Biobjective Optimization Problems on Matroids with Binary Costs

Jochen Gorski[a] and Kathrin Klamroth[b] and Julia Sudhoff[c]

[a]TH Nürnberg, Nürnberg, Germany; [b,c]University of Wuppertal, Wuppertal, Germany

ABSTRACT
Like most multiobjective combinatorial optimization problems, biobjective optimization problems on matroids are in general intractable and their corresponding decision problems are in general NP-hard. In this paper, we consider biobjective optimization problems on matroids where one of the objective functions is restricted to binary cost coefficients. We show that in this case the problem has a connected efficient set with respect to a natural definition of a neighborhood structure and hence, can be solved efficiently using a neighborhood search approach. This is, to the best of our knowledge, the first non-trivial problem on matroids where connectedness of the efficient set can be established.

The theoretical results are validated by numerical experiments with biobjective minimum spanning tree problems (graphic matroids) and with biobjective knapsack problems with a cardinality constraint (uniform matroids). In the context of the minimum spanning tree problem, coloring all edges with cost 0 green and all edges with cost 1 red leads to an equivalent problem where we want to simultaneously minimize one general objective and the number of red edges (which defines the second objective) in a Pareto sense.

1. Introduction

Optimization problems on matroids have been frequently studied in the literature. Early references date back to the middle 1930's, see, for example, [1]. A well-known example are *graphic matroids*, i.e., minimum spanning tree problems in simple connected graphs. Single objective optimization problems on matroids can be solved efficiently by a simple greedy strategy. We refer to the books of Kung [2] and Oxley [3] for a more detailed introduction into this field.

While the literature on single objective matroid optimization is relatively rich, the work on multiobjective optimization on matroids mostly focuses on multiobjective spanning tree problems. See, for example, Ruzika and Hamacher [4] for a survey and Benabbou and Perny [5] for a more recent reference on this topic. Evolutionary methods for multiobjective spanning tree problems were suggested, among others, in Zhou and Gen [6], Knowles and Corne [7], Neumann and Witt [8] and Bossek et al. [9] as well as references therein. Loera at al. [10] describe heuristic approaches to general multiobjective matroid optimization problems that rely on adjacency relations

---

and nonlinear scalarizations. The methods are implemented in the MOCHA software package [11]. Approximation schemes were suggested, for example, in Grandoni et al. [12] and Bazgan et al. [13].

Multiobjective optimization problems on matroids are a special case of *multiobjective combinatorial optimization* (MOCO) problems which are known to be notoriously hard. We refer to [14] for a recent discussion of the prevalent difficulties in MOCO problems. The decision problem of multiobjective matroid optimization is proven to be $\mathcal{NP}$-complete in general, see [15]. For multiobjective spanning tree problems it was shown in [16] that already in the biobjective case the cardinality of the non-dominated set may grow exponentially with the size of the instance. This result applies also to multiobjective optimization problems on matroids. As a consequence, for such instances the complete enumeration of the non-dominated set is impractical since it requires an exponential amount of time. Bökler et al. [17] recently suggested to consider the concept of output sensitive complexity in the context of MOCO problems and analysed various problem classes. In the dissertation of Bökler [18] the output sensitive complexity of the biobjective spanning tree problem was related to that of biobjective unconstrained combinatorial optimization (BUCO) which is, however, also still open. Despite the general intractability of multiobjective spanning tree problems, it was shown in the dissertation of Seipp [19] that the number of extreme supported non-dominated outcome vectors grows only polynomially with the size of the instance.

The above mentioned hardness results usually refer to MOCO instances with 'large' cost coefficients that may grow exponentially with the instance size. For problems with 'small' cost coefficients the situation is different. When coefficients are small, then the ranges of possible outcome values are bounded, which limits the size of the non-dominated set. For example, the biobjective minimum spanning tree problem has only supported efficient solutions when all cost coefficients take only values from the set $\{0, 1, 2\}$, see again [19]. This implies that all efficient solutions of this problem are connected, i.e., the complete efficient set can be generated by only performing simple swap operations (e.g., pivot operations in an associated linear programming formulation) among efficient solutions. In the same work, [19] show that tri-objective optimization problems on uniform matroids with one general cost function and two binary cost functions have a connected efficient set. However, in general even comparably simple problems like BUCO may possess a non-connected efficient set, see [20].

In this paper we focus on biobjective optimization problems on matroids that have binary coefficients in *one* of the objectives. While the first objective may take arbitrary non-negative integer values, we assume that the second objective takes only values from the set $\{0, 1\}$. Note that binary coefficients allow for an alternative interpretation of the problem: When associating a cost of 0, for example, with the color 'green', and a cost of 1 with the color 'red', then we are interested in the simultaneous minimization of the cost of a solution (w.r.t. the first objective) *and* of the number of its red elements.

A related problem is the *multicolor matroid problem* that was discussed by Rendl and Leclerc [21] and by Brezovec et al. [22]. In this problem, a minimum cost solution is sought that does not exceed a given bound on the number of elements from different colors. Srinivas [23] extended the results from Brezovec et al. [22] to the case that the number of elements of different colors is constrained by linear inequalities. Hamacher and Rendl [24] generalized the multicolor matroid problem to combinatorial optimization problems, now allowing for elements having more than one color. Similar to [22] the goal is to find minimum cost solutions not exceeding given bounds on the number of elements in each color. A different optimization objective was considered in Climaco

2

et al. [25], who discussed a biobjective minimum cost / minimum label spanning tree problem in a graph where each edge is associated with a cost value and a label (i.e., a color). While the first objective is a classical cost objective that is to be minimized, the second objective is to find a solution with a minimal number of *different* labels (i.e., colors). Since it is already $\mathcal{NP}$-hard to determine the minimum label spanning tree on a given graph due to a result of Chang and Leu [26], this problem is also $\mathcal{NP}$-hard.

From an application point of view, MOCO problems with one general objective function and one (or several) binary objectives are closely related to $k - \max$ optimization where the kth largest cost coefficient of a solution vector is to be minimized. Such problems can be translated into a series of problems with binary sum objectives in a thresholding framework, see, e.g, [27] for more details.

Contribution. This paper extends results from Chapter 10 of the dissertation of Gorski [28]. It is shown that the non-dominated set of biobjective optimization problems on matroids with one general and binary objective function contains only supported efficient solutions and is connected. This is the foundation for an efficient exact algorithm that enumerates the non-dominated set using a neighborhood search approach, i.e., using simple swaps between elements contained in different (efficient) bases of the problem. This *Efficient Swap Algorithm* ESA can be interpreted as an extension of the algorithm of [29] for a constrained version of the problem that is guaranteed to generate the complete non-dominated set. To the best of our knowledge, this is the first non-trivial optimization problem on matroids for which connectedness of the efficient set is established.

Organization of the paper. The remainder of this paper is organized as follows. In Section 2 we recall basic concepts from matroid theory and from multiobjective optimization that are relevant for the subsequent sections. The biobjective matroid optimization problem with one binary cost objective is introduced in Section 3. The neighborhood search algorithm ESA is presented in Section 4, and connectedness of the efficient set is proven in Section 5. The numerical results presented in Section 6 confirm the efficiency of the algorithm introduced in Section 4. The paper is concluded in Section 7 with some ideas for future research.

2. Matroid and Multiobjective Optimization Preliminaries

We first review basic concepts from matroid theory and multiobjective optimization. For more details on matroid theory we refer to the books of Kung [2] and Oxley [3]. For an introduction into the field of multiobjective optimization, see, e.g., the books of Ehrgott [30] and Miettinen [31].

2.1. Matroids

Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be a finite *ground set* with $n \in \mathbb{N}$ elements and let $\mathcal{I}$ be a subset of the power set $\mathcal{P}(\mathcal{E})$ of $\mathcal{E}$. The ordered pair $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ is called a *matroid* if the

following three conditions are satisfied:

$$\emptyset \in \mathcal{I} \tag{M1}$$

$$(I \in \mathcal{I} \ \wedge \ I' \subseteq I) \quad \Rightarrow \quad I' \in \mathcal{I} \tag{M2}$$

$$\forall I_1, I_2 \in \mathcal{I} \text{ with } |I_1| < |I_2| \ \exists e \in I_2 \setminus I_1 : \ I_1 \cup \{e\} \in \mathcal{I}. \tag{M3}$$

$|I|$ denotes the cardinality of a finite set I. If $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ is a matroid, then all sets $I \in \mathcal{I}$ are called *independent sets.* Conversely, a subset of $\mathcal{E}$ is called *dependent* if it is not contained in $\mathcal{I}$.

An independent set $I \in \mathcal{I}$ is called *maximal* when $I \cup \{e\} \notin \mathcal{I}$ for all $e \in \mathcal{E} \setminus I$. Similarly, a dependent set $D \in \mathcal{P}(\mathcal{E}) \setminus \mathcal{I}$ is called *minimal* if $D \setminus \{e\} \in \mathcal{I}$ for all $e \in D$. Maximal independent sets are called *bases* of the matroid, and minimal dependent sets are called *circuits* of the matroid. All bases of a matroid have the same cardinality, which is referred to as the *rank* of $\mathcal{M}$. We denote the set of all bases of a given matroid by $\mathcal{X}$.

Given a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$, a basis $B \in \mathcal{X}$, and an element $e \in \mathcal{E} \setminus B$, then $B \cup \{e\}$ contains a uniquely determined circuit $C(e, B)$ containing e. This circuit is also called the *fundamental circuit* of e w.r.t. B. An important property of matroids is the *basis exchange property*:

$$\forall E, F \in \mathcal{X} \ \forall e \in E \setminus F \ \exists f \in F \setminus E : \ (E \cup \{f\}) \setminus \{e\} \in \mathcal{X}. \tag{B}$$

The following stronger version of the basis exchange property was proven in [32].

Lemma 2.1 ([32])**.** *Let $E, F \in \mathcal{X}$. For all $e \in E \setminus F$ there exists $f \in F \setminus E$ such that both $(E \cup \{f\}) \setminus \{e\}$ and $(F \setminus \{f\}) \cup \{e\}$ are bases in $\mathcal{X}$.*

In this context, two bases of a matroid are called *adjacent* if they have $m-1$ elements in common, assuming that the matroid is of rank m. According to the basis exchange property (B) and Lemma 2.1, a given basis can be transformed into an adjacent basis by exactly one basis exchange. We refer to this as a *swap operation* in the following.

If a subset $S \subseteq \mathcal{E}$ of the ground set $\mathcal{E}$ is deleted from $\mathcal{E}$, we obtain the *restriction* of $\mathcal{M}$ to $\mathcal{E} \setminus S$. The ground set of this matroid is the set $\mathcal{E} \setminus S$ and its independent sets are those independent sets of $\mathcal{M}$ that are completely contained in $\mathcal{E} \setminus S$, i.e., that do not contain any elements from S. We write $\mathcal{M} - S$ for short.

Moreover, if an independent set I of $\mathcal{M}$ is *contracted* we obtain the *contraction* of $\mathcal{M}$ to I denoted by $\mathcal{M}/I$. The ground set of $\mathcal{M}/I$ is given by $\mathcal{E} \setminus I$, and its independent sets are the sets $I' \subseteq (\mathcal{E} \setminus I)$ such that $I' \cup I$ is an independent set of $\mathcal{M}$.

A classical example for a matroid is the *uniform matroid* of rank k, denoted by $\mathcal{U}_{k,n}$. The independent sets of $\mathcal{U}_{k,n}$ are all subsets of $\mathcal{E} = \{e_1, \ldots, e_n\}$ that have at most k elements, and the bases of $\mathcal{U}_{k,n}$ are all subsets of $\mathcal{E}$ that have exactly k elements. A subset of $\mathcal{E}$ is a circuit of $\mathcal{U}_{k,n}$ if it contains exactly $k + 1$ elements of $\mathcal{E}$. Another common example is the *graphic matroid.* Given a finite undirected graph $G = (V, E)$ with node set V and edge set E, the independent sets of $\mathcal{M}(G)$ are all forests in G, and the bases of $\mathcal{M}(G)$ are all spanning forests of G. When G is connected, then $\mathcal{X}$ is the set of all spanning trees of G. In this case, a circuit is referred to as a cycle. It is easy to verify that uniform matroids and graphic matroids satisfy the conditions (M1), (M2) and (M3). We will use graphic matroids to illustrate the results throughout this paper.

