Computation in Multicriteria Matroid Optimization

JESÚS A. DE LOERA †
University of California, Davis
DAVID C. HAWS †
University of California, Davis
JON LEE ‡
IBM T.J. Watson Research Center
ALLISON O'HAIR †
University of California, Davis

Motivated by recent work on algorithmic theory for nonlinear and multicriteria matroid optimization, we have developed algorithms and heuristics aimed at practical solution of large instances of some of these difficult problems. Our methods primarily use the local adjacency structure inherent in matroid polytopes to pivot to feasible solutions which may or may not be optimal. We also present a modified breadth-first-search heuristic that uses adjacency to enumerate a subset of feasible solutions. We present other heuristics, and provide computational evidence supporting our techniques. We implemented all of our algorithms in the software package MOCHA.

Categories and Subject Descriptors: G.2.1 [**Discrete Mathematics**]: Combinatorics—*Combinatorial Algorithms*; G.2.2 [**Mathematics of Computing**]: Discrete Mathematics—*Graph Theory*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Matroids, Matroid Optimization, Multicriteria Optimization, Local Search, Tabu Search, Non-linear combinatorial optimization, Multiobjective Optimization

1. INTRODUCTION

Let $\mathcal{M}$ be a matroid on the ground set $[n] := \{1, \ldots, n\}$ with set of bases $\mathcal{B}_{\mathcal{M}}$. Consider d vectors $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^n$ where each vector applies a weighting to the ground set $[n]$. That is, every $\mathbf{w}_i$ assigns a real-value to each element of $[n]$. We let $W \in \mathbb{R}^{d \times n}$ be the matrix with rows $\mathbf{w}_1, \ldots, \mathbf{w}_d$. We use the standard notation where $\mathbf{e}_i \in \mathbb{R}^n$ means the vector with one in the ith position and zeros in the rest. For each base $B \in \mathcal{B}_{\mathcal{M}} \subseteq 2^{[n]}$, we define the *incidence vector* of B as $\mathbf{e}(B) := \sum_{i \in B} \mathbf{e}_i \in \mathbb{R}^n$, and we let $\mathbf{e}(\mathcal{B}_{\mathcal{M}}) := \{\, \mathbf{e}(B) \mid B \in \mathcal{B}_{\mathcal{M}} \,\}$. Then $W\mathbf{e}(B)$ is the vector of evaluations of the base B under the different weightings $\mathbf{w}_1, \ldots, \mathbf{w}_d$, and $W\mathbf{e}(\mathcal{B}_{\mathcal{M}}) := \{\, W\mathbf{e}(B) \mid B \in \mathcal{B}_{\mathcal{M}} \,\} \subseteq \mathbb{R}^d$. We also define $\mathcal{P}_{\mathcal{M}} := conv(\, \mathbf{e}(B) \mid B \in \mathcal{B}_{\mathcal{M}} \,) \subseteq \mathbb{R}^n$, where *conv* stands for the convex hull, and where

† Department of Mathematics, University of California, Davis, CA 95616, USA
‡ IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
Authors from University of California, Davis were supported by NSF-DMS 0608785 and 0914107, NSF-VIGRE grant 0636297, and an IBM Open Collaborative Research Grant.

$vert(\mathcal{P}) = \{\text{vertices of } \mathcal{P}\}$ for a polytope $\mathcal{P}$. Finally $W\mathcal{P}_{\mathcal{M}} := \{ W\mathbf{x} \mid \mathbf{x} \in \mathcal{P}_{\mathcal{M}} \}$.

Matroids are undeniably one of the fundamental structures in combinatorial optimization (e.g., see [Schrijver 2003; Lee 2004]). In this paper, we consider techniques aimed at four generalizations of the classical single-criterion linear-objective matroid optimization problem.

Nonlinear Matroid Optimization: Given a matroid $\mathcal{M}$ on $[n]$ with set of bases $\mathcal{B}_{\mathcal{M}}$, $W \in \mathbb{R}^{d\times n}$, and a function $f : \mathbb{R}^d \to \mathbb{R}$, find a base $B \in \mathcal{B}_{\mathcal{M}}$ such that $f(W\mathbf{e}(B)) = \min(f(W\mathbf{e}(B')) \mid B' \in \mathcal{B})$.

The motivation for nonlinear matroid optimization is that the function f that we seek to optimize trades off the competing d linear objectives described by the rows of W. When $d = 1$ and f is the identity (or any monotone) function, then we have classical linear-objective matroid optimization which is solvable via the greedy method.

Two important special cases of nonlinear matroid optimization are as follows.

Convex Matroid Optimization: Given a matroid $\mathcal{M}$ on $[n]$ with set of bases $\mathcal{B}_{\mathcal{M}}$, $W \in \mathbb{R}^{d\times n}$ where d is fixed, and a convex function $f : \mathbb{R}^d \to \mathbb{R}$, find a base $B \in \mathcal{B}_{\mathcal{M}}$ such that $f(W\mathbf{e}(B)) = \max(f(W\mathbf{e}(B')) \mid B' \in \mathcal{B}_{\mathcal{M}})$. Similarly we consider the minimization problem $f(W\mathbf{e}(B)) = \min (f(W\mathbf{e}(B')) \mid B' \in \mathcal{B}_{\mathcal{M}})$

Min-Max Multi-criteria Matroid Optimization: Given a matroid $\mathcal{M}$ on $[n]$ with set of bases $\mathcal{B}_{\mathcal{M}}$, $W \in \mathbb{R}^{d\times n}$, find a base $B \in \mathcal{B}_{\mathcal{M}}$ such that

$$\max_{i=1\ldots d}((W\mathbf{e}(B))_i) = \min(\max_{i=1\ldots d}((W\mathbf{e}(B'))_i) \mid B' \in \mathcal{B}_{\mathcal{M}}).$$

Also, we investigate the following problem.

Pareto Multi-criteria Matroid Optimization: Given a matroid $\mathcal{M}$ on n-elements with set of bases $\mathcal{B}_{\mathcal{M}}$, $W \in \mathbb{R}^{d\times n}$, find a base $B \in \mathcal{B}_{\mathcal{M}}$ such that $B \in argmin_{Pareto}((W\mathbf{e}(B')) \mid B' \in \mathcal{B}_{\mathcal{M}})$.

Above, $\min_{Pareto}$ is understood in the sense of Pareto optimality for problems with multiple objective functions, that is, we adopt the convention that for vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, we have $\mathbf{a} \leq \mathbf{b}$ if and only if $a_i \leq b_i$ for all entries of the vectors. Further, we say that $\mathbf{a} < \mathbf{b}$ if $\mathbf{a} \leq \mathbf{b}$ and $\mathbf{a} \neq \mathbf{b}$. Given a set $S \subseteq \mathbb{R}^n$ we say $\mathbf{a} \in S$ is a Pareto optimum if there does not exist $\mathbf{b} \in S$ such that $\mathbf{b} < \mathbf{a}$. We note that an optimum of the min-max problem will be a Pareto optimum. This is easy to see because if $\mathbf{a} \leq \mathbf{b}$ then $\max_i(a_i) \leq \max_i(b_i)$. The Pareto multi-criteria matroid optimization problem has been studied by several authors before. For example, Ehrgott [Ehrgott 1996] investigated two optimization problems for matroids with multiple objective functions, and he pioneered a study of Pareto bases via the base-exchange property of matroids. See [Ehrgott and Gandibleux 2000] for a detailed introduction to some aspects of multicriteria combinatorial optimization.

The matroid optimization problems we consider here have wide applicability. For

example, in [Berstein et al. 2008] the authors consider the "minimum aberration model fitting problem" in statistics, which can be reduced to a nonlinear matroid optimization problem. Multi-criteria problems concerning minimum spanning trees of graphs are common in applications (see [Ehrgott and Gandibleux 2000; Knowles and Corne 2002] and references therein).

