Combining Preference Elicitation with Local Search and Greedy Search for Matroid Optimization

Nawal Benabbou, Cassandre Leroy, Thibaut Lust, Patrice Perny

Combining Preference Elicitation with Local Search and Greedy Search for Matroid Optimization

Nawal Benabbou, Cassandre Leroy, Thibaut Lust, Patrice Perny
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6, LIP6
F-75005 Paris, France

Abstract

In this paper we propose two incremental preference elicitation methods for interactive preference-based optimization on weighted matroid structures. More precisely, for linear objective (utility) functions, we propose an interactive greedy algorithm interleaving preference queries with the incremental construction of an independent set to obtain an optimal or near-optimal base of a matroid. We also propose an interactive local search algorithm based on sequences of possibly improving exchanges for the same problem. For both algorithms, we provide performance guarantees on the quality of the returned solutions and the number of queries. Our algorithms are tested on the uniform, graphical and scheduling matroids to solve three different problems (committee election, spanning tree, and scheduling problems) and evaluated in terms of computation times, number of queries, and empirical error.

Introduction

Local search is a very general heuristic approach standardly used in AI to determine good solutions to hard combinatorial optimization problems (Russell and Norvig 2002; E.H. Aarts 2003). Local search algorithms start from a candidate solution and then iteratively move to a neighbor solution with the aim of finding the optimal one. A typical instance of local search algorithm is *hill climbing*, which consists in choosing the neighbor solution among the ones that locally optimize the objective function in the neighborhood.

In many cases, local search delivers sub-optimal solutions to the global problem and the search method needs to be modified to continue the search beyond local optimality. However, there are some well known examples of problems for which local search algorithms deliver optimal solutions. For example, in continuous optimization, the simplex algorithm used to solve linear programs is essentially a local search algorithm passing from vertices of a polytope to neighbor vertices until reaching a local optimum, which is known to be a global optimum. In discrete optimization, the minimum spanning tree problem can also be solved by local search from any arbitrary initial spanning tree using neighborhoods and improving sequences based on edge swaps. This result is strongly related to the correctness of the greedy algorithm in the spanning tree problem (Kruskal algorithm (Kruskal 1956)) and more generally to the weighted matroid structure underlying the problem.

Matroids are indeed of special interest both for greedy search and local search. First, they are the only non-empty hereditary structures for which the greedy algorithm provides optimal solutions (Oxley 2006). But local search also benefits from matroid structures. In a combinatorial optimization problem where the admissible solutions are the bases of a weighted matroid, a local search algorithm can indeed be used from any base to progressively improve the current base using swaps of elements until reaching the optimal one, as suggested in (Lee 2004) with a simple exchange algorithm. To perform useful swaps in a local search or optimal greedy choices in the construction of a solution, we need to know the relative values of the elements composing the feasible solutions. When this preference information is not available at the beginning of the search, it can be acquired during the search via preference elicitation procedures to reduce the elicitation burden. The goal of this paper is to make some propositions in this direction for matroid optimization.

The standard assumption in preference-based optimization is that preferences have been elicited beforehand and are known when the optimization procedure is launched. Nevertheless, some recent works in AI and algorithmic decision theory have relaxed this assumption to extend search procedures to the case of imprecisely known objective functions. The idea is to combine preference elicitation methods aiming to progressively specify the objective with the exploration of the set of solutions in order to focus the elicitation task on acquiring the part of preference information that is necessary to solve the instance under consideration. Thus, the optimal solution can be found after a reduced number of preference queries. This approach which is both incremental and adaptive to answers provided by the Decision Maker was first implemented in non-combinatorial problems, in various contexts such as multi-attribute utility theory (White III, Sage, and Dozono 1984; Braziunas and Boutilier 2007), decision making under risk (Ha and Haddawy 1997; Chajewska, Koller, and Parr 2000; Wang and Boutilier 2003), collective decision making (Lu and Boutilier 2011) and multicriteria decision making (Benabbou, Perny, and Viappiani 2017). Elicitation for decision making on combinatorial domains is more challenging and has motivated several contributions in various contexts, e.g.,

constraint satisfaction problems (Gelain et al. 2010), sequential decision making under risk (Regan and Boutilier 2009; Weng and Zanuttini 2013), stable matching (Drummond and Boutilier 2014), and multiobjective state space search (Benabbou and Perny 2015).

Then, some interactive versions of dynamic programming, branch and bound, greedy algorithms have been proposed for specific problems, combining preference elicitation and optimization on combinatorial domains (Benabbou and Perny 2018). Recently, some metaheuristics have also been proposed to solve harder optimization problems (see e.g., (Benabbou, Leroy, and Lust 2020) for the traveling salesman problem). Our aim here is to take advantage of matroid structures to propose general interactive local search and greedy algorithms yielding *exact* solutions or approximate solutions *with performance guarantees*.

The paper is organized as follows: First, we recall some basic notions on weighted matroids and on regret-based preference elicitation. Then, we present an interactive greedy algorithm and interactive local search method for linear optimization on matroids. We establish a performance guarantee in terms of worst-case regrets for these algorithms. Finally, we present three applications of the proposed algorithms and the results of numerical tests.

Weighted Matroids and Preference Elicitation

Matroids appear in various combinatorial optimization problems such as subset selection, task ordering, minimum spanning trees (Lee and Ryan 1992; Lee 2004) with various possible applications in AI, e.g., in the context of social choice and fair division (Gourvès, Monnot, and Tlilane 2013; Gourvès, Monnot, and Tlilane 2014). We recall here some basic definitions related to matroids, as well as some well-known examples that will be used for application and numerical tests later in the paper.

Matroids. A matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ is a finite ground set $\mathcal{S}$ of size n, together with a collection of sets $\mathcal{I} \subseteq 2^{\mathcal{S}}$ known as the *independent sets*, satisfying the following axioms:

- (A_1) if $Y \in \mathcal{I}$ and $X \subseteq Y$ then $X \in \mathcal{I}$,
- (A_2) if $X \in \mathcal{I}$, $Y \in \mathcal{I}$ and $|Y| > |X|$ then there exists $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{I}$.

