm 1 introduced in Section 6 (used with $\Delta = 0$ to obtain exact solutions). To generate instances, utility and weights are generated at random using an uniform density. The capacity W is randomly drawn around a reference value defined as the half of the weight of the entire set of items. This allows to favor the generation of hard knapsack instances. We generate instances of different sizes involving up to 100 items and up to 10 agents. The DM's answers are simulated using an hidden f_α function (α is randomly chosen). The linear programs formulated for regrets computations are solved using the Gurobi library of Python. The set A^{ext} is computed (lines 2 and 6 of Algorithm 1) using the cdd Python's library, which is an implementation of the *double description method* introduced in (Fukuda and Prodon 1996). The tests are performed on a Intel(R) Core(TM) i7-4790 CPU with 15 GB of RAM.

We first compare the performances of the two algorithms both in terms of computation time (in seconds) and number of preference queries (see Tables 1 and 2). The results are averaged over 30 runs with a timeout set to 20 minutes for every run. Due to the time constraint, the algorithms may be stopped before finding an optimal solution. In this case, they return the best solution found within the time window (anytime property). In particular, Algorithm 1 returns the current

Computation times (s)

n \ p	20	50	100
3	10.45	139.22 (60 %)	198.62 (37 %)
5	25.8	170.86 (67 %)	274.31 (23 %)
10	94.21 (97 %)	166.62 (27 %)	374.03 (17 %)

Number of queries

n \ p	20	50	100
3	3.3	4.7 (60 %)	5.73 (37 %)
5	3.9	5.9 (67 %)	8.0 (23 %)
10	7.1 (97 %)	7.5 (27 %)	9.6 (17 %)

Table 1: Results for Interactive Branch & Bound (IBB)

Computation times (s)

n \ p	20	50	100
3	0.2	0.9	0.6
5	0.5	1.9	5.8
10	8.3	81.8	411.7

Number of queries

n \ p	20	50	100
3	2.7	3.9	5.3
5	2.8	6.8	11.2
10	4.7	11.2	23.1

Table 2: Results for Algorithm 1 with strategy S1

MMR solution which represents a robust choice since the gap to optimality is guaranteed to be lower than the current MMR value. In the tables below, we provide both the average solution time on solved instances and the percentage of solved instances within 20 minutes (when $< 100\%$).

Tables 1 and 2 show that Algorithm 1 mostly outperforms IBB. It not only solves 100% of instances within the time constraint but runs faster on average. It also requires fewer preference queries for small or middle size instances. For big instances IBB seems to generate less preference queries but this evaluation is optimistic. The average performance only includes instances solved within the time constraint.

We next assess and compare the performances of the two elicitation strategies S1 and S2 in Algorithm 1. First S1 and S2 are compared to a random strategy S3 for the selection of preference queries. The results are given in Figure 3 that represents, for every strategy, the average MMR value (computed over 30 runs) as a function of the number of preference statements collected during the elicitation process.

In Figure 3 MMR values are given as a percentage of the initial MMR value computed before running the elicitation procedure. We can see that strategies S1 and S2 are significantly more efficient that the random strategy S3, the decrease in MMR value being faster for these two strategies. For instance, to obtain a MMR smaller than 10% of the initial value, we need about 5 queries for S1 and approximatively 17 queries for S2 while 30 preference queries would not be sufficient with S3. Wa also observed that the actual regret defined as the optimality gap achived by the current MMR solution decrease faster than the MMR values. It very quickly drops to 0, which explains that we did not plot the curve on Figure 3. Moreover, we can see that S1 is significantly faster than S2 on average. This can be explained as follows: Algorithm 1 implemented with S1 looks for preference queries aiming at quickly discriminating the solutions of the instance to be solved without trying to learn

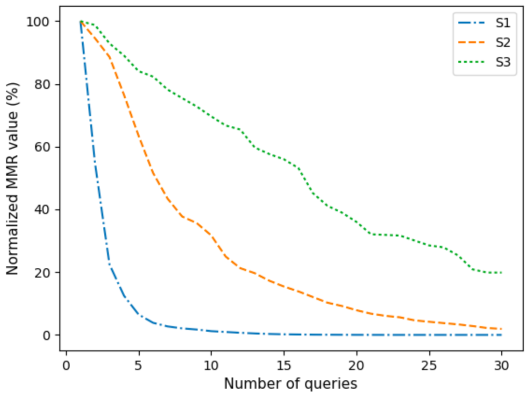

Figure 3: Elicitation with S1, S2, S3 (10 agents, 50 items)

precisely vector α. Very often, a necessary winner can be identified while some components α_i remain largely imprecise. This is the key point explaining practical efficiency. Unfortunately, this approach does not provide any control on the reduction of uncertainty and prevents to derive non-trivial formal bounds on the number of queries. Algorithm 1 implemented with S2 generates queries designed to systematically reduce the uncertainty attached to weights (the range of possible value is divided by two at every step), and the optimal solution is obtained as a byproduct. It allows to bound the number of queries from above but the drawback is that the generated queries may not be very efficient to determine the optimal choice because queries are not taylored specifically to discriminate between the solutions appearing in the particular instance to be solved. This explains why S1 empirically performs better than S2. Nevertheless, strategy S2 is also interesting because we have obtained a theoretical bound on the maximum number of queries (see Proposition 4) while keeping good empirical performances.

