Interactive Optimization of Submodular Functions under Matroid Constraints

Nawal Benabbou, Cassandre Leroy, Thibaut Lust, Patrice Perny

Interactive Optimization of Submodular Functions under Matroid Constraints

Nawal Benabbou, Cassandre Leroy, Thibaut Lust, Patrice Perny

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
firstname.lastname@lip6.fr

Abstract. Various practical optimization problems can be formalized as the search of an optimal independent set in a matroid. When the set function to be optimized is additive, this problem can be exactly solved using a greedy algorithm. However, in some situations, the set function is not exactly known and must be elicited before or during the optimization process. Moreover, the set function is not always additive due to possible interactions between the elements of the set. Here we consider the problem of maximizing a submodular set function under a matroid constraint. We propose two interactive approaches aiming at interweaving the elicitation of the submodular set function with the construction of an optimal independent subset subject to a matroid constraint. The first one is based on a greedy algorithm and the other is based on local search. These algorithms are tested on practical problems involving a matroid structure and a submodular function to be maximized.

Keywords: Submodular function, matroid, preference elicitation, greedy search, local search.

1 Introduction

In many problems studied in combinatorial optimization, admissible solutions are defined as subsets of a ground set satisfying a structural property. A set function representing the utility or the cost of any subset is generally used to model preferences and the selection problem consists in determining an admissible subset having the maximal utility or the minimal cost. In particular, the optimization of a set function under a matroid constraint has received much attention since the seminal work of Edmonds [5]. This problem has multiple applications in various contexts such as recruitment, committee election, combinatorial auctions, scheduling, resource allocation, facility location and sensor placement, just to give a few examples. Various algorithms are now available to solve this problem either to optimality or approximately, for specific classes of set functions, see e.g., [4, 10–12, 14, 15, 17].

When the set function is additive (i.e., the value of any set is defined as the sum of the values of its elements), it is well known, after Edmonds [5], that the problem can be efficiently solved by a greedy algorithm. However, preferences are not always representable by additive functions due to possible interactions

among elements. In decision theory, the additivity of utilities is often relaxed and submodular utility functions are frequently used to guarantee a principle of diminishing returns [1, 9, 17]. This principle states that adding an element to a smaller set has more value than adding it to a larger set, formally the set function w should satisfy the following property: $w(X \cup \{i\}) - w(X) \geq w(Y \cup \{i\}) - w(Y)$ whenever $X \subseteq Y$ and $i \notin Y$. This is known to be equivalent to submodularity of function w defined by: $w(X \cup Y) + w(X \cap Y) \leq w(X) + w(Y)$ for all X, Y. Various set functions naturally considered in practical problems appear to be both submodular and monotonic with respect to set inclusion. Let us mention, for example, the budgeted-additive set function defined by $w(X) = \min\{\sum_{i \in X} w_i, B\}$, but also the coverage measure defined by $w(X) = |\bigcup_{i \in X} E_i|$ (where E_i is the list of elements covered by i) and satisfaction measures of the form $w(X) = \sum_{i \in I} p_i \max_{j \in X}\{u_{ij}\}$ used in the facility location problem (where I denotes a set of clients and u_{ij} the utility of location j for client i).

Although the problem of minimization of submodular functions is known to be polynomially solvable [14], the maximization of a submodular function is hard in general because it includes max-cut as special case. Approximate greedy and local search algorithms have been proposed for the maximization problem and some interesting worst case bounds on the quality of the approximations returned are known, see, e.g., [4, 12, 15, 17]. In this paper we stay on this problem of finding a global maximum of a general submodular and monotonic set function under a matroid constraint, but in a different perspective. We propose an active learning approach aiming to iteratively collect preference information over sets to progressively infer new preference statements over other sets until an optimal subset can be determined. More precisely, our approach interleaves preference queries with some optimization steps of a combinatorial algorithm aiming to construct an optimal set. In particular, we propose an interactive greedy algorithm and an interactive local search algorithm combining preference elicitation and search to determine an optimal (or near-optimal) subset. The approach proposed in the paper extends to non-additive submodular functions a recent approach proposed in [2] for additive set functions.

The paper is organized as follows: in Section 2 we recall some background on matroids and regret-based incremental preference elicitation. Then in Section 3 we propose a near-admissible interactive greedy search algorithm for submodular optimization. In Section 4 we introduce an interactive local search algorithm based on improving sequences of element swaps for the same problem. In Section 5, both algorithms are tested on optimization problems involving different submodular set functions and different matroid constraints. We analyse and compare their performance and obtain various possible tradeoffs between the number of preference queries and the quality of the final solution returned.

2 Background

In this work, we consider the problem of finding a maximum weight independent set in a matroid. A matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$ where E is a set of size n

(called the *ground set*) and $\mathcal{I} \subseteq 2^E$ is a non-empty collection of sets (called the *independent sets*) such that, for all $X, Y \in 2^E$, the following properties hold:

(A_1): $(Y \in \mathcal{I}$ and $X \subseteq Y) \Rightarrow X \in \mathcal{I}$.
(A_2): $(X \in \mathcal{I}$ and $Y \in \mathcal{I}$ and $|Y| > |X|) \Rightarrow \exists e \in Y \setminus X$ s.t. $X \cup \{e\} \in \mathcal{I}$.