Since the efficiency of the methods developed in this paper depends on the structure

of the considered matroid, we briefly review some further matroids in the following. First consider the *matching matroid* that is also defined on a finite undirected graph $G = (V, E)$. The ground set of the matching matroid is a subset of the vertices of G, i.e., $\mathcal{E} \subseteq V$, and all subsets of $\mathcal{E}$ that can be covered by a matching of G are independent. A special case of the matching matroid is the *transversal matroid,* that is a matching matroid on a bipartite graph $G = (V_1 \cup V_2, E)$ with bipartition $V = V_1 \cup V_2$, where $\mathcal{E}$ equals V_1 or V_2. The *partition matroid* is defined on a groundset $\mathcal{E} = \{e_1, \ldots, e_n\}$ of n elements that is partitioned into $p \geq 1$ subsets E_i, $i = 1, \ldots, p$. For given non-negative bounds $b_i \geq 0$, $i = 1, \ldots, p$, a set $I \subseteq \mathcal{E}$ is independent whenever $|E_i \cap I| \leq b_i$ for all $i = 1, \ldots, p$. Note that the uniform matroid is a special case of the partition matroid with $p = 1$.

2.2. Multiobjective Optimization

Now suppose that a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ is given and that $p \geq 2$ cost coefficients $w_i(e) \geq 0$, $i = 1, \ldots, p$, are associated with each element $e \in \mathcal{E}$ of the ground set $\mathcal{E}$. The cost of a subset $S \subseteq \mathcal{E}$ in the ith objective is computed as $w_i(S) = \sum_{e \in S} w_i(e)$, $i = 1, \ldots, p$. Then the *multiple objective matroid problem* (MOMP) can be formulated as

$$\min w(B) = (w_1(B), \ldots, w_p(B)) \qquad \text{(MOMP)}$$
$$\text{s.t. } B \in \mathcal{X}.$$

The feasible solutions of (MOMP) are the bases $B \in \mathcal{X}$ of the matroid, and $y = w(B) \in \mathbb{R}^p$ denotes the *cost vector* or *outcome vector* of the basis B. In the following, we will enumerate different outcome vectors by using superscripts and refer to their components by subscripts. The minimization in problem (MOMP) is understood w.r.t. the Pareto concept of optimality that is based on the componentwise ordering in $\mathbb{R}^p$:

$$y^1 \leqq y^2 :\Leftrightarrow y_i^1 \leq y_i^2, \ i = 1, \ldots, p,$$
$$y^1 \leqslant y^2 :\Leftrightarrow y_i^1 \leq y_i^2, \ i = 1, \ldots, p \text{ and } y^1 \neq y^2,$$
$$y^1 < y^2 :\Leftrightarrow y_i^1 < y_i^2, \ i = 1, \ldots, p.$$

We say that an outcome vector y^1 *dominates* another outcome vector y^2 if and only if $y^1 \leqslant y^2$, and y^1 *strongly dominates* y^2 if and only if $y^1 < y^2$. A feasible solution $B \in \mathcal{X}$ (i.e., a feasible basis) is called *efficient* or *Pareto optimal* if there does not exist another feasible solution $\bar{B} \in \mathcal{X}$ that *dominates* B, i.e., for which $w(\bar{B}) \leqslant w(B)$. Similarly, $B \in \mathcal{X}$ is called *weakly efficient* or *weakly Pareto optimal* if there does not exist $\bar{B} \in \mathcal{X}$ that *strongly dominates* B, i.e., for which $w(\bar{B}) < w(B)$ (cf. also Figure 4 in Section 5 below). We are interested in finding the *efficient set* (or the *weakly efficient set*, respectively) of problem (MOMP) given by

$$\mathcal{X}_E := \{B \in \mathcal{X} : \text{there exists no } \bar{B} \in \mathcal{X} \text{ with } w(\bar{B}) \leqslant w(B)\},$$
$$\mathcal{X}_{wE} := \{B \in \mathcal{X} : \text{there exists no } \bar{B} \in \mathcal{X} \text{ with } w(\bar{B}) < w(B)\}.$$

Given $\mathcal{X}_E$ and $\mathcal{X}_{wE}$, the images of these two sets under the vector-valued mapping w are called *non-dominated set* and *weakly non-dominated set*, respectively:

$$\mathcal{Y}_N := w(\mathcal{X}_E)$$
$$\mathcal{Y}_{wN} := w(\mathcal{X}_{wE}).$$

A subset $\mathcal{X}_{cE}$ of $\mathcal{X}_E$ satisfying $w(\mathcal{X}_{cE}) = \mathcal{Y}_N$ is called a *complete set of efficient solutions*. Note that in general $\mathcal{X}_{cE} \subsetneq \mathcal{X}_E$. If in addition $|w(\mathcal{X}_{cE})| = |\mathcal{Y}_N|$ holds true, we say that the set $\mathcal{X}_{cE}$ is of minimal cardinality or just minimal, for short. Note that in this case, $\mathcal{X}_{cE}$ contains exactly one efficient solution for each vector in the non-dominated set. Algorithms designed to solve (MOMP) often aim to compute $\mathcal{Y}_N$ and $\mathcal{X}_{cE}$ rather than $\mathcal{Y}_N$ and $\mathcal{X}_E$. An efficient basis B is called *supported efficient* if it is a minimizer of the non-trivial weighted sum problem $\min\{\sum_{i=1}^{p} \lambda_i w_i(B), B \in \mathcal{X}\}$ with $\lambda_i \in (0,1)$, $i = 1, \ldots, p$ and $\sum_{i=1}^{p} \lambda_i = 1$. Note that the image $w(B)$ of a supported efficient basis B is called supported non-dominated outcome vector and lies on the boundary of the convex hull $\mathrm{conv}(\mathcal{Y})$ of the set $\mathcal{Y} = w(\mathcal{X})$ of feasible outcome vectors in the objective space. Moreover, if a basis B is supported efficient and if $w(B)$ is an extreme point of $\mathrm{conv}(\mathcal{Y})$ then B is called an *extreme supported efficient basis* and $w(B)$ is called an *extreme supported non-dominated point*. See Figure 4 in Section 5 for an illustration.

An important subset of the efficient set is the set of lexicographically optimal solutions: An outcome vector y^1 is *lexicographically optimal* if for all other outcome vectors y^2 it holds that $y_i^1 < y_i^2$ with $i = \min\{j \in \{1, \ldots, p\} : y_j^1 \neq y_j^2\}$.

Based on the concept of adjacent bases, the *adjacency graph* $G = (V, E)$ of efficient bases of Problem (MOMP) is defined analogous to [20]. The node set V consists of all efficient bases of (MOMP). An (undirected) edge is introduced between all pairs of vertices corresponding to adjacent bases of the underlying problem. These edges form the set E. The set $\mathcal{X}_E$ is said to be connected if its corresponding adjacency graph G is connected, i.e., if every pair of vertices in V is connected by a path. As shown in [28], the adjacency graph G is not connected in general, even if it is extended to include weakly efficient bases. Nevertheless, the adjacency graph always contains a connected component given by the supported efficient bases of (MOMP), see [15]. Although the adjacency graph is not connected in general, many solution methods make use of the adjacency structure of matroids. Some examples for such solution strategies are described in [10].

3. Problem Formulation and Notation

Let $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ be a matroid and let $\mathcal{X}$ denote the set of all bases of $\mathcal{M}$. We assume that $\mathrm{rank}(\mathcal{M}) = m > 0$, i.e., the cardinality $|B|$ of all bases $B \in \mathcal{X}$ is equal to m. In the following we consider two different types of cost functions on the ground set $\mathcal{E}$. While the first function $c : \mathcal{E} \to \mathbb{N}$ is given by arbitrary non-negative integer coefficients, we assume that the second cost function $b : \mathcal{E} \to \{0, 1\}$ only takes binary values on the elements of the ground set. According to these definitions the two different costs of a basis $B \in \mathcal{X}$ are given by $c(B) = \sum_{e \in B} c(e)$ and $b(B) = \sum_{e \in B} b(e)$, respectively. The related *biobjective matroid problem with binary costs* (BBMP) is given by

$$\min_{B \in \mathcal{X}} \left(c(B), b(B) \right). \qquad\qquad (BBMP)$$

Since the second cost function b has binary coefficients for all elements $e \in \mathcal{E}$, the corresponding objective function values of feasible bases $B \in \mathcal{X}$ are lower bounded by zero and upper bounded by m. In other words, $b(\mathcal{X}) := \{b(B) : B \in \mathcal{X}\} \subseteq \{0, \dots, m\}$ is of size $\mathcal{O}(m)$, and thus the same bound also holds for $\mathcal{Y}_N$.

For solving Problem $(BBMP)$ we introduce the following two associated ε-constraint versions of the problem. The first is given by

$$\begin{aligned} \min \; & c(B) \\ \text{s.t. } \; & b(B) \leq k, \qquad\qquad (BMP_\leq) \\ & B \in \mathcal{X}, \end{aligned}$$

where $k \in \{0, \dots, m\}$ is a fixed integer bound on the binary cost function b. From the theory of multiple criteria optimization (see e.g. [33]) we know that each optimal solution of Problem $(BMP_\leq)$ is at least weakly efficient for Problem $(BBMP)$. Note that this is not true in general when the inequality constraint in Problem $(BMP_\leq)$ is replaced by an equality constraint. Indeed, given an optimal solution of the equality constrained problem

$$\begin{aligned} \min \; & c(B) \\ \text{s.t. } \; & b(B) = k, \qquad\qquad (BMP_=) \\ & B \in \mathcal{X}, \end{aligned}$$

this solution may be dominated in Problem $(BBMP)$. An example for this situation can be seen in Figure 4. There, each basis $B \in \mathcal{X}$ that maps to the outcome vector $(c(B), b(B)) = (18, 5)$ is optimal for Problem $(BMP_=)$ with $k = 5$, while it is dominated by all bases that map to the outcome vector $(17, 4)$ for the biobjective problem. Nevertheless, we will use Problem $(BMP_=)$ to generate a sequence of optimal solutions by varying $k \in \{0, \dots, m\}$ and show that there exists a critical index j such that for all $k \leq j$ all generated bases that are optimal for Problem $(BMP_=)$ correspond to efficient bases of Problem $(BBMP)$.

Note that the binary cost function b introduced above also allows for another interpretation as used, for example, in [29] and [34]: Given a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ and a (first) cost function $c : \mathcal{E} \to \mathbb{N}$, one of the two colors red and green is assigned to each element of $\mathcal{E}$. In [29] and [34] algorithms are presented that determine a minimum cost basis $B \in \mathcal{X}$ that contains exactly k red elements from $\mathcal{E}$ (here, k is a predetermined parameter). To establish a connection between the problem discussed in [29] and [34] and the problems considered here, we simply identify the red elements $r \in \mathcal{E}$ from the ground set $\mathcal{E}$ with the binary costs $b(r) = 1$, while all green elements $g \in \mathcal{E}$ are considered to have binary cost $b(g) = 0$. Hence, determining a minimum cost basis $B \in \mathcal{X}$ containing at most or exactly k red elements from $\mathcal{E}$ corresponds to solving Problem $(BMP_\leq)$ and $(BMP_=)$, respectively. In this context, especially Problem $(BMP_=)$ can be seen as a generalized version of a single objective matroid problem with an additional constraint, where the original problem is obtained when $\mathcal{E}$ only consists of red elements and $k = m$. Note that for a better illustration, we will make use of the idea of red and green elements in the further sections.