8 Conclusion

We proposed two active learning algorithms for the elicitation of weights in GSFs and the determination of f_α-optimal knapsacks. Algorithm 1 used with strategy S1 or S2 appears to be quite efficient, both in terms of preference queries and computation times. One research direction to further improve the computation of regrets concerns the control of the number of extreme points $|A^{\text{ext}}|$ in the uncertainty set. Although the exploration of A^{ext} was easily tractable in the experiments we made, an interesting challenge would be to design new questionnaires enabling a closer control of $|A^{\text{ext}}|$ while keeping the informative value of our preference queries. Besides this perspective, the techniques presented here could be adapted to other fair multiagent combinatorial optimization problems (e.g., fair assignment problems, fair multiagent scheduling problems) and more generally to elicit other decision models (linear in their parameter).

9 Acknowledgments

This work is supported by the ANR project CoCoRICo-CoDec ANR-14-CE24-0007-01.

References

Aziz, H.; Gaspers, S.; Mackenzie, S.; and Walsh, T. 2014. Fair assignment of indivisible objects under ordinal preferences. In *Proceedings of AAMAS-14*, 1305–1312.

Benabbou, N., and Perny, P. 2015a. Combining preference elicitation and search in multiobjective state-space graphs. In *Proceedings of IJCAI-15*, 297–303.

Benabbou, N., and Perny, P. 2015b. Incremental weight elicitation for multiobjective state space search. In *Proceedings of AAAI-15*, 1093–1099.

Benabbou, N., and Perny, P. 2016. Solving multi-agent knapsack problems using incremental approval voting. In *Proceedings of ECAI-16*, 1318–1326.

Benabbou, N.; Perny, P.; and Viappiani, P. 2017. Incremental elicitation of choquet capacities for multicriteria choice, ranking and sorting problems. *Artificial Intelligence* 246:152–180.

Benade, G.; Nath, S.; Procaccia, A. D.; and Shah, N. 2017. Preference elicitation for participatory budgeting. In *Proceedings of AAAI*, 376–382.

Blackorby, C., and Donalson, D. 1978. Measures of relative equality and their meaning in terms of social welfare. *Journal of Economic Theory* 2:59–80.

Bourdache, N., and Perny, P. 2017. Anytime algorithms for adaptive robust optimization with OWA and WOWA. In *Proceedings of ADT-17*, 93–107.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* 170(8-9):686–713.

Bouveret, S., and Lang, J. 2008. Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *Journal of Artificial Intelligence Research* 32:525–564.

Bouveret, S.; Cechlrov, K.; Elkind, E.; Igarashi, A.; and Peters, D. 2017. Fair division of a graph. In *Proc. of IJCAI-17*, 135–141.

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

Brero, G.; Lubin, B.; and Seuken, S. 2018. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of IJCAI'18*, 128–136.

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

Chevaleyre, Y.; Endriss, U.; and Maudet, N. 2017. Distributed fair allocation of indivisible goods. *Artificial Intelligence* 242:1–22.

Elkind, E., and Ismaili, A. 2015. Owa-based extensions of the chamberlin-courant rule. In *Proc. of ADT-15*, 486–502.

Fluschnik, T.; Skowron, P.; Triphaus, M.; and Wilker, K. 2017. Fair knapsack. *arXiv:1711.04520*.

Freeman, R.; Zahedi, S. M.; and Conitzer, V. 2017. Fair and efficient social choice in dynamic settings. In *Proceedings of IJCAI-17*, 4580–4587.

Fukuda, K., and Prodon, A. 1996. Double description method revisited. In *Comb. and Computer Science*. Springer. 91–111.

Gal, Y. K.; Mash, M.; Procaccia, A. D.; and Zick, Y. 2017. Which is the fairest (rent division) of them all? *J. ACM* 64(6):39:1–39:22.

Galand, L., and Spanjaard, O. 2007. Owa-based search in state space graphs with multiple cost functions. In *Proceedings of FLAIRS-07*, 86–91.

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

Golden, B., and Perny, P. 2010. Infinite order Lorenz dominance for fair multiagent optimization. In *Proceedings of AAMAS-10*, 383–390.

Goldsmith, J.; Lang, J.; Mattei, N.; and Perny, P. 2014. Voting with rank dependent scoring rules. In *Proceedings of AAAI-14*, 698–704.

Ha, V., and Haddawy, P. 1997. Problem-focused incremental elicitation of multi-attribute tility models. In *Proceedings of UAI-97*, 215–222. Morgan Kaufmann Publishers Inc.

Heinen, T.; Nguyen, N. T.; and Rothe, J. 2015. Fairness and rank-weighted utilitarianism in resource allocation. In *Proc. of ADT-15*, 521–536.

Lesca, J., and Perny, P. 2010. LP solvable models for multiagent fair allocation problems. In *Proc. of ECAI-10*, 393–398.

Ogryczak, W., and Śliwiński, T. 2003. On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research* 148(1):80–91.

Ogryczak, W. 2009. Inequality measures and equitable locations. *Annals of Operations Research* 167(1):61–86.

Skowron, P.; Faliszewski, P.; and Lang, J. 2016. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence* 241:191–216.

Wang, T., and Boutilier, C. 2003. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of IJCAI-03*, 309–318.

Weng, P., and Zanuttini, B. 2013. Interactive value iteration for markov decision processes with unknown rewards. In *Proceedings of IJCAI-13*, 2415–2421.

Weymark, J. 1981. Generalized gini inequality indices. *Math. Social Sciences* 1(4):409–430.

White, C. C.; Sage, A. P.; and Dozono, S. 1984. A model of multiattribute decision making and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics* 14(2):223–229.

Yager, R. 1998. On ordered weighted averaging aggregation operators in multicriteria decision making. In *IEEE Trans. Systems, Man and Cybern.*, volume 18, 183–190.