Axiom A_1 is sometimes called the *hereditary property* (or the *downward-closed property*) whereas A_2 is known as the *augmentation property* (or the *independent set exchange property*). 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 *basis* of the matroid, and the set of all bases will be denoted by $\mathcal{B}$ in the sequel. The cardinality of a basis is called the *rank* of the matroid and it will be denoted by $r(\mathcal{M})$ in the sequel. In this paper, a special focus will be given to the *uniform matroid*, which is defined by $\mathcal{I} = \{X \subseteq E : |X| \leq k\}$ for a given positive integer $k \leq n$. In the numerical tests, we will also consider the *partition matroid* which is defined by a collection $\mathcal{D} = \{D_1, \ldots, D_q\}$ of q disjoints subsets of E, a positive integer $d_i \leq |D_i|$ for all $i \in \{1, \ldots, q\}$ and $\mathcal{I} = \{X \subseteq \cup_{i=1}^{q} D_i : \forall i \in \{1, \ldots, q\}, |X \cap D_i| \leq d_i\}$.

The problem of finding a maximum weight independent set in a matroid can be defined as follows: given a matroid $\mathcal{M} = (E, \mathcal{I})$, we want to compute $\max_{X \in \mathcal{I}} w(X)$ where w is a positive set function defined on 2^E measuring the weight (or utility) of any subset of E. Here we assume that $w(\emptyset) = 0$, w is submodular (i.e., $w(X \cup Y) + w(X \cap Y) \geq w(X) + w(Y)$ for all $X, Y \subseteq E$) and w is monotonic with respect to set inclusion (i.e., $w(X) \leq w(Y)$ for all $X \subset Y \subseteq E$). Note that the latter assumption implies that we can focus on the bases of the matroid when searching for an optimal independent subset.

In this paper, we assume that w is a set function representing the subjective preferences of a Decision Maker (DM): for any two sets $X, Y \in 2^E$, X is preferred to Y if and only if $w(X) \geq w(Y)$. Hence finding a maximum weight basis amounts to determining an optimal basis according to the DM's preferences. Moreover, we assume that w is initially not known. 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 (i.e., by asking the DM to compare two subsets and state which one is preferred). Let W be the *uncertainty set* implicitly 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 $(X, Y) \in \mathcal{P}$. The problem is now to determine the most promising basis under preference imprecision. To this end, we consider the minimax regret decision criterion which is commonly used to make robust recommendations under preference imprecision in various decision contexts. The minimax regret (MMR) can be defined using pairwise max regrets (PMR) and max regrets (MR) as follows:

Definition 1 *For any collection of sets $\mathcal{S} \subseteq 2^E$ and for any two sets $X, Y \in \mathcal{S}$:*
$PMR(X, Y, W) = \max_{w \in W}\{w(Y) - w(X)\}$
$MR(X, \mathcal{S}, W) = \max_{Y \in \mathcal{S}} PMR(X, Y, W)$
$MMR(\mathcal{S}, W) = \min_{X \in \mathcal{S}} MR(X, \mathcal{S}, W)$

Thus $PMR(X, Y, W)$ is the worst-case loss when choosing X instead of Y. $MR(X, \mathcal{S}, W)$ is the worst-case loss incurred when selecting X instead of any other set $Y \in \mathcal{S}$. The set $\arg\min_{X \in \mathcal{S}} MR(X, \mathcal{S}, W)$ is the set of all optimal sets according to the minimax regret decision criterion. By definition, recommending any of these optimal sets allows to minimize the worst-case loss. Moreover, if $MMR(\mathcal{S}, W) = 0$, then we know that these sets are necessarily optimal according to the DM's preferences.

Note that, depending on the available preference statements, the MMR value (representing the worst-case loss) might still be at an unacceptable level for the DM. As the MMR value can only decrease when adding new preference statements in $\mathcal{P}$, the minimax regret decision criterion can be used within an incremental elicitation process that progressively asks preference queries to the DM until the MMR value drops below a given *tolerance* threshold $\delta \geq 0$ (representing the maximum allowable gap to optimality) [3]. At that time, recommending any optimal basis for the minimax regret criterion ensures that the loss incurred by not choosing the preferred basis is bounded above by that threshold. This approach is sometimes referred to as *regret-based incremental elicitation* in the literature. Note that if we set $\delta = 0$, then the returned basis is necessarily optimal according to the DM's preferences. However, using $\delta > 0$ allows to reduce the number of generated preference queries in practice.

For matroid optimization problems, computing the MMR value at every step of the elicitation procedure may induce prohibitive computation times as 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, preference queries are generated during the search so as to progressively reduce the set W until being able to determine a (near-)optimal basis.

3 An interactive greedy algorithm

For problems where w is exactly observable, good approximate solutions can be constructed using the following simple greedy algorithm: starting from $X = \emptyset$, the idea is to select an element $e \in E \backslash X$ that maximizes the marginal contribution to X, i.e.,

$$\Delta(e|X) = w(X \cup \{e\}) - w(X) \tag{1}$$

without loosing the independence property. The algorithm stops when no more element can be added to X (set X is a basis at the end of the procedure). For monotonic submodular set functions, this greedy algorithm has an approximation ratio of $(1 - \frac{1}{e}) \approx 0.63$ for the uniform matroid and an approximation ratio of $\frac{1}{2}$ in the general case [6, 12]. For problems where the set function w is imprecisely known, we propose an interactive version of the greedy algorithm that generates preference queries only when it is necessary to discriminate between some elements. More precisely, queries are generated only when the available preference data is not sufficient to identify an element that could be added to set X so as to ensure that the returned basis is a good approximate solution with

provable guarantees. We implement this idea by computing minimax regrets on sets $\mathcal{S} = \{X \cup \{e\} : e \in E \backslash X \text{ s.t. } X \cup \{e\} \in \mathcal{I}\}$, asking preference queries at step i until $MMR(\mathcal{S}, W)$ drops below a given threshold $\delta_i \geq 0$, where δ_i is a fraction of the tolerance threshold δ such that $\sum_{i=1}^{r(\mathcal{M})} \delta_i = \delta$ (see Algorithm 1).