4. Solving Biobjective Matroid Problems with Binary Costs

In this section we present an algorithm that computes the complete non-dominated set of Problem $(BBMP)$ in polynomial time. The method is based on the ideas stated in [29] and can be used to establish a connectedness result for the adjacency graph of Problem $(BBMP)$. In more detail, the algorithm generates a sequence of optimal solutions of Problem $(BMP_=)$ for decreasing right-hand side values k. In Subsection 4.1 we formulate the theoretical results that are needed to prove the correctness of the method, and in Subsection 4.2 we present the algorithm itself and illustrate it at a graphic matroid.

To simplify the discussion, we use the following notation for set operations throughout this and the following sections: Let S denote a subset of the finite ground set $\mathcal{E}$ and let $e, f \in \mathcal{E}$. We write $S + e$ to denote the set $S \cup \{e\}$ and $S - f$ to denote the set $S \setminus \{f\}$. Furthermore, let $S^c := \mathcal{E} \setminus S$ denote the complement of S in $\mathcal{E}$. To further simplify the notation we assume throughout this section that set operations are executed from left to right. Given an instance of Problem $(BBMP)$, we denote by $E_0 := \{e \in \mathcal{E} : b(e) = 0\}$ the subset of $\mathcal{E}$ containing all elements with binary cost 0 (green elements) while $E_1 := \{e \in \mathcal{E} : b(e) = 1\} = E_0^c$ denotes the set of elements with binary cost 1 (red elements). By definition, E_0 and E_1 form a partition of $\mathcal{E}$.

Throughout this section, we consider Problem $(BBMP)$ on a given matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ with set of feasible bases $\mathcal{X}$.

4.1. Minimal Swaps

The idea of our approach to generate the complete non-dominated set of Problem $(BBMP)$ is based on the stronger version of the basis exchange property for matroids stated in Lemma 2.1. Given this property we define *swaps* between elements from E_0 and E_1.

Definition 4.1. Let $B \in \mathcal{X}$. Then the *swap* (e, f) w.r.t. B is an ordered pair of elements such that $e \in E_1 \cap B$, $f \in E_0 \setminus B$ and $B - e + f \in \mathcal{X}$ is a basis. The *cost* of the swap (e, f) is defined as $c(e, f) := c(f) - c(e)$. A swap (e, f) is called *minimal* w.r.t. B if $c(e, f) \leq c(e', f')$ for all $e' \in E_1 \cap B$ and $f' \in E_0 \setminus B$ with $B - e' + f' \in \mathcal{X}$.

By definition, a swap always improves the binary cost function by one unit since a red element from E_1 is replaced by a green element from E_0. The idea of the *efficient swap algorithm* (ESA) is to generate a sequence of minimal swaps that yields all non-dominated outcome vectors of Problem $(BBMP)$, as outlined in Algorithm 4.1.

A detailed description of this approach will be given in Algorithm 4.2 below, after a thorough analysis of the individual steps.

For this purpose, let $i \in \{0, \ldots, m\}$ and $\mathcal{X}_i := \{B \in \mathcal{X} : |B \cap E_0| = i\}$ be the set of all bases with exactly i green elements. Note that $\mathcal{X}_i$ might be empty for low or high values of i, respectively. Furthermore, let

$$\mathcal{S}_i := \{B \in \mathcal{X}_i : c(B) \leq c(B') \, \forall B' \in \mathcal{X}_i\}$$

denote the set of all bases with minimal costs containing exactly i green elements from E_0. By construction, $B \in \mathcal{S}_i$ is an optimal basis of Problem $(BMP_=)$ with right hand side value $k = m - i$.

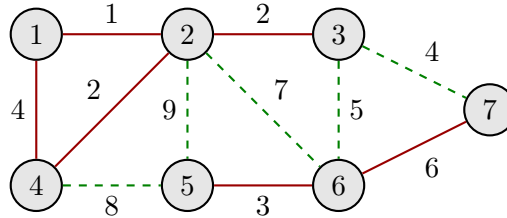From [29] we recall that given an optimal solution $B \in \mathcal{S}_{i-1}$, a minimal swap can

Algorithm 4.1 Outline of the Efficient Swap Algorithm (ESA) for Biobjective Matroid Problems with one Binary Cost Function

Input: An instance $((\mathcal{M}, \mathcal{X}, (c, b))$ of Problem $(BBMP)$.
Output: $\mathcal{Y}_N$ and a complete set $\mathcal{X}_{cE}$ of efficient solutions.
  1: Determine a basis B_j, which is optimal with respect to c, and a basis B_u, which is optimal with respect to b, such that both bases have as many elements as possible in common.
  2: Compute a sequence of minimal swaps, which describes the necessary swaps to get from basis B_j to basis B_u.
  3: Sort the swaps in non-decreasing order with respect to their costs.
  4: Compute from the sorted swap sequence a complete set $\mathcal{X}_{cE}$ of efficient solutions and the corresponding outcome vectors $\mathcal{Y}_N$.
  5: **return** $\mathcal{X}_{cE}$ and $\mathcal{Y}_N$.

Figure 1. Graph $G = (V, E)$ with costs $c(e)$, $e \in E$, for the graphic matroid considered in Example 4.4. Dashed green lines correspond to edges $e \in E$ with $b(e) = 0$ and solid red lines correspond to edges with $b(e) = 1$, respectively.

be used to generate an optimal solution contained in $\mathcal{S}_i$ whenever $\mathcal{S}_i$ is non-empty.

Theorem 4.2 (see [29], Augmentation Theorem 3.1)**.** *Let $B \in \mathcal{S}_{i-1}$ for an $i \in \{1, \ldots, m\}$ and assume that $\mathcal{S}_i \neq \emptyset$. If the swap (e, f) is minimal w.r.t. B, then $B - e + f$ is contained in $\mathcal{S}_i$.*

The following result is an immediate consequence of Theorem 4.2.

Corollary 4.3. *Let $s, t \in \mathbb{N}$ with $0 \leq s < t \leq m$ such that $\mathcal{S}_s \neq \emptyset \neq \mathcal{S}_t$. Then $\mathcal{S}_i \neq \emptyset$ for all $i \in \{s, \ldots, t\}$.*

Note that Corollary 4.3 does not state that Problem $(BMP_=)$ is feasible for all right-hand side values $k \in \{0, \ldots, m\}$. However, it implies that there exist fixed lower and upper bounds $l, u \in \mathbb{N}$ (satisfying $0 \leq l \leq u \leq m$) such that $\mathcal{S}_i \neq \emptyset$ for all $i \in \{l, \ldots, u\}$ while $\mathcal{S}_j = \emptyset$ for all $j \in \{0, \ldots, m\} \setminus \{l, \ldots, u\}$.

The results of Theorem 4.2 and Corollary 4.3 imply a simple algorithm that allows to generate a superset of the non-dominated set for a given instance of Problem $(BBMP)$ by swapping between the optimal bases contained in $\mathcal{S}_i$ for $i = \{l, \ldots, u\}$. In this method, a sequence of minimal swaps has to be generated. The algorithm presented in [29] uses a recursive procedure to generate this sequence. Further details on the generation of minimal swaps are given in Subsection 4.2 below. Example 4.4 illustrates the idea of sequential minimal swaps at a graphic matroid.

Example 4.4. We consider the graphic matroid induced by the graph $G = (V, E)$ given in Figure 1. Note that $\mathcal{X}$ is the set of all spanning trees of G, and that the

matroid has rank $m = 6$. The objective coefficients of the first objective function c are depicted next to each edge. For the second objective b, a solid red edge is used to indicate a cost of 1, while a dashed green edge indicates a cost of 0.

The spanning trees $T_1, \ldots, T_5$ given in Figure 2 correspond to optimal solutions for Problem $(BMP_=)$ for the right-hand side values $k \in \{1, \ldots, 5\}$. We have that $T_i \in \mathcal{S}_i$, $i \in \{1, \ldots, 5\}$ while $\mathcal{S}_0 = \mathcal{S}_6 = \emptyset$, i.e. $l = 1$ and $u = 5$. The objective vector $(c(T_i), b(T_i))$ of tree T_i, $i = 1, \ldots, 5$, is stated in the first column, below the name of the respective tree. The corresponding trees are shown in the second column. The tables in the right-most column list relevant swaps w.r.t. the tree T_i, $i = 1, \ldots, 5$, together with the respective cost, where minimal swaps are highlighted in bold. Here, the "in"-column goes through the list of all dashed green edges that are not yet contained in T_i and that may hence potentially be included. Adding the respective edges induces a unique cycle, and the best possible outgoing edge is shown in the "out"-column. It is selected as a solid red edge in this cycle with maximum cost. Since we exchange a red against a green edge, the swap with minimal cost $c(e, f)$ w.r.t. T_i leads to an optimal spanning tree $T_{i+1} \in \mathcal{S}_{i+1}$. While the spanning tree T_1 is dominated by T_2 (the implemented swap decreases each objective by one unit), the remaining trees form a complete set of efficient solutions and we conclude that $\mathcal{Y}_N = \{(17, 4), (22, 3), (27, 2), (34, 1)\}$.

Note that the procedure that is used to iteratively determine minimal swaps in Example 4.4 originates from [34]. In the following, we will present an improved procedure that avoids the computation of many unnecessary swaps. Example 4.4 further shows that not all optimal spanning trees for Problem $(BMP_=)$ result in an efficient solution for Problem $BBMP$. However, we will show in the following that there exists a fixed index $j \in \{l, \ldots, u\}$ such that $B \in \mathcal{S}_i$ is efficient whenever $i \geq j$. Having a closer look at the example, it can be recognized that the minimal swaps that lead from T_1 to T_5 have non-decreasing costs. To prove that this property holds in general, we need the following lemma from [29].
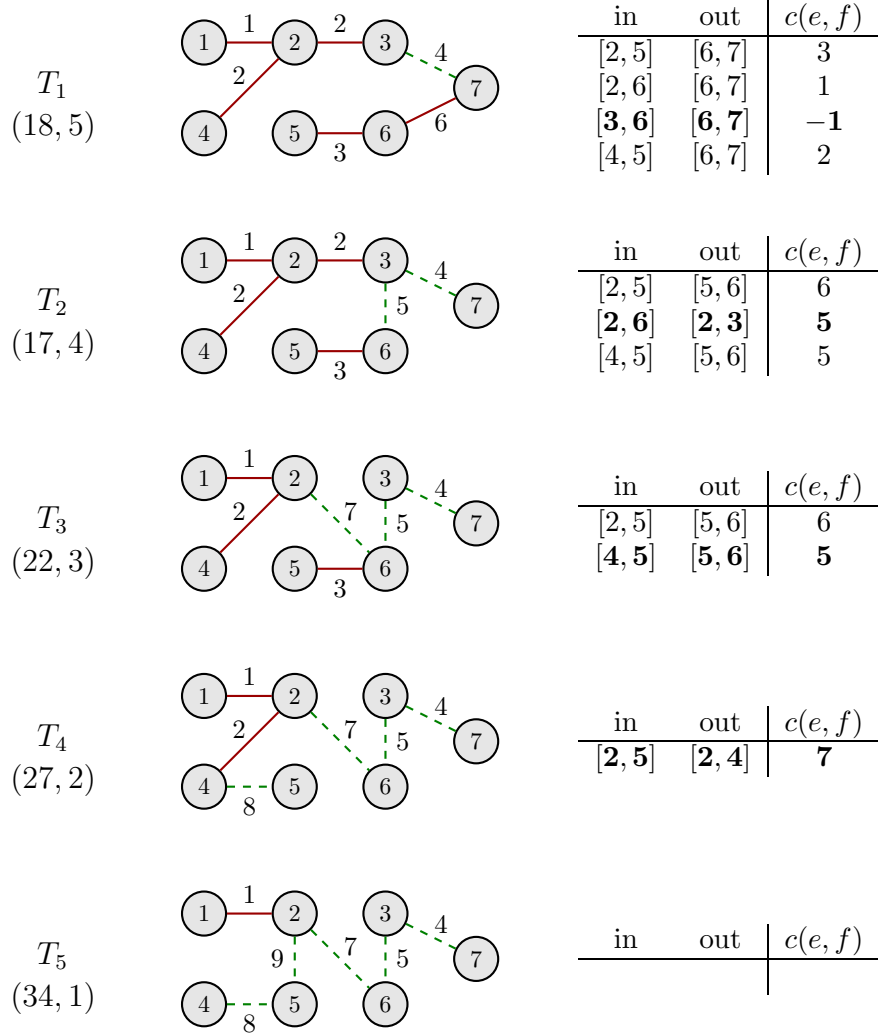
Lemma 4.5 (see [29], Lemma 3.2). *Let B be a basis containing the element $e \in E_1 \cap B$. Let (e, f) be a swap w.r.t. B that has minimal cost among all swaps w.r.t. B involving e, and set $B' = B - e + f$. Given $g \in E_1 \cap (B - e)$ arbitrary but fixed, let (g, h) and (g, h') denote swaps w.r.t. B and B', respectively, that have minimal costs w.r.t. B and B', respectively, and that involve g. Then it holds that $c(g, h) \leq c(g, h')$.*

Using Lemma 4.5 it can now be shown that the sequence of costs induced by a sequence of minimal swaps is non-decreasing for increasing $i \in \{l, \ldots, u\}$.