Axiom A_2 implies that all maximal independent sets (w.r.t set inclusion) have the same cardinality. A maximal independent set is called a *base* of the matroid; the set of all bases will be denoted by $\mathcal{B}$ hereafter. The cardinality of a base is called the *rank* of the matroid and it will be denoted by $r(\mathcal{M})$ in the sequel. Note that the notion of matroid clearly generalizes the notion of linear independence over vectors.

In this work, we consider the problem of finding a maximum weight independent set in a matroid. More precisely, given a matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$, we want to compute $\max_{X \in \mathcal{I}} w(X)$ where w is a set function defined on $2^{\mathcal{S}}$ measuring the weight (or utility) of any subset of $\mathcal{S}$. Here we consider the case of an additive set function, characterized by the fact that $w(X) = \sum_{e \in X} w(e)$ for all $X \subseteq \mathcal{S}$. Function w is therefore completely characterized by the weights $w(e)$, $e \in \mathcal{S}$. Without loss of generality, we assume that $w(e) > 0$ for all $e \in S$ (the elements with negative weights can be omitted because they will not appear in the optimal subset). Under this assumption, w is monotonic with respect to set inclusion, and therefore any optimal independent subset is necessarily a base of the matroid.

Finding the optimal base in a weighted matroid is a very general problem that appears in various practical contexts. Let us give some examples of matroid optimization problems in the context of combinatorial optimization and algorithmic decision theory:

Uniform matroid. It is defined by $\mathcal{I} = \{X \subseteq \mathcal{S} : |X| \leq k\}$ for a given positive integer $k \leq n$. A base is any set of cardinality k. This structure appears in social choice when the problem is to determine the best committee of size k. In this context $\mathcal{S}$ is the set of candidates and w measures the social utility of any candidate and any group of candidates.

Graphic matroid. Given a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges, the independent sets are the subsets of edges that do not contain any cycle (i.e. the forests). If the graph G is connected, any base will correspond to a spanning tree T of the graph. Let $c : E \to \mathbb{R}_+$ be a function defining the cost of every edge $e \in E$ and let c^* be any constant strictly greater than $\max_{e \in E} c(e)$. Now if we define $w(e) = c^* - c(e)$ then finding the w-optimal base amounts to solving the minimum spanning tree problem.

Scheduling matroid. We are given n jobs that each takes one unit of processing time. All jobs are available at time 0, and job j has a profit w_j and a deadline d_j. The profit for job j will only be earned if the job completes by time d_j. The problem is to find an ordering of the jobs that maximizes the total profit. If $\mathcal{S} = \{1, \ldots, n\}$ and $\mathcal{I} = \{X \subseteq \mathcal{S} : X \text{ can be completed on time}\}$ then $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ is a matroid. If we set $w(X) = \sum_{j \in X} w_j$ for all $X \in \mathcal{I}$ then any optimal base gives a set of jobs maximizing the total profit.

Here we consider a matroid optimization problem where w is a set function representing the subjective preferences of a Decision Maker (DM). Moreover, we assume that w is initially not known which precludes the use of standard solving methods. Instead, we are given a (possibly empty) set $\mathcal{P}$ of pairs $(X, Y) \in \mathcal{I} \times \mathcal{I}$ such that X is known to be preferred to Y by the DM; such preference data can be obtained by asking comparison queries to the DM. Let W be the *uncertainty set* defined as the set of all functions w that are compatible with $\mathcal{P}$, i.e. such that $w(X) \geq w(Y)$ for all $(X, Y) \in \mathcal{P}$. In this context, we consider the minimax regret decision criterion which is commonly used within the AI community to make robust recommendations under preference imprecision. The minimax regret (MMR) can be defined using pairwise max regrets (PMR) and max regrets (MR) as follows:

Definition 1. *For any two sets $B, B' \in \mathcal{B}$:*
$PMR(B, B', W) = \max_{w \in W} \{w(B') - w(B)\}$
$MR(B, \mathcal{B}, W) = \max_{B' \in \mathcal{B}} PMR(B, B', W)$
$MMR(\mathcal{B}, W) = \min_{B \in \mathcal{B}} MR(B, \mathcal{B}, W)$

Applying this criterion in a matroid context, an optimal base is a base that achieves the minimax regret, i.e., a base

$B \in \mathcal{B}$ such that $MR(B, \mathcal{B}, W) = MMR(\mathcal{B}, W)$. By definition, recommending any of these solutions enables to minimize the worst-case loss (regret). Note that these solutions are necessarily optimal when $MMR(\mathcal{B}, W) = 0$ holds.

Since the MMR value can only decrease when adding new preference statements in $\mathcal{P}$, it has been proposed to use the following incremental elicitation approach: progressively ask preference queries to the DM until the MMR value drops below a given threshold $\delta \geq 0$ which represents the maximum allowable gap to optimality (Boutilier et al. 2006). Note that if we set $\delta = 0$, then we obtain an optimal base at the end of the execution. This approach, sometimes referred to as *regret-based incremental elicitation*, was efficiently used in various decision contexts, such as multicriteria decision making and voting problems (e.g., (Lu and Boutilier 2011; Benabbou, Perny, and Viappiani 2017)).

To implement this approach, one needs to compute the value $MMR(\mathcal{B}, W)$ at every iteration step of the procedure. For matroid optimization, this may induce prohibitive computation times since it may require to compute the pairwise max regrets for all pairs of distinct bases in $\mathcal{B}$. Therefore, we propose instead to combine search and regret-based incremental elicitation to reduce both computation times and number of queries. More precisely, we generate queries during the search so as to progressively reduce the set W until being able to determine a (near-)optimal base.