As a concrete example related to spanning trees, consider a graph G with three linear criteria on the edges described as the three rows of a matrix W.

1. the first row of W encodes the *fixed installation cost* of each edge of G;
2. the second row of W encodes the *monthly operating cost* of each edge of G;
3. assuming that the edge j fails independently with probability $1 - p_j$, then by having the $\log p_j$ as the third row of W (possibly scaled and rounded suitably), $\sum_{j \in T} \log p_j$ captures the *reliability* of the spanning tree T of G.

It can be difficult for a decision maker to balance these three competing objectives in selecting a best spanning tree. There are many issues to consider such as the time horizon, repairability, fault tolerance, etc. These issues can be built into a concrete function f, for example a weighted norm, or can be thought of as determining a black-box f.

Unfortunately, although useful, the problems we are considering are also very difficult in general. Multicriteria matroid optimization is generally NP-complete [Ehrgott 1996]. The Min-Max optimization problem includes the NP-complete partition problem (see [Garey and Johnson 1979]), certain multi-processor scheduling problems (see [Graham et al. 1979]), and specific worst-case stochastic optimization problems (see [Warburton 1985]).

Nevertheless, recently there has been considerable progress on algorithmic and complexity theory for nonlinear matroid optimization. In particular, algorithms with polynomial worst-case complexity bounds have been developed under nice assumptions on W, f and d. For instance, it has been shown that although multicriteria matroid optimization is NP-complete in general, it is polynomial-time solvable under certain restrictions on W and with fixed d [Berstein et al. 2009]. We refer the reader to the recent series of papers on nonlinear matroid optimization [Berstein et al. 2009; Berstein and Onn 2008; Onn 2003; Berstein et al. 2008] which serve as background for the algorithms and strategies implemented here. The present paper reports on some of the current computational possibilities by comparing various heuristics and algorithms.

In Section 2, we present several heuristics and algorithms for nonlinear matroid optimization and multicriteria matroid optimization problems. At a glance, Table I shows the four problems described above indicating which of our algorithms or heuristics presented in this paper concern them.

Section 2 begins with a description of our implementation of a speed-up of the algorithm proposed in [Onn 2003] for convex maximization and two primal heuristics, Heuristic 1 (Local Search) and Heuristic 2 (Tabu Search), for convex minimization. Using either Local Search or Tabu Search, we present Heuristic 3 (Pivot Test) that is aimed at finding a large subset of $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$. We also present Algorithm 4 (Projected Boundary) that finds all vertices of $W\mathcal{P}_{\mathcal{M}}$. Combining Algorithm 4 (Projected Boundary) and Heuristic 3 (Pivot Test) we derive Heuristic 5 (Boundary and Triangular Region Pareto Test) which finds approximate solutions to the

Nonlinear Matroid Optimization	Heuristic 6 (DFBFS) Heuristic 3 (Pivot Test)
Convex Matroid Optimization	Heuristic 1 (Local Search) Heuristic 2 (Tabu Search) Heuristic 6 (DFBFS) Heuristic 3 (Pivot Test)
Min-Max Matroid Optimization	Heuristic 6 (DFBFS) Heuristic 3 (Pivot Test) Heuristic 5 (Boundary and Triangular Region Pareto Test)
Pareto Multi-criteria Matroid Optimization	Heuristic 6 (DFBFS) Heuristic 3 (Pivot Test) Heuristic 5 (Boundary and Triangular Region Pareto Test)

Table I. Our four optimization problems and our algorithms and heuristics applicable to them.

Pareto optimization problem or Min-Max problem. Finally, we present a modified breadth-first-search heuristic, Heuristic 6 (DFBFS), that is also aimed at finding a large subset of $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$. We especially note that Heuristic 3 (Pivot Test), Heuristic 6 (DFBFS), and Heuristic 5 (Boundary and Triangular Region Pareto Test) do not explicitly optimize a function, but instead find a subset of the feasible points, the projected bases $W\mathbf{e}(\mathcal{B})$. From there, one can easily scan through the subset to determine the optimum (with respect to that subset) using any objective.

We describe our software MOCHA in Section 3. In Section 4, we present our computational results followed by a discussion.

2. DESCRIPTION OF THE ALGORITHMS AND HEURISTICS

Throughout the paper we refer to a piece pseudocode as an algorithm if it is guaranteed to output an optimal solution, otherwise we refer to the pseudocode as a heuristic. Before we start our description of the algorithms and heuristics, we remark that all of them rely on the geometry of the 1-skeleton graph of the matroid polytope (see [Schrijver 2003]). The edges correspond to base exchanges. These graphs have a lot of special structure. For example, these graphs are always Hamiltonian (see [Holtzmann and Harary 1972]), and it is known that each two-dimensional face of every matroid polytope is either a triangle or a quadrilateral. This implies that the graph of the 1-skeleton of $W\mathcal{P}_{\mathcal{M}}$ is quite dense and easy to traverse (see [Borovik et al. 2007]).

Given $\mathbf{v}_1, \ldots, \mathbf{v}_k$, we define $cone(\mathbf{v}_1, \ldots, \mathbf{v}_k) := \left\{ \sum_{i=1}^{k} \lambda_i \mathbf{v}_i \mid \lambda_i \geq 0 \right\}$. If $\mathcal{P} \subseteq \mathbb{R}^n$ is a polytope, we say two vertices $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{P}$ are *adjacent* if they are contained in a one-dimensional face. Next we give a vital but elementary proposition necessary for our algorithms.

PROPOSITION 1. *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a polytope, $\mathbf{v}_1, \mathbf{v}_2$ vertices of $\mathcal{P}$, and $W \in \mathbb{R}^{d\times n}$. Then there exists $\lambda_v \geq 0$ such that*

$$W\mathbf{v}_2 = W\mathbf{v}_1 + \sum_{\mathbf{v} \text{ adjacent to } \mathbf{v}_1} \lambda_{\mathbf{v}}(W\mathbf{v} - W\mathbf{v}_1).$$

PROOF. Let $\widetilde{\mathbf{v}}_\mathbf{1}, \ldots, \widetilde{\mathbf{v}}_\mathbf{k}$ be adjacent to $\mathbf{v}_1$. It follows from convexity of $\mathcal{P}$ that there exist $\lambda_1, \ldots, \lambda_k \geq 0$ such that

$$\begin{aligned} \mathbf{v}_2 &= \lambda_1(\widetilde{\mathbf{v}}_\mathbf{1} - \mathbf{v}_1) + \cdots + \lambda_k(\widetilde{\mathbf{v}}_\mathbf{k} - \mathbf{v}_1) + \mathbf{v}_1 \\ \implies W\mathbf{v}_2 &= \lambda_1 W(\widetilde{\mathbf{v}}_\mathbf{1} - \mathbf{v}_1) + \cdots + \lambda_k W(\widetilde{\mathbf{v}}_\mathbf{k} - \mathbf{v}_1) + W\mathbf{v}_1. \end{aligned}$$

□

Let $W\mathcal{P} := \{\, Wx \mid x \in \mathcal{P} \,\}$. Proposition 1 implies the following lemma.

LEMMA 1. *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a polytope, $\mathbf{v}$ a vertex of $\mathcal{P}$, and $W \in \mathbb{R}^{d\times n}$. Then $W\mathbf{v} \in relint(W\mathcal{P})$ if and only if $cone(W\widetilde{\mathbf{v}} - W\mathbf{v} \mid \widetilde{\mathbf{v}}$ adjacent to $\mathbf{v}) = \mathbb{R}^d$.*

Next we state a vital lemma which fully characterizes adjacency on $\mathcal{P}_\mathcal{M}$.