Theorem 4.6. *Let $u \geq l + 2$. For $i \in \{l, \ldots, u - 1\}$ let $B_i \in \mathcal{S}_i$ and let (e_i, f_i) denote a minimal swap w.r.t. B_i leading to B_{i+1}. Then the sequence of costs of minimal swaps $\{c(e_i, f_i)\}_{i=l}^{u-1}$ is non-decreasing, i.e. $c(e_i, f_i) \leq c(e_{i+1}, f_{i+1})$ for all $i \in \{l, \ldots, u - 2\}$.*

Proof. Let $\{c(e_i, f_i)\}_{i=l}^{u-1}$ be a cost sequence of minimal swaps and let $i \in \{l, \ldots, u-2\}$ arbitrary but fixed. Note that $e_i \neq e_{i+1}$ since $e_i \in B_i \setminus B_{i+1}$. Moreover, $e_{i+1} \in B_i \cap B_{i+1}$ since otherwise e_{i+1} would be contained in $B_{i+1} \setminus B_i$, i.e. $e_{i+1} = f_i$. But since e_{i+1} is a red element of $\mathcal{E}$ while f_i is a green element, this is impossible.

Now consider a swap (e_{i+1}, f) w.r.t. B_i that has minimal cost among all swaps w.r.t. B_i that involve e_{i+1}. Note that the existence of a swap w.r.t. B_i involving the edge e_{i+1} follows from the basis exchange property (B): For the two bases B_i, B_{i+2} we have $e_{i+1} \in B_i \setminus B_{i+2}$ and hence there exists an element $f \in B_{i+2} \setminus B_i$ such that

10

Figure 2. Sequence of optimal spanning trees $\{T_1, \ldots, T_5\}$ for $(BMP_=)$ for the graphic matroid defined in Example 4.4. Left column: tree T_i and corresponding objective vector. Center column: associated tree. Right column: computation of a minimal swap $c(e, f)$ w.r.t. T_i, $i = 1, \ldots, 5$.

$B_i - e_{i+1} + f \in \mathcal{X}$. Hence, (e_{i+1}, f) is a feasible swap w.r.t. B_i involving e_{i+1} and with $f \in B_{i+2} \setminus B_i = \{f_i, f_{i+1}\}$.

Since (e_i, f_i) is a minimal swap w.r.t. B_i it follows that $c(e_i, f_i) \leq c(e_{i+1}, f)$. If $f = f_{i+1}$, we are done. Otherwise, we conclude from Lemma 4.5 that $c(e_{i+1}, f) \leq c(e_{i+1}, f_{i+1})$, since the swap (e_{i+1}, f_{i+1}) is minimal w.r.t. B_{i+1}. Combining these results we get $c(e_i, f_i) \leq c(e_{i+1}, f_{i+1})$, which completes the proof. $\qquad\square$

Since we have that

$$c(B_{i+1}) - c(B_i) = c(f_i) - c(e_i) = c(e_i, f_i), \tag{1}$$

Theorem 4.6 implies that the minimum costs of bases $B \in \mathcal{S}_i$ define a convex function for $i = |B \cap E_0| \in \{l, \ldots, u\}$. Furthermore, if c and b are conflicting, then there must exist an index $j \in \{l, \ldots, u\}$ such that, starting from this index, all subsequent bases contained in the sequence $\{B_i\}_{i=j}^u$ correspond to efficient solutions of Problem $(BBMP)$.

This holds since the value of the binary objective function b is decreased by one unit when a swap from B_i to B_{i+1} is performed, while the corresponding value of the cost function c remains constant or is increased. By construction, the index j is the first index from $\{l, \ldots, u-1\}$ for which $c(e_i, f_i) > 0$ holds true. This implies the following result.

Theorem 4.7. *Let $\{B_i\}_{i=l}^u$ denote the sequence of minimum cost bases such that $B_i \in \mathcal{S}_i$ for $i \in \{l, \ldots, u\}$. Assume that $u \geq l + 2$. If there exists an index $j \in \{l+1, \ldots, u\}$ such that $c(B_{j-1}) < c(B_j)$, then $c(B_i) < c(B_{i+1})$ holds true for all $i \in \{j-1, \ldots, u-1\}$.*

Proof. Let $j \in \{l+1, \ldots, u\}$ denote the index where $c(B_{j-1}) < c(B_j)$ holds true for the first time. If $j = u$ then there is nothing to show. So, let $j < u$. It suffices to prove that $c(B_j) < c(B_{j+1})$ holds true. From Equation (1) it follows that $c(e_{j-1}, f_{j-1}) > 0$. Furthermore, Theorem 4.6 implies that

$$c(B_{j+1}) - c(B_j) = c(e_j, f_j) \geq c(e_{j-1}, f_{j-1}) > 0,$$

which shows that $c(B_j) < c(B_{j+1})$ is valid. $\square$

Note that the basis B_j where j is the index such that $c(e_j, f_j) > 0$ holds true for the first time is lexicographically optimal w.r.t. c (with secondary optimization w.r.t. b). This means that B_j is optimal w.r.t. c and additionally satisfies $b(B_j) \leq b(B)$ for all $B \in \mathcal{X}$ with $c(B) = c(B_j)$. A lexicographically optimal basis B_j can be computed efficiently using a greedy algorithm by computing an optimal basis w.r.t. the costs $w(e) = (m+1) \cdot c(e) + b(e)$ for all $e \in E$, where m is the rank of $\mathcal{M}$.

Theorem 4.7 induces a method that generates a minimal complete set $\mathcal{X}_{cE}$ of efficient bases. Starting from a lexicographically optimal basis contained in $\mathcal{S}_j$, we compute a sequence of minimal swaps $\{(e_i, f_i)\}_{i=j}^{u-1}$ which is called *swap sequence* in the following. By construction, we have that each of the generated bases B_i is contained in $\mathcal{S}_i$ for $i \in \{j+1, \ldots, u\}$. The basis B_j as well as all subsequently generated bases correspond to efficient solutions of Problem $(BBMP)$. Note that starting with basis B_j rather than with B_l has the advantage that all generated bases are efficient. For example, for the graphic matroid from Example 4.4 the basis B_l corresponds to T_1 in Figure 2 while basis B_j is given by T_2. Therefore, one unnecessary swap is omitted. Nevertheless, in the worst case $B_l = B_j$ holds and all swaps have to be calculated.

Since the binary objective b decreases by one unit in each iteration of this procedure, it is ensured that no non-dominated outcome vector is missed in the objective space and hence $B_j, \ldots, B_u$ form a minimal complete set of efficient bases. Hence, we have proven the following result:

Theorem 4.8. *Let $\{B_i\}_{i=j}^u$ denote a sequence of bases generated by a swap sequence. Then $X = \{B_j, \ldots, B_u\}$ forms a minimal complete set of efficient solutions and $\mathcal{Y}_N = \{(c(B_i), b(B_i)), i = j, \ldots, u\}$.*

4.2. The Efficient Swap Algorithm

The *Efficient Swap Algorithm* (ESA) presented in this section utilizes swap sequences to efficiently generate a minimal complete set of efficient solutions for Problem $(BBMP)$. Note that ESA can be interpreted as an extension of the algorithm stated in [29] for the solution of Problem $(BMP_=)$ for fixed k. Indeed, it was shown in [29] that this algorithm generates a complete swap sequence $\{(e_i, f_i)\}_{i=l}^{k-1}$ starting from

$B_l \in \mathcal{S}_l$ and leading to $B_k \in \mathcal{S}_k$. Setting $k = u$ and starting from a lexicographically optimal basis B_j thus induces ESA, and hence we omit detailed proofs for the correctness of this part of the algorithm. We rather focus on explaining how a complete swap sequence is generated without calculating a multiplicity of unnecessary swaps that do not lead to new efficient bases of the biobjective problem ($BBMP$). At the end of this subsection we apply our algorithm to the graphic matroid from Example 4.4 to show how ESA works in practice.

We use the ideas from Theorem 4.2 and Corollary 4.3 to avoid the calculation of unnecessary swaps. In a first step, we generate bases $B_j \in \mathcal{S}_j$ and $B_u \in \mathcal{S}_u$ such that these two bases have as many elements as possible in common. The following two properties hold if and only if B_j and B_u coincide in a maximal number of elements:

(a) $B_j \cap E_0 \subseteq B_u$, i.e., B_u contains all green elements from B_j.
(b) $B_u \cap E_1 \subseteq B_j$, i.e., B_j contains all red elements from B_u.

Note that properties (a) and (b) imply that $U := B_u \setminus B_j \subseteq E_0$ and that $J := B_j \setminus B_u \subseteq E_1$, respectively, and that $|U| = |J|$.

If both properties (a) and (b) hold, then all elements of the matroid that are neither contained in B_j nor in B_u are redundant for ESA and can be removed from the ground set of the problem, i.e., we continue by considering the restriction $\mathcal{M} - (B_j \cup B_u)^c$. Furthermore, only those elements have to be swapped that are not contained in both bases simultaneously (see [29] for a detailed proof of this fact). This means that it is sufficient to consider the contraction of the matroid w.r.t. all elements that are contained in both bases. ESA works on this reduced problem $(\mathcal{M} - (B_j \cup B_u)^c)/(B_j \cap B_u)$ and uses a recursive swap sequence generation procedure (SSG) to generate a swap sequence. We will illustrate the main aspects of this procedure in the following, assuming that B_j and B_u satisfy properties (a) and (b) above and that we start from B_j.

If we add a green element f from $B_u \setminus B_j \subseteq E_0$ with minimal costs to B_j, then a uniquely defined circuit $C(f, B_j)$ is generated. Note that all elements of this circuit, with the only exception of f, are elements of B_j. If these elements are all red, then a minimal swap (e^*, f) w.r.t. B_j containing f, where $e^* \in C(f, B_j) \setminus f$ and $c(e^*, f) \leq c(e, f)$ for all $e \in C(f, B_j) \setminus f$, has to be contained in a swap sequence. The reason for this is that no other element of this circuit will lead to a better swap than the swap (e, f) does, when f is added to B_j. Otherwise, if the circuit $C(f, B_j)$ contains red and green elements from B_j, then a minimal swap w.r.t. B_j containing f that is contained in a swap sequence cannot be deduced immediately. The idea in this case is to generate two smaller subproblems by contraction that do not intersect on the original ground set $\mathcal{E}$. The reduction to two subproblems is repeated until adding f leads to a circuit with only red edges besides f.