Algorithm 1: Interactive Greedy Algorithm

```
1  X ← ∅;
2  E_c ← E;
3  for i = 1 ... r(M) do
4  |   S ← {X ∪ {e} : e ∈ E_c};
5  |   while MMR(S, W) > δ_i do
6  |   |   Ask the DM to compare two elements of S;
7  |   |   Update W according to the DM's answer;
8  |   end
9  |   Select e ∈ E_c such that MR(X ∪ {e}, S, W) ≤ δ_i and move e from E_c to X;
10 |   Remove from E_c all elements e such that X ∪ {e} ∉ I;
11 end
12 return X;
```

Note that Algorithm 1 generates no more than a polynomial number of queries. At every step, the number of queries is indeed bounded above by $|E|^2$ as comparison queries are generated until $MMR(\mathcal{S}, W) \leq \delta_i$, where $\mathcal{S} \subseteq \{X \cup \{e\} : e \in E\}$ (in the worst-case scenario, the DM is asked to compare all the elements of $\mathcal{S}$). Hence the number of steps of the while loop is also polynomial. Note however that the implementation of Algorithm 1 may differ significantly from one application context to another. In particular, checking whether $X \cup \{e\} \in \mathcal{I}$ can be more or less complex depending on the matroid under consideration. For example, when considering the uniform and partition matroids, the independence tests (line 10) can be performed in polynomial time. A second source of complexity is the computation of MMR values, which can be more or less simple depending on the assumptions made on w. An interesting option is to focus on parametric functions that are linear in their 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, defining w by a parametric function enables to reduce the number of queries in practice, since any preference statement of type $w(X) \geq w(X')$ translates into a constraint on the parameter space, reducing possible preferences over other subsets.

We now provide theoretical guarantees on the quality of the returned solution. Before considering the general case, let us focus on the uniform matroid.

Proposition 1. *Let W_f be the final set W when Algorithm 1 stops. For the uniform matroid, Algorithm 1 is guaranteed to return a basis X such that:*

$$\forall w \in W_f,\ w(X) \geq \big(1 - \frac{1}{e}\big) w(X^*) - \delta, \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} w(Y).$$

Proof. Let $w \in W_f$ and let $X^* \in \arg\max_{Y \in \mathcal{I}} w(Y)$. We want to prove that $w(X) \geq \big(1 - \frac{1}{e}\big) w(X^*) - \delta$ holds. Let e_i, $i \in \{1, \ldots, r(\mathcal{M})\}$, be the ith element inserted in X during the execution of Algorithm 1. Let X_i be the set X at the end of the ith iteration step (i.e., $X_i = \{e_1, \ldots, e_i\}$). Let W_i (resp. $\mathcal{S}_i$) denote the uncertainty set W (resp. the set $\mathcal{S}$) at the end of the ith iteration step. Let e_i^*, $i \in \{1, \ldots, r(\mathcal{M})\}$, denote the ith element of X^* in an arbitrary order.

For any step $i \in \{1, \ldots, r(\mathcal{M})\}$, we have $MR(X_{i-1} \cup \{e_i\}, \mathcal{S}_i, W_i) \leq \delta_i$ due to line 9. Since $w \in W_f \subseteq W_i$, we know that $w(X_{i-1} \cup \{e\}) - w(X_{i-1} \cup \{e_i\}) \leq \delta_i$ for all $e \in E_c$, where $E_c = E \backslash X_{i-1}$ for the uniform matroid (see lines 2 and 10). Then, from Equation (1), we can derive $\Delta(e|X_{i-1}) - \Delta(e_i|X_{i-1}) \leq \delta_i$ for all $e \in E \backslash X_{i-1}$. Note that the last inequality also holds for all $e \in X_{i-1}$ as $\Delta(e|X_{i-1}) = 0$. Hence, for any step $i \in \{1, \ldots, r(\mathcal{M})\}$, we have:

$$\Delta(e|X_{i-1}) - \Delta(e_i|X_{i-1}) \leq \delta_i, \, \forall e \in E \tag{2}$$

Then we obtain:

$$\begin{aligned}
w(X^*) &\leq w(X_{i-1} \cup X^*) \text{ (since } w \text{ is monotonic)} \\
&= w(X_{i-1}) + \sum_{j=1}^{r(\mathcal{M})} \big(w(X_{i-1} \cup \{e_1^*, \ldots, e_j^*\}) - w(X_{i-1} \cup \{e_1^*, \ldots, e_{j-1}^*\}) \big) \\
&= w(X_{i-1}) + \sum_{j=1}^{r(\mathcal{M})} \Delta(e_j^*|X_{i-1} \cup \{e_1^*, e_2^*, \ldots, e_{j-1}^*\}) \text{ (by Equation (1))} \\
&\leq w(X_{i-1}) + \sum_{j=1}^{r(\mathcal{M})} \Delta(e_j^*|X_{i-1}) \text{ (since } w \text{ is submodular)} \\
&\leq w(X_{i-1}) + \sum_{j=1}^{r(\mathcal{M})} (\Delta(e_i|X_{i-1}) + \delta_i) \text{ (by Equation (2))} \\
&= w(X_{i-1}) + r(\mathcal{M}) \times \big(\Delta(e_i|X_{i-1}) + \delta_i \big)
\end{aligned}$$