An Interactive Greedy Algorithm

When function w is known, the so-called Rado-Edmons theorem states that the problem of finding a maximum weight independent set in the matroid can be solved using a greedy algorithm (Edmonds 1971). Starting from the empty set, the idea is to iteratively select an element of maximal weight among those that could be inserted to the current independent subset without loosing the independence property. In practice, one first sorts the elements of the ground set by decreasing order and try to insert each of them successively without loosing independence. When no element can be added anymore to the current subset, the algorithm stops and the elements selected so far form the optimal base.

When w is not known or imperfectly known, we can ask preference queries to the DM in order to determine his/her preference order over the elements of the ground set, or to approximate it. For the sake of efficiency, the preference elicitation can be performed during the greedy search in order to focus the elicitation task on preference information that is directly necessary to implement the algorithm. To this end we propose an interactive version of the greedy algorithm for matroid optimization, described in Algorithm 1.

Our algorithm uses the rank $r(\mathcal{M})$ in lines 2-3 to reduce the number of preference queries in practice. The rank can be easily obtained for various matroids (e.g., uniform or graphic). For more complex matroids, we can use any upper bound (by default n) or precalculate $r(\mathcal{M})$ by running the standard greedy algorithm for an arbitrary function w.

Note that the inner loop (lines 3-6) stops after a finite number of steps since $MMR(E, W)$ equals 0 when all the elements of E have been compared by the DM. Note also

Algorithm 1: The Interactive Greedy Algorithm

```
1  X ← ∅; E ← S;
2  while |X| < r(M) do
3  |  while MMR(E, W) > δ/r(M) do
4  |  |  Ask the DM to compare two elements of E;
5  |  |  Update W according to the DM's answer;
6  |  end
7  |  Select e_i ∈ E such that MR({e_i}, E, W) ≤ δ/r(M);
8  |  if X ∪ {e_i} ∈ I then X ← X ∪ {e_i} end;
9  |  E ← E \ {e_i}
10 end
11 return X;
```

that Algorithm 1 with $\delta = 0$ selects an element that is necessarily optimal at every step (i.e., the selected element is optimal for all admissible set functions $w \in W$). Therefore, our algorithm with $\delta = 0$ is nothing else but the standard greedy algorithm (Edmonds 1971), and it returns an optimal base. However, asking queries until minimax regrets decrease to zero may entail an important number of queries in practice (as will be observed in the numerical section). For a practical implementation of this procedure, Algorithm 1 can be used with any positive δ to construct a base with bounded regrets. This adaptation is not straightforward, as shown below:

Proposition 1. *Algorithm 1 returns a base $B \in \mathcal{B}$ such that $MR(B, \mathcal{B}, W) \leq \delta$ holds at the end of the execution.*

Proof. We want to prove that $MR(B, \mathcal{B}, W) \leq \delta$ holds at the end of the execution. By definition of max regrets, it is sufficient to prove that $w(B^*) - w(B) \leq \delta$ is true for any base $B^* \in \mathcal{B}$ and for any function $w \in W$ that is still admissible at the end of the execution. To do so, let us prove by induction that the following statement, denoted by $P(i)$, holds at the end of iteration step $i \in \{1, \ldots, m\}$ of the outer loop (lines 2-10):

$$\exists B_i \in \mathcal{B} \text{ s. t. } \begin{cases} X_i \subseteq B_i & \text{(1a)} \\ B_i \backslash X_i \subseteq E_i & \text{(1b)} \\ w(B^*) - w(B_i) \leq |X_i| \times \delta / r(\mathcal{M}) & \text{(1c)} \end{cases}$$

where m is the number of iteration steps and X_i (resp. E_i) denotes the set X (resp. E) at the end of step i. In other words, we want to prove that X_i can be extended into a base B_i with a regret bounded above by $|X_i| \times \delta / r(\mathcal{M})$. For step $i = 0$ (before entering the first loop), we have $X_i = \emptyset$ and $E_i = \mathcal{S}$ (see line 1). Consequently if we set $B_i = B^*$ then Equations (1a-1c) are obviously satisfied. Hence $P(0)$ holds.

Now we assume that $P(i-1)$ holds for some step $i \in \{1, \ldots, m\}$ and we want to prove that $P(i)$ is true. In other words, assuming that Equations (1a-1c) hold for some base B_{i-1}, we need to identify some base B_i such that Equations (1a-1c) are satisfied by B_i. According to line 8, two cases may occur: either $X_{i-1} \cup \{e_i\} \notin \mathcal{I}$ or $X_{i-1} \cup \{e_i\} \in \mathcal{I}$.

Case $X_{i-1} \cup \{e_i\} \notin \mathcal{I}$: In that case, we have $X_i = X_{i-1}$ (see lines 8-9). Let us prove that we can simply set $B_i = B_{i-1}$ to establish the result. First, note that the induction hypothesis (IH) directly implies that Equations (1a) and (1c) hold for B_i since $X_i = X_{i-1}$ and $B_i = B_{i-1}$.

Then, for Equation (1b), we need to prove that we have $B_i \backslash X_i \subseteq E_i$, i.e. $B_{i-1} \backslash X_{i-1} \subseteq E_i$. By the IH, we have $B_{i-1} \backslash X_{i-1} \subseteq E_{i-1}$, i.e. $B_{i-1} \backslash X_{i-1} \subseteq E_i \cup \{e_i\}$ (see line 9). Moreover, we can derive $e_i \notin B_{i-1}$ from axiom A_1 since $X_{i-1} \cup \{e_i\} \notin \mathcal{I}$ and $X_{i-1} \subseteq B_{i-1}$ (by the IH). Hence $B_{i-1} \backslash X_{i-1} \subseteq E_i$ holds and therefore $P(i)$ is true.