As will be explained in the following, problem splitting can be realised such that all swaps that are already guaranteed to be contained in a final swap sequence by the criterion given above are preserved (see [29] for further details). Moreover, the problem can be split until adding f leads to a circuit with only red edges besides f. Note that this is always satisfied when the respective ground sets of the contracted matroids consist of two elements $e \in B_j$ and $f \in B_u$ only. In this case, the swap (e, f) must be contained in a final swap sequence since this swap is minimal.

More formally, the split of the reduced matroid $(\mathcal{M} - (B_j \cup B_u)^c)/(B_j \cap B_u)$ into smaller parts is induced by a bisection of the sets $U = B_u \setminus B_j \subseteq E_0$ and $J = B_j \setminus B_u \subseteq E_1$. At first, the sets U and J are partitioned into two subsets U_1, U_2 and J_1, J_2 satisfying the following two conditions:

13

Algorithm 4.2 Efficient Swap Algorithm (ESA) for Biobjective Matroid Problems with one Binary Cost Function

Input: An instance $((\mathcal{M}, \mathcal{X}, (c, b))$ of Problem $(BBMP)$.
Output: $\mathcal{Y}_N$ and a complete set $\mathcal{X}_{cE}$ of efficient solutions.

1: $\mathcal{X}_{cE} = \emptyset$, $\mathcal{Y}_N = \emptyset$.
2: Determine a lexicographically optimal basis B_j
3: Determine a minimum basis B_u with respect to c such that B_u contains a maximal number of elements from E_0, all elements from $B_j \cap E_0$ and only those elements from E_1 that are also contained in B_j.
4: Call $\text{SSG}((\mathcal{M} - (B_j \cup B_u)^c)/(B_j \cap B_u), B_j \backslash B_u, B_u \backslash B_j)$ to generate a swap sequence.
5: Let $\{(e_i, f_i)\}_{i=j}^{u-1}$ denote the swap sequence found by Procedure SSG, where the swaps are sorted in non-decreasing order with respect to their costs.
6: Set $B = B_j$, $\gamma = c(B_j)$ and $\beta = b(B_j)$.
7: $\mathcal{X}_{cE} = \{B\}$ and $\mathcal{Y}_N = \{(\gamma, \beta)\}$
8: **for** $i = j$ to $u - 1$ **do**
9:     Set $B = B - e_i + f_i$, $\gamma = \gamma + c(e_i, f_i)$ and $\beta = \beta - 1$.
10:     Set $\mathcal{X}_{cE} = \mathcal{X}_{cE} \cup \{B\}$ and $\mathcal{Y}_N = \mathcal{Y}_N \cup \{(\gamma, \beta)\}$.
11: **end for**
12: **return** $\mathcal{X}_{cE}$ and $\mathcal{Y}_N$.

Algorithm 4.3 Swap Sequence Generation $\text{SSG}(\mathcal{M}, J, U)$ ([29])

Input: A matroid $\mathcal{M}$ and two sets of elements $J \subseteq B_j \backslash B_u$ and $U \subseteq B_u \backslash B_j$, $|J| = |U|$.
Output: A minimal swap (e, f) or two recursive calls of the procedure SSG.

1: **if** $|U| = 1$ **then**
2:     **return** the swap (e, f), where $J = \{e\}$ and $U = \{f\}$.
3: **else**
4:     Let U_1 be the set of $\lfloor |U|/2 \rfloor$ smallest elements with respect to c (contained in E_0) and set $U_2 = U \backslash U_1$.
5:     Determine J_1 such that $B = J_1 \cup U_1$ forms a minimal basis for $\mathcal{M}$ with respect to c satisfying $B \cap E_0 = U_1$ and set $J_2 = J \backslash J_1$.
6:     Call $\text{SSG}((\mathcal{M} - U_2)/J_1, J_2, U_1)$ to find the swaps for the elements in U_1.
7:     Call $\text{SSG}((\mathcal{M} - J_2)/U_1, J_1, U_2)$ to find the swaps for the elements in U_2.
8: **end if**

---

(1) The set $U_1 \subseteq E_0$ consists of the $\lfloor |U|/2 \rfloor$ smallest elements of U with respect to c.
(2) The set $B = J_1 \cup U_1$ is a minimum basis for $\mathcal{M}$ with respect to c satisfying $B \cap E_0 = U_1$.

In a second step, the given problem is split into two different subproblems and the procedures $\text{SSG}((\mathcal{M} - U_2)/J_1, J_2, U_1)$ and $\text{SSG}((\mathcal{M} - J_2)/U_1, J_1, U_2)$ are executed.

Applying this procedure, it can be shown (cf. [29]) that all involved matroid problems remain feasible and that all swaps contained in the final swap sequence are preserved. Furthermore, if a subproblem consists of exactly one red element $e \in J \subseteq (B_j \backslash B_u) \cap E_1$ and one green element $f \in U \subseteq (B_u \backslash B_j) \cap E_0$, it is guaranteed that the swap (e, f) is in the swap sequence.

The Efficient Swap Algorithm ESA for the solution of Problem $(BBMP)$ is summarized in Algorithm 4.2. The associated bisection procedure SSG that is recursively called during the course of ESA is outlined in Algorithm 4.3. At the beginning of Algorithm 4.2 the two bases B_j and B_u are calculated. Then, using Algorithm 4.3, a

swap sequence for the (contracted) matroid $(\mathcal{M} - (B_j \cup B_u)^c)/(B_j \cap B_u))$ with ground set $(B_j \cup B_u) \setminus (B_j \cap B_u)$ is generated recursively. Finally, the generated swaps are sorted in non-decreasing order of their costs and, based on the result of Theorem 4.8, the non-dominated set $\mathcal{Y}_N$ as well as a minimal complete set $\mathcal{X}_{cE}$ of efficient solutions are determined.

Note that during the course of Algorithm 4.3 it may happen that swaps (or elements) with the same cost occur. So, a rule how to cope with ties in Line 4 of Algorithm 4.3 has to be given. We follow the approach suggested in [29]: First assume that the elements of E_0 are sorted and indexed according to their costs c in non-decreasing order. Then, in Line 4 of Algorithm 4.3 we always choose the first $\lfloor |U|/2 \rfloor$ elements from U. When there are ties in the costs of the swap sequence, then the affected swaps are arranged in increasing order of the indices with respect to the elements that are contained in E_0. The following theorem summarizes the results.

Theorem 4.9. *Algorithm 4.2 is correct and returns the non-dominated set and a minimal complete set of efficient solutions.*

Proof. The correctness of the algorithm follows from Theorem 4.8 and from the correctness of the algorithm for solving Problem $(BMP_=)$ stated in [29]. □

Note that the complexity of Algorithm 4.2 depends on the considered matroid problem. For graphic matroids with $G = (V, E)$, for example, it is shown in [29] that their basic algorithm solves Problem $(BMP_=)$ within $\mathcal{O}(m \log \log_{(2+m/n)} n + n \cdot \log(n))$ time, where $|V| = n$ and $|E| = m$. Hence, Algorithm 4.2 has the same time bound in this case, since the additional construction of $\mathcal{X}_{cE}$ and $\mathcal{Y}_N$ takes at most $\mathcal{O}(m)$ time. For a matching matroid and a transversal matroid the time bound is $\mathcal{O}(n \log n + m\ell)$, which follows again from a corresponding result in [29], where n is the number of vertices of a graph, ℓ is the number of edges in a maximum matching and m is the number of edges in the graph. Again, Algorithm 4.2 has the same time bound, since the additional construction of $\mathcal{X}_{cE}$ and $\mathcal{Y}_N$ takes at most $\mathcal{O}(\ell)$ time. Furthermore, it is proven in [29] that the Problem $(BMP_=)$ can be solved in linear time, i.e. $\mathcal{O}(n)$, for a partition matroid (and therefore also for a uniform matroid) on a groundset $\mathcal{E}$ which consists of n elements. In this case the construction of $\mathcal{X}_{cE}$ and $\mathcal{Y}_N$ takes at most $\mathcal{O}(n)$ time and hence Algorithm 4.2 has the same time bound.

Example 4.10. We apply ESA to the graphic matroid introduced in Example 4.4. To simplify the notation, the edges of the graph G (see Figure 1) are identified by their associated costs c rather than by their respective end nodes. This only induces ambiguity in the case of the edges $[2, 3]$ and $[2, 4]$ which both have cost 2, and in the case of the edges $[1, 4]$ and $[3, 7]$ which both have cost 4. We will refer to the edge $[2, 3]$ by writing $2'$ and to the edge $[1, 4]$ by writing $4'$ in the following to distinguish between these edges.

In a first step the optimal bases B_j and B_u are determined. This leads to the spanning trees T_2 and T_5, respectively, shown in Figure 2, i.e. $B_j = \{1, 2, 2', 3, 4, 5\}$ and $B_u = \{1, 4, 5, 7, 8, 9\}$. Hence, $U = B_u \setminus B_j = \{7, 8, 9\} \subseteq E_0$, $J = B_j \setminus B_u = \{2, 2', 3\} \subseteq E_1$, and $B_j \cap B_u = \{1, 4, 5\}$. This implies that the edges 1, 4 and 5, i.e., the edges $[1, 2]$, $[3, 6]$ and $[3, 7]$, are contained in every efficient spanning tree in the set $\mathcal{X}_{cE}$ generated by ESA, and the edges $4'$ and 6, i.e., the edges $[1, 4]$ and $[6, 7]$, can be removed from the problem since they are not contained in $B_j \cup B_u$. The contracted matroid $\mathcal{M}^1 := (\mathcal{M} - (B_j \cup B_u)^c)/(B_j \cap B_u)$ is shown in Figure 3. From now on, we will enumerate (contracted) matroids and their respective subsets by superscripts, while

Figure 3. Contracted graphic matroids $\mathcal{M}^1$, $\mathcal{M}^2$ and $\mathcal{M}^3$ from Example 4.10. Solid red lines correspond to edges e with $b(e) = 1$ while the green dashed lines correspond to edges with $b(e) = 0$. The edges are identified by their associated cost value c, where ambiguities are resolved by using the notation 2 and 2' to refer to the edges $[2, 4]$ and $[2, 3]$, respectively.

referring to the corresponding subsets U_1, U_2, J_1, J_2 by subscripts, as before.

Then the procedure SSG is called with SSG$(\mathcal{M}^1, J^1, U^1)$, where $J^1 := J = \{2, 2', 3\}$ and $U^1 := U = \{7, 8, 9\}$. Since $|U^1| = 3 > 1$, the matroid $\mathcal{M}^1$ has to be split into two smaller matroids. We first determine the $\lfloor |U^1|/2 \rfloor$ smallest elements of U^1 as $U_1^1 = \{7\}$ and set $U_2^1 := U^1 \setminus U_1^1 = \{8, 9\}$. Now we determine J_1^1 such that $B^1 = J_1^1 \cup U_1^1$ is a minimum basis for $\mathcal{M}^1$ with respect to c satisfying $B^1 \cap E_0 = U_1^1$. This implies that $J_1^1 = \{2, 3\}$, $J_2^1 := J^1 \setminus J_1^1 = \{2'\}$ and $B^1 = \{2, 3, 7\}$. Now the procedure SSG is called recursively with SSG$(\mathcal{M}^2, J^2, U^2)$ and SSG$(\mathcal{M}^3, J^3, U^3)$, respectively, where $\mathcal{M}^2$ and $\mathcal{M}^3$ correspond to the contracted matroids shown in Figure 3, and $J^2 = J_2^1$, $J^3 = J_1^1$, $U^2 = U_1^1$ and $U^3 := U^1 \setminus U_1^1 = \{8, 9\}$. SSG$(\mathcal{M}^2, J^2, U^2)$ returns immediately the swap $(2', 7)$ while SSG$(\mathcal{M}^3, J^3, U^3)$ needs another recursion to compute the swaps $(3, 8)$ and $(2, 9)$. Sorting these swaps in non-decreasing order of their costs leads to the swap sequence $\{(2', 7), (3, 8), (2, 9)\}$ with costs $5, 5, 7$. This immediately leads to the final result $\mathcal{Y}_N = \{(17, 4), (22, 3), (27, 2), (34, 1)\}$ and $\mathcal{X}_{cE} = \{T_2, T_3, T_4, T_5\}$, see also Figure 2.