LEMMA 2 (See [GELFAND ET AL. 1987]). *Let $\mathcal{M}$ be a matroid on the ground set $[n]$. Two vertices $\mathbf{e}(B_1), \mathbf{e}(B_2)$ of $\mathcal{P}_\mathcal{M}$ are adjacent if and only if $\mathbf{e}(B_1) - \mathbf{e}(B_2) = \mathbf{e}_i - \mathbf{e}_j$ for some i, j.*

Define $Adj(B) := \{\, \widetilde{B} \in \mathcal{B}_\mathcal{M} \mid \mathbf{e}(\widetilde{B}) - \mathbf{e}(B) = \mathbf{e}_i - \mathbf{e}_j \,\}$. Using Lemma 2 we get the following corollaries of Proposition 1 and Lemma 1.

COROLLARY 1. *Let $\mathcal{M}$ be a matroid on the ground set $[n]$, $W \in \mathbb{R}^{d\times n}$, $B, \widetilde{B} \in \mathcal{B}_\mathcal{M}$. Then*

$$W\mathbf{e}(\widetilde{B}) = W\mathbf{e}(B) + \sum_{B' \in Adj(B)} \lambda_{B'}(W\mathbf{e}(B') - W\mathbf{e}(B))$$

for some $\lambda_{B'} \geq 0$.

COROLLARY 2. *Let $\mathcal{M}$ be a matroid on the ground set $[n]$, $W \in \mathbb{R}^{d\times n}$, and $B \in \mathcal{B}_M$. Then $W\mathbf{e}(B) \in relint(W\mathcal{P}(\mathcal{M}))$ if and only if cone $(\, W\mathbf{e}(B') - W\mathbf{e}(B) \mid B' \in Adj(B) \,) = \mathbb{R}^d$.*

In order to present a general situation for which some of our heuristics are efficient, we consider the "generalized unary encoding" of the weight matrix $W \in \mathbb{R}^{d\times n}$ introduced in [Berstein et al. 2009]. We consider weights $W_{i,j}$ of the form

$$W_{i,j} = \sum_{k=1}^{p} W_{i,j}^k a_k \,,$$

where $p \geq 1$ is a fixed integer, $a = (a_1, \ldots, a_p)$ is a p-tuple of distinct positive integers a_k that are binary encoded, and the integers $W_{i,j}^k$ (unrestricted in sign) are unary encoded. For each k, we can organize the $W_{i,j}^k$ into a matrix $W^k := ((W_{i,j}^k))$. Then we have $W := ((W_{i,j})) = \sum_{k=1}^p a_k W^k$. We say that W has a *generalized unary encoding over a*. The *length* of the generalized unary encoded $W\,(= \sum_{k=1}^p a_k W^k)$ is the sum of the lengths of the binary encoded a and the lengths of the unary encoded W^k.

The generalized unary encoding includes important special cases:

1. **Unary-encoded weights:** With $p = 1$ and $a_1 = 1$, we get the ordinary model of unary-encoded $W = W^1$.

2. **Binary-encoded $\{a_1, \ldots, a_p\}$-valued weights:** With $W^k \geq 0$ for all k, and $\sum_{k=1}^{p} W^k_{i,j} \leq 1$, for all i, j, we get the case of all $W_{i,j}$ in the set $\{a_1, \ldots, a_p\}$ having binary-encoded elements.

The following lemma is a variation on facts pointed out in [Berstein et al. 2009]. With W having a generalized unary encoding, it is simple to bound the size of $\#W\mathbf{e}(\mathcal{B}_\mathcal{M})$.

LEMMA 3. *Let d and p be fixed, $\mathcal{M}$ a matroid on n elements, and a generalized unary encoded matrix $W := ((W_{i,j})) = \sum_{k=1}^{p} a_k W^k \in \mathbb{Z}^{d\times n}$. Then $\#W\mathbf{e}(\mathcal{B}_\mathcal{M})$ is polynomially bounded in n and the lengths of the unary encodings of the W^k, for $1 \leq k \leq p$.*

PROOF. Let $\omega := \max W^k_{i,j}$. We have

$$W\mathbf{e}(\mathcal{B}_\mathcal{M}) = \{\, W\mathbf{e}(B) \mid B \in \mathcal{B}_\mathcal{M} \,\} \subseteq \left\{ \sum_{k=1}^{p} \lambda_k a_k \mid \lambda \in \{0, \pm 1, \ldots, \pm\omega \cdot rank(\mathcal{M})\}^p, \right\}^d$$

Therefore, $\#W\mathbf{e}(\mathcal{B}_\mathcal{M}) \leq (2\omega \cdot rank(\mathcal{M}) + 1)^{pd}$. □

A fortiori, $\#W\mathbf{e}(\mathcal{B}_\mathcal{M})$ is bounded by the length of a *generalized unary encoding* of W. This is useful for us, but some of our heuristics depend on the number of integer points in the smallest rectangular region containing $W\mathbf{e}(\mathcal{B}_\mathcal{M})$ being polynomially bounded in order to prove efficiency. In such situations, we can only make the claim that the number of integer points in the smallest rectangular region containing $W\mathbf{e}(\mathcal{B}_\mathcal{M})$ is polynomially bounded, when W has a *unary encoding*, as the number of such points depends exponentially on the lengths of the binary encodings of the a_k, $1 \leq k \leq p$.

2.1 Local and Tabu Search

First we present Heuristic 1 that starts at any base of $\mathcal{M}$ and proceeds by pivoting to its neighbors on $\mathcal{P}_\mathcal{M}$ as long as the pivot decreases the value of the given function f through the weighting W.

In the majority of our experiments f is concave or convex. In the case where f is concave, there will be a minimum on the boundary of $W\mathcal{P}_\mathcal{M}$. If f is convex, then the minima may be in the interior of $W\mathcal{P}_\mathcal{M}$. We emphasize that Heuristic 1 is not guaranteed to terminate at an optimum.

Heuristic 1 (Local Search)

$LS(\mathcal{M}, W, f, B)$
Input: Matroid $\mathcal{M}$ on n elements, $B \in \mathcal{B}_\mathcal{M}$, $W \in \mathbb{R}^{d\times n}$, $f : \mathbb{R}^d \to \mathbb{R}$.
Output: A base $B' \in \mathcal{B}_\mathcal{M}$ such that $f(W\mathbf{e}(\widetilde{B})) \geq f(W\mathbf{e}(B'))\ \forall\, \widetilde{B} \in Adj(B')$
begin
  $B' := B$
  repeat
    $OLDB' := B'$
    if $f(W\mathbf{e}(\widetilde{B})) < f(W\mathbf{e}(B'))$ *for some* $\widetilde{B} \in Adj(B')$ **then**
      $B' := \widetilde{B}$
    end
  until $B' = OLDB'$;
  return B'
end

If the objective function f is linear, then Heuristic 1 follows non-degenerate steps of the simplex method for some pivot rule. The following is a well-known result concerning an efficient algorithm for finding a minimum-weight base of a matroid.

LEMMA 4. *Let $\mathcal{M}$ be a matroid on $[n]$, $B \in \mathcal{B}_\mathcal{M}$, $W \in \mathbb{R}^{d\times n}$, and $f(\mathbf{x}) = \mathbf{c}\cdot\mathbf{x}$, where $c \in \mathbb{R}^d$. $LS(\mathcal{M}, W, g, B)$ terminates at $\widehat{B}$ where $f(W\mathbf{e}(\widehat{B})) = \min\{\, f(W\mathbf{e}_B) \mid B \in \mathcal{B}_\mathcal{M}\,\}$.*

Lemma 4 is used by MOCHA in the implementation of our heuristics. For example, we can easily compute a tight rectangular region containing $W\mathcal{P}$. We also use Lemma 4 to obtain an integer point on the boundary of $W\mathbf{e}(\mathcal{B}_\mathcal{M})$, a prerequisite for some of our heuristics.