Case $X_{i-1} \cup \{e_i\} \in \mathcal{I}$: In that case, we have $X_i = X_{i-1} \cup \{e_i\}$ (see lines 8-9). Two cases can be distinguished:

- Case $e_i \in B_{i-1}$: let us show that we can simply set $B_i = B_{i-1}$. More precisely, for Equation (1a), we have:

$$\begin{aligned} X_i &= X_{i-1} \cup \{e_i\} \text{ (by hypothesis)} \\ &\subseteq B_{i-1} \cup \{e_i\} \text{ (by the IH)} \\ &= B_{i-1} \text{ (since } e_i \in B_{i-1}) \\ &= B_i \text{ (by definition)} \end{aligned}$$

  For Equation (1b), we have:

$$\begin{aligned} B_i \backslash X_i &= B_{i-1} \backslash (X_{i-1} \cup \{e_i\}) \\ &\subseteq E_{i-1} \backslash \{e_i\} \text{ (by the IH)} \\ &= E_i \text{ (see line 9)} \end{aligned}$$

  For Equation (1c), we have:

$$\begin{aligned} w(B^*) - w(B_i) &= w(B^*) - w(B_{i-1}) \text{ (by definition)} \\ &\leq |X_{i-1}| \times \delta / r(\mathcal{M}) \text{ (by the IH)} \\ &\leq |X_i| \times \delta / r(\mathcal{M}) \text{ (since } |X_i| = |X_{i-1}| + 1) \end{aligned}$$

  Hence $P(i)$ holds.
- Case $e_i \notin B_{i-1}$: note that we have $|X_i| \leq |B_{i-1}|$ since B_{i-1} is a base. Hence by iteratively applying axiom A_2, we can conclude that there exists $Y \subseteq B_{i-1} \backslash X_i$ such that $|X_i \cup Y| = |B_{i-1}|$ and $X_i \cup Y \in \mathcal{I}$. Now we set $B_i = X_i \cup Y$ and we want to prove that Equations (1a-1c) are satisfied by B_i. Note that Equation (1a) is obviously true since $B_i = X_i \cup Y$ (by definition). For Equation (1b), since $B_i \backslash X_i = Y$, we only need to prove that $Y \subseteq E_i$. We have:

$$\begin{aligned} Y &\subseteq B_{i-1} \backslash X_i \text{ (by definition)} \\ &= B_{i-1} \backslash (X_{i-1} \cup \{e_i\}) \text{ (since } X_i = X_{i-1} \cup \{e_i\}) \\ &\subseteq E_{i-1} \backslash \{e_i\} \text{ (by the IH)} \\ &= E_i \text{ (see line 9)} \end{aligned}$$

  For Equation (1c), we need to prove that inequality $w(B^*) - w(B_i) \leq |X_i| \times \delta / r(\mathcal{M})$ holds. We have:

$$\begin{aligned} B_i \backslash B_{i-1} &= (X_i \cup Y) \backslash B_{i-1} \text{ (by definition)} \\ &= (X_{i-1} \cup \{e_i\} \cup Y) \backslash B_{i-1} \text{ (by hypothesis)} \\ &= (\{e_i\} \cup Y) \backslash B_{i-1} \text{ (since } X_{i-1} \subseteq B_{i-1} \text{ by the IH)} \\ &= \{e_i\} \text{ (since } Y \subseteq B_{i-1} \text{ and } e_i \notin B_{i-1}) \end{aligned}$$

  As a consequence, since we also have $|B_i| = |B_{i-1}|$, we know that there exists $e_j \in \mathcal{S} \backslash \{e_i\}$ such that $B_i = (B_{i-1} \cup \{e_i\}) \backslash \{e_j\}$. Now let us prove that $e_j \in E_{i-1}$:

$$\begin{aligned} \{e_j\} &= B_{i-1} \backslash B_i = B_{i-1} \backslash (X_i \cup Y) \\ &= B_{i-1} \backslash (X_{i-1} \cup \{e_i\} \cup Y) \subseteq B_{i-1} \backslash X_{i-1} \subseteq E_{i-1} \end{aligned}$$

  where the last inclusion is obtained by the IH. Consequently we can derive $w(e_i) - w(e_j) \leq \delta / r(\mathcal{M})$ from the definition of max regrets (see line 7). Hence we have:

$$\begin{aligned} & w(B^*) - w(B_i) \\ &= w(B^*) - w((B_{i-1} \cup \{e_i\}) \backslash \{e_j\}) \text{ (by definition)} \\ &= w(B^*) - w(B_{i-1}) + (w(\{e_i\}) - w(\{e_j\})) \text{ (by additivity)} \\ &\leq |X_{i-1}| \times \frac{\delta}{r(\mathcal{M})} + (w(\{e_i\}) - w(\{e_j\})) \text{ (by the IH)} \\ &\leq (|X_{i-1}| + 1) \times \delta / r(\mathcal{M}) \text{ (since } w(e_i) - w(e_j) \leq \delta / r(\mathcal{M})) \\ &= |X_i| \times \delta / r(\mathcal{M}) \text{ (since } X_i = X_{i-1} \cup \{e_i\}) \end{aligned}$$

  Hence Equation (1c) holds and therefore $P(i)$ is true.

Thus, statement $P(i)$ holds for every step $i \in \{1, \ldots, m\}$, and in particular $P(i)$ is true for $i = m$. Since X_m is a base, then we necessarily have $X_m = B_m$ due to Equation (1a). Then, using Equation (1c), we obtain $w(B^*) - w(X_m) \leq |X_m| \times \delta / r(\mathcal{M}) = r(\mathcal{M}) \times \delta / r(\mathcal{M}) = \delta$. Finally, since Algorithm 1 returns $B = X_m$, the result is established. □

Note that Algorithm 1 generates no more than a polynomial number of queries since in the worst-case the DM is asked to compare all the elements of $\mathcal{S}$ (see line 4). Consequently the total number of steps of the inner loop (lines 3-6) is also polynomial. However, despite the existence of a general formalism, the implementation of Algorithm 1 may differ significantly from one application context to another.

More precisely, checking whether $X \cup \{e_i\} \in \mathcal{I}$ can be more or less complex depending on the matroid under consideration; for the three examples used in the paper (the uniform, graphic and scheduling matroids), the independence test can be performed in polynomial time.