5. Connectedness of the Efficient Set

In the following we show that the set of efficient bases $\mathcal{X}_E$ for Problem $(BBMP)$ is always connected. We recall from Section 2.1 that the set $\mathcal{X}_E$ is said to be connected if its corresponding adjacency graph is connected. Recall also that two efficient bases of a matroid of rank m are called adjacent if they have $m - 1$ elements in common. Our proof is based on the fact that the set of *supported* efficient bases is always connected with respect to the above given definition of adjacency for efficient bases. For more details on this topic we refer to [15]. In the following we show that every efficient basis of Problem $(BBMP)$ is a supported efficient solution which implies that the adjacency graph of the problem is always connected.

To do so, we first formulate a sufficient condition that guarantees that the non-dominated set of a general biobjective combinatorial minimization problem only consists of supported non-dominated outcome vectors. Given the non-dominated set $\mathcal{Y}_N = \{z_1, \ldots, z_n\} \subset \mathbb{R}^2$ of the problem, where $n \geq 3$ and $z_i = (x_i, y_i) \in \mathbb{R}^2$, with

$x_1 < \ldots < x_n$ and $y_1 > \ldots > y_n$, we define the *sequence of slopes* $\{m_i\}_{i=1}^{n-1}$ of subsequent points of $\mathcal{Y}_N$ by setting

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad i = 1, \ldots, n-1.$$

Note that $m_i \in (-\infty, 0)$ holds for all $i \in \{1, \ldots, n-1\}$.

Lemma 5.1. *Consider a biobjective combinatorial minimization problem and suppose that the sequence of slopes $\{m_i\}_{i=1}^{n-1}$ is non-decreasing. Then all non-dominated outcome vectors in the set $\mathcal{Y}_N$ are supported.*

Proof. Suppose that, to the contrary, there is a non-supported non-dominated outcome vector $z_t \in \mathcal{Y}_N$, $t \in \{2, \ldots, n-1\}$. Since a non-dominated outcome vector is supported if and only if it is an element of the convex hull of $\mathcal{Y}$, it follows that there exist supported non-dominated outcome vectors $z_i, z_j \in \mathcal{Y}_N$ and a weight $\lambda \in (0, 1)$ such that the point $z_\lambda = (x_\lambda, y_\lambda) := \lambda z_i + (1 - \lambda) z_j \in \mathbb{R}^2$ strongly dominates z_t, where $1 \leq i < t < j \leq n$ holds. Note that z_λ can not be an element of $\mathcal{Y}_N$ since otherwise it would dominate z_t. Without loss of generality we may assume that $i = 1$ and $t = 2$. Since $x_1 < x_\lambda < x_2$ and $y_\lambda < y_2 < y_1$ holds, it follows that

$$(y_\lambda - y_1) \cdot (x_2 - x_1) < (y_2 - y_1) \cdot (x_2 - x_1) < (y_2 - y_1) \cdot (x_\lambda - x_1) < 0.$$

Since z_λ is an element of the straight line connecting z_1 and z_j, it follows that

$$m^\star := \frac{y_j - y_1}{x_j - x_1} = \frac{y_\lambda - y_1}{x_\lambda - x_1} < \frac{y_2 - y_1}{x_2 - x_1} = m_1.$$

This is impossible, since by assumption $m_1 \leq m_i$ for all $i \in \{1, \ldots, n\}$, and hence

$$\begin{aligned}
y_j &= y_1 + \sum_{i=1}^{j-1}(y_{i+1} - y_i) = y_1 + \sum_{i=1}^{j-1} m_i \cdot (x_{i+1} - x_i) \\
&\geq y_1 + m_1 \cdot \sum_{i=1}^{j-1}(x_{i+1} - x_i) = y_1 + m_1 \cdot (x_j - x_1).
\end{aligned}$$

Therefore, it has to hold that $m^\star \geq m_1$, which is a contradiction. $\qquad\square$

We combine the results of Theorem 4.7, Theorem 4.8 and Lemma 5.1 to conclude that all nondominated outcome vectors of biobjective optimization problems on matroids with one binary objective function are supported.

Lemma 5.2. *Consider a feasible instance of Problem $(BBMP)$, i.e., assume that $\mathcal{Y} \neq \emptyset$. Then the non-dominated set $\mathcal{Y}_N$ consists only of supported non-dominated outcome vectors.*

Proof. Using the notation introduced in Section 4, we denote by $\{(e_i, f_i)\}_{i=j}^{u-1}$ a swap sequence starting from a lexicographically optimal basis B_j that induces a set of efficient bases $B_i \in \mathcal{S}_i$ for Problem $(BBMP)$, $i = j, \ldots, u$. According to Theorem 4.8 we have that $\mathcal{Y}_N = \{(c(B_i), b(B_i)), i = j, \ldots, u\}$.

Note that the result is trivial when $|\mathcal{Y}_N| \leq 2$. When $|\mathcal{Y}_N| \geq 3$ we know from Lemma 5.1 that it suffices to show that the sequence of slopes $\{m_i\}_{i=j}^{u-1}$, where

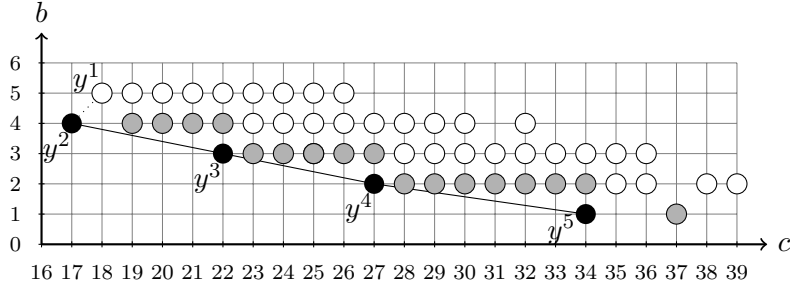$$m_i = \frac{b(B_{i+1}) - b(B_i)}{c(B_{i+1}) - c(B_i)} = \frac{-1}{c(B_{i+1}) - c(B_i)}$$

is non-decreasing. Since in this case $|\mathcal{Y}_N| \geq 3$ we have that $j \leq u - 2$. For an arbitrary but fixed index $i \in \{j, \ldots, u-2\}$ it follows from Theorem 4.6 and Theorem 4.7 that

$$c(B_{i+2}) - c(B_{i+1}) = c(e_{i+1}, f_{i+1}) \geq c(e_i, f_i) = c(B_{i+1}) - c(B_i) > 0.$$

This implies that

$$m_{i+1} = \frac{-1}{c(B_{i+2}) - c(B_{i+1})} \geq \frac{-1}{c(B_{i+1}) - c(B_i)} = m_i,$$

and hence the sequence of slopes $\{m_i\}_{i=j}^{u-1}$ is non-decreasing. This implies that $\mathcal{Y}_N$ contains only supported non-dominated outcome vectors. $\square$

Figure 4. Non-dominated set of the graphic matroid introduced in Figure 1. Non-dominated outcome vectors are shown in black and weakly non-dominated points shown in grey. The remaining (white) points are all dominated outcome vectors. Note that the points y^2, y^4 and y^5 are extreme supported non-dominated outcome vectors while y^3 is a non-extreme supported non-dominated outcome vector. The nodes y^i correspond to the trees T_i from Figure 2.

Note that not every supported non-dominated outcome vector must be extreme supported. Indeed, when the costs of two consecutive swaps (e_i, f_i) and (e_{i+1}, f_{i+1}) in a swap sequence are equal, then the point $(c(B_{i+1}), b(B_{i+1}))$ is not an extreme point of $\text{conv}(\mathcal{Y})$. To see this we consider again the graphic matroid introduced in Figure 1, c.f. Example 4.4. The set of feasible outcome vectors $\mathcal{Y}$ in the objective space is shown in Figure 4. In this example, the supported non-dominated point y^3 (which is the image of the spanning tree T_3 from Figure 2) is not an extreme point of $\text{conv}(\mathcal{Y})$ and thus not extreme supported.

We finally conclude that the set of efficient bases is connected.

Theorem 5.3. *Consider a feasible instance of Problem (BBMP). Then the set of efficient solutions $\mathcal{X}_E$ is connected.*

Proof. Lemma 5.2 implies that all non-dominated outcome vectors in $\mathcal{Y}_N$ are supported, and hence all efficient solutions in $\mathcal{X}_E$ are supported. Using the fact that the

(sub)graph of all supported efficient solutions is always connected (see [15]) implies the result. $\qquad\square$

6. Numerical Results

The main advantage of ESA is its computational efficiency. In this section we present numerical results that validate this statement for the examples of graphic and uniform matroids. In addition we address the question whether, and if yes, how far the results on the connectedness of the efficient set can be extended to more general cases. Towards this end, we randomly generated instances of uniform matroids with more than two (integer) values for the coefficients in the second objective. For all instances we compute the complete efficient set $\mathcal{X}_E$ and count the number of instances for which $\mathcal{X}_E$ is non-connected.

We note that other generalizations have been investigated for specific matroids. [19], for example, analyzes uniform matroids with one general cost function and two binary cost functions in a tri-criteria model. They suggest an exact solution method that is, similar to ESA, based on neighborhood search. In contrast to ESA their algorithm may generate dominated solutions. Nevertheless, they show that a complete set of efficient solutions can be generated in polynomial time with this method and that the efficient set consists only of supported solutions and is thus connected.

6.1. Performance of the Efficient Swap Algorithm

In this section, we present numerical results on randomly generated instances of graphic matroids and of uniform matroids to validate the efficiency of ESA.

6.1.1. Graphic Matroids

For graphic matroids on undirected connected graphs $G = (V, E)$, i.e., for biobjective minimum spanning tree problems with one general and one binary cost function, we evaluate the computational time needed by ESA to compute the non-dominated set $\mathcal{Y}_N$. To set this time in relation to the combinatorial complexity of the respective instances, we also provide the total number of feasible solutions, i.e., of spanning trees of the graph, and evaluate the time needed to determine all efficient trees from this set by total enumeration. This complete enumeration approach (CE) is implemented by using the matlab code by Matthias Hotz [35] for the generation of all spanning trees that is based on an algorithm described in [36]. For a recent survey and numerical comparison of exact algorithms for general multiobjective minimum spanning tree problems we refer to [37]. Note that the problem could also be solved by $n = |V|$ restarts of the method of Gabow and Tarjan [29] with appropriately chosen constraints on the number of green edges. ESA avoids these restarts as well as the computation of dominated solutions by initializing the swap sequence with a lexicographically optimal basis. The induced savings depend on the considered instance and are most significant when the non-dominated set is rather small compared to $|V|$.

The efficiency tests are run on a computer with an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz processor, 12MB Cache and 32 GB RAM. Both algorithms are implemented in MATLAB Version R2020a.