From the last inequality, we can derive:

$$\frac{1}{r(\mathcal{M})} \big(w(X^*) - w(X_{i-1}) \big) - \delta_i \leq \Delta(e_i|X_{i-1})$$

which can be rewritten as follows:

$$\frac{1}{r(\mathcal{M})} \big(w(X^*) - w(X_{i-1}) \big) - \delta_i \leq w(X^*) - w(X_{i-1}) - \big(w(X^*) - w(X_i) \big)$$

since $X_i = X_{i-1} \cup \{e_i\}$. Therefore we have $\frac{\Pi_{i-1}}{r(\mathcal{M})} - \delta_i \leq \Pi_{i-1} - \Pi_i$ or equivalently:

$$\Pi_i \leq \big(1 - \frac{1}{r(\mathcal{M})}\big) \Pi_{i-1} + \delta_i$$

where Π_i is simply defined by $\Pi_i = w(X^*) - w(X_i)$ for all $i \in \{0, \ldots, r(\mathcal{M})\}$. By recursively applying this inequality, we obtain:

$$\Pi_{r(\mathcal{M})} \leq \left(1 - \frac{1}{r(\mathcal{M})}\right)^{r(\mathcal{M})} \times \Pi_0 + \sum_{i=1}^{r(\mathcal{M})} \delta_i \left(1 - \frac{1}{r(\mathcal{M})}\right)^{r(\mathcal{M})-i}$$

Then, since $\Pi_0 = w(X^*)$ and $\Pi_{r(\mathcal{M})} = w(X^*) - w(X)$, we obtain:

$$w(X^*) - w(X) \leq \left(1 - \frac{1}{r(\mathcal{M})}\right)^{r(\mathcal{M})} \times w(X^*) + \sum_{i=1}^{r(\mathcal{M})} \delta_i \left(1 - \frac{1}{r(\mathcal{M})}\right)^{r(\mathcal{M})-i}$$

or equivalently:

$$w(X) \geq \left(1 - \big(1 - \frac{1}{r(\mathcal{M})}\big)^{r(\mathcal{M})}\right) w(X^*) - \sum_{i=1}^{r(\mathcal{M})} \delta_i \left(1 - \frac{1}{r(\mathcal{M})}\right)^{r(\mathcal{M})-i}$$

Finally, using $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$, and $1 - \frac{1}{x} \leq 1$ for all $x \in \mathbb{R}_+^*$, we obtain:

$$w(X) \geq \big(1 - \frac{1}{e}\big) w(X^*) - \sum_{i=1}^{r(\mathcal{M})} \delta_i = \big(1 - \frac{1}{e}\big) w(X^*) - \delta$$

□

Note that Proposition 1 cannot be extended to the case of general matroid, as inequalities of type $\Delta(e_i|X_{i-1}) + \delta_i \geq \Delta(e_j^*|X_{i-1})$ may not hold anymore ($E_c \neq E \backslash X_i$ in the general case). We now establish a more general result.

Proposition 2. *Let W_f be the final set W when Algorithm 1 stops. Algorithm 1 is guaranteed to return a basis X such that:*

$$\forall w \in W_f,\ w(X) \geq \frac{1}{2}\big(w(X^*) - \delta\big), \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} w(Y).$$

Proof. Let $w \in W_f$ and let $X^* \in \arg\max_{Y \in \mathcal{I}} w(Y)$. We want to prove that $w(X) \geq \frac{1}{2}(w(X^*) - \delta)$ holds. Let e_i, $i \in \{1, \ldots, r(\mathcal{M})\}$, be the ith element inserted in X during the execution of Algorithm 1. Let X_i be the set X at the end of the ith iteration step (i.e., $X_i = \{e_1, \ldots, e_i\}$). Let W_i (resp. $\mathcal{S}_i$) denote the uncertainty set W (resp. the set $\mathcal{S}$) at the end of the ith iteration step.

Due to a well-known multiple exchange theorem [7], there exists a one-to-one correspondence $\sigma : X \to X^*$ such that $B_i = (X \backslash \{e_i\}) \cup \{\sigma(e_i)\}$ is a basis of the matroid for every element $e_i \in X$. Then we can derive $X_{i-1} \cup \{\sigma(e_i)\} \in \mathcal{I}$ from $X_{i-1} \cup \{\sigma(e_i)\} \subseteq B_i$ (using Axiom A_1), and therefore we necessarily have $X_{i-1} \cup \{\sigma(e_i)\} \in \mathcal{S}_i$ at step i. Since $MR(X_{i-1} \cup \{e_i\}, \mathcal{S}_i, W_i) \leq \delta_i$ (line 9), we obtain $w(X_{i-1} \cup \{\sigma(e_i)\}) - w(X_{i-1} \cup \{e_i\}) \leq \delta_i$, which can be rewritten:

$$\Delta(\sigma(e_i)|X_{i-1}) - \Delta(e_i|X_{i-1}) \leq \delta_i \tag{3}$$

Then we obtain:

$$
\begin{aligned}
w(X^*) &\leq w(X \cup X^*) \text{ (since } w \text{ is monotonic)} \\
&= w(X) + \sum_{i=1}^{r(\mathcal{M})} \big(w(X \cup \{\sigma(e_1), \ldots, \sigma(e_i)\}) - w(X \cup \{\sigma(e_1), \ldots, \sigma(e_{i-1})\})\big) \\
&= w(X) + \sum_{i=1}^{r(\mathcal{M})} \Delta(\sigma(e_i) | X \cup \{\sigma(e_1), \ldots, \sigma(e_{i-1})\}) \text{ (by Equation (1))} \\
&\leq w(X) + \sum_{i=1}^{r(\mathcal{M})} \Delta(\sigma(e_i) | X_{i-1}) \text{ (since } w \text{ is submodular and } X_{i-1} \subseteq X) \\
&\leq w(X) + \sum_{i=1}^{r(\mathcal{M})} (\Delta(e_i | X_{i-1}) + \delta_i) \text{ (by Equation (3))} \\
&= 2w(X) + \sum_{i=1}^{r(\mathcal{M})} \delta_i \text{ (by Equation (1))} \\
&= 2w(X) + \delta \text{ (which establishes the result)} \qquad \square
\end{aligned}
$$