Moreover, computing regrets can be more or less complex depending on the assumptions made on w. In this respect, an interesting option is to define w by a parametric function which is linear in its parameters (e.g., a linear combination of spline functions, or a linear multiattribute utility, or an ordered weighted average of criterion values). In that case, regret optimization can be performed in polynomial time using linear programming. Moreover, using a parametric definition of w enables to save preferences queries because any preference statement of type $w(e) > w(e')$ translates into a constraint on the parameter space that will reduce possible preferences over other elements.

We now present an execution of Algorithm 1 on a small instance of the scheduling matroid:

Example 1. Consider an instance of the scheduling matroid with $\mathcal{S} = \{1, \ldots, 8\}$ (a set of 8 jobs of unitary length). The utility $w(j)$ of completing job j on time is defined from an attribute vector $y(j) = (y_1(j), y_2(j), y_3(j))$ by

$$w(j) = \lambda_1 y_1(j) + \lambda_2 y_2(j) + \lambda_3 y_3(j) \qquad (2)$$

for some unknown $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ representing the value system of the DM. The attribute vectors attached to jobs are given below, together with their deadlines $d_j, j \in \mathcal{S}$:

j	1	2	3	4	5	6	7	8
$y_1(j)$	6	2	5	8	1	6	3	2
$y_2(j)$	8	4	2	7	2	3	4	3
$y_3(j)$	8	7	5	1	8	3	6	1
d_j	4	1	2	4	1	3	4	1

Initially, the set Λ of admissible weighting vectors λ is defined by $\Lambda = \{\lambda \in [0,1]^3 : \sum_{i=1}^{3} \lambda_i = 1\}$ and induces a set W of admissible set functions w by Equation (2). To simulate the DM's answers during the execution of Algorithm 1, we assume that the actual set function w of the DM is defined by $\lambda^* = (\frac{6}{9}, \frac{2}{9}, \frac{1}{9})$. In Figure 1, Λ is represented by triangle ABC in the space (λ_1, λ_2), λ_3 being implicitly defined by $\lambda_3 = 1 - \lambda_1 - \lambda_2$, and λ^* is represented by a star inside the triangle. For this instance, we applied the standard greedy algorithm with an arbitrary w and obtained $r(\mathcal{M}) = 4$. Now, let us execute Algorithm 1 with $\delta = 0$.

Step 1: Since $E = \mathcal{S}$ and $MMR(E, W) = 2 > 0$, the DM is asked to compare two jobs, say 1 and 4. We have $w(4) = 7 \geq w(1) = 6.7$ (by Equation (2) with $\lambda = \lambda^*$). Therefore the DM's answers: "job 4 is more important than job 1". Then W is updated by imposing the constraint $w(4) \geq w(1)$ which amounts to restricting Λ by inserting $\lambda_2 \geq \frac{7}{6} - \frac{3}{2}\lambda_1$. Now Λ is represented by the triangle CDE in Figure 1. Moreover, we have $MMR(E, W) = MR(\{4\}, E, W) = 0$ and therefore job 4 is added to the current independent set X.

Step 2: We have $E = \mathcal{S}\backslash\{4\}$ and $MMR(E, W) = MR(\{1\}, E, W) = 0$. Hence we do not need to solicit the DM at this. Then job 1 is added to X ($X = \{4, 1\}$) since it can be completed on time.

Step 3: We have $E = \mathcal{S}\backslash\{4, 1\}$. Again no query is needed since $MMR(E, W) = MR(\{6\}, E, W) = 0$, and job 6 is added to X ($X = \{4, 1, 6\}$) as schedule $(6, 1, 4)$ is feasible.

Step 4: We have $E = \mathcal{S}\backslash\{4, 1, 6\}$ and $MMR(E, W) = 0.67 > 0$. Therefore the DM is asked to compare two jobs, say 3 and 7. Since we have $w(3) = 4.3 \geq w(7) = 3.6$, the DM answers: "job 3 is more important than job 7". Then, W is updated by imposing the constraint $w(3) \geq w(7)$, which amounts to further restricting Λ by inserting $\lambda_2 \leq 3\lambda_1 - 1$. Now Λ is represented by the polyhedron CFGE in Figure 2. Moreover, since $MMR(E, W) = MR(\{3\}, E, W) = 0$, job 3 is added to X ($X = \{4, 1, 6, 3\}$) as schedule $(3, 6, 1, 4)$ is feasible. Finally the algorithm stops since $|X| = r(\mathcal{M}) = 4$ and $X = \{4, 1, 6, 3\}$ is optimal. After only two queries, we know that the actual weights of the DM is within the area CFGE in Figure 2 and this is sufficient to determine an optimal solution without any ambiguity.

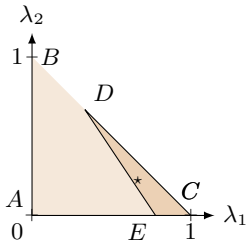

Figure 1: Λ after 1 query.

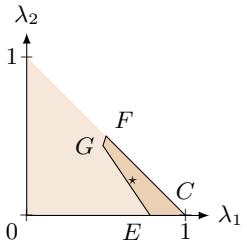

Figure 2: Λ after 2 queries.

An Interactive Local Search Algorithm

Local Search is another efficent way of constructing optimal solutions to matroid optimization problems (Lee 2004). More precisely, starting from an arbitrary base, one can look for an element that can be profitably replaced by an element out of the base while preserving independence. This simple exchange principle can be iterated to progressively improve the current solution until reaching a local optimum. When w is not known, this principle can be combined with a preference elicitation algorithm to collect the information that is necessary to identify improving swaps. To implement this idea, we propose Algorithm 2 where $X \Delta Y$ denotes the symmetric difference defined by $(X \setminus Y) \cup (Y \setminus X)$ and N_B denotes the neighborhood of base B, i.e., the set of adjacent bases that differ from B by exactly one element (thus the symmetric difference with B is of cardinality 2). The procedure `ComputeInitialBase` called in line 1 can be any heuristic providing a good starting solution. For instance, we can compute an optimal base for a sample of arbitrary set functions and asks the DM to compare them.