Recall from Section 3 that the objective values of the binary objective b can only take values between 0 and m, i.e., $b(B) \in \{0, 1, \ldots, m\}$ where m is the rank of the underlying

matroid and B is an arbitrary basis. As a consequence, we have that $|\mathcal{Y}_N| = \mathcal{O}(m)$. For an instance of the graphic matroid on a connected graph $G = (V, E)$ with n vertices and m edges, this implies that $b(T) \in \{0, 1, \ldots, n-1\}$ for all spanning trees T of G. Note that due to the common notation that $|V| = n$ and $|E| = m$ for graphic matroids, the rank of a graphic matroid is thus $n - 1$. The CE approach determines all efficient spanning trees by maintaining a list with n entries, one for each potential value of $b(T) \in \{0, 1, \ldots, n-1\}$ that stores the currently best cost value $c(T)$ together with all corresponding trees that were enumerated so far.

Table 1 summarizes the times needed to compute all non-dominated outcome vectors with ESA for instances with up to $n = 1000$ nodes and $m = 45\,000$ edges. To randomly generate connected graphs, we use a code from [38] that first constructs a random spanning tree for the required number of nodes and afterwards the remaining edges are randomly added. For all instances, the cost coefficients of the first objective were uniformly distributed random integers between 1 and $50\,000$ that were linearly transformed such that the smallest cost value is always equal to zero. For the second objective, the cost coefficients were uniformly distributed random integers from the set $\{0, 1\}$. To reduce the effect of fluctuations due to varying processor loads, all times are averaged over ten runs on the same instance. Despite the exponentially growing cardinality $|\mathcal{X}|$ of the feasible set, which was computed using Kirchhoff's matrix tree theorem (see, e.g., [39]) for instances up to $n = 100$, the computational time of ESA always remains below one minute. It can be observed that both the number $|\mathcal{Y}_N|$ of non-dominated outcome vectors as well as the computational time needed by ESA grow mainly with n, and only marginally with m and with the number $|E_1|$ of edges that have cost 1, i.e., which are in the set $E_1 = \{e \in \mathcal{E} : b(e) = 1\}$ in the second objective b.

The numerical results shown in Table 1 confirm the expected efficiency of ESA. Indeed, since ESA computes the set of non-dominated outcome vectors $\mathcal{Y}_N$ (which has at most n elements) rather than the set of all efficient solutions $\mathcal{X}_E$, the number of iterations of ESA is bounded by n. Moreover, each iteration requires a simple swap operation that can be implemented very efficiently.

Note that ESA generates only one pre-image, i.e., one feasible tree for each non-dominated outcome vector, while the number of efficient trees may be substantially larger. As an example, consider an instance where all edges have the same coefficients in both objectives. Then all spanning trees map to the same outcome vector in the objective space, i.e., $|\mathcal{Y}_N| = 1$, and are thus efficient, i.e., $|\mathcal{X}_E| = |\mathcal{X}|$. In order to test whether this is a common situation also in randomly generated instances, we computed the complete set $\mathcal{X}_E$ with the CE approach for the smaller instances from Table 1. It turns out that this is not the case for randomly generated instances on small graphs with a rather large range for the objective coefficients.

Note also that since ESA exploits the fact that the non-dominated set $\mathcal{Y}_N$ solely consists of supported non-dominated outcome vectors when one of the objective functions has binary coefficients only (c.f. Lemma 5.2), a numerical comparison with general solvers for bi- and multiobjective minimum spanning tree problems is not meaningful. In two-phase methods, for example, the search for unsupported non-dominated outcome vectors could be omitted, leading to an implementation that is somewhat similar to ESA. On the other hand, algorithms that generalize classical methods for the single objective minimum spanning tree problem to the multiobjective case cannot be expected to be competitive with ESA since they generally enumerate far too many irrelevant trees.

Table 1. Computational results for randomly generated graphs with n vertices, m edges, and $|E_1|$ edges with cost 1 in the binary cost function b. $|\mathcal{X}_E|$ is the number of efficient solutions, $|\mathcal{Y}_N|$ is the number of non-dominated points and $|\mathcal{X}|$ is the number of spanning trees for this instance. The last two columns give the time in seconds for ESA and for CE, respectively.

(n, m)	$|E_1|$	$|\mathcal{Y}_N|$	$|\mathcal{X}_E|$	$|\mathcal{X}|$	ESA [s]	CE [s]
(7, 10)	2	1	1	76	0.065	0.010
(7, 10)	4	4	4	66	0.032	0.001
(7, 15)	8	2	2	1 615	0.010	0.013
(7, 15)	12	3	3	1 807	0.014	0.015
(7, 20)	8	4	4	12 005	0.019	0.080
(7, 20)	11	5	5	12 005	0.023	0.081
(7, 20)	12	5	5	12 005	0.022	0.081
(10, 20)	10	6	6	26 646	0.027	0.203
(10, 20)	11	5	5	21 560	0.023	0.167
(10, 20)	15	2	2	18 956	0.011	0.139
(10, 30)	15	3	3	$1.85 \cdot 10^6$	0.016	11.808
(10, 30)	17	2	2	$1.62 \cdot 10^6$	0.012	10.215
(10, 30)	20	5	5	$1.60 \cdot 10^6$	0.021	10.202
(10, 40)	17	6	6	$3.06 \cdot 10^7$	0.029	172.213
(10, 40)	18	7	7	$3.01 \cdot 10^7$	0.032	171.100
(10, 40)	20	3	3	$3.01 \cdot 10^7$	0.016	168.403
(15, 30)	13	5	5	$5.35 \cdot 10^6$	0.025	38.684
(15, 30)	15	5	5	$4.11 \cdot 10^6$	0.023	29.293
(15, 30)	16	5	5	$4.66 \cdot 10^6$	0.024	33.935
(15, 60)	27	5	-	$2.97 \cdot 10^{11}$	0.028	-
(15, 60)	28	7	-	$3.95 \cdot 10^{11}$	0.034	-
(15, 60)	34	10	-	$2.86 \cdot 10^{11}$	0.034	-
(15, 100)	46	6	-	$9.46 \cdot 10^{14}$	0.034	-
(15, 100)	48	8	-	$9.35 \cdot 10^{14}$	0.040	-
(15, 100)	51	7	-	$9.35 \cdot 10^{14}$	0.036	-
(20, 40)	19	7	-	$1.18 \cdot 10^9$	0.034	8 663.920
(20, 40)	20	10	-	$7.42 \cdot 10^8$	0.044	5 419.526
(20, 100)	47	12	-	$5.76 \cdot 10^{17}$	0.058	-
(20, 100)	48	13	-	$4.43 \cdot 10^{17}$	0.062	-
(20, 100)	52	13	-	$4.15 \cdot 10^{17}$	0.059	-
(20, 180)	83	7	-	$8.91 \cdot 10^{22}$	0.047	-
(20, 180)	84	10	-	$8.89 \cdot 10^{22}$	0.057	-
(20, 180)	103	11	-	$8.94 \cdot 10^{22}$	0.057	-
(100, 200)	93	36	-	$4.94 \cdot 10^{44}$	0.198	-
(100, 200)	101	32	-	$5.06 \cdot 10^{45}$	0.179	-
(100, 200)	102	36	-	$1.06 \cdot 10^{45}$	0.193	-
(100, 1 000)	476	39	-	$2.20 \cdot 10^{125}$	0.296	-
(100, 1 000)	488	48	-	$5.74 \cdot 10^{125}$	0.341	-
(100, 1 000)	494	50	-	$2.14 \cdot 10^{125}$	0.345	-
(100, 2 000)	981	48	-	$3.00 \cdot 10^{156}$	0.452	-
(100, 2 000)	988	43	-	$3.00 \cdot 10^{156}$	0.427	-
(100, 2 000)	1 047	52	-	$2.32 \cdot 10^{156}$	0.457	-
(100, 4 000)	1 960	49	-	$5.39 \cdot 10^{186}$	0.700	-
(100, 4 000)	1 976	46	-	$5.43 \cdot 10^{186}$	0.681	-
(100, 4 000)	1 998	55	-	$5.53 \cdot 10^{186}$	0.723	-
(1 000, 2 000)	970	355	-	-	3.695	-
(1 000, 2 000)	976	320	-	-	3.445	-
(1 000, 2 000)	1 008	353	-	-	3.695	-
(1 000, 15 000)	7 419	489	-	-	8.101	-
(1 000, 15 000)	7 441	478	-	-	7.999	-
(1 000, 15 000)	7 541	511	-	-	8.325	-
(1 000, 30 000)	14 894	494	-	-	12.805	-
(1 000, 30 000)	14 947	482	-	-	12.416	-
(1 000, 30 000)	14 988	510	-	-	12.708	-
(1 000, 45 000)	22 430	470	-	-	17.375	-
(1 000, 45 000)	22 548	514	-	-	17.996	-
(1 000, 45 000)	22 632	517	-	-	18.011	-

Table 2. Computational results for randomly generated instances of uniform matroids $\mathcal{U}_{k,n}$. The two last columns show the accumulated average computation time over all $k = 1, \ldots, n/2$ in seconds, rounded over 10 repetitions for each instance, for ESA (for the computation of $\mathcal{Y}_N$) and for DP (for the computation of $\mathcal{X}_E$), respectively. For instances with 80 or more elements DP needs more than 600 seconds.

n	$|E_1|$	$|\mathcal{Y}_N|$	$|\mathcal{X}_E|$	ESA [s]	DP [s]
20	9	64	83	0.02	0.08
20	10	65	65	0.02	0.06
20	11	64	74	0.03	0.06
30	12	129	129	0.03	0.44
30	15	135	166	0.03	0.50
30	16	134	134	0.02	0.30
50	21	340	364	0.06	35.33
50	21	340	340	0.07	27.29
50	29	340	413	0.06	14.43
60	27	489	523	0.09	131.83
60	30	495	627	0.09	130.62
60	32	492	502	0.09	80.36
70	34	664	690	0.12	327.33
70	35	665	716	0.12	336.23
70	38	659	705	0.12	248.97
80	38	857	935	0.21	> 600
80	44	850	899	0.27	> 600
80	50	805	860	0.21	> 600
90	45	1 080	1 181	0.20	> 600
90	46	1 079	1 213	0.21	> 600
90	47	1 077	1 159	0.19	> 600
100	46	1 315	1 361	0.25	> 600
100	50	1 325	1 483	0.46	> 600
100	51	1 324	1 441	0.27	> 600

6.1.2. Uniform matroids

As a second test case we consider uniform matroids $\mathcal{U}_{k,n}$ on the ground set $\mathcal{E} = \{e_1, \ldots, e_n\}$, from which exactly k elements have to be selected in a basis. Rather than minimizing the cost of a basis we aim at maximizing its profit w.r.t. one general and one binary cost function to reflect the similarity of this problem to biobjective knapsack problems with bounded cardinality.

To determine the profit vectors of each element $e_i \in \mathcal{E}$, we generated n uniformly distributed random values from the set $\{0, 1, \ldots, 10n\}$ (for the first objective) and n values from the set $\{0, 1\}$ (for the second objective). After sorting the values for the first objective in non-decreasing order and the values of the binary objective in non-increasing order, the coefficients were combined into profit vectors for the elements $e_1, \ldots, e_n$.

For each instance on n elements, Table 2 shows the accumulated results over all values of $k \in \{1, \ldots, \frac{n}{2}\}$. In order to analyse the relation between $|\mathcal{Y}_N|$ and $|\mathcal{X}_E|$, we applied a simple implementation of a dynamic programming algorithm (DP) for multiobjective knapsack problems as described, for example, in [40]. Different from the biobjective minimum spanning tree instances described above, we consistently observe that the number of efficient solutions exceeds the number of non-dominated outcome vectors, however, not by very much. As was to be expected, ESA easily solves larger instances within fractions of a second, while the computational time required by DP grows significantly with the size of the instance. The efficency tests are run on a computer with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz processor and 8 GB RAM. The algoithms are implemented in MATLAB, Version R2019b.

Table 3. Number of observed nc-instances in 300 000 randomly generated instances of $\mathcal{U}_{k,20}$, cumulated for all $k \in \{1, \ldots, 10\}$, for different values of β.

β	2	3	4	5	6	7	8	9	11	13	15	20	25	30
nc-instances	0	0	0	0	0	0	2	4	3	2	8	10	12	13

6.2. Connectedness for more General Cost Functions

The proof of the connectedness of the efficient set of Problem ($BBMP$) (c.f. Theorem 5.3) relies on two basic properties: On one hand, this is the matroid structure of the considered problem, and on the other hand it is the fact that one of the two objective functions has only binary cost coefficients. While the first property ensures the feasibility of elementary swap operations, the latter implies that two adjacent non-dominated outcome vectors always differ by exactly one unit in the binary objective function.

In general, i.e., when the objective coefficients can be chosen freely, biobjective optimization problems on uniform matroids may have non-connected efficient sets. Corresponding examples are provided in [20] indicating that such non-connected instances (*nc-instances*) are very rare in randomly generated instances. The question remains whether non-connected instances already exist when the cost coefficients in the second objective are restricted to $\{0, 1, 2\}$ (rather than $\{0, 1\}$), or, more generally, to $\{0, 1, \ldots, \beta\}$ with $\beta \geq 2$.
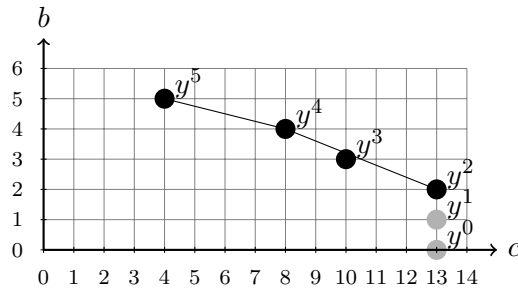
The frequency in which nc-instances occurred for different values of β in a large numerical study are reported in Table 3. For each value of $\beta \in \{2, \ldots, 9, 11, 13, 15, 20, 25, 30\}$ we randomly generated 300 000 instances of uniform matroids $\mathcal{U}_{k,n}$ with $n = 20$ elements. The profit vectors were chosen as described in Section 6.1 above, where the coefficients for the second objective were now drawn from the set $\{0, 1, \ldots, \beta\}$. All instances were solved for all $k \in \{1, \ldots, \frac{n}{2}\}$ using a DP approach for multiobjective knapsack problems, see [40].

Table 3 indicates that it seems to become more likely to find nc-instances the larger the range for the coefficients in the second objective function is, i.e., the larger the value of β is. Nevertheless, we suspect that nc-instances also exist for smaller values of β but that such instances are extremely rare. While nc-instances may be more likely for larger values of n, analysing large data sets becomes more and more challenging since this requires the exact computation of the complete efficient set for each instance (without the possibility of using ESA). We note that preliminary tests with $n = 30$ and $n = 50$ did not provide further insight on this topic.

A necessary condition for the existence of nc-instances is the existence of non-dominated non-supported outcome vectors. But in contrast to nc-instances it is quite easy to generate knapsack problems with such outcome vectors. An example is given in Table 4. In the first column of the left table the number of an item, in the second column the first weight and in the third column the second weight with $\beta = 2$ is given. In the right table, the efficient bases for $k = 3$ are given with their outcome vectors. Recall that we interpret the uniform matroid as a special case of a knapsack problem with bounded cardinality and thus consider both objective functions as maximization objectives. As can be seen in Figure 5, the basis $\{e_1, e_4, e_5\}$ is non-dominated and non-supported. Nevertheless, the efficient set of this instance is connected.

Table 4. Left: Elements e_i $i = 1, \ldots, k$ of a knapsack $\mathcal{U}_{3,6}$ with their weights with $\beta = 2$. Right: Efficient bases for the knapsack.

e	$c(e)$	$b(e)$		B	$c(B)$	$b(B)$
e_1	6	0		$\{e_1, e_2, e_3\}$	13	0
e_2	5	0		$\{e_1, e_2, e_4\}$	13	1
e_3	2	0		$\{e_1, e_2, e_5\}$	13	2
e_4	2	1		$\{e_1, e_4, e_5\}$	10	3
e_5	2	2		$\{e_1, e_5, e_6\}$	8	4
e_6	0	2		$\{e_4, e_5, e_6\}$	4	5

Figure 5. Non-dominated set of the uniform matroid introduced in Table 4. Non-dominated outcome vectors are shown in black and weakly non-dominated points shown in grey. The point y^3 corresponds to a non-dominated and non-supported outcome vector.

7. Conclusions

In this paper we investigate biobjective matroid problems involving one binary cost objective. We present an efficient swap algorithm (ESA) that solves this special kind of biobjective matroid problem efficiently, although the decision problem of the general version of this problem is known to be NP-complete (cf. [15]). The idea of ESA is based on a method of [29] for a constrained version of single-objective matroid optimization problems. The complexity of ESA depends on the matroid type. For a graphic matroid on a graph $G = (V, E)$, for example, it is given by $\mathcal{O}(m + n \cdot \log(n))$, where $|V| = n$ and $|E| = m$. Numerical experiments confirm the efficiency of this approach.

The efficient swap algorithm can be interpreted as a neighborhood search approach with an efficient strategy for the identification of relevant swaps. The correctness of this approach is based on the proof of the connectedness of the efficient set in this special case, which is in turn based on the insight that the non-dominated set consists only of supported non-dominated outcome vectors. This is surprising since it was shown in [28] that the efficient set is in general non-connected for biobjective matroid problems. To the best of our knowledge this is the first class of problems where connectedness of $\mathcal{X}_E$ can be established even though the non-dominated set is not contained in a hyperplane.

References

[1] Whitney H. On the abstract properties of linear dependence. American Journal of Mathematics. 1935;57(3):509–533.

[2] Kung JPS. A source book in matroid theory. Boston: Birkhäuser; 1986.

[3] Oxley JG. Matroid theory. Oxford University Press, NJ; 1992.

[4] Ruzika S, Hamacher H. A survey on multiple objective minimum spanning tree problems.

In: Lerner J, Wagner D, Zweig KA, editors. Algorithmics. (Lecture Notes in Computer Science; Vol. 5515/2009). Springer Berlin/Heidelberg; 2009. p. 104–116.

[5] Benabbou N, Perny P. On possibly optimal tradeoffs in multicriteria spanning tree problems. In: Walsh T, editor. Algorithmic Decision Theory; Cham. Springer International Publishing; 2015. p. 322–337.

[6] Zhou G, Gen M. Genetic algorithm approach on multi-criteria minimum spanning tree problem. European Journal of Operational Research. 1999;114:141–152.

[7] Knowles JD, Corne DW. A comparison of encodings andalgorithms for multiobjective minimum spanning tree problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC '01). IEEE Press; 2001. p. 544–551.

[8] Neumann F, Witt C. Multi-objective minimum spanning trees. In: Neumann F, Witt C, editors. Bioinspired computation in combinatorial optimization. Berlin, Heidelberg: Springer; 2010. Natural Computing Series; p. 149–159.

[9] Bossek J, Grimme C, Neumann F. On the benefits of biased edge-exchange mutation for the multi-criteria spanning tree problem. In: Proceedings of the 21th Genetic and Evolutionary Computation Conference (GECCO); Prague, Czech Republic. ACM; 2019. p. 516–523.

[10] Loera JAD, Haws DC, Lee J, et al. Computation in multicriteria matroid optimization. J Exp Algorithmics. 2010;14.

[11] Loera JAD, Haws DC, Lee J, et al. MOCHA – matroids optimization combinatorics heuristics and algorithms [`https://github.com/coin-or/MOCHA`]; 2009.

[12] Grandoni F, Ravi R, Singh M, et al. New approaches to multi-objective optimization. Mathematical Programming. 2014;146:525–554.

[13] Bazgan C, Ruzika S, Thielen C, et al. The power of the weighted sum scalarization for approximating multiobjective optimization problems. CoRR. 2019;abs/1908.01181. Available from: `http://arxiv.org/abs/1908.01181`.

[14] Figueira JR, Fonseca CM, Halffmann P, et al. Easy to say they're hard, but hard to see they're easy - toward a categorization of tractable multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis. 2017;24:82–98.

[15] Ehrgott M. On matroids with multiple objectives. Optimization. 1996;38(1):73–84.

[16] Hamacher HW, Ruhe G. On spanning tree problems with multiple objectives. Annals of Operations Research. 1994;52:209–230.

[17] Bökler F, Ehrgott M, Morris C, et al. Output-sensitive complexity of multiobjective combinatorial optimization. Journal of Multi-Criteria Decision Analysis. 2017;24:25–36.

[18] Bökler F. Output-sensitive complexity of multiobjective combinatorial optimization with an application to the multiobjective shortest path problem [dissertation]. TU Dortmund; 2018.

[19] Seipp F. On adjacency, cardinality, and partial dominance in discrete multiple ob- jective optimization [dissertation]. TU Kaiserslautern; 2013.

[20] Gorski J, Klamroth K, Ruzika S. Connectedness of efficient solutions in multiple objective combinatorial optimization. Journal of Optimization Theory and Applications. 2011; 150:475–497.

[21] Rendl F, Leclerc M. A multiply constrained matroid optimization problem. Discrete Mathematics. 1988/89;73:207–212.

[22] Brezovec C, Cornuéjols G, Glover F. A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. Mathematical Programming. 1988;42:471–487.

[23] Srinivas MA. Matroid optimization with generalized constraints. Discrete Applied Mathematics. 1995;63:161–174.

[24] Hamacher HW, Rendl F. Color constrained combinatorial optimization problems. Operations Research Letters. 1991;10:211–219.

[25] Climaco JCN, Captivo ME, Pascoal MMB. On the bicriterion - minimal cost/minimal label - spanning tree problem. European Journal of Operational Research. 2010;204:199–205.

[26] Chang R, Leu SJ. The minimum labeling spanning trees. Information Processing Letters. 1997;63(6):277–282.

[27] Gorski J, Ruzika S. On k-max optimization. Operations Research Letters. 2009;37(1):23–26.

[28] Gorski J. Multiple objective optimization and implications for single objective optimization. Shaker Verlag; 2010.

[29] Gabow HN, Tarjan RE. Efficient algorithms for a family of matroid intersection problems. Journal of Algorithms. 1984;5:80–131.

[30] Ehrgott M. Multicriteria optimization. Berlin, Heidelberg: Springer Verlag; 2005.

[31] Miettinen K. Nonlinear multiobjective optimization. Boston: Kluwer Academic Publishers; 1999.

[32] Brualdi RA. Comments on bases in dependence structures. Bulletin of the Australian Mathematical Society. 1969;1(2):161–167.

[33] Chankong V, Haimes YY. Multiobjective decision making: Theory and methodology. Elsevier Science Publishing, New York; 1983.

[34] Gusfield D. Matroid optimization with the interleaving of two ordered sets. Discrete Applied Mathematics. 1984;8(1):41–50.

[35] Hotz M. generatespanningtrees(a) ; 2016. Matlab implementation for MST computation, MATLAB Central File Exchange, downloaded on February 19, 2020; Available from: `https://www.mathworks.com/matlabcentral/fileexchange/53787-generatespanningtrees-a`.

[36] Knuth DE. The art of computer programming. Vol. 4A (Combinatorial Algorithms, Part 1). Boston: Pearson Education, Inc.; 2012.

[37] Fernandes I, Goldbarg E, Maia S, et al. Empirical study of exact algorithms for the multiobjective spanning tree. Computational Optimization and Applications. 2020;75:561–605.

[38] Schnepper T, Klamroth K, Puerto J, et al. A local analysis to determine all optimal solutions of p-k-max location problems on networks. Discrete Applied Mathematics. 2021; 296:217–234.

[39] Russell M. Laplacian Matrices of Graphs: A Survey. Linear Algebra and its Applications. 1994;197-198:143–176.

[40] Klamroth K, Wiecek M. Dynamic programming approaches to the multiple criteria knapsack problem. Naval Research Logistics. 2000;47:57–76.