Example 1. We now present an execution of our algorithm. Consider an instance of the maximum coverage problem over a uniform matroid with a set $V = \{v_1, \ldots, v_q\}$, $q = 10$, and a family of $n = 8$ subsets $E = \{S_1, \ldots, S_n\}$ defined by:

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
v_3	v_1	v_6	v_2	v_7	v_6	v_2	v_1
v_4	v_3	v_{10}	v_8	v_9	v_7	v_8	v_3
v_5					v_{10}		v_5

Table 1. Subsets used in Example 1.

A feasible solution is a collection of subsets $X \subseteq E$ such that $|X| \leq k$ (here we set $k = 2$), and the goal is to identify a feasible solution X maximizing $w(X)$ for a given set function w defined on 2^E. Here we assume that w is defined by:

$$w(X) = \sum_{v \in \bigcup_{S \in X} S} u(v) \tag{4}$$

where $u(v) \geq 0$ is the utility of element $v \in V$. In that case, it can be proved that w is monotone and submodular [8]. We further assume that all elements $v \in V$ are evaluated with respect to 3 criteria (denoted by u_1, u_2, and u_3), and their evaluations are given in Table 2. Then, the utility of any element $v \in V$ is:

$$u(v) = \sum_{i=1}^{3} \lambda_i u_i(v) \tag{5}$$

where $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}_+$ represents the value system of the DM.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
u_1	4	2	2	3	7	6	8	7	7	1
u_2	5	7	1	2	3	1	5	1	9	1
u_3	4	5	3	7	2	5	3	8	4	4

Table 2. Performance vectors attached to elements in Example 1.

In this example, we assume that the DM's preferences can be represented by the set function w^* defined by the hidden parameter $\lambda^* = (0.2, 0.5, 0.3)$. Here we start the execution with no preference data, and therefore we have to consider all weighting vectors λ in the set $\Lambda = \{\lambda \in [0,1]^3 : \sum_{i=1}^{3} \lambda_i = 1\}$, which implicitly defines the uncertainty set W using Equations (4-5). In Figure 1, Λ is represented by triangle ABC in the space (λ_1, λ_2), λ_3 being implicitly defined by $\lambda_3 = 1 - \lambda_1 - \lambda_2$. Now, let us execute Algorithm 1 with $\delta = 0$. Note that only two iteration steps are needed as the rank of the uniform matroid is equal to k.

First iteration step: We have $X = \emptyset$ and $E_c = E$, and therefore $\mathcal{S} = E$. Since $MMR(\mathcal{S}, W) = 6 > 0$, the DM is asked to compare two elements of $\mathcal{S}$, say S_5 and S_7. Since we have $w^*(\{S_5\}) = 12.1 \geq 9.7 = w^*(\{S_7\})$, the answer is: "subset S_5 is better than subset S_7". Then W is updated by imposing the constraint $w(\{S_5\}) \geq w(\{S_7\})$ which amounts to restricting Λ by imposing $\lambda_2 \geq \frac{1}{2} - \lambda_1$. Now Λ is represented by the polyhedron BCDE in Figure 2. Since $MMR(\mathcal{S}, W) = 2.5$, the DM is asked to compare two subsets, say S_5 and S_6. Since we have $w^*(\{S_5\}) = 12.1 \geq w^*(\{S_6\}) = 10.1$, the DM answers: "subset S_5 is better than subset S_6". Then, W is updated by imposing the constraint $w(\{S_5\}) \geq w(\{S_6\})$, which amounts to further restricting Λ by imposing $\lambda_2 \geq \frac{5}{12} - \frac{5}{12}\lambda_1$. Now Λ is represented by the polyhedron BCFE in Figure 3. We have $MMR(\mathcal{S}, W) = MR(\{S_5\}, \mathcal{S}, W) = 0$, and therefore S_5 is added to X.

Second iteration step: We have $X = \{S_5\}$ and $E_c = E \backslash \{S_5\}$, and therefore $\mathcal{S} = \{\{S_5\} \cup \{S\} : S \in E_c\}$. Since $MMR(\mathcal{S}, W) = 1.5$, we ask the DM to compare two elements of $\mathcal{S}$, say $\{S_5, S_8\}$ and $\{S_5, S_7\}$. The DM prefers the former option as $w^*(\{S_5, S_8\}) = 21.9 \geq w(\{S_5, S_7\}) = 21.8$. The uncertainty set W is therefore updated by imposing $w(\{S_5, S_8\}) \geq w(\{S_5, S_7\})$, i.e., $\lambda_2 \geq 1 - \frac{8}{3}\lambda_1$. Now Λ is represented by the triangle BGC in Figure 4. Since we have $MMR(\mathcal{S}, W) = MR(\{S_5, S_8\}, \mathcal{S}, W) = 0$, then subset S_8 is added to X.

As $|X| = k = 2$, the algorithm stops and returns $X = \{S_5, S_8\}$ which is the optimal solution for this instance. This shows that we are able to make good recommendations without knowing λ^* precisely (here only 3 queries are needed).

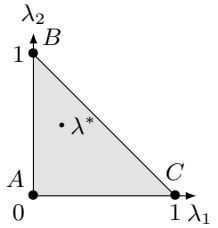

Fig. 1. Initial set Λ.

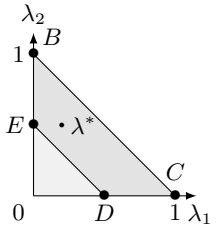

Fig. 2. Λ after 1 query.

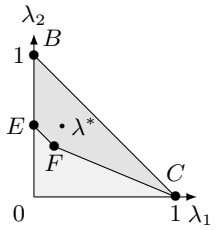

Fig. 3. Λ after 2 queries.

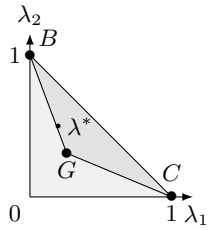

Fig. 4. Λ after 3 queries.

4 An interactive local search

In this section, we consider another efficient way of constructing a good approximate solution to matroid optimization problems with monotonic submodular functions. More precisely, we focus on the following simple local search approach: starting from an arbitrary basis X, the idea is to replace one element $e \in X$ by an element $e' \in E \backslash X$ such that $X \cup \{e'\} \backslash \{e\}$ belongs to $\mathcal{I}$ and is better than X. This simple exchange principle can be iterated until reaching a local optimum. When w is exactly observable, the local search algorithm has an approximation ratio of $1/2$, even in the special case of the uniform matroid [6]. When w is not known, the local search algorithm can be combined with a preference elicitation method which collects preference data only when it is necessary to identify improving swaps. To implement this idea, we propose Algorithm 2 where N_X is the neighborhood of basis X (i.e., the set of bases that differ from X by exactly one element). The procedure `ComputeInitialBasis` called in line 1 can be any heuristic providing a good starting solution (see the numerical tests).

Algorithm 2: The Interactive Local Search Algorithm

```
1  X ← ComputeInitialBasis(M);
2  improve ← true;
3  while improve do
4  |   N_X ← {X' ∈ B : |{X \ X'} ∪ {X' \ X}| = 2};
5  |   S ← N_X ∪ {X};
6  |   while MMR(S, W) > δ/r(M) do
7  |   |   Ask the DM to compare two elements of S;
8  |   |   Update W according to the DM's answer;
9  |   end
10 |   if MR(X, S, W) ≤ δ/r(M) then
11 |   |   improve ← false
12 |   else
13 |   |   X ←RandomSelect(arg min_{X'∈N_X} MR(X', S, W))
14 |   end
15 end
16 return X;
```

The following proposition shows that the basis returned by Algorithm 2 is a good approximate solution.

Proposition 3. *Let W_f be the final set W when Algorithm 2 stops. Algorithm 2 is guaranteed to return a basis X such that:*

$$\forall w \in W_f,\ w(X) \geq \frac{1}{2}\big(w(X^*) - \delta\big), \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} w(Y).$$

Proof. Let $w \in W_f$ and let $X^* \in \arg\max_{Y\in\mathcal{I}} w(Y)$. We want to prove that $w(X) \geq \frac{1}{2}(w(X^*) - \delta)$ holds. Let e_i, $i \in \{1, \ldots, r(\mathcal{M})\}$, denote the ith element of X in an arbitrary order. Let X_i be the set defined by $X_i = \{e_1, \ldots, e_i\}$. Due to the multiple exchange theorem, there exists a one-to-one correspondence $\sigma : X \to X^*$ such that $B_i = (X \backslash \{e_i\}) \cup \{\sigma(e_i)\}$ is a basis of the matroid for every element $e_i \in X$. Note that $B_i \in N_X$ (the neighborhood of X) for all $i \in \{1, \ldots, r(\mathcal{M})\}$ since B_i differs from X by exactly one element. Moreover, we have $MR(X, N_X, W) \leq \delta / r(\mathcal{M})$ at the end of the execution (due to line 10). Therefore, $w(B_i) - w(X) \leq \delta / r(\mathcal{M})$ holds by definition of max regrets, which can be rewritten:

$$\Delta(\sigma(e_i)|X\backslash\{e_i\}) - \Delta(e_i|X\backslash\{e_i\}) \leq \frac{\delta}{r(\mathcal{M})} \tag{6}$$

using Equation (1). Then, we obtain:

$$\begin{aligned}
w(X^*) &\leq w(X) + \sum_{e\in X^*} \Delta(e|X) \text{ (by submodularity, see [12] for a proof)} \\
&= w(X) + \sum_{i=1}^{r(\mathcal{M})} \Delta(\sigma(e_i)|X) \\
&\leq w(X) + \sum_{i=1}^{r(\mathcal{M})} \Delta(\sigma(e_i)|X \setminus \{e_i\}) \text{ (by submodularity)} \\
&\leq w(X) + \sum_{i=1}^{r(\mathcal{M})} \big(\Delta(e_i|X \setminus \{e_i\}) + \frac{\delta}{r(\mathcal{M})}\big) \text{ (by Equation (6))} \\
&\leq w(X) + \sum_{i=1}^{r(\mathcal{M})} \big(\Delta(e_i|X_{i-1}) + \frac{\delta}{r(\mathcal{M})}\big) \text{ (since } X_{i-1} \subseteq X\backslash\{e_i\}) \\
&= 2w(X) + \delta \text{ (which establishes the result)}
\end{aligned}$$

□

Contrary to the greedy algorithm, we cannot prove that the local search algorithm generates a polynomial number of queries and ends after a polynomial number of iterations. More precisely, when $\delta \neq 0$, some cycles of type $(X_1, \ldots, X_t)$ with $X_{i+1} \in N_{X_i}$ and $X_1 = X_t$ can even occur. Fortunately, when a cycle is detected, it can be easily broken by iteratively dividing δ by two (while

still guaranteeing the near-optimality of the returned basis). Despite these poor theoretical properties, we will see in the experimental section that the local search algorithm achieves good results in practice.

5 Experimental results

In this section, we report the results obtained by our algorithms on two problems: the maximum coverage problem and the collective selection of items. Two matroid constraints have been considered for each problem: the uniform matroid and the partition matroid. The algorithms are evaluated through three performance indicators: number of queries, computation times (given in seconds) and empirical error, expressed as a percentage from the optimal solution. For the local search algorithm, we also report the number of iteration steps (NbI) performed by the algorithm. In our tests, two tolerance thresholds have been used: $\delta = 0$ and $\delta = 20\%$ of the initial maximum regret (to reduce the number of preference queries). Results are averaged over 30 runs. All the results have been obtained with a program written in C++ and 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).

To generate preference queries during the execution of our algorithms, we use the well-known query selection strategy called the Current Solution Strategy (CSS)[3] which consists in asking the DM to compare a solution X minimizing the max regret to one of its best challengers arbitrary chosen in the set $\arg\max_Y PMR(X, Y, W)$.

5.1 The maximum coverage problem

Here we consider instances of the maximum coverage problem with a set $V = \{v_1, \ldots, v_q\}$ of $q = 100$ elements, and a family E of $n = 80$ subsets of V. The family of subsets are generated as suggested in [13]. The utility of an element $v \in V$ is defined by a weighted sum $u^\lambda(v) = \sum_{i=1}^{p} \lambda_i u_i(v)$ where u_i is the evaluation of v on criterion $i \in \{1, \ldots, p\}$. Utilities are randomly generated within $[1, 10]$ and three values of p are considered: $p = 4, 6$, and 8. The DM's preferences are then represented by a submodular monotone set function w defined by:

$$w(X) = \sum_{v \in \bigcup_{S \in X} S} u^\lambda(v)$$

for any $X \subseteq E$. Here we assume that λ is initially unknown. Answers to queries are simulated using a hidden vector λ randomly generated before running the algorithms. For the uniform matroid, we focus on subsets of size at most $k = 16$, i.e., $\mathcal{I} = \{X \subset E : |X| \leq 16\}$. For the partition matroid, set E is randomly partitioned into $q = 4$ sets $\mathcal{D} = \{D_1, \ldots, D_q\}$, and at most $d_i = 4$ elements can be selected for all $i \in \{1, \ldots, q\}$, i.e., $\mathcal{I} = \{X \subseteq E : \forall i \in \{1, \ldots, 4\}, |X \cap D_i| \leq 4\}$. The results are given in Table 3 and Table 4

		Greedy			Local search (random start)				Local search (greedy start)			
δ	p	time(s)	queries	error(%)	time(s)	queries	error(%)	NbI	time(s)	queries	error(%)	NbI
0	4	4.4	19.4	1.1	0.2	17.2	0.9	12.6	0.2	12.1	0.5	2.9
	6	6.9	36.2	1.3	0.8	32.4	1.0	12.6	0.3	17.1	0.4	2.5
	8	9.9	48.3	1.1	1.4	42.9	0.8	12.5	0.5	28.5	0.3	3.0
0.2	4	2.8	7.2	1.3	0.1	7.6	6.1	7.7	0.1	9.5	0.5	2.9
	6	3.9	13.7	1.5	0.3	13.1	4.6	8.9	0.3	13.7	0.4	2.4
	8	5.0	17.3	1.3	0.6	19.5	4.0	9.4	0.4	21.8	0.3	2.9

Table 3. Results obtained for the maximum coverage problem with uniform matroid.

		Greedy			Local search (random start)				Local search (greedy start)			
δ	p	time(s)	queries	error(%)	time(s)	queries	error(%)	NbI	time(s)	queries	error(%)	NbI
0	4	3.8	20.5	4.2	0.2	16.2	4.1	11.1	0.1	9.0	2.2	2.8
	6	5.4	33.4	3.5	0.6	25.5	4.5	10.7	0.2	13.4	1.8	2.7
	8	6.8	44.3	4.1	0.9	35.9	4.3	10.6	0.3	17.7	2.2	3.0
0.2	4	2.4	7.4	4.3	0.1	7.5	6.8	7.7	0.1	7.2	2.3	2.7
	6	3.1	12.4	3.9	0.2	10.6	7.2	7.6	0.2	10.3	1.9	2.7
	8	4.1	17.8	4.5	0.4	15.9	5.9	8.8	0.3	14.6	2.2	3.0

Table 4. Results obtained for the maximum coverage problem with partition matroid.

respectively. For our local search algorithm, we consider two implementations of the procedure `ComputeInitialBasis`: we generate a basis at random (random start), or we use the standard greedy algorithm with the uniform weighting vector $\lambda = (1/p, \ldots, 1/p)$ (greedy start).

For $\delta = 0$, we observe that the interactive greedy algorithm is outperformed by the interactive local search procedures: the interactive greedy algorithm is about 10 times slower on average and asks more preference queries. Moreover, we observe that the local search performs better when considering the greedy start heuristic instead of the random start heuristic. We also observe that using $\delta = 0.2$ allows to significantly reduce the number of queries, without increasing the error too much (except for the local search with a random starting point). Finally, we observe that our algorithms perform better on the uniform matroid than on the partition matroid which is a little more complex.