In Section 4, we will show the practical limitations of Heuristic 1, and we give a theoretical bound on the running time of the previous algorithm given restrictions on W and d.

LEMMA 5. *Let d and p be fixed, $\mathcal{M}$ a matroid on $[n]$ given by an independence oracle, a generalized unary encoded matrix $W := ((W_{i,j})) = \sum_{k=1}^{p} a_k W^k \in \mathbb{Z}^{d\times n}$, and $f : \mathbb{R}^d \to \mathbb{R}$ given by comparison oracle. Heuristic 1 takes in as input M, a base $B \in \mathcal{B}_\mathcal{M}$, W, f and in time polynomial in n and the length of the generalized unary encoding of W outputs a base $B' \in \mathcal{B}_\mathcal{M}$ such that $f(W\mathbf{e}(\widetilde{B})) \geq f(W\mathbf{e}(B'))\ \forall\, \widetilde{B} \in Adj(B')$.*

PROOF. For any $B \in \mathcal{B}_\mathcal{M}$ we can enumerate $Adj(B)$ in polynomial time via an oracle for $\mathcal{M}$ by testing which of $\{\, B\backslash\{i\}\cup\{j\} \mid i \in B,\ j \in [n]\backslash B\,\}$ are bases. Thus, we can pivot in polynomial time in the input. Because Heuristic 1 always pivots to an adjacent base that is smaller, through f and W, than the current base, each point of $W\mathbf{e}(\mathcal{B}_\mathcal{M})$ will be visited at most once. By Lemma 3, $\#W\mathbf{e}(\mathcal{B}_\mathcal{M}) \cap \mathbb{Z}^d$ is polynomially bounded in the input. Also, the arithmetic in Heuristic 1 is polynomial in n and the binary encoding of a_k. □

Tabu search was first presented in [Glover 1986] and has been widely used in combinatorial optimization. It begins at base $B \in \mathcal{B}_\mathcal{M}$ and pivots to an adjacent base B' if it is smaller than some other adjacent base, through f and the weighting

W. This differs from Heuristic 1 in that we allow pivots that are not necessarily smaller than the current base, through f and the weighting W. We record the smallest value $f(W\mathbf{e}(B))$ encountered and terminate after L pivots with no update to the minimum encountered. As for Heuristic 1, for the majority of our experiments f is concave or convex. We emphasize that Heuristic 2 is not guaranteed to terminate at the optimum.

Heuristic 2 (Tabu Search)

$TS(\mathcal{M}, B, W, L)$
Input: Matroid $\mathcal{M}$ on $[n]$, $B \in \mathcal{B}_{\mathcal{M}}$, $W \in \mathbb{R}^{d\times n}$, $L \in \mathbb{N}$.
Output: $B' \in \mathcal{B}_{\mathcal{M}}$ s.t. $f(W\mathbf{e}(B')) \leq f(W\mathbf{e}(\widetilde{B}))$ for all bases and neighbors $\widetilde{B}$ for L pivots.
begin
  $B' := B$
  VIS $:= \{B\}$
  CURMIN $:= f(W\mathbf{e}(B'))$
  repeat
    if $f(W\mathbf{e}(\widetilde{B})) < f(W\mathbf{e}(\widehat{B}))$ *for some* $\widetilde{B}, \widehat{B} \in Adj(B') \setminus$ *VIS* **then**
      VIS $:=$ VIS $\cup \{\widetilde{B}\}$
      $B' := \widetilde{B}$
    end
    if $f(W\mathbf{e}(B')) <$ *CURMIN* **then**
      CURMIN $:= f(W\mathbf{e}(B'))$
    end
    return B'
  until L *pivots with* CURMIN *unchanged* ;
end

Heuristic 2 can also be modified such that we mark $W\mathbf{e}(B)$ as visited, not B, and only pivot to B such that $W\mathbf{e}(B)$ is unvisited. In this way, we could also prove that the running time of the modified algorithm is polynomial in the input, given the same restrictions for Lemma 5. Though, restricting to visiting new $W\mathbf{e}(B)$ could limit the possibilities of quickly converging to the optimum though.

We observed through our computational experiments that Heuristic 1 and Heuristic 2 work very well when optimizing concave or convex functions f over projected matroid polytopes as we will show in Section 4. However, when f is arbitrary then the previous algorithms may not perform well.

2.2 Listing Heuristics

In the remainder of the present section, we describe algorithms and heuristics aimed at tackling problems with an arbitrary function f or using Pareto or minmax optimization. For maximum generality, the following algorithms do not explicitly evaluate f but are presented as *listing algorithms*. Note that if we have all of the projected matroid bases $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$, then it is simple to extract the Pareto optima through a straight forward pairwise comparison. Likewise for optimizing f or with a min-max objective, we simply evaluate over all projected bases found as we go through them. This is the same methodology as [Berstein et al. 2009] where the

authors prove efficient deterministic algorithms, given sufficient conditions on the input, to list all projected bases. Efficiency follows due to assumptions on the input, i.e. d is fixed and W is a generalized unary encoded matrix.

In [Berstein et al. 2009], the authors used matroid intersections to solve the problem: Given $\mathbf{x} \in \mathbb{R}^d$ find $B \in \mathcal{B}_\mathcal{M}$ such that $W\mathbf{e}(B) = \mathbf{x}$ if such a B exists. Guided by the success of our previous heuristics, we had the idea of using either Heuristic 2 or Heuristic 1 to find a $B \in \mathcal{B}_\mathcal{M}$ such that $W\mathbf{e}(B) = \mathbf{x}$. The novelty of our heuristic is that we use convex optimization as a subroutine to solve nonlinear and Pareto optimization problems.

Given an $\mathbf{x}' \in \mathbb{Z}^d$, Heuristic 3 forms a convex function $f_{\mathbf{x}'}$ that is minimized at $\mathbf{x}'$ and $f_{\mathbf{x}'}(\mathbf{x}') = 0$. Our heuristic calls Heuristic 1 or Heuristic 2 with the function $f_{\mathbf{x}'}$ and if it returns a base $B \in \mathcal{B}_\mathcal{M}$ such that $f_{\mathbf{x}'}(W\mathbf{e}(B)) = 0$ then $W\mathbf{e}(B) = x$.

Heuristic 3 (Pivot Test)

$PT(\mathcal{M}, B, W, t, S)$
Input: Matroid $\mathcal{M}$ on n elements, $B \in \mathcal{B}_\mathcal{M}$, $W \in \mathbb{R}^{d\times n}$, $t \in \mathbb{N}$, finite set $S \subseteq \mathbb{Z}^d$.
Output: $\mathtt{PT} \subseteq \mathcal{B}_\mathcal{M}$ such that $\{\, W\mathbf{e}(B) \mid B \in \mathtt{PT} \,\} \subseteq S$.
begin
  $\mathtt{PT} := \emptyset$
  for *each* $\mathbf{x}' \in S$ **do**
    $f_{\mathbf{x}'}(\mathbf{x}) := \sum_{i=1}^{d} (x_i - x'_i)^2$
    for $1, \ldots, t$ **do**
      $B' :=$ random base of $\mathcal{M}$
      $B := LS(\mathcal{M}, W, f_{x'}, B')$ *(Or use Heuristic 2)*
      if $f_{\mathbf{x}'}(W\mathbf{e}(B)) = 0$ **then**
        $\mathtt{PT} := \mathtt{PT} \cup \{B\}$
        Break for loop
      end
    end
  end
  return *PT*
end

Lemma 6. *Let d and p be fixed, $\mathcal{M}$ a matroid on $[n]$ given by an independence oracle, a generalized unary encoded matrix $W := ((W_{i,j})) = \sum_{k=1}^{p} a_k W^k \in \mathbb{Z}^{d\times n}$, and a finite set $S \subset \mathbb{Z}^d$. Heuristic 3 takes in as input M, W, S and in time polynomial in n, t, the size of S, and the length of the generalized unary encoding of W outputs $\mathtt{PT} \subseteq \mathcal{B}_\mathcal{M}$ such that $\{\, W\mathbf{e}(B) \mid B \in \mathtt{PT}\} \subseteq S$.*

Proof. This follows from the fact that Heuristic 1 and Heuristic 2 are polynomial under these assumptions. Moreover, we call Heuristic 1 or Heuristic 2 at most $t|S|$ times. □

Typically we wish to use Heuristic 3 to try to enumerate a subset of $W\mathcal{P}_\mathcal{M}$. Thus, if we further assume W is unary encoded, then the smallest box containing $W\mathcal{P}_\mathcal{M}$ is polynomial which we use as our input S in Heuristic 3.

For general nonlinear multi-criteria matroid optimization problems, when the function f we are minimizing is concave, then some optimum will be a vertex of $W\mathcal{P}_{\mathcal{M}}$. Thus, in such a case, it is sufficient to enumerate the vertices of $W\mathcal{P}_{\mathcal{M}}$ and test f over those points. Furthermore, the vertices are also Pareto optimal, but in general, the vertices are not all of the Pareto optima. Later we will give an algorithm that uses the vertices of $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$ to facilitate finding other Pareto optima. In [Okamoto and Uno 2007] the authors develop an output-sensitive polynomial time algorithm using the well-known Avis-Fukuda's reverse search algorithm which outputs all bases which project to the vertices of $W\mathcal{P}_{\mathcal{M}}$. We will prove that the following algorithm will enumerate all vertices of $W\mathcal{P}_{\mathcal{M}}$ and possibly other integral points on the boundary of $W\mathcal{P}_{\mathcal{M}}$. The following algorithm not only finds all the vertices, but also finds a base which projects to each vertex. Algorithm 4 starts at a base $B \in \mathcal{B}_{\mathcal{M}}$ such that $W\mathbf{e}(B)$ is on the boundary of $W\mathcal{P}_{\mathcal{M}}$. If $B' \in Adj(B)$, $W\mathbf{e}(B')$ is a newly seen point and is on the boundary of $W\mathcal{P}_{\mathcal{M}}$, then we record B' and $W\mathbf{e}(B')$, pivot to B', and continue.

Algorithm 4 (Projected Boundary)

Input: Matroid $\mathcal{M}$ on n elements, $W \in \mathbb{R}^{d\times n}$, $B \in \mathcal{B}_{\mathcal{M}}$ such that $W\mathbf{e}(B)$ is an extreme point of $W\mathcal{P}_{\mathcal{M}}$.
Output: $\texttt{CH} \subseteq \mathcal{B}_{\mathcal{M}}$ such that $\{\, W\mathbf{e}(B) \mid B \in \texttt{CH} \,\} \supseteq vert(W\mathcal{P}_{\mathcal{M}})$.
begin
  $\texttt{CH} := \{B'\}$
  Mark B' as unvisited in CH
  $\texttt{PB} := \{W\mathbf{e}(B')\}$
  repeat
    $B' :=$ first unvisited base in CH
    Mark B' as visited.
    for $\widehat{B} \in Adj(B')$ **do**
      if $W\mathbf{e}(\widehat{B})$ *is an extreme point of* $W\mathcal{P}_{\mathcal{M}}$ **then**
        if $W\mathbf{e}(\widehat{B}) \notin$ *PB* **then**
          $\texttt{CH} := \texttt{CH} \cup \{\widehat{B}\}$ Mark $\widehat{B}$ as unvisited. $\texttt{PB} := \texttt{PB} \cup \{W\mathbf{e}(\widehat{B})\}$
        end
      end
    end
  until *PB unchanged.* ;
  return *CH*
end

In Section 4 we will show that Algorithm 4 works well. Empirical observations seem to suggest this because the projected bases are highly clustered and with much fewer projected bases near the boundary of $W\mathcal{P}_{\mathcal{M}}$. To see how large the pre-image of $\mathbf{x}$ via W can be, for $\mathbf{x} \in \mathbb{R}^d$, see Figure 1 in [Gunnels et al. 2008]. The following lemma proves that the output of Algorithm 4 will contain all vertices of the convex hull of $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$.

LEMMA 7. *Let $\mathcal{M}$ be a matroid on $[n]$, $W \in \mathbb{R}^{d\times n}$, $B \in \mathcal{B}_{\mathcal{M}}$ such that $W\mathbf{e}(B)$ is an extreme point of $W\mathcal{P}_{\mathcal{M}}$. Then*

$$vert(W\mathcal{P}_{\mathcal{M}}) \subseteq W(\, PB(\mathcal{M}, W, B)\,) \subseteq (W\mathbf{e}(\mathcal{B}_{\mathcal{M}}) \cap \partial W\mathcal{P}_{\mathcal{M}}).$$

PROOF. Let $B', \widehat{B} \in \mathcal{B}_{\mathcal{M}}$ where $W\mathbf{e}(B')$ is an extreme point and $W\mathbf{e}(\widehat{B})$ is a vertex of $W\mathcal{P}_{\mathcal{M}}$. Moreover let $W\mathbf{e}(B')$ and $W\mathbf{e}(\widehat{B})$ be on a 1-face F of $W\mathcal{P}_{\mathcal{M}}$. By Corollary 1 there exists $B'' \in Adj(B')$ such that either $W\mathbf{e}(B'') = W\mathbf{e}(\widehat{B})$ or $W\mathbf{e}(B'')$ is a positive convex combination of $W\mathbf{e}(B')$ and $W\mathbf{e}(\widehat{B})$, i.e. $W\mathbf{e}(B'') \in F$. Thus $W\mathbf{e}(B'')$ is extreme and Algorithm 4 can always pivot towards every vertex of $W\mathcal{P}_{\mathcal{M}}$ and will output all vertices of $W\mathcal{P}_{\mathcal{M}}$. □

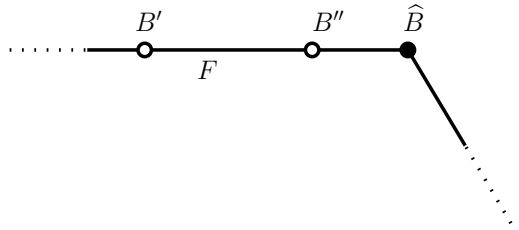

Fig. 1. A pivot from B' to B'' towards $\widehat{B}$ in Algorithm 4.

In [Onn 2003], it was shown that the number of vertices of $W\mathcal{P}_{\mathcal{M}}$ is polynomially bounded when then number of weightings d is fixed (see Proposition 2.1 and Lemma 2.3). Moreover the author showed that the vertices of $W\mathcal{P}_{\mathcal{M}}$ can be found in polynomial time. A similar result can be found in [Berstein et al. 2009]. Algorithm 4 can (and does in practice) pick up non-vertex integral boundary points. Hence, a bound on the number of vertices is not sufficient to guarantee efficiency of the algorithm.

LEMMA 8. *Let d and p be fixed, $\mathcal{M}$ a matroid on $[n]$ given by an independence oracle, and a generalized unary encoded matrix $W := ((W_{i,j})) = \sum_{k=1}^{p} a_k W^k \in \mathbb{Z}^{d\times n}$. Algorithm 4 takes in as input M, W, $B \in \mathcal{B}_{\mathcal{M}}$ such that $W\mathbf{e}(B)$ is an extreme point of $W\mathcal{P}_{\mathcal{M}}$, and runs in time polynomial in n, and the length of the generalized unary encoding of W outputs $\mathtt{CH} \subseteq \mathcal{B}_{\mathcal{M}}$ such that $\{\, W\mathbf{e}(B) \mid B \in \mathtt{CH}\} \supseteq vert(W\mathcal{P}_{\mathcal{M}})$.*

PROOF. Because $\#W\mathbf{e}(\mathcal{B}_{\mathcal{M}}) \cap \mathbb{Z}^d$ is polynomially bounded and $\#Adj(B) \leq n^2$ we only have to show that deciding if $W\mathbf{e}(\widetilde{B})$ is an extreme point of $W\mathcal{P}_{\mathcal{M}}$ is polynomial in the input. This is equivalent to deciding if $cone\{\, W\mathbf{e}(B) - W\mathbf{e}(\widetilde{B}) \mid B \in Adj(\widetilde{B})\,\}$ is pointed. We can decide this by solving the feasibility problem:

$$\Big\{\, \mathbf{x}, -\mathbf{x} \in cone\{\, W\mathbf{e}(B) - W\mathbf{e}(\widetilde{B}) \mid B \in Adj(\widetilde{B})\,\} \mid \mathbf{x} > 0 \Big\}$$

□

We will show in Section 4 the effectiveness of Heuristic 3 (PT) in enumerating all projected bases. When finding Pareto optima though it is not necessary to enumerate *all* projected bases. The following heuristic combines Algorithm 4 and Heuristic 3 to find a set of points which could be Pareto optima of $W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$, and

hence could be a min-max optimum of $W\mathbf{e}(\mathcal{B}_\mathcal{M})$. The main idea is that if we have the boundary points, then the regions where other Pareto optima lay is determined by the convex hulls of points derived from a triangulation of the boundary.

Heuristic 5 (Boundary and Triangular Region Pareto Test)

$BTRPT(\mathcal{M}, W, t)$
Input: Matroid $\mathcal{M}$ on n elements, $W \in \mathbb{R}^{d\times n}$, $t \in \mathbb{N}$.
Output: PO $\subseteq \mathcal{B}_\mathcal{M}$, Pareto optima of Heuristic 4 (PB) and Heuristic 3 (PT) on certain convex regions.
begin
  PO := $PB(\mathcal{M}, W)$
  for $B \in$ *PO* **do**
    for $B' \in$ *PO* $\setminus B$ **do**
      if $W\mathbf{e}(B') \geq_{Pareto} W\mathbf{e}(B)$ **then**
        PO := PO $\setminus B'$
      end
    end
  end
  PPO := WPO projected on the hyperplane $H := \{ x \mid \sum x_i = 1 \}$.
  Triangulate PPO.
  for *each facet* F *of* *PPO* **do**
    TRIPOINTS := {Bases in PO corresponding to vertices of F}
    for $i = 1, \ldots d$ **do**
      $X_i^{\max} := \max((W\mathbf{e}(B))_i \mid B \in$ TRIPOINTS $)$
    end
    PO := $PT(\mathcal{M}, B, W, t, conv($TRIPOINTS $\cup\ X^{\max}))\ \cup$ PO
  end
  for $B \in$ *PO* **do**
    for $B' \in$ *PO* $\setminus B$ **do**
      if $W\mathbf{e}(B') \geq_{Pareto} W\mathbf{e}(B)$ **then**
        PO := PO $\setminus B'$
      end
    end
  end
  return *PO*
end

Except for the boundary points, Heuristic 5 is not guaranteed to return points that are Pareto optima because Heuristic 3 (PT) may miss some projected bases in the test regions.

Motivated by the need of optimizing any function f and exploring *all* Pareto or minmax optima, we devised a variation of the breadth-first search algorithm that would limit the search to some depth of the tree. Our intuition is that the graph of base exchanges is highly connected, thus small depth is all that is necessary for listing a large proportion, as we will see in Section 4, of all projected bases:

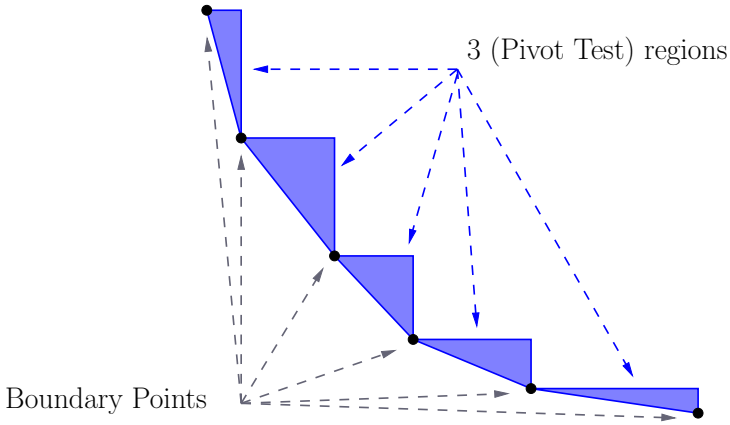

Fig. 2. Pareto optima boundary points and blue regions where other Pareto optima may lay.

Heuristic 6 (Different Fiber BFS)

$DFBFS(\mathcal{M}, W, B, d, l, \texttt{PB})$
Input: Matroid $\mathcal{M}$ on n elements, $W \in \mathbb{R}^{d\times n}$, $B \in \mathcal{B}_\mathcal{M}$, $d, l \in \mathbb{N}$, PB $\subseteq W\mathbf{e}(\mathcal{B}_\mathcal{M})$.
Output: DFBFS $\subseteq \{\, W\mathbf{e}(B) \mid B \in \mathcal{B}_\mathcal{M} \,\}$, an approximation of all projected bases $WP_\mathcal{M}$
begin
  if $l \geq d$ **then**
    return $\emptyset$
  end
  PB := PB $\cup \{W\mathbf{e}(B)\}$
  PTT := $\emptyset$
  for *each* $B' \in Adj(B)$ **do**
    if $W\mathbf{e}(B') \notin$ *PB* **then**
      if $W\mathbf{e}(B') \neq W\mathbf{e}(B)$ **then**
        PB := PB $\cup W\mathbf{e}(B')$
        PTT := PTT $\cup B'$
      end
    end
  end
  for *each* $B' \in$ *PTT* **do**
    PB := PB $\cup DFBFS(\mathcal{M}, W, B', d, l+1, \texttt{PB})$.
  end
  return *PB*
end

Heuristic 6 begins at a base $B \in \mathcal{B}_\mathcal{M}$. It adds the projected point $W\mathbf{e}(B)$ to the set PB, short for projected bases. It then enumerates all adjacent bases $Adj(B)$ of B. If a neighbor B''s projected point $W\mathbf{e}(B') \notin$ PB and $W\mathbf{e}(B) \neq W\mathbf{e}_{B'}$, then we add $W\mathbf{e}(B')$ to PB and recursively call $DFBFS$ on B'. We allow a parameter d

which determines the recursive depth allowed. Heuristic 6 takes its name, different fiber breadth-first-search, from the fact that we do not allow pivots that evaluate the same under the weighting W. As a small consequence, this guarantees that the number of times $DFBFS$ is called is bounded by $\#W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$ a useful fact if the input is bounded as in Lemma 3.

3. SOFTWARE IMPLEMENTATION

Our heuristics are implemented in C++ and take advantage of the object-oriented paradigm. The software MOCHA[1] [De Loera et al. 2009] reads in either a vectorial matroid, represented by an $m \times n$ floating-point matrix or a graphic matroid, represented by an $n \times n$ adjacency matrix. The weightings are read in as a $d \times n$ floating point matrix.

For vectorial matroids, the rank of a subset of columns is computed using LAPACK ([Anderson et al. 1999]), a standard and robust linear algebra package. For vectorial matroids with elements in $\mathbb{Z}$, our software has the option of using the GMP arbitrary-precision software ([Granlund and et al. 2009]) to perform Gaussian elimination using exact arithmetic. Enumeration of the neighbors of a base B is done by calculating the rank of $B\backslash i \cup j$ for all $i \in B$ and $j \in [n]\backslash B$. If it is full rank then it is returned as a neighboring base. Random bases are determined by randomly choosing a $rank(\mathcal{M})$ sized subset $A \subseteq [n]$ and checking if $rank(A) = rank(\mathcal{M})$, repeating until such an A is found.

For graphic matroids, the rank of the matroid, and any $A \subseteq [n]$, is determined by calculating the size of a spanning forest by breadth-first-search. To enumerate the neighbors of $B \in \mathcal{B}_{\mathcal{M}}$, we first calculate all paths in B using dynamic programming. Adding any element (edge) $j \notin B$ to B will create a cycle C. We use our precalculation of all paths of B to quickly determine C. Then all subsets $B\backslash i \cup j$ where $i \in C$ will be an adjacent base to B. This is not the most efficient method for adjacency enumeration with respect to a graphic matroid, but it is straight forward and good enough.

The Projected Boundary algorithm (Algorithm 4) is only implemented for $d = 2$. This is due to the computational expense of determining if $W\mathbf{e}(\widehat{B})$ is an extreme point of $W\mathcal{P}_{\mathcal{M}}$ on line 8. When $d = 2$ this is easy to check by sorting the vectors $W\mathbf{e}(\widehat{B}) - W\mathbf{e}(B)$ with respect to their angle to the positive x-axis, where $\widehat{B}$ is a neighbor of B. If there exists two sorted vectors with angle larger than π then $W\mathbf{e}(\widehat{B})$ is extreme, due to Corollary 2.

For correctness, we compare the number of projected spanning trees found using our methods versus the actual total number of projected spanning trees. We used an algorithm for generating all of the spanning trees in undirected graphs presented by Matsui (see [Matsui 1997]). The algorithm requires $O(n + m + \tau n)$ time when the given graph has n vertices, m edges, and τ spanning trees. For outputting all of the spanning trees explicitly, this time complexity is optimal. We also implemented the asymptotic $0\backslash 1$ polytope vertex-estimation presented in [Barvinok and Samorodnitsky 2007]. This gives us the ability to estimate the number of bases of matroid polytopes in order to better understand the ratio of bases to projected bases for problems where full enumeration is intractable.

[1]Matroid Optimization: Combinatorial Heuristics and Algorithms

4. COMPUTATIONAL RESULTS

Now we present our experiments. We performed many more experiments as presented here but for compactness and space limitation we present the full collection at [De Loera et al. 2009]. In the experiments we used six roughly comparable machines (see Table XVII). Heuristic 1 uses the pivot rule that $f(W\mathbf{e}(\widetilde{B})) < f(W\mathbf{e}(B'))$ for all $\widetilde{B} \in Adj(B')$ in line 4. Heuristic 2 uses the pivot rule that $f(W\mathbf{e}(\widetilde{B})) < f(W\mathbf{e}(\widehat{B}))$ for all $\widetilde{B}, \widehat{B} \in Adj(B') \setminus$ `VIS` in line 5.

4.1 Calibration Set

Table II. Calibration Graphs

Name	Nodes	Edges	#Spanning Trees
gn10e22	10	22	53,357
gn10e28	10	28	800,948
gn10e33	10	33	3,584,016
gn11e41	11	41	90,922,271
gn13e39	13	39	131,807,934

Our first goal was to perform experiments on matroids for which we can compute all bases in order to better understand our heuristics and algorithms. We generated five connected random graphs: *gn10e22, gn10e28, gn10e33, gn11e41, gn13e39* (see II) which we will refer to as our *calibration set*. We consider two, three and five criteria, i.e. number of weightings. We further consider three different ranges of integral weights for each criterion. For the calibration set we adopt the following nomenclature

gn[#**nodes**]*e*[#**edges**]*d*[#**criteria**]*w*[**low weight**]*w*[**high weight**]

where we generated random integral weightings between [**low weight**] and [**high weight**]. First we simply compare the number of spanning trees of our calibration set to the number of projected spanning trees.

—Table III shows the calibration set with two, three, and five weightings (criteria) and various integral weights.

We give the exact number of projected spanning trees and compare versus the exact number of spanning trees. We also generated ten additional graphs and generated complete tables for all fifteen graphs which can be found at [De Loera et al. 2009].

4.1.1 *Calibration Set - Heuristic 6.* There are four parameters to our implementation of Heuristic 6 ($DFBFS$): number of searches N, BFS depth, boundary retry limit and random retry limit. First we attempt to find a new boundary projected base using Heuristic 1 (Local Search) and a random direction. For every new projected base, we run Heuristic 6 ($DFBFS$) with the given depth parameter.

We attempt to find N new boundary projected bases and give up if we exceed the boundary retry limit. Next our algorithm will generate a random base and project it by the weighting. If it is a new projected base we run Heuristic 6 ($DFBFS$) with the given depth parameter. We attempt to find N new random projected bases and give up if we exceed the random retry limit.

—Table IV shows Heuristic 6 ($DFBFS$) on the calibration set in two, three, and five criteria.

We list the machine used, seconds required, number of searches, BFS depth, boundary retry limit and the random retry limit. We noticed a decrease in the ratio of projected trees found to all projected trees as the dimension and weights increased. It is also important to note, as seen in Table III, as the dimension and weights increase, the ratio of all projected trees to all spanning trees increases.

In our experiments, in almost all cases, Heuristic 6 ($DFBFS$) terminated by exceeding the boundary and random retry limit and not the number of searches. This can be attributed to the phenomenon where most bases are projected in a relatively tight area, with few projected bases near the boundary. Other conditions for picking the initial base for Heuristic 6 ($DFBFS$) could yield better results. For example, keeping a random leaf of the previous truncated BFS. Or we could compute the boundary of the projected bases and attempt to find a new initial base using Heuristic 3 (PT) on the "holes" in the convex hull.

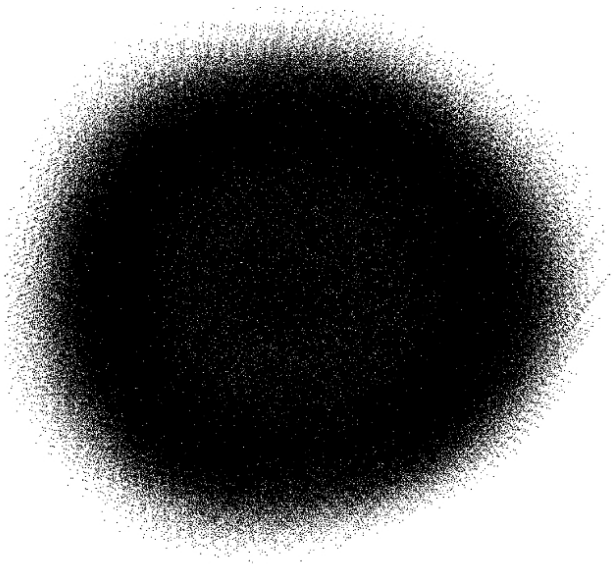

Fig. 3. 96.18% projected bases of *gn11e41d2w0w100* using Heuristic 6 ($DFBFS$)

4.1.2 *Calibration Set - Heuristic 3.* Our implementation of Heuristic 3 (PT) uses Heuristic 1 (LS) with the $2d$ directions $\{ \pm\mathbf{e}_i \mid i \in \{1, \ldots, d\} \}$ to compute a bounding box containing all projected base.

—Table V shows the result of Heuristic 3 (PT) on our calibration test set with two criteria and weights $0-20$ and $0-100$ using Heuristic 1 (LS) as our test method subroutine. We give the percentage of projected trees found versus all projected trees. Naturally with larger weight values, the projected trees will be contained in a larger box, requiring more time.

—Table VI shows the result of Heuristic 3 (PT) on our calibration test set with two criteria and weights $0-20$ using Heuristic 2 (TS) as our test method subroutine. The run times using Heuristic 2 (TS) are longer than Heuristic 1 (LS), but we find nearly all the projected trees.

Heuristic 3 (PT) has two major advantages;

—Heuristic 3 uses very little memory since Heuristic 2 (TS) and Heuristic 1 (LS) use little memory;

—Heuristic 3 can be distributed since one can partition up test regions and run each region as a separate instance.

4.1.3 *Calibration Set - Algorithm 4.* Table VIII shows the number of projected spanning trees found on the boundary using Algorithm 4 (PB). All computations took less than one second. To find a starting base, Algorithm 4 (PB) generates a random direction and calls Heuristic 1 (LS). It is interesting to note that the graph *gn13e39d2w0w1000* has $15,037,589$ projected spanning trees, yet there are only 26 points found by Algorithm 4 (PB).

4.1.4 *Calibration Set - Heuristic 5.* Table IX shows the results of Heuristic 5 ($BTRPT$) on the calibration set with weightings $0-20$ and $0-100$ in two criteria. We use Heuristic 2 (TS) as our subroutine in Heuristic 3 (PT). We give the seconds, the exact number of Pareto optima and the number of Pareto optima found. Heuristic 5 ($BTRPT$) not only found the correct number of Pareto optima, it found the correct Pareto optima in all cases. Heuristic 5 ($BTRPT$) can be distributed: once the boundary is computed and the regions are found, Heuristic 3 (PT) can be run as separate instances for each region (or subdivided further).

4.1.5 *Calibration - Local and Tabu Search For Convex Minimization.* For our experiments of Heuristic 1 (LS) and Heuristic 2 (TS) on the calibration set we minimized over the convex function $(\mathbf{x} - \widehat{\mathbf{x}})^2$. First we choose $\widehat{\mathbf{x}} \in W\mathbf{e}(\mathcal{B}_{\mathcal{M}})$ such that it is an interior point of $W\mathcal{P}_{\mathcal{M}}$. Second we consider $\widehat{\mathbf{x}}$ to be a rational non-integer point which is an interior point of $W\mathcal{P}_{\mathcal{M}}$. For the integral case, we can easily detect if we are at the minimum because our objective will evaluate to zero if so. For the rational case, we verify the global minimum by evaluating our objective on all projected spanning trees. For all tests we perform 1000 minimizations with random starts and record the number of successes.

—Table X show Heuristic 1 (LS) where the minimum is the integer point described above minimizing $(\mathbf{x} - \widehat{\mathbf{x}})^2$.

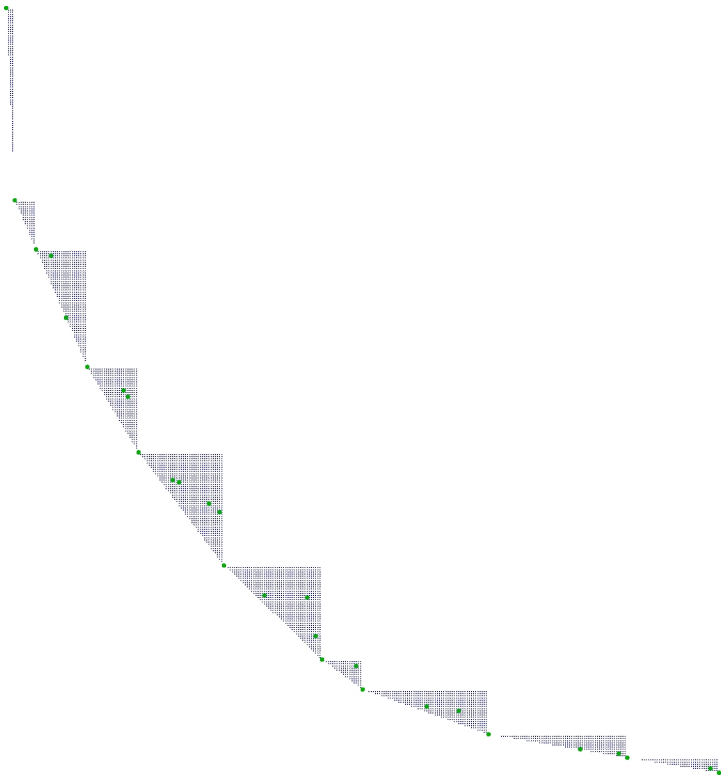

Fig. 4. Green Pareto optima and blue test regions of *gn11e41d2w0w100* using Heuristic 5 ($BTRPT$)

—Table XII show Heuristic 1 (LS) where the minimum is the rational point described above minimizing $(\mathbf{x} - \widehat{\mathbf{x}})^2$.

—Table XI show Heuristic 2 (TS) where the minimum is the integer point described above minimizing $(\mathbf{x} - \widehat{\mathbf{x}})^2$.

—Table XIII show Heuristic 2 (TS) where the minimum is the rational point described above minimizing $(\mathbf{x} - \widehat{\mathbf{x}})^2$.

We first observe that the success of Heuristic 1 (LS) and Heuristic 2 (TS) decreases as the proportion of projected spanning trees to all spanning trees increases. Second, for Heuristic 2 (TS) we see a noticeable increase in the number of successes as the Tabu Limit ranges from 1 to 100 indicating that in many cases, a low Tabu Limit is sufficient to reach the global minimum.

4.1.6 *Calibration Set - Non-convex Optimization.* We emphasize that Heuristic 6 (DFBFS) and Heuristic 3 (Pivot Test) do not rely on any particular objective being optimized and we propose that their effectiveness be gauged by how many projected bases they find. For completeness though, we minimize the following

three non-convex functions:

$$f(x,y) = x\sin(3y^2+x^3) + x^2y^3 + \frac{1}{(xy-3)^2+1},$$

$$g(x,y,z) = xy^2\cos^2(xy-z^3) + 3x - 2y + z^2 + \frac{1}{(2x^3-3y-z)^2+1},$$

$$h(u,v,x,y,z) = uv\sin(x)\cos(y^2)\cos(\frac{1}{z^2+1}) - x^3z^2 - u^2y + 2vx.$$

Table VII shows the minimal value of the three functions f, h, g, (applied to the problems of appropriate dimension) over all projected bases and those found by Heuristic 6 (DFBFS), Heuristic 3 (Pivot Test) using Local Search, and Heuristic 3 (Pivot Test) using Tabu Search. Table VII only shows minimal values for Heuristic 3 for the data computed in Table V and VI (which was limited due to running times).

4.2 Pushing the Limits

4.2.1 *Sparse Graphs - Solids.* We now compare sparse graphs versus the dense graphs presented in the calibration set. We chose fifteen planar graphs of the 1-skeleton of three-dimensional polytopes. Table XIV shows the result of Heuristic 6 ($DFBFS$) on these examples. For some of these graphs, the number of spanning trees can be explicitly enumerated, hence we give the exact number of projected spanning trees. When the weights take value from $0-1000$ we omit the last five graphs as $DFBFS$ exceeded over a day of computation or the machines (*Fuzzy, Truth*) ran out of memