# Dual-Mode Greedy Algorithms Can Save Energy

**Barbara Geissmann**
Department of Computer Science, ETH Zürich, Switzerland
barbara.geissmann@inf.ethz.ch

**Stefano Leucci**
Department of Algorithms and Complexity, Max Planck Institute for Informatics, Germany*
stefano.leucci@mpi-inf.mpg.de

**Chih-Hung Liu**
Department of Computer Science, ETH Zürich, Switzerland
chih-hung.liu@inf.ethz.ch

**Paolo Penna**
Department of Computer Science, ETH Zürich, Switzerland
paolo.penna@inf.ethz.ch

**Guido Proietti**
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università dell'Aquila, Italy
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR, Roma, Italy
guido.proietti@univaq.it

## Abstract

In real world applications, important resources like energy are saved by deliberately using so-called low-cost operations that are less reliable. Some of these approaches are based on a *dual mode* technology where it is possible to choose between high-energy operations (always correct) and low-energy operations (prone to errors), and thus enable to trade energy for correctness.

In this work we initiate the study of algorithms for solving optimization problems that in their computation are allowed to choose between two types of operations: *high-energy comparisons* (always correct but expensive) and *low-energy comparisons* (cheaper but prone to errors). For the errors in low-energy comparisons, we assume the *persistent* setting, which usually makes it impossible to achieve optimal solutions without high-energy comparisons. We propose to study a natural complexity measure which accounts for the number of operations of either type separately.

We provide a new family of algorithms which, for a fairly large class of maximization problems, return a *constant approximation* using only *polylogarithmic* many high-energy comparisons and only $O(n \log n)$ low-energy comparisons. This result applies to the class of *p-extendible systems* [24], which includes several *NP-hard* problems and *matroids* as a special case ($p = 1$).

These algorithmic solutions relate to some fundamental aspects studied earlier in different contexts: (i) the approximation guarantee when only *ordinal information* is available to the algorithm; (ii) the fact that even such ordinal information may be *erroneous* because of low-energy comparisons and (iii) the ability to approximately sort a sequence of elements when comparisons are subject to *persistent errors*. Finally, our main result is quite general and can be parametrized and adapted to other error models.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis

**Keywords and phrases** matroids, $p$-extendible systems, greedy algorithm, approximation algorithms, high-low energy

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2019.64

**Funding** Research supported by the Swiss National Science Foundation (SNFS project 200021_165524).

* Part of this work was completed while the author was affiliated with ETH Zürich.

**Acknowledgements** We are grateful to Peter Widmayer for many inspiring discussions.

# 1 Introduction

Classical computational problems have been studied under two (somewhat extreme) settings: the one in which every operation is always *correct* and the one in which operations are prone to *errors* (see, e.g., [10, 28, 11, 20]). The latter scenario represents not only faults in hardware, but also measurement errors, or even errors that are deliberately introduced in the system in order to save important resources. For instance, several approaches to save energy in computation consists in designing systems which are in part inaccurate but use substantially less energy [22, 1, 21].

Here we consider the scenario in which these two types of operations *coexist* and they may be *combined* in a clever way in order to save resources and still achieve a certain goal. Interestingly, this is already done in some practical applications. At a hardware level, the *dual mode logic* [21] allows each single gate to switch between a static mode, which uses *low energy* but suffers from some performance degradation, and a dynamic mode which uses a *higher energy* but reaches the better performance of standard CMOS gates. Similarly, in *probabilistic CMOS* [1] one can reduce the energy spent by a single gate at the price of increasing the probability of *error* in the corresponding output. In [7], the authors propose a *probabilistic adder* for image processing purposes where *high energy* is used in the most significant bits and *lower energy* in the less significant bits. In computational geometry, [12] suggests a model in which the algorithm uses either *cheap* operations (floating-point arithmetic), whose result may be erroneous in some circumstances, and *expensive* operations (exact arithmetic) whose result is always correct.

In all these examples, a good solution is obtained by combining both high-energy and low-energy (expensive and cheap) operations in a suitable way. This suggests the following *algorithmic* question regarding the *trade-offs* between high-energy and low-energy operations:

> *Suppose an algorithm can use both high-energy and low-energy operations, the latter being erroneous, according to some error model. How many such high-energy and low-energy operations are needed to obtain a good solution for a given problem?*

## 1.1 Our contribution

We propose to evaluate algorithms according to a simple and natural measure that we call *high-low* energy complexity. In this model, algorithms can operate at *low energy* or at *high energy*. The former mode is cheap, but introduces some *errors* in the result of the operation, while the latter is more expensive but *always correct*. We evaluate the performance of the algorithm by essentially distinguishing between the two types of operations on inputs of size $n$:

- Total number $h(n)$ of *high-energy* operations;
- Total number $l(n)$ of *low-energy* operations.

In this case, we say that the algorithm has $\langle h(n), l(n)\rangle$*-high-low energy complexity*. Because high-energy operations are more expensive, the classical notion of time complexity does not fully capture the possible *trade-offs* that may result in a lower total energy. Indeed, it may be possible to have algorithms which use *significantly fewer* high-energy operations, and essentially the same number of low-energy operations, and still obtain (nearly) optimal solutions.

In this work, we show that such trade-offs are indeed possible for a large class of problems. Specifically, we consider the setting in which the basic operations are *comparisons* between the input *weights*, which is the part of the input necessary to determine if a solution is optimal or not. For the errors in the comparisons at low energy, we assume the classical model of *persistent* errors [6, 18, 15]: comparisons between distinct pairs of elements are independent and return the wrong answer with some (small) constant probability independently across the pairs; however, comparing the same pair of elements multiple times will *always* give the same result. We consider the following setting:

- The input elements that can be part of feasible solutions have a *weight*, but the algorithm can only work with *ordinal information* meaning that it can only compare the weight of two elements, but does not know the actual weights. The computed solution is however evaluated with respect to the weights and it is compared with the optimum.
- The ordinal information is accessible to the algorithm via two types of (comparison) operations: *low-energy* operations which are *cheap* but may contain *errors*, or *high-energy* operations which are always correct, though more expensive.

We study the high-low energy complexity of a wide class of optimization problems for which greedy algorithms are guaranteed to return optimal or nearly optimal solutions: Using only high-energy operations it is possible to find such solution, but this would require already $\boldsymbol{\Theta(n \log n)}$ **high-energy operations**. On the other hand, with **no high-energy operations**, errors during this sorting phase are likely to produce some *dislocation*, which in turn may cause the greedy algorithm to have an **unbounded approximation ratio**. This is true also for those problems where greedy computes the optimum like, e.g., the minimum spanning tree (as we will further discuss in Section 2). Perhaps it may be surprising if, for some problem, one could devise an algorithm with the following performance:

> *Compute a* ***constant approximation*** *using only* $\boldsymbol{O(\text{polylog } n)}$ ***high-energy*** *operations and* $\boldsymbol{O(n \log n)}$ ***low-energy*** *operations.*

We show that this is indeed the case for the rich class of maximization problems in so called *p-extendible systems*, a generalization of matroids introduced in [24], and recently reconsidered in [8] (see Section 2 for a formal definition). We consider the case of *additive* optimization functions, where the goal is to optimize (maximize or minimize) the sum of the weights in the solution (basis). This class includes, among others, *maximum profit scheduling* ($p = 1$ or 2 depending on the version), *maximum weight b-matching* ($p = 2$), *maximum asymmetric travelling salesman* problem ($p = 3$), *weighted* $\Delta$*-independent set* ($p = \Delta =$ maximum degree) [24, 8]. Interestingly, greedy (without errors) is only a constant factor away from the actual optimum: in any $p$-extendible system greedy is *p-approximate* [24], and it even returns the *optimum* if the input instance is *p-stable* [8], that is, if the $p$-extendible system admits an unique optimal solution which remains unique even when the elements' weights are perturbed by a factor between 1 and $p$. The special case of $p$-extendible systems with $p = 1$ are *matroids* [24], where greedy also computes the optimum (the minimum spanning tree is a classical example).

Our results show a direct connection between the number of *high-energy* operations required to obtain a provably good solution and the ability to *approximately sort* a sequence of $n$ elements at *low energy*. For a given error model, suppose we can compute a sequence of the input elements so that the *maximum dislocation is at most* $d = d(n)$: each element appears at most $d$ positions away from its position in the sorted sequence (depending on the weights). The results of this work are summarized in Table 1 (upper part) where for the moment we do not account for the number of operations used to approximately sort

**Table 1** A summary of the results for a generic input with initial dislocation at most $d$ (**upper part**), and their instantiation for the case of *persistent* comparison errors (**lower part**). In the latter case, an additional $O(n \log n)$ low-energy operations are needed to produce the sequence with dislocation $d = \Theta(\log n)$ w.h.p. [15]. The approximation guarantee of the lower part holds w.h.p.

| Problem | High-Energy | Low-Energy | Approximation |
|---|---|---|---|
| Matroids (min/max) | $O(d \log^2 d)$ | $O(n)$ | 2 (Thm 4) |
| Matroids (min/max) | $O(d^2/\epsilon)$ | $O(n)$ | $1+\epsilon$ (Thm 4) |
| $p$-Ext. Sys. (max, $p \geq 2$) | $O(d + \frac{d^2}{p})$ | $O(n)$ | $p$ times greedy $\leq p^2$ (Thm 15) |
| Matroids (min/max) | $O(\log n \cdot (\log\log n)^2)$ | $O(n \log n)$ | 2 |
| Matroids (min/max) | $O(\frac{\log^2 n}{\epsilon})$ | $O(n \log n)$ | $1+\epsilon$ |
| $p$-Ext. Sys. (max, $p \geq 2$) | $O(\log n + \frac{\log^2 n}{p})$ | $O(n \log n)$ | $p$ times greedy $\leq p^2$ |

the sequence (i.e., assume that such a sequence is provided in input). If such a sequence can be constructed in time $t_{sort}(n)$ using only unreliable (low-energy) comparisons, then the low-energy operations of the whole algorithm (first approximately sort and then run our algorithms) is $O(n + t_{sort}(n))$. For the case of *persistent* comparison errors, [15] showed that it is possible to approximately sort a sequence in $t_{sort}(n) = O(n \log n)$ time so that the maximum dislocation is $d = O(\log n)$ *with high probability* [15].[1] This immediately yields the bounds in Table 1 (lower part) showing that *polylogarithmic* high-energy operations suffice to compute w.h.p. an approximate solution, where the approximation guarantee depends on the problem version as shown in Table 1 (lower part).

Interestingly, since the greedy algorithm has approximation guarantee $p$ for maximization problems restricted to $p$-extendible systems [24], our results for $p \geq 2$ yield a constant approximation for any $p = O(1)$, including the aforementioned NP-hard optimization problems. Moreover, for the special case of *matroids* ($p = 1$) where greedy returns the actual optimum, we show that $O(\frac{\log^2 n}{\epsilon})$ high-energy operations suffice to compute a $(1+\epsilon)$-approximation, and extend this result in two ways: we consider *minimization* problems as well and algorithms which use even *fewer high energy* operations and still achieve an approximation factor of 2.

Regarding problems where greedy is guaranteed to return the optimum solution, we recall that this is also the case on *p-stable instances* of $p$-extendible systems [8]. There we show that our algorithm will in some cases return a solution which is a factor $p$ worse than the optimal (greedy) solution, thus showing that the analysis in Theorem 15 is tight (see Theorem 19).

Due to space limitations, the analysis of our result concerning maximization in matroids is omitted and will appear in the full version of the paper.

## 1.2 Related work

The standard greedy algorithm performs optimally or nearly optimally in several interesting classes of problems.

- Maximization of *submodular* optimization functions under *cardinality constraint*. In this case, greedy has an approximation guarantee of $\frac{e}{e-1}$ [27, 29]. An even better guarantee for greedy holds for *modular* functions and for functions that are "close" to being modular

[1] Technically this requires a comparisons error probability $p_{err} < 1/16$ for which the $O(n \log n)$-time algorithm in [15] applies. Alternatively, if low-energy operations account only for comparisons, the same high-low energy complexity can be achieved for larger $p_{err} < 1/2$ using [6] which uses $O(n \log n)$ comparisons.

[9, 30]. Finally, constant approximations also hold for functions that are close to be submodular [5].

- Maximization in problems with *more complex constraints* has been also considered. For $p$-extendible systems, greedy has an approximation of $p+1$ for submodular functions and $p$ for additive ones (our case). For the additive case, a small constant approximation can be achieved using only *ordinal information*, i.e., without knowing the actual weights and solely based on comparisons [3, 4]. Another line of research deals with *stable* instances of $p$-extendible systems where greedy *recovers the optimal solution* [8].
- Minimization problems are generally harder, with the exception of matroids, including *minimum spanning tree*. Other examples are those problems where the minimum spanning tree itself is a good approximation of the optimum, and thus greedy automatically provides the same guarantee (see for example, the 2-approximation for the metric *travelling salesman problem*, and [17, 2] for connectivity problems in wireless networks). Bicriteria results for supermodular functions with cardinality constraints are given in [23], which considers extending a given solution in a greedy fashion. Finally, also for the metric travelling salesman problem, greedy recovers the *optimum* in the case of *stable* instances [26].

Theoretical models for algorithms that use *two-level* operations are not completely new, though they have been studied with different objectives. In particular, [12] distinguishes between *cheap* and *expensive* comparisons, and errors occur only in the cheap comparisons according to a *threshold error model*: a cheap comparison between two elements whose position in the sorted sequence differ by at most $\tau$ is prone to errors, and all other comparisons are correct. They suggested to use a suitable "concatenation" of two algorithms to *sort perfectly* in this error model. In our terminology, their approach has $\langle O(\tau n), O(n \log n)\rangle$-high-low energy complexity (see also [16] for further results on this setting). Note that the techniques and the results in [12, 16] are not applicable to our setting as their error model is *different* (see next paragraph) and because our primary objective is not to sort. Indeed, our results say that sorting exactly is *not* energy-optimal for many optimization problems (while our algorithms use $O(\log^2 n)$ high-energy operations, sorting exactly w.h.p. requires $\Omega(n)$ high-energy complexity).

The error model with *persistent* errors is different from the threshold error model [12, 16], and somewhat more difficult. In the model with persistent errors, there is no bound on $\tau$, and thus every comparison result is wrong with some probability $p_{err} > 0$, independently of the other results. In this case, the best bound on the maximum dislocation which is possible to guarantee (with high probability) is $d = \Theta(\log n)$ and, among the various known algorithms that achieve this performance [6, 18, 13, 14, 15], the fastest has running time $\Theta(n \log n)$ [15].

## 2 Preliminaries

### Independence systems and matroids

An *independence system* is a pair $M = (E, \mathcal{I})$ where $E$ is a collection of $n$ elements called *ground set*; $\mathcal{I} \subseteq \mathcal{P}(E)$ is a family of *independent sets*[2] which is *downward closed* (also known as the *hereditary property*): if $B \in \mathcal{I}$ and $A \subseteq B$, then $A \in \mathcal{I}$; and $\emptyset \in \mathcal{I}$. A maximal (w.r.t. inclusion) independent set is called a *base* or *feasible solution*.

[2] Where $\mathcal{P}(E)$ denotes the power-set of $E$, i.e., the set of all subsets of $E$.

We consider *maximization* and *minimization* problems involving independence systems where each element $e \in E$ has a non-negative weight $w(e) \geq 0$. In maximization problems (resp., minimization problems) the goal is to compute a base $B$ maximizing (resp., minimizing) the total weight $w(B) = \sum_{b \in B} w(b)$, the so called additive case. We shall restrict our attention to the following two important classes of independence systems, for which the simple greedy algorithm (see below) returns either a constant approximation or even the optimum.

**Matroids:** A *matroid* is an independence system $(E, \mathcal{I})$ that satisfies the *augmentation property*: If $A, B \in \mathcal{I}$ and $|A| < |B|$, then $\exists x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

**$p$-extendible systems** : A *$p$-extendible system* is an independence system $(E, \mathcal{I})$ such that if $A \subseteq B \in \mathcal{I}$ and $A \cup \{e\} \in \mathcal{I}$, then $(B \setminus R) \cup \{e\} \in \mathcal{I}$ for some $R \subseteq B \setminus A$ of cardinality $|R| \leq p$.

It follows from the above definitions that the cardinalities of any two bases of a $p$-extendible system are at most a factor $p$ apart. Since any matroid $M$ is a 1-extendible system [25], it follows that all bases of $M$ have the same cardinality, denoted by $\text{rank}(M)$. Finally, a *circuit* $C$ of an independence system is a minimal (w.r.t. inclusion) non-independent set, i.e., $C \in \mathcal{P}(E) \setminus \mathcal{I}$.

## Greedy algorithm

A well-known algorithm is the greedy algorithm. Starting from the empty set, it iteratively extends the current solution $A \in \mathcal{I}$ with the "best" element $x \in E \setminus A$ such that $A \cup \{x\}$ is still independent; For maximization problems, greedy considers the elements from the largest weight to the smallest one, while in minimization it follows the opposite order. The algorithm stops and returns the current set $A$ when no such $x$ element can be added. In the following we denote by $\text{greedy}(M)$ the set returned by (an unspecified implementation of) the greedy algorithm on $M$, and by $\text{greedy}(M, \widetilde{S})$ the set returned by the variant of the greedy algorithm that considers the elements of $M$ as they appear in a (non necessarily sorted) sequence $\widetilde{S}$.