5.2 The collective subset selection problem

In the collective subset selection problem, we are given a set A of m agents, and a set $E = \{e_1, \ldots, e_n\}$ of n items. Every agent $a \in A$ gives a score $s_a(e) \geq 0$ to each item $e \in E$, and the utility that agent a derives from a set $X \subseteq E$ is defined by an ordered weighted average (OWA) [18]. More precisely, for any $X = \{x_1, \ldots, x_\ell\} \subseteq E$ of size $\ell \leq n$, the utility of agent a is defined by:

$$u_a^\lambda(X) = \sum_{i=1}^{\ell} \lambda_i s_a(x_{(i)})$$

where $(\cdot)$ is a permutation of $\{1,\ldots,\ell\}$ sorting the elements of X is non-increasing order (i.e., $s_a(x_{(1)}) \geq \ldots \geq s_a(x_{(\ell)})$), and $\lambda = (\lambda_1,\ldots,\lambda_n) \in [0,1]^n$ is a non-increasing normalized vector. Here the set function w is simply defined by:

$$w(X) = \sum_{a\in A} u_a^{\lambda}(X)$$

Note that function w is a submodular as functions u_a^{λ}, $a \in A$, are submodular whenever vector λ is non-increasing (see Skowron *et al* [16]). Here also we assume that λ is initially unknown, and answers to queries are simulated using a hidden weighting vector. We consider instances with $m = 50$ agents and $n = 50$ items, and scores are randomly generated within in $[1, 100]$. For the uniform matroid, two values of k have been tested: $k = 5$, and $k = 10$. For the partition matroid, E is randomly partitioned into 4 sets, and at most $d/4$ items can be selected in each set; we consider two values of d ($d = 8$ and $d = 16$). The results are given in Table 5 and Table 6 respectively. For this problem, we observe that our interactive greedy algorithm outperforms our interactive local search procedure. With $\delta = 0.2$, the greedy algorithm is very efficient: the error is less than 0.2% and the number of queries does not exceed 7.

		Greedy			Local search (random start)				Local search (greedy start)			
δ	d	time(s)	queries	error(%)	time(s)	queries	error(%)	NbI	time(s)	queries	error(%)	NbI
0	5	0.9	8.2	0.2	3.1	14.4	0.1	5.8	1.6	11.4	0.0	3.1
	10	2.7	25.1	0.1	34.4	45.6	0.0	9.3	31.2	35.5	0.0	4.3
0.2	5	0.6	3.0	0.2	0.8	6.8	0.5	5.0	1.1	6.9	0.1	2.9
	10	1.2	4.3	0.1	12.8	16.6	0.6	7.5	17.7	18.5	0.0	4.1

Table 5. Collective selection of items problem under uniform matroid constraint.

		Greedy			Local search (random start)				Local search (greedy start)			
δ	d	time(s)	queries	error(%)	time(s)	queries	error(%)	NbI	time(s)	queries	error(%)	NbI
0	8	1.3	14.0	0.2	2.2	24.0	0.0	8.1	1.5	17.7	0.0	3.8
	16	3.1	38.9	0.1	21.5	56.9	0.0	12.0	0.4	34.0	0.0	4.1
0.2	8	0.8	4.2	0.2	1.8	18.5	0.1	8.1	1.2	13.4	0.1	3.7
	16	1.4	6.3	0.1	16.0	43.4	0.2	11.4	6.9	25.7	0.0	4.0

Table 6. Collective selection of items problem under partition matroid constraint.

6 Conclusion

We have proposed two interactive algorithms (greedy and local search) combining the elicitation of a submodular utility function and the determination of the optimal independent subset in a weighted matroid. Both algorithms admit performance guarantees on the quality of the returned basis. The tradeoff between the quality of solutions and the number of preference queries used in the process can be controlled by the parameter δ used to define admissible max regrets. Our approach has been tested on two specific problems, but many others could be solved with similar performances due to the generality of matroids.

Note also that a counterpart of this interactive approach could be proposed for the optimization of supermodular functions.

References

1. S. Ahmed and A. Atamtürk. Maximizing a class of submodular utility functions. *Mathematical programming*, 128(1):149–169, 2011.
2. N. Benabbou, C. Leroy, T. Lust, and P. Perny. Combining preference elicitation with local search and greedy search for matroid optimization. In *35th AAAI Conference on Artificial Intelligence (AAAI'21)*, 2021.
3. C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8–9):686–713, 2006.
4. G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
5. J. Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1):127–136, 1971.
6. M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. *An analysis of approximations for maximizing submodular set functions-II*, volume 8, pages 73–87. Springer, 1978.
7. C. Greene and T.L. Magnanti. Some abstract pivot algorithms. *SIAM Journal on Applied Mathematics*, 29(3):530–539, 1975.
8. D.S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.
9. B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.
10. M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.
11. G.L. Nemhauser and L.A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *North-Holland Mathematics Studies*, volume 59, pages 279–301. Elsevier, 1981.
12. G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical programming*, 14(1):265–294, 1978.
13. M. Resende. Computing approximate solutions of the maximum covering problem with GRASP. *Journal of Heuristics*, 4(2):161–177, 1998.
14. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.
15. P. Skowron. FPT approximation schemes for maximizing submodular functions. *Information and Computation*, 257:65–78, 2017.
16. P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artif. Intell.*, 241:191–216, 2016.
17. J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.
18. R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.