Algorithm 2: The Interactive Local Search Algorithm

```
1  B ← ComputeInitialBase(M);
2  improve ← true;
3  while improve do
4      N_B ← {B' ∈ B : |B Δ B'| = 2};
5      while MMR(N_B ∪ {B}, W) > δ/r(M) do
6          Ask one preference query to the DM;
7          Update W according to the DM's answer;
8      end
9      if MR(B, N_B, W) ≤ δ/r(M) then improve ← false ;
10     else
           B ← Select(arg min_{B' ∈ N_B} MR(B', N_B ∪ {B}, W)) ;
11 end
12 return B;
```

The quality of the base returned by Algorithm 2 is guaranteed by the following:

Proposition 2. *Algorithm 2 returns a base $B \in \mathcal{B}$ such that $MR(B, \mathcal{B}, W) \leq \delta$ holds at the end of the execution.*

Proof. We want to prove that $MR(B, \mathcal{B}, W) \leq \delta$ holds when the algorithm stops. By definition of max regrets, we only need to prove that $w(B^*) - w(B) \leq \delta$ holds for any base $B^* \in \mathcal{B}$ and for any function $w \in W$ that is still admissible at the end of the execution. Due to a well-know multiple exchange theorem (Greene and Magnanti 1975), there exists a one-to-one correspondence $\sigma : B \to B^*$ such that $B_i = (B\backslash\{e_i\}) \cup \{\sigma(e_i)\}$ is a base of the matroid for every element $e_i \in B$. Note that $B_i \in N_B$ (the neighborhood of B) for all $e_i \in B$ since we have $|B \Delta B_i| = 2$ (see line 4). Note also that we necessarily have $MR(B, N_B, W) \leq \delta/r(\mathcal{M})$ at the end of the execution due to line 9. Therefore, $w(B_i) - w(B) \leq \delta/r(\mathcal{M})$ holds by definition of max regrets. Then, since w is additive and $B_i = (B\backslash\{e_i\}) \cup \{\sigma(e_i)\}$, we obtain $w(\sigma(e_i)) - w(e_i) \leq \delta/r(\mathcal{M})$. Hence we have:

$$w(B^*) - w(B) = w\Big(\bigcup_{i=1}^{r(\mathcal{M})} \{\sigma(e_i)\}\Big) - w(B) \text{ (by definition of } \sigma)$$

$$= \sum_{i=1}^{r(\mathcal{M})} \Big(w(\sigma(e_i)) - w(e_i)\Big) \text{ (by additivity of } w)$$

$$\leq \sum_{i=1}^{r(\mathcal{M})} \frac{\delta}{r(\mathcal{M})} = \delta. \qquad \square$$

Note that when $\delta = 0$, Proposition 2 ensures that Algorithm 2 yields an optimal base. Note also that Algorithm 2 generates no more than a polynomial number of queries when w is additive as computing the PMR values between two neighbors amounts to comparing two elements of the ground set. We now present an execution of Algorithm 2:

Example 2. Consider the same scheduling problem as in Example 1. Let us execute Algorithm 2 with $\delta = 0$. Assume that `ComputeInitialBase`($\mathcal{M}$) returns $B = \{1, 2, 4, 7\}$ corresponding to the schedule $(2, 1, 4, 7)$ whose attribute vector is $y(B) = (19, 23, 22)$.

Step 1: Base B only has 7 candidate neighbors and their attribute vectors are as follows:

	$B_{2,3}$	$B_{2,5}$	$B_{2,6}$	$B_{4,3}$	$B_{4,6}$	$B_{7,3}$	$B_{7,6}$
y_1	22	18	23	16	17	21	22
y_2	21	21	22	18	19	21	22
y_3	20	23	18	26	24	21	19

where $B_{e,e'}$ is the base obtained by replacing e by e' in B. Since $MMR(N_B, W) = 4 > 0$, the DM is asked to compare two bases, say B and $B_{2,6}$. We have $w(B_{2,6}) = 22.2 \geq w(B) = 20.2$ (by Equation (2)), and therefore the DM answers: "$B_{2,6}$ is better than B". Then the algorithm updates W by imposing the constraint $w(B_{2,6}) \geq w(B)$, which amounts to restricting Λ by inserting $\lambda_2 \geq \frac{4}{3} - \frac{8}{3}\lambda_1$ (see Figure 4 where Λ is represented by CDE). Now $MMR(N_B, W) = 0.5 > 0$ which requires to compare two other bases, say $B_{2,6}$ and $B_{2,3}$. Here we have $w(B_{2,6}) = 22.2 \geq w(B_{2,3}) = 21.5$. Therefore W is updated by imposing $w(B_{2,6}) \geq w(B_{2,3})$, i.e. Λ is restricted by adding $\lambda_2 \geq \frac{2}{3} - \lambda_1$ (see Figure 5 where Λ is represented by CFGD). Now the algorithm stops asking queries since $MMR(N_B, W) = MR(B_{2,6}, N_B, W) = 0$ and then we move from solution B to solution $B_{2,6}$ for the next step corresponding to schedule $(6, 1, 4, 7)$.

Step 2: Here N_B only includes 2 candidate bases, namely $B = B_{2,6}$ and $B' = \{1, 4, 6, 3\}$ corresponding to the schedule $(3, 6, 1, 4)$ whose vector is $(25, 20, 17)$. Since $MMR(N_B, \Lambda) = 1.2 \geq 0$, the DM is asked to compare B and B'. We have $w(B') = 23 \geq w(B) = 22.2$ and the DM answers: "B' is better than B". Then W is updated by imposing $w(B') \geq w(B)$ and Λ is restricted by $\lambda_2 \leq 3\lambda_1 - 1$ (see Figure 6 where Λ is represented by CFIH). Now $MMR(N_B, W) = MR(B', N_B, W) = 0$ and we set $B = B'$ for the next step.