## Our setting (high and low energy operations)

The greedy algorithm needs access to the ground set of $M$, to the function $w$, and to an *independence oracle* $\mathcal{O}$ that reports whether a subset of elements of $E$ is independent. Notice, however, that only *ordinal* information about $w$ are needed, i.e., we can replace $w$ with a *comparison oracle* $C_H : E \times E \to \{\leq, >\}$: a query $C_H(x, y)$ to $C_H$ reports whether $x$ is smaller than or equal to $y$.

In this paper we consider the scenario in which algorithms have access to an additional comparison oracle $C_L$ that can sometimes return *incorrect answers*. More precisely, there is a (small) constant probability $p_{err} < 1/16$ that $C_L$ returns the wrong answer to a query. Errors between comparisons that involve different pairs of elements are *independent*. However the answers of $C_L$ are *persistent*, i.e., they do not change if the pair of elements is queried multiple times. Notice that the algorithm is still allowed to query $C_H$ in order to determine with certainty the correct order relation between two elements (even if a query to $C_L$ involving the same pair of elements has already been performed).

▶ **Definition 1.** *We say that an algorithm has* high-low energy complexity $\langle h(n), \ell(n) \rangle$ *if it performs at most $h(n)$ queries to $C_H$, and at most $\ell(n)$ other operations (including queries to $C_L$).*

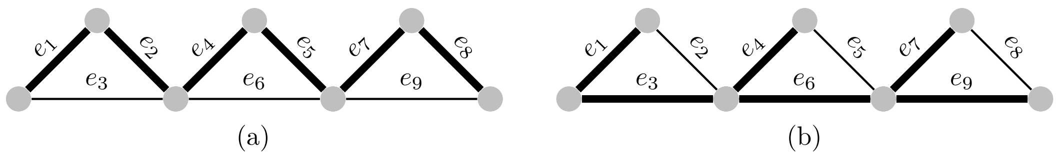


**Figure 1** The graph $G$ along with (a) its minimum spanning tree, and (b) the spanning tree returned by greedy$(M, \widetilde{S})$.

▶ **Example 2.** The greedy algorithm has high-low energy complexity $\langle O(n \log n), O(n \log n)\rangle$ as it performs $O(n \log n)$ queries to $C_H$ in order to sort the elements of $E$ according to $w$.[3]

### Greedy on nonsorted sequences

As already mentioned, the standard greedy algorithm first orders the elements in $E$ according to their weights, i.e., it constructs a sorted sequence

$$S = (e_1, e_2, \dots, e_n) \qquad \text{where} \qquad w(e_1) \geq w(e_2) \geq \cdots \geq w(e_n) \tag{1}$$

for *maximization* problems (the opposite order is considered for *minimization problems*). The greedy algorithm then considers the elements of $S$ in order and iteratively maintains an independent set. Because sorting exactly the elements requires $\Omega(n)$ high-energy operations (see Example 2 above), we consider running greedy with respect to a different *almost-sorted* sequence $\widetilde{S} = (\tilde{e}_1, \tilde{e}_2, \dots, \tilde{e}_n)$. The sequence $\widetilde{S}$ is a permutation of the elements which has some bound $d$ on the *dislocation*: we say that $\widetilde{S}$ has dislocation at most $d$ if there exists a sorted sequence $S$ as in (1) such that[4] $|t(e, S) - t(e, \widetilde{S})| \leq d$, where $t(e, S)$ and $t(e, \widetilde{S})$ denote the positions of $e$ in $S$ and in $\widetilde{S}$, respectively.

One might wonder whether running greedy on the almost-sorted sequence $\tilde{S}$ already results in good approximation guarantees. Unfortunately, this turns out not to be the case: the solution computed by the greedy algorithm on input a sequence with *very small dislocation* can be *arbitrarily far from the optimum*, as the following example for the minimum spanning tree problem shows.

▶ **Example 3** (minimum spanning tree). Let $\epsilon \in (0, \frac{1}{8})$ and consider the graph $G$ shown in Figure 1 (a) along with its corresponding *graphic matroid* $M$.[5] The edges of $G$ are weighted as follows: $w(e_i) = i\epsilon$ for $i = 1, \dots, 8$ and $w(e_9) = 1$, so that the sorted version of the ground set of $M$ w.r.t. $w$ is $S = \langle e_1, e_2, \dots, e_9\rangle$. The minimum-weight base of $M$ has weight $27\epsilon$ and consists of the edges in the (unique) minimum spanning tree of $G$ (in bold). For a sequence $\widetilde{S}$ of dislocation 1, in which we swap the order between $e_2$ and $e_3$, $e_5$ and $e_6$, and $e_8$ and $e_9$, i.e., $\widetilde{S} = \langle e_1, e_3, e_2, e_4, e_6, e_5, e_7, e_9, e_8\rangle$, greedy$(M, \widetilde{S})$ would select the suboptimal tree shown in Figure 1 (b). The cost of the resulting tree can be significantly higher than cost of a MST (i.e., more than $1 = w_9$ as opposed to $27\epsilon$) and, for tiny $\epsilon$, there is no approximation guarantee.

Even if greedy$(M, S)$ returns the optimal solution for a minimization/maximization problems in matroids, the above example above shows that greedy$(M, \widetilde{S})$ cannot approximate

[3] Here, the bound on the low-energy complexity accounts for up to $O(n \log n)$ non-comparison operations.
[4] For distinct elements (weights) the sorted sequence is unique. In general, we consider $S$ and $\widetilde{S}$ to agree on the relative order between elements with identical weight.
[5] The ground-set of $M$ is the set of edges of $G$, while a subset of edges is independent in $M$ if it induces a forest in $G$.

the greedy solution within any factor (to apply the previous example to the problem of computing a maximum-weight base of $M$ it suffices to exchange the roles of $S$ and $\widetilde{S}$).

## 3 The general scheme for matroids

In this section we describe a general scheme which leads to different approximation algorithms for computing a minimum- or maximum-weight base in a matroid. In particular, we will then be able to prove the following theorem.

▶ **Theorem 4.** *Consider any maximization/minimization problem in a matroid, where the input elements are given as a sequence with dislocation at most $d$. There exists an algorithm which returns a $2$-approximate solution and that has $\langle O(d\log^2 d), O(n)\rangle$-high-low energy complexity. Moreover, for every $\epsilon \in (0,1)$, there exists an algorithm which returns a $(1+\epsilon)$-approximate solution and that has $\langle O(d^2/\epsilon), O(n)\rangle$-high-low energy complexity.*

Our algorithm and analysis are based on standard notions of "submatroid". If $M = (E, \mathcal{I})$ is a matroid and $X \subseteq E$, the *restriction* $M|X$ of $M$ to $X$ is the matroid having $X$ as its ground set and $\{Y \in \mathcal{I} : Y \subseteq X\}$ as its independent sets. If $X \in \mathcal{I}$, the *contraction* $M/X$ of $M$ by $X$ is the matroid having $E \setminus X$ as its ground set and $\{Y \in \mathcal{P}(E \setminus X) : Y \cup X \in \mathcal{I}\}$ as its independent sets. A *minor* of $M$ is matroid that can be obtained from $M$ by a sequence of restrictions and contractions. We denote by $\text{Opt}(M) = w(\text{greedy}(M))$ the weight of a base of $M$ having minimum (resp. maximum) weight in the case of minimization (resp. maximization) problems.

### The algorithm

The inputs of our algorithm are a matroid $M$, given in the form of a set $E$ of $n$ elements and an independence oracle, and an approximately-sorted sequence $\widetilde{S}$ of the elements in $E$ having dislocation at most $d$.[6] If such a sequence is not readily available, one with $d = O(\log n)$ can be computed in a pre-processing step using $O(n \log n)$ low-energy operations [15].
Our algorithm, whose pseudocode is shown in Algorithm 1, performs the following steps:

1. First, we run the greedy algorithm by considering the elements as they appear in $\widetilde{S}$. Let $A = \{a_1, a_2, \dots, a_k\} = \text{greedy}(M, \widetilde{S})$ be the resulting (now possibly suboptimal) base, where $a_i$ represents the $i$-th element added during the execution of the algorithm and $k = \text{rank}(M)$.
2. Next, select a suitable subset $F$ of elements of $A$ that will be part of our final solution. The exact details of this step will be specified later.
3. Let $E'$ be the set of all elements $x \in E \setminus F$ such that $|t(x, \widetilde{S}) - t(y, \widetilde{S})| \leq 2d$ for some $y \in A \setminus F$. We define $M'$ as the matroid having $E'$ as its ground set, and all the sets $X \in \mathcal{P}(E')$ such that $X \cup F$ is independent in $M$ as its independent sets. Notice that $M'$ is a minor of $M$ as it can be obtained by first contracting $M$ by $F$, and then restricting the resulting matroid $M/F$ to $E'$, i.e., $M' = (M/F)|E'$.
4. Finally, we compute a minimum-weight base $A'$ of $M'$ using the greedy algorithm and high-energy queries to $C_H$. We return $F \cup A'$.

When minimization (resp. maximization) problems are concerned, the high level intuition is that the initial greedy solution $A$, which can be far from optimal, contains a "large" subset

[6] Elements are (approximately) sorted in non-decreasing or non-increasing order of weights depending on whether we are interested in a minimum-weight or maximum-weight base of $M$, respectively.

**Algorithm 1** Dual-Mode Greedy Scheme($M$, $\widetilde{S}$, $d$).

$A \leftarrow \text{greedy}(M, \widetilde{S})$;
$F \leftarrow$ Select a suitable subset of $A$;
$E' \leftarrow \{x \in E \setminus F : \exists y \in A \setminus F, |t(x, \widetilde{S}) - t(y, \widetilde{S})| \leq 2d\}$;
$A' \leftarrow \text{greedy}((M/F)|E')$; `// high energy part`
**return** $F \cup A'$;

$F$ of elements whose weight is comparable to the optimum. We fix these elements and isolate a "small" set $E'$ of candidates to complete the solution. As this set is "small" we can run greedy at high energy, and hope that it will only contribute a small (resp. large enough) additional weight to the final solution, which is the union of the solutions of the two parts.

## Some properties

For the sake of the analysis, we define $S$ to be the (correctly) sorted sequence containing the elements in $E$. Whenever ties between two elements arise they are broken by preserving their relative order in $\widetilde{S}$. Let $B = \{b_1, b_2, \ldots, b_k\} = \text{greedy}(M, S)$ be an optimal base of $M$ as computed by the greedy algorithm that considers the elements in the same order as $S$.

▶ **Lemma 5.** *For all $i = 1, \ldots, k$ we have $|t(b_i, S) - t(a_i, \widetilde{S})| \leq d$.*

**Proof.** Let $S_t$ (resp. $\widetilde{S}_t$) be the sequence consisting of the first $t$ elements of $S$ (resp. $\widetilde{S}$). Similarly, let $A_t = \text{greedy}(M, \widetilde{S}_t)$ (resp. $B_t = \text{greedy}(M, S_t)$) be the set of elements included in the independent set maintained by $\text{greedy}(M, \widetilde{S})$ (resp. $\text{greedy}(M, S)$) at *time* $t$, i.e., immediately after the $t$-th element of $\widetilde{S}$ (resp. $S$) is considered.

Notice that, for every $t = 0, \ldots, n$, each element in $S_t$ must also be contained in $\widetilde{S}_{\min\{t+d,n\}}$ due to the bound on the dislocation of $\widetilde{S}$. This implies that $|B_t| \leq |A_{\min\{t+d,n\}}|$. By choosing $t = t(b_i, S)$ we obtain $i = |B_{t(b_i,S)}| \leq |A_{\min\{t(b_i,S)+d,n\}}|$ and therefore the $i$-th element $a_i$ of $A$ must have been added at a time (i.e., position in $\widetilde{S}$) of at most $t(b_i, S) + d$, i.e., $t(a_i, \widetilde{S}) \leq t(b_i, S) + d$, or equivalently $t(b_i, S) \geq t(a_i, \widetilde{S}) - d$.

Similarly, for $t = 0, \ldots, n$, $S_t$ is a superset of $\widetilde{S}_{\max\{0,t-d\}}$, implying $|B_t| \geq |A_{\max\{0,t-d\}}|$. By choosing $t = t(b_i, S)$ we obtain $i = |B_{t(b_i,S)}| \geq |A_{\max\{0,t(b_i,S)-d\}}|$, implying that the $i$-th element $a_i$ in $A$ has been added at a time of at least $t(b_i, S) - d$, i.e., $t(a_i, \widetilde{S}) \geq t(b_i, S) - d$, or equivalently $t(b_i, S) \leq t(a_i, \widetilde{S}) + d$. ◀

The above lemma, together with the bound on the dislocation of $\widetilde{S}$, immediately implies:

▶ **Corollary 6.** *$|t(b_i, S) - t(a_i, S)| \leq 2d$ and $|t(b_i, \widetilde{S}) - t(a_i, \widetilde{S})| \leq 2d$.*

Next, we show that, regardless of the choice of $F$, Algorithm 1 returns a base of $M$.

▶ **Lemma 7.** *The set $A' \cup F$ returned by Algorithm 1 is a base of $M$.*

**Proof.** We start by defining $M''$ as the matroid having $E$ as its ground set and such that $X \in \mathcal{P}(E \setminus F)$ is an independent set of $M''$ iff $X \cup F$ is independent in $M$. Notice how $M''$ is closely related to the *contraction* of $M$ by $F$ (which is an independent set of $M$ since $F \subseteq A$): the only difference is that the ground set of $M''$ still contains the elements in $F$, even though they do not belong to any independent set.

Since $F$ and $B$ are, respectively, an independent set and a base of $M$, we can iteratively invoke the augmentation property of matroids to select a set $B'' \subset B \setminus F$ of $k - |F|$ elements

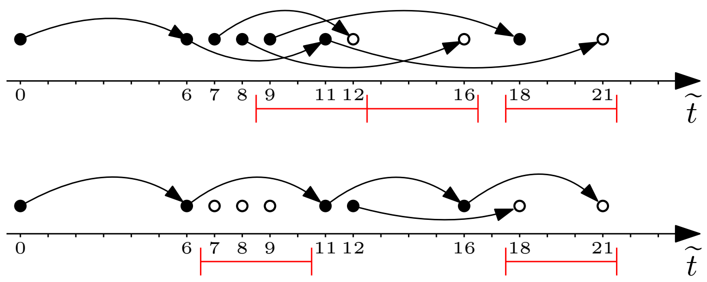


**Figure 2** Top: an example of a $\langle \tau, \lambda \rangle$-min-mapping with $\tau = 4$ and $\lambda = 3$. The elements in the set $X$ are depicted as dots and the horizontal axis represents the function $\widetilde{t}$. Filled (resp. hollow) dots correspond to mapped (resp. unmapped) elements. Three intervals of size $\tau$ whose union contains the values $\widetilde{t}(x)$ of all unmapped elements $x$ are shown in red. The mapping has been obtained by greedily assigning each element of $X$ to the first suitable element, in increasing order of $\widetilde{t}(\cdot)$. Bottom: A $\langle 4, 2 \rangle$-min-mapping for the same set of elements $X$ and values of $\widetilde{t}(\cdot)$.

such that $F \cup B''$ is an independent set of $M$. This implies that $B''$ is also an independent set in $M''$. In particular, since all the independent sets $X$ of $M''$ are such that $X \cap F = \emptyset$ and $X \cup F$ is independent in $M$, it follows that $|X| \leq k - |F| = |B''|$ therefore that $B''$ is maximal w.r.t. inclusion in $M''$ and hence it is a base.

Notice that $M'$ is exactly the *restriction* of $M''$ to the set $E'$, and that the set greedy$(M'', \widetilde{S})$ coincides with $A \setminus F$. By Corollary 6, the position in $\widetilde{S}$ of each element in $B' = \text{greedy}(M'', S)$ differs by at most $2d$ from the position of a suitable element in $A \setminus F$, implying that $B' \subseteq E'$. To summarize, we have: (i) $B' = \text{greedy}(M'', S) = \text{greedy}(M', S) = A'$, (ii) $B' \cup F$ is independent in $M$, (iii) $|B' \cup F| = |B'| + |F| = \text{rank}(M') + |F| = |B''| + |F| = (k - |F|) + |F| = k$. ◀

## 4 Minimization in Matroids

In this section, we instantiate our general scheme in order to prove Theorem 4 in the case of *minimization in matroids* (the maximization counterpart will appear in the full version of the paper). Recall that $\widetilde{S}$ has dislocation $d$ w.r.t. a sequence $S$ in which elements are sorted in *non-decreasing* order of weight.

We start by proving the following lemma, which will hold for all our choices of $F$.

▶ **Lemma 8.** $w(A') \leq \sum_{i=|F|+1}^{k} w(b_i) \leq \text{Opt}(M)$.

**Proof.** Let $M''$, and $B''$ be defined as in Lemma 7. We have that $|A' \cup F| = k$ and hence $|A'| = k - |F|$ as $A \cap F = \emptyset$. Since $A' = \text{greedy}(M', S)$ is a minimum-weight base of $M'$ and $M''$ while $B'' \subseteq B$ is a base of $M''$, we have $w(A') = \text{Opt}(M'') \leq w(B'')$.

By definition, $B''$ is a subset of $B$ and, since $|B''| = \text{rank}(M'') = |A'| = k - |F|$, we can upper bound the total weight of the $k - |F|$ elements in $B''$ with that of the $k - |F|$ elements of $B$ of largest weight, i.e., $w(B'') \leq \sum_{i=|F|+1}^{k} w(b_i)$. Combining all the previous inequalities, we can write: $w(A') \leq w(B'') \leq \sum_{i=|F|+1}^{k} w(b_i) \leq w(B) = \text{Opt}(M)$. ◀

### 4.1 A 2-Approximation (for Minimization in Matroids)

To instantiate the general scheme of Algorithm 1, we need to specify how the subset $F$ of elements of Step 2 is selected. To this aim we *map* some of the elements of the initially computed solution $A$ into some other elements of $A$. The set of mapped elements will be our set $F$, while the mapping shall satisfy the following definition.

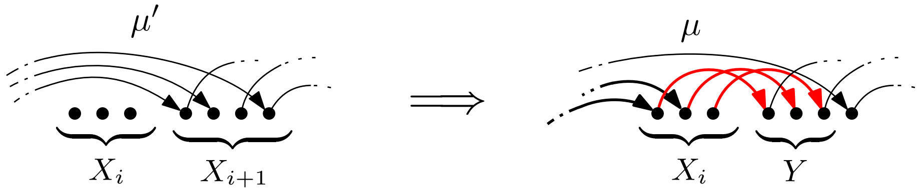


**Figure 3** A graphical representation of the local transformation that allows us to assume that $|X_i| > |X_{i+1}|$. Function $\mu'$ (whose domain does not include the elements in $X_i$) is shown on the left, while function $\mu$ is shown on the right. Recall that $Y \subseteq X_{i+1}$ is chosen arbitrarily such that $|Y| = |X_i|$. The differences between $\mu'$ and $\mu$ are highlighted in bold. The newly-added mappings $\mu(x)$ for $x \in X_i$ are shown in red.

▶ **Definition 9** (*$\langle\tau,\lambda\rangle$-min-mapping*). *A* $\langle\tau,\lambda\rangle$-min-mapping *of a set of elements $X$ w.r.t. an injective function $\widetilde{t} : X \to \mathbb{N}$ is an injective partial function $\mu : X \to X$ such that:*

- *For every element $x$ in the domain $\mathcal{D}(\mu)$ of $\mu$ it holds $\widetilde{t}(\mu(x)) \geq \widetilde{t}(x) + \tau$;*
- *The integers in $\{\widetilde{t}(x) : x \in X \setminus \mathcal{D}(\mu)\}$ are all contained in the union of at most $\lambda$ intervals of contiguous integers, each of size at most $\tau$.*

In other words, if we think of $\widetilde{t}()$ as a function that associates a *time* to each element of $X$, the above definition guarantees that an element $x$ of $X$ is either mapped to some other element $y \in X$ that appears sufficiently later in time, or it belongs to a small set of at most $\lambda$ *time intervals*, each of size $\tau$. Figure 2 shows an example of a $\langle 4, 3\rangle$- and a $\langle 4, 2\rangle$-min-mapping.

## Finding a $\langle\tau, 4\log\tau + 4\rangle$-min-mapping

We now show how a $\langle\tau,\lambda\rangle$-min-mapping with $\lambda = O(\log\tau)$ can be found, which will turn out to be the best asymptotic trade-off between $\tau$ and $\lambda$ one can hope to obtain. In particular, we will set $\lambda = 4\log\tau + 4$.

To this aim, it is useful to consider a relaxed variant of the problem. Intuitively, we get rid of $\widetilde{t}()$ by grouping the elements of $X$ into a collection of sets $\langle X_1, X_2, \ldots, X_m\rangle$. Instead of requiring $\widetilde{t}(\mu(x)) \geq \widetilde{t}(x) + \tau$, it will be enough for $x \in X_i$ to be mapped to some element $\mu(x) \in X_j$ with $j > i$. Moreover, we allow up to $2\log\tau + 2$ sets $X_i$ to contain unmapped elements.

Formally, we are given a sequence of $m$ pairwise-disjoint sets $\langle X_1, X_2, \ldots, X_m\rangle$ each containing at most $\tau$ elements, as we want to find:

- A subset $N$ of $\{X_1, X_2, \ldots, X_m\}$ of size at most $2\log\tau + 2$.
- An injective function $\mu : \bigcup_{X_i \notin N} X_i \to \bigcup_i X_i$ such that $x \in X_i \implies \mu(x) \in X_j$ for a $j > i$.

W.l.o.g. we can restrict ourselves to the case in which the cardinalities of the sets $X_i$ are monotonically decreasing and $|X_m| > 0$. Indeed, if $i$ is an index such that $|X_i| \leq |X_{i+1}|$, we can consider the modified instance consisting of the sets $\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m\}$ instead. Any solution $(N', \mu')$ to this modified instance yields a solution $(N, \mu)$ for the original instance. Indeed, if $X_i = \{x_1, x_2, \ldots\}$, it suffices to pick $N = N'$, select an arbitrary subset $Y = \{y_1, y_2, \ldots\}$ of $|X_i|$ elements from $X_{i+1}$, and define $\mu$ as follows (see also Figure 3):

$$\mu(x) = \begin{cases} y_j & \text{if } x \in X_i \text{, where } j \text{ is such that } x = x_j, \\ x_j & \text{if } x \notin X_i \text{ and } \mu'(x) \in Y \text{, where } j \text{ is such that } \mu'(x) = y_j, \\ \mu'(x) & \text{otherwise.} \end{cases}$$

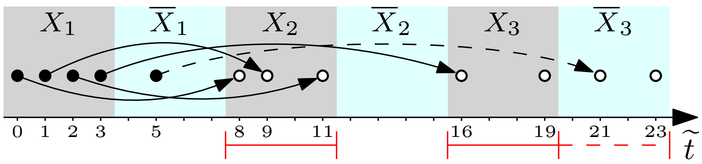


**Figure 4** Decomposition of a set $X$ into the families of sets $X_1, X_2, \dots$ and $\overline{X}_1, \overline{X}_2, \dots$ used in the relaxed version of our mapping problem for $\tau = 4$. The elements in the set $X$ are depicted as dots and the horizontal axis represents the function $\widetilde{t}$. Filled (resp. hollow) dots correspond to elements that belong to (resp. do not belong to) $\mathcal{D}(\mu)$. The solution $(N_1, \mu_1)$ is shown with solid lines, while the solution $(N_2, \mu_2)$ is shown with dashed lines. Their combination yields the $\langle \tau, \lambda \rangle$-min-mapping $\mu$ with $\lambda = 4 \log \tau + 4$. The (at most $\lambda$) red intervals span all the elements in $N_1 \cup N_2 = X \setminus \mathcal{D}(\mu)$.

We henceforth assume that $|X_i| > |X_{i+1}|$ for all $i = 1, \dots, m-1$. Our algorithm starts by letting $i = 1$ and $N = \emptyset$, then it iteratively looks for the largest index $j \geq i$ such that $|X_j| \geq |X_i|/2$ and performs one of the following two steps, depending on the value of $j$:

- If $j = i$, then $X_i$ is added to $N$.
- If $j > i$, we assign consecutive indices $1, 2, \dots,$ to the elements in $X_i, \dots, X_j$ (in order). Let $x_h$ be the element with index $h$. We define $\mu(x_h) = x_{h+|X_i|}$ for all $h = 1, \dots, \sum_{\ell=1}^{j-2} |X_\ell|$. Notice that, for every $x_h \in \bigcup_{\ell=i}^{j-2} X_\ell$, we have that $\mu(x_h) \in \bigcup_{\ell=i+1}^{j} X_\ell$, that $\mu(x_h)$ is necessarily in a successive set, and that the mapping is injective. Finally, we add $X_j$ and $X_{j-1}$ to $N$.

If $j < m$ we set $i = j + 1$ and continue with the next iteration, otherwise we return the pair $(N, \mu)$.

Observe that, in every iteration, the cardinality of $N$ increases by at most 2. Moreover, if $j < m$, the set $X_i$ has cardinality least twice the one of the set $X_{j+1}$, i.e., the set considered at the next iteration. This means that there will be at most $\log |X_1| + 1$ iterations, thus $|N| \leq 2 \log |X_1| + 2 \leq 2 \log \tau + 2$.

We now argue that a $\langle \tau, 4 \log \tau + 4 \rangle$-min-mapping of a set $X$ w.r.t. $\widetilde{t} : X \to \mathbb{N}$ can be found by solving *two instances* of the relaxed problem. Namely, for $i = 1, 2, \dots,$ we define $X_i = \{x \in X : 2(i-1)\tau \leq \widetilde{t}(x) < (2i-1)\tau\}$ and $\overline{X}_i = \{x \in X : (2i-1)\tau \leq \widetilde{t}(x) < 2i\tau\}$. Let $(N_1, \mu_1)$ and $(N_2, \mu_2)$ be two solutions to the instances of the above problem consisting of the sets $X_i$ and $\overline{X}_i$, respectively. Then, the mapping $\mu$ defined as $\mu(x) = \mu_1(x)$ if $x \in \mathcal{D}(\mu_1)$ and $\mu(x) = \mu_2(x)$ if $x \in \mathcal{D}(\mu_2)$ is a $\langle \tau, 4 \log \tau + 4 \rangle$-min-mapping. Notice indeed that each element in $X \setminus \mathcal{D}(\mu)$ is contained in one of the at most $4 \log \tau + 4$ sets in $N_1 \cup N_2$, and – by our definition of $X_i$ and $\overline{X}_i$ – all the elements in a set $Y \in N_1 \cup N_2$ are such that all $\widetilde{t}(x)$ for $x \in Y$ belong to a single interval of size $\tau$. Moreover, since each element in $x \in X_i \notin N_1$ is mapped to an element $\mu(x)$ in some $X_j$ with $j \geq i + 1$, we know that $\widetilde{t}(x) < (2i-1)\tau$ while $\widetilde{t}(\mu(x)) \geq 2i\tau$, i.e., $\widetilde{t}(\mu(x)) - \widetilde{t}(x) > \tau$, as desired (a symmetrical arguments holds for $x \in \overline{X}_i \notin N_2$). See Figure 4 for an example.

It is not hard to see that such a mapping can be computed in $O(|X|)$ time if a sorted version of $X$ w.r.t. $\widetilde{t}(\cdot)$ is known, as it will be the case in the sequel.

### There is no $\langle \tau, o(\log \tau) \rangle$-min-mapping

We point out that the above construction of a $\langle \tau, \lambda \rangle$-min-mapping essentially achieves the best attainable trade-off between $\tau$ and $\lambda$,

▶ **Lemma 10.** *In general, there exists no $\langle \tau, o(\log \tau) \rangle$-min-mapping.*

**Proof.** Let $h = \lfloor \log \tau \rfloor$ and consider a set $X$ of $2^{h+1} - 1$ elements which is partitioned into $h+1$ sets $X_0, \ldots, X_h$ where $X_i = \{x_1^{(i)}, x_2^{(i)}, \ldots\}$ contains $2^{h-i}$ elements. We define $\widetilde{t}(x_j^{(i)}) = 2^i + j + i\tau - 1$.

Let $\mu$ be any $\langle \tau, h \rangle$-min-mapping of $X$ w.r.t. $\widetilde{t}$. We say that a set $X_i$ is *covered* by $\mu$ if $X_i \subseteq \mathcal{D}(\mu)$. We claim that, in $\mu$, no set can be covered. Indeed, assume towards a contradiction that at least one set $X_i$ is covered. Since the value of $\widetilde{t}$ for any two elements in $X_i$ differs by at most $|X_i| - 1 \le 2^{h-i} - 1 \le 2^h - 1 \le \tau - 1$, we have that, for every $x_j^{(i)} \in X_i$, $\mu(x_j^{(i)}) \in X_\ell$ for some $\ell > i$. However, $|\bigcup_{\ell=i+1}^{h} X_\ell| = \sum_{\ell=0}^{h-i-1} 2^\ell = 2^{h-i} - 1$ and, since $|X_i| = 2^{h-i}$ this contradicts the fact that $\mu$ is an injective function.

To conclude the proof it suffices to notice that $h + 1 = \lfloor \log \tau \rfloor + 1 > \log \tau$ intervals of length at most $\tau$ are necessary for their union to include all the integers in $X \setminus \mathcal{D}(\mu)$. ◀

### Analysis for the 2-Approximation

In order to obtain a 2-approximate minimum-weight base of $M$, we compute a $\langle 2d, 4\log d + 8 \rangle$-min-mapping $\mu$ of $A$ w.r.t. $\widetilde{t}(x) = t(x, \widetilde{S})$ and we choose $F$ as the domain $\mathcal{D}(\mu)$ of $\mu$. Intuitively, for $\tau = 2d$, the first condition of Definition 9 gives an *implicit* partial injective mapping into elements of $B$ of non smaller weight, thus yielding $w(F) \le \mathrm{Opt}(M)$. The second condition can be used to show that the set $F$ can be extended in an optimal way by looking at a "small" subset of elements (Steps 3 and 4 of Algorithm 1), and thus the number of high-energy operations is not too large. As this part of the solution also contributes at most another factor $\mathrm{Opt}(M)$, we get a 2-approximation. We formalize this intuition in the following lemmas:

▶ **Lemma 11.** *The set returned by Algorithm 1 with $F = \mathcal{D}(\mu)$ has weight at most $2\,\mathrm{Opt}(M)$.*

**Proof.** We associate each $a_i \in A \cap \mathcal{D}(\mu) = \mathcal{D}(\mu)$ to an element $b_j \in B$, where $j$ is the index such that $a_j = \mu(a_i)$. By definition of $\langle 2d, 4\log d + 8 \rangle$-min-mapping we know that $\widetilde{t}(a_j) \ge \widetilde{t}(a_i) + 2d$ and, by Lemma 5, $\widetilde{t}(a_j) = t(a_j, \widetilde{S}) \le t(b_j, S) + d$. Combining the previous inequalities we have $t(b_j, S) \ge \widetilde{t}(a_j) - d \ge \widetilde{t}(a_i) + d = t(a_i, \widetilde{S}) + d \ge t(a_i, S)$, thus implying that $w(b_j) \ge w(a_i)$, as $b_j$ appears no earlier than $a_j$ in $S$ (which is sorted in nondecreasing order of weights). Moreover, since $\mu$ is injective, our mapping between $\mathcal{D}(\mu)$ and $B$ is also injective. Therefore: $w(\mathcal{D}(\mu)) \le w(B) = \mathrm{Opt}(M)$. We can now use the above inequality together with Lemma 8 to conclude that $w(F \cup A') = w(\mathcal{D}(\mu)) + w(A') \le 2\,\mathrm{Opt}(M)$. ◀

▶ **Lemma 12.** *The high-low energy complexity of Algorithm 1 when $F = \mathcal{D}(\mu)$ and the elements of $E$ are given in a almost-sorted sequence $\widetilde{S}$ having dislocation at most $d$ is $\langle O(d \cdot \log^2 d), O(n) \rangle$.*

**Proof.** Once the base $A$ of $M$ is found using $O(n)$ low-energy operations, both the function $\mu$ and the set $F = \mathcal{D}(\mu)$ can be computed in linear time w.r.t. $|A|$ without requiring access to the oracle $C_H$, as we discussed in Section 4.1. This, in turn, allows to construct $E'$ using $O(n)$ additional low-energy operations.

Finally, finding the set $A' = \mathrm{greedy}(M')$ requires $O(d \cdot \log^2 d)$ high- and low-energy operations. Indeed, since $\mu$ is a $\langle 2d, 4\log d + 8 \rangle$-min-mapping, the set $E'$ will contain at most $6d+1$ elements for each of the $4\log d + 8$ unmapped intervals, i.e., $|E'| \le (4\log d + 8) \cdot (2d+1) =$

$O(d \log d)$ implying that $E'$ can be sorted using $O(d \log d \cdot \log(d \log d)) = O(d \cdot \log^2 d)$ *high-energy* queries to $C_H$. ◀

## 4.2 A $(1+\epsilon)$-Approximation (for Minimization in Matroids)

In order to improve the approximation guarantee from 2 to $1+\epsilon$, for any $\epsilon \in (0,1)$, we shall define the set $F$ in Step 2 of Algorithm 1 as the first $k - cd$ elements of the initially computed solution $A$, for $c = \lceil 2/\epsilon \rceil$. The approximation guarantee and the high-low energy complexity are given by the next two lemmas, respectively. Recall that $a_i$ (resp. $b_i$) is the $i$-th element added to the independent set maintained by greedy$(M, \widetilde{S})$ (resp. greedy$(M, S)$).

▶ **Lemma 13.** *The set returned by Algorithm 1 with $F = \{a_1, \ldots, a_{k-cd}\}$ has weight at most $(1+\epsilon)\operatorname{Opt}(M)$.*

**Proof.** By Corollary 6, $t(a_i, S) \le t(b_i, S) + 2d \le t(b_{i+2d}, S)$, which implies $w(a_i) \le w(b_{i+2d})$ (notice that $c \ge 2$, therefore $i + 2d \le k - (c-2)d \le k$ and $b_{i+2d}$ always exists). We thus get the following bound: $w(F) = \sum_{i=1}^{k-cd} w(a_i) \le \sum_{i=1+2d}^{k-(c-2)d} w(b_i)$.

From Lemma 8 we also have $w(A') \le \sum_{i=k-cd+1}^{k} w(b_i)$ and, combining the above inequalities, we obtain:

$$\begin{aligned} w(F \cup A') &\le \sum_{i=1+2d}^{k-(c-2)d} w(b_i) + \sum_{i=k-cd+1}^{k} w(b_i) = \sum_{i=1+2d}^{k} w(b_i) + \sum_{i=k-cd+1}^{k-(c-2)d} w(b_i) \\ &\le \operatorname{Opt}(M) + \frac{2}{c} \sum_{i=k-cd+1}^{k} w(b_i) \le (1+\epsilon)\operatorname{Opt}(M). \end{aligned}$$ ◀

▶ **Lemma 14.** *For any $\epsilon > 0$, the high-low energy complexity of Algorithm 1 when $F = \{a_1, \ldots, a_{k-cd}\}$ and the elements of $E$ are given in a almost-sorted sequence $\widetilde{S}$ having dislocation at most $d$ is $\langle O(\epsilon^{-1} d^2), O(n) \rangle$.*

**Proof.** Similarly to the proof of Lemma 12, $A$ can be computed using at most $O(n)$ low-energy operations. Notice that $F$ can trivially be found in linear time in $|A|$ and that the set $E'$ contains at most $|A \setminus F| \cdot d = O(\frac{1}{\epsilon} d^2)$ elements.

We now simulate the greedy algorithm in order to compute an optimal base $A'$ of $M'$ as follows: We start with $A' = \emptyset$ and we consider the elements $x$ in $A \setminus F$ in increasing order of $t(x, \widetilde{S})$ until we find an element $x'$ such that $A' \cup \{x'\}$ is independent. We then perform a linear search for the minimum-weight element $x^*$ among the ones in $\{y \in E : A' \cup \{y\}$ is independent in $M'$ and $t(x', \widetilde{S}) \le t(y, \widetilde{S}) \le t(x', \widetilde{S}) + d\}$ using $O(d)$ high-energy queries to $C_H$, we add $x^*$ to $A'$ and we resume considering the elements $x$ in $A \setminus D(\mu)$ such that $t(x, \widetilde{S}) \ge t(x', \widetilde{S})$.[7]

Since $\operatorname{rank}(M') = k - |F| = cd$, the total number of high-energy operations is $O(cd^2) = O(\frac{d^2}{\epsilon})$. ◀

To conclude this section, we remark that Lemmas 7, 11, 12, 13, and 14 together prove Theorem 4 when minimization problems are concerned.

[7] Notice how the next element to be considered will again be $x'$.

**Algorithm 2** Dual-Mode $p$-Extendible-System Maximization($M$, $p$, $\widetilde{S}$, $d$).

```
1 γ ← 1 + ⌈2d/(p−1)⌉;
2 B* ← First γ elements included in the independent set maintained by greedy(M);
3 A ← greedy(M/B*, S̃);
4 return A ∪ B*;
```

## 5 Maximization in $p$-Extendible Systems

In this section we show the following theorem which yields a $p^2$-approximation for the general problem of computing a maximum-weight base of a $p$-extendible system $M$, and a $p$-approximation if $M$ is $p$-stable.

▶ **Theorem 15.** *Consider any maximization problem in $p$-extendible systems, with $p \geq 2$, where the input elements are given as a sequence with maximum dislocation at most $d$. There exists an algorithm which returns a $p$-approximation of the base returned by the greedy algorithm and that has $\langle O(d + \frac{d^2}{p}), O(n)\rangle$-high-low energy complexity.*

Similarly to matroids, we define the contraction $M/X$ of $M = (E, \mathcal{I})$ by $X \in \mathcal{I}$ as the independence system having $E \setminus X$ as its ground set and all $Y \in \mathcal{P}(E \setminus X)$ such that $Y \cup X \in \mathcal{I}$ as its independent sets. It is easy to check that $M/X$ is a $p$-extendible system.

Our Algorithm, whose pseudocode is shown in Algorithm 2, computes an independent set $B^*$ consisting of the first $\gamma = 1 + \lceil \frac{2d}{p-1} \rceil$ elements that greedy($M$) would select, and then completes the solution with the base $A$ of $M' = M/B^*$ obtained by greedily adding the elements in $\widetilde{S}$. We start our analysis by proving a generalization of Lemma 5 to $p$-extendible systems.

▶ **Lemma 16.** *Let $M$ be a $p$-extendible system and $k = |\text{greedy}(M, S)|$. Let $S_t$ (resp. $\widetilde{S}_t$) be the sequence containing the first $t$ elements of $S$ (resp. $\widetilde{S}$), $A_t = \{a_1, a_2, \dots\} = \text{greedy}(M, \widetilde{S}_t)$, and $B_t = \{b_1, b_2, \dots\} = \text{greedy}(M, S_t)$. For all $i = 1, \dots, \lfloor k/p \rfloor$ it holds $t(a_i, \widetilde{S}) \leq t(b_{i \cdot p}, S) + d$.*

**Proof.** For any time $t = 0, \dots, n$ we have $S_t \subseteq \widetilde{S}_{\min\{n, t+d\}}$. This implies that $|A_{\min\{t+d,n\}}| \geq |B_t|/p$. By choosing $t = t(b_{i \cdot p}, S)$ we obtain $|A_{\min\{t(b_{i\cdot p},S)+d,n\}}| \geq |B_{t(b_{i\cdot p},S)}|/p = ip/p = i$, meaning that greedy($M, \widetilde{S}$) must have already considered $a_i$ by the time it finished considering the $(t(b_{i \cdot p}, S) + d)$-th element of $\widetilde{S}$, i.e., $t(a_i, \widetilde{S}) \leq t(b_{i\cdot p}, S) + d$. ◀

We can now lower bound the weight of the base returned by Algorithm 2. Since if $p = 1$ the results of the previous sections apply, we henceforth assume $p \geq 2$.

▶ **Lemma 17.** *Algorithm 2 returns a base $A \cup B^*$ of $M$ of weight at least $\frac{1}{p} w(\text{greedy}(M))$.*

**Proof.** Let $B = \text{greedy}(M, S)$, $M' = M/B^*$, and $B' = \{b'_1, \dots, b'_k\} = \text{greedy}(M', S)$. Since a contraction of a $p$-extendible system is again a $p$-extendible system and $|A| \geq k/p$, we can invoke Lemma 16 on $M'$ to write:

$$w(A) = \sum_{a_i \in A} w(a_i) \geq \sum_{i=1}^{\lfloor \frac{k-2d}{p} \rfloor} w(a_i) \geq \sum_{i=1}^{\lfloor \frac{k-2d}{p} \rfloor} w(b'_{ip+2d}) \geq \frac{1}{p} \sum_{i=p+2d}^{k} w(b'_i).$$

Notice that $B = B^* \cup B'$ and that all the elements in $B^*$ weigh at least $w(b'_1)$, therefore:

$$\sum_{i=1}^{p+2d-1} w(b'_i) \leq (p+2d-1)w(b'_1) \leq (p+2d-1)\cdot\frac{1}{\gamma}w(B^*) \leq \frac{p+2d-1}{1+\frac{2d}{p-1}}w(B^*) = (p-1)w(B^*).$$

Combining the above inequalities:

$$\begin{aligned} w(A \cup B^*) &= w(A) + w(B^*) \geq \frac{1}{p}\sum_{i=p+2d}^{k} w(b'_i) + w(B^*) \\ &= \frac{1}{p}\sum_{i=1}^{k} w(b'_i) - \frac{1}{p}\sum_{i=1}^{p+2d-1} w(b'_i) + w(B^*) \geq \frac{1}{p}w(B') - \frac{p-1}{p}w(B^*) + w(B^*) \\ &= \frac{1}{p}\left(w(B') + w(B^*)\right) = \frac{1}{p}w(B). \end{aligned}$$

To conclude the proof we need to show that $A \cup B^*$ is a base of $M$. Since $A \cup B^*$ is an independent set of $M$ by construction, we only need to show that it is maximal. In order to do so suppose towards a contradiction that there exists an element $x \in E \setminus (A \cup B^*)$ such that $A \cup B^* \cup \{x\}$ is independent. If we let $A'$ be the independent set maintained by greedy$(M/B^*, \widetilde{S})$ immediately before $x$ is considered, then we have that $A' \cup \{x\} \subseteq A \cup \{x\} \subseteq A \cup B^* \cup \{x\}$ must also be independent, contradicting $x \neq A$. ◀

We now bound the high-low energy complexity of Algorithm 2 which, when combined with Lemma 17, immediately yields Theorem 15.

▶ **Lemma 18.** *The high-low energy complexity of Algorithm 2 when the elements of $E$ are given in a almost-sorted sequence $\widetilde{S}$ having dislocation at most $d$ is $\langle O(d + \frac{d^2}{p}), O(n)\rangle$.*

**Proof.** Since the overall number of the low-energy operations is $O(n)$ we only need to bound the number of high-energy operations, i.e., the ones needed to select $B^*$. By using a technique similar to the one described in the proof of Lemma 14, we can select the first $\gamma = O(1 + \frac{d}{p})$ elements of greedy$(M, S)$ by performing $O(d)$ high-energy queries to $C_H$ per element. The overall high-energy complexity is therefore $O(d + \frac{d^2}{p})$. ◀

Since in any $p$-extendible system, the greedy algorithm recovers the optimum whenever the instance is $p$-stable, Lemma 17 implies that Algorithm 2 computes a $p$-approximation. The next result, whose proof will appear in the full version of this paper, shows that our analysis is actually tight whenever $d \geq p$ (we recall that the best dislocation that sorting algorithms can achieve with high probability is $\Omega(\log n)$).

▶ **Theorem 19.** *For every $d \geq p \geq 2$, there exists a $p$-stable instance of a $p$-extendible system for which Algorithm 2 is no better than $p$-approximate.*

One might wonder whether our techniques can be extended to larger classes of independence system, e.g., to $p$-systems [19] (i.e., independence system such that the ratio between the cardinality of any two maximal independent sets is at most $p$). Unfortunately, the answer is negative: our analysis uses the fact that $p$-extendible systems are closed under restrictions and contractions. This is no longer true when $p$-systems are considered, as the following counterexample shows: Fix any $n \geq 2$ and let $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$ be two disjoint sets. We define the independence system $M = (\mathcal{E}, \mathcal{I})$ where $\mathcal{E} = A \cup B$ and $\mathcal{I} = \mathcal{P}(A) \cup \mathcal{P}(B)$. Notice that $M$ is a 1-system as the only two maximal independent sets of $M$ are $A$ and $B$. Consider now $M' = M|(A \cup \{b_1\})$. The only two maximal independent sets of $M'$ are $A$ and $\{b_1\}$, showing that $M'$ is a $n$-system but not a $(n-1)$-system.

## References


1. Bilge ES Akgul, Lakshmi N Chakrapani, Pinar Korkmaz, and Krishna V Palem. Probabilistic CMOS technology: A survey and future directions. In *Very Large Scale Integration, 2006 IFIP International Conference on*, pages 1–6. IEEE, 2006.
2. Christoph Ambühl. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In *Proc. of International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1139–1150. Springer, 2005.
3. Elliot Anshelevich and Shreyas Sekar. Blind, Greedy, and Random: Algorithms for Matching and Clustering Using Only Ordinal Information. In *Proc. of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 390–396, 2016.
4. Elliot Anshelevich and Shreyas Sekar. Truthful mechanisms for matching and clustering in an ordinal world. In *International Conference on Web and Internet Economics*, pages 265–278. Springer, 2016.
5. Andrew An Bian, Joachim M. Buhmann, Andreas Krause, and Sebastian Tschiatschek. Guarantees for Greedy Maximization of Non-submodular Functions with Applications. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 498–507, International Convention Centre, Sydney, Australia, 06–11 August 2017. PMLR.
6. Mark Braverman and Elchanan Mossel. Noisy Sorting Without Resampling. In *Proc. of the 19th Annual Symposium on Discrete Algorithms (SODA)*, pages 268–276, 2008. `arXiv:0707.1051`.
7. Lakshmi N. B. Chakrapani, Jason George, Bo Marr, Bilge E. S. Akgul, and Krishna V. Palem. Probabilistic Design: A Survey of Probabilistic CMOS Technology and Future Directions for Terascale IC Design. In Giovanni De Micheli, Salvador Mir, and Ricardo Reis, editors, *VLSI-SoC: Research Trends in VLSI and Systems on Chip*, pages 101–118, Boston, MA, 2008. Springer US.
8. Vaggos Chatziafratis, Tim Roughgarden, and Jan Vondrák. Stability and Recovery for Independence Systems. In *Proc. of the 25th Annual European Symposium on Algorithms (ESA)*, volume 87 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.ESA.2017.26`.
9. Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete applied mathematics*, 7(3):251–274, 1984.
10. Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with Noisy Information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. `doi:10.1137/S0097539791195877`.
11. Irene Finocchi, Fabrizio Grandoni, and Giuseppe F. Italiano. Optimal resilient sorting and searching in the presence of memory faults. *Theor. Comput. Sci.*, 410(44):4457–4470, 2009. `doi:10.1016/j.tcs.2009.07.026`.
12. Stefan Funke, Kurt Mehlhorn, and Stefan Näher. Structural filtering: a paradigm for efficient and exact geometric programs. *Comput. Geom.*, 31(3):179–194, 2005.
13. Barbara Geissmann, Stefano Leucci, Chih-Hung Liu, and Paolo Penna. Sorting with Recurrent Comparison Errors. In *ISAAC*, volume 92 of *LIPIcs*, pages 38:1–38:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
14. Barbara Geissmann, Stefano Leucci, Chih-Hung Liu, and Paolo Penna. Optimal Dislocation with Persistent Errors in Subquadratic Time. In *Proc. of the 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96 of *LIPIcs*, pages 36:1–36:13, 2018.
15. Barbara Geissmann, Stefano Leucci, Chih-Hung Liu, and Paolo Penna. Optimal Sorting with Persistent Comparison Errors. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, pages 49:1–49:14, 2019. `doi:10.4230/LIPIcs.ESA.2019.49`.
16. Barbara Geissmann and Paolo Penna. Inversions from Sorting with Distance-Based Errors. In *Proc. of the 44th InternationalConference on Current Trends in Theory and Practice of*

*Computer Science (SOFSEM)*, volume 10706 of *Lecture Notes in Computer Science*, pages 508–522. Springer, 2018. doi:10.1007/978-3-319-73117-9_36.

17 Lefteris M Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243(1-2):289–305, 2000.

18 Rolf Klein, Rainer Penninger, Christian Sohler, and David P. Woodruff. Tolerant Algorithms. In *Proc. of the 19th Annual European Symposium on Algorithm (ESA)*, pages 736—-747, 2011.

19 Bernhard Korte and Dirk Hausmann. An Analysis of the Greedy Heuristic for Independence Systems. In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 65–74. Elsevier, 1978. doi:10.1016/S0167-5060(08)70322-4.

20 Stefano Leucci, Chih-Hung Liu, and Simon Meierhans. Resilient Dictionaries for Randomly Unreliable Memory. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, pages 70:1–70:16, 2019. doi:10.4230/LIPIcs.ESA.2019.70.

21 Itamar Levi and Alexander Fish. Dual mode logic—Design for energy efficiency and high performance. *IEEE access*, 1:258–265, 2013.

22 Sven Leyffer, Stefan M. Wild, Mike Fagan, Marc Snir, Krishna V. Palem, Kazutomo Yoshii, and Hal Finkel. Doing Moore with Less - Leapfrogging Moore's Law with Inexactness for Supercomputing. *CoRR*, abs/1610.02606, 2016. arXiv:1610.02606.

23 Edo Liberty and Maxim Sviridenko. Greedy Minimization of Weakly Supermodular Set Functions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM-17)*, volume 81 of *LIPIcs*, pages 19:1–19:11, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.19.

24 Julián Mestre. Greedy in approximation algorithms. In *European Symposium on Algorithms*, pages 528–539. Springer, 2006.

25 Julián Mestre. Greedy in Approximation Algorithms. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 528–539, 2006. doi:10.1007/11841036_48.

26 Matúš Mihalák, Marcel Schöngens, Rastislav Šrámek, and Peter Widmayer. On the complexity of the metric tsp under stability considerations. In *Proc. of International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 382–393. Springer, 2011.

27 George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(1):265–294, 1978.

28 Andrzej Pelc. Searching games with errors - fifty years of coping with liars. *Theor. Comput. Sci.*, 270(1-2):71–109, 2002.

29 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proc. of the fortieth Annual ACM Symposium on Theory of computing (STOC)*, pages 67–74. ACM, 2008.

30 Jan Vondrák. Submodularity and curvature: The optimal algorithm (combinatorial optimization and discrete algorithms). Mathematical Analysis Laboratory of Kyoto University, 2010.