Step 3: Here $MMR(N_B, W) = MR(B, N_B, W) = 0$ and therefore no question is needed. The algorithm ends by returning base B associated to $(3, 6, 1, 4)$ which is optimal.

Experimental Results

We have implemented our algorithms on three problems corresponding to three different matroids (uniform, graphic, and scheduling). Their performances are evaluated in terms of number of queries (for local search, it includes those generated by `ComputeInitialBase`), computation times (given in seconds), and empirical error measured as a percentage of the optimal value. Results are averaged over 30 runs. Standard deviations are given in the appendix. Before analyzing the results, let us present our experimental setting.

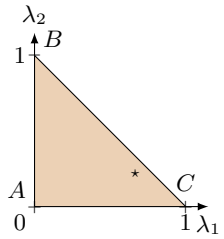

Figure 3: Initial set Λ.

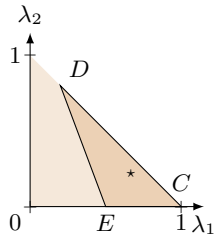

Figure 4: Λ after 1 query.

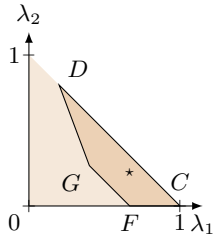

Figure 5: Λ after 2 queries.

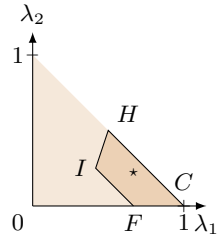

Figure 6: Λ after 3 queries.

We consider here multi-objective optimization problems in which every matroid's element $e \in \mathcal{S}$ is evaluated by a vector $x_e \in \mathbb{R}^p$ where p is the number of objectives to be maximized. The criterion values are randomly drawn between 1 and 1000. The DM's preferences over elements are defined using a scalarizing function $f_\lambda : \mathbb{R}^p \to \mathbb{R}$ (linear in its parameter λ) by $w(e) = f_\lambda(x_e)$. Hence, any preference information of type "e is as least as good as e'" translates into the constraint $f_\lambda(e) \geq f_\lambda(e')$ which is linear in λ. Examples of admissible functions for f_λ are weighted sums, ordered weighted average or Choquet integrals. Here we assume that f_λ is a weighted sum and λ is initially unknown.

Recall that, in our algorithms, we ask preference queries to the DM until MMR values drops below $\delta / r(\mathcal{M})$. In our tests, two values of δ have been used: $\delta = 0\%$ (which guarantees that the returned solution is optimal) and $\delta = 20\%$ of the initial regret (to reduce the number of preference queries). To generate informative queries, we use the Current Solution Strategy (CSS) to efficiently reduce the set of admissible weights (Boutilier et al. 2006): at each step, the DM is asked to compare a subset X achieving the minimax regret to one of its best challengers chosen in $\arg\max_Y PMR(X, Y, W)$. Answers to queries are simulated using a function w^* generated from a weighting vector λ^* randomly generated before running the algorithms.

We now provide the results obtained with our program written in C++ tested on an Intel Core i7-9700, 3.00 GHz with 15,5 GB of RAM. Pairwise max regret optimizations were performed by CPLEX (https://www.ibm.com/analytics/cplex-optimizer).

Subset Selection Problem (Uniform Matroid). We first consider the subset selection problem corresponding to the uniform matroid. Here the size of the ground set is $|\mathcal{S}| = 50$ and $k = 25$, i.e. $\mathcal{I} = \{X \subseteq \mathcal{S} : |X| \leq 25\}$. For this matroid,

		Local Search			Greedy		
δ	p	time	queries	error(%)	time	queries	error(%)
0%	4	3.99	18.5	0.00	5.15	43.9	0.00
	6	13.61	46.2	0.00	10.42	99.8	0.00
	8	45.26	81.0	0.00	13.16	124.4	0.00
20%	4	3.08	13.7	0.01	2.84	17.6	0.00
	6	7.60	28.9	0.01	4.52	37.4	0.00
	8	35.97	36.0	0.02	3.99	38.1	0.01

Table 1: Greedy vs local search for the subset selection problem.

		Local Search			Greedy		
δ	p	time	queries	error(%)	time	queries	error(%)
0%	4	17.79	41.7	0.00	19.51	64.0	0.00
	6	61.29	105.1	0.00	151.83	154.9	0.00
	8	256.07	195.0	0.00	583.93	262.27	0.00
20%	4	16.82	13.2	0.14	10.12	13.4	0.28
	6	33.01	24.7	0.17	40.92	26.2	0.24
	8	68.88	37.0	0.25	99.20	40.9	0.28

Table 2: Greedy vs local search for the spanning tree problem.

		Local Search			Greedy		
δ	p	time	queries	error(%)	time	queries	error(%)
0%	4	4.15	19.9	0.00	4.43	42.6	0.00
	6	8.21	42.7	0.00	8.33	79.9	0.00
	8	21.64	86.1	0.00	14.47	122.9	0.00
20%	4	4.12	12.5	0.02	2.20	14.2	0.00
	6	6.46	23.2	0.03	3.04	27.2	0.00
	8	11.97	38.9	0.01	4.52	41.6	0.00

Table 3: Greedy vs local search for the scheduling problem.

the independence test simply consists in checking the cardinality constraint. In Table 1, we observe that local search is more efficient than greedy in terms of number of queries. For example, the local search algorithm performs about two times better for $p = 6$ (46 queries against 99). However, greedy is faster than local search in terms of computation times. This is due to the fact that minimax regrets are computed on subsets of the ground set (of size bounded by 50) for greedy search, whereas it is computed on neighborhoods (of size bounded above by 25^2 since all elementary swaps are allowed) for local search. The gap becomes larger as the number of objectives increases due to the growing number of Pareto non-dominated neighbors. With δ = 20%, we notice a significant reduction of the number of queries for both algorithms while keeping good quality solutions (the error is at most equal to 0.02%).

Spanning Tree Problem (Graphic Matroid). Here the elements of the ground set are the edges of a connected graph. The independence test consists in checking that the selected edges do not form a cycle. We have generated graphs with 50 nodes and a density equal to 50% with the LEMON (https://lemon.cs.elte.hu/trac/lemon). In Table 2, we observe that local search is much faster and asks less queries than greedy with $\delta = 0$. This is mainly due to a larger size of the ground set (about 600 here against 50 for the previous problem) and to the fact that greedy constructs the optimal base from the empty set, whereas local search starts from a good complete solution (obtained with `ComputeInitialBase`). For both algorithms, we observe a significant improvement in terms of number of queries and times for $\delta = 20\%$, without causing a significant reduction in the quality of the returned base.

Scheduling Problem (Scheduling Matroid). We consider instances involving a ground set of 50 jobs, with deadlines randomly drawn between 1 and 25.The independence test consists in checking that there exists an order of the selected jobs compatible with their deadlines. As already observed, local search generates less queries than greedy, and the empirical error remains very small for both algorithms. Execution times are close for $\delta = 0$, but greedy is twice as fast as local search for $\delta = 20\%$. Note that this problem is very close to the subset selection problem but the independence test here further restricts the number of possible swaps in the local search procedure (which explains why local search is faster here).

Conclusion

We have introduced two interactive optimization methods combining preference elicitation with the exploration of independent sets in a matroid, one based on local search and the other on greedy search. The common principle used in both methods is to interleave preference queries with optimization steps to concentrate the elicitation effort on obtaining the preferential information needed to determine an optimal solution (or a near-optimal solution if we want to save some preference queries). The interest of our proposal is that it is quite general and can be implemented in various optimization problems involving a matroid structure. We have implemented these methods on three problems (subset selection, spanning tree and scheduling) and presented numerical tests showing the practical efficiency of our algorithms, in terms of computation times, number of preference queries and empirical error.

A natural continuation of this work is to extend our approach to non linear set functions. For example, submodular set functions (i.e., such that $w(X \cup Y) + w(X \cap Y) \leq w(X) + w(Y)$ for all subsets X, Y of the ground set $\mathcal{S}$) are of particular interest as they naturally appear in various contexts (e.g., coverage problems). Submodular utility functions are also used to guarantee a *principle of diminishing returns* stating that adding an element to a smaller set has more value than adding it to a larger set. Moreover, submodularity is of special interest in maximization problems over combinatorial domains because it plays a role that is analogous to concavity in continuous optimization.

The optimization of a submodular function is hard in general (it includes max-cut as special case) but approximate greedy and local search algorithms have been proposed for the optimization of submodular functions under a matroid constraint and some interesting worst case bounds on the quality of the approximations returned are known, see e.g., (Nemhauser, Wolsey, and Fisher 1978). The interactive approach proposed here, based on interleaving preference queries with the construction of the optimal solution, can also be implemented for submodular optimization. We did some preliminary tests which are very encouraging. The detailed analysis of such algorithms is left for further research.

References

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

[Benabbou and Perny 2018] Benabbou, N., and Perny, P. 2018. Interactive resolution of multiobjective combinatorial optimization problems by incremental elicitation of criteria weights. *EURO journal on decision processes* 6(3-4):283–319.

[Benabbou, Leroy, and Lust 2020] Benabbou, N.; Leroy, C.; and Lust, T. 2020. An interactive regret-based genetic algorithm for solving multi-objective combinatorial optimization problems. In *Proceedings of AAAI'20*, 2335–2342.

[Benabbou, Perny, and Viappiani 2017] 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.

[Boutilier et al. 2006] 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.

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

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

[Drummond and Boutilier 2014] Drummond, J., and Boutilier, C. 2014. Preference elicitation and interview minimization in stable matchings. In *Proceedings of AAAI'14*, 645–653.

[Edmonds 1971] Edmonds, J. 1971. Matroids and the greedy algorithm. *Mathematical programming* 1(1):127–136.

[E.H. Aarts 2003] E.H. Aarts, J. L. 2003. *Local search in combinatorial optimization*. Princeton University Press.

[Gelain et al. 2010] 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.

[Gourvès, Monnot, and Tlilane 2013] Gourvès, L.; Monnot, J.; and Tlilane, L. 2013. A matroid approach to the worst case allocation of indivisible goods. In *proceedings of IJCAI'13*, 136–142.

[Gourvès, Monnot, and Tlilane 2014] Gourvès, L.; Monnot, J.; and Tlilane, L. 2014. Near fairness in matroids. In *proceedings of ECAI'14*, volume 263, 393–398.

[Greene and Magnanti 1975] Greene, C., and Magnanti, T. L. 1975. Some abstract pivot algorithms. *SIAM Journal on Applied Mathematics* 29(3):530–539.

[Ha and Haddawy 1997] Ha, V., and Haddawy, P. 1997. Problem-focused incremental elicitation of multi-attribute utility models. In *UAI*, 215–222.

[Kruskal 1956] Kruskal, J. B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society* 7(1):48–50.

[Lee and Ryan 1992] Lee, J., and Ryan, J. 1992. Matroid applications and algorithms. *ORSA Journal on Computing* 4(1):70–98.

[Lee 2004] Lee, J. 2004. *A first course in combinatorial optimization*, volume 36. Cambridge University Press.

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

[Nemhauser, Wolsey, and Fisher 1978] Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming* 14(1):265–294.

[Oxley 2006] Oxley, J. G. 2006. *Matroid theory*, volume 3. Oxford University Press, USA.

[Regan and Boutilier 2009] Regan, K., and Boutilier, C. 2009. Regret-based reward elicitation for Markov decision processes. In *UAI*, 444–451.

[Russell and Norvig 2002] Russell, S., and Norvig, P. 2002. Artificial intelligence: a modern approach.

[Wang and Boutilier 2003] Wang, T., and Boutilier, C. 2003. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. 309–316.

[Weng and Zanuttini 2013] Weng, P., and Zanuttini, B. 2013. Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In *International Joint Conference on Artificial Intelligence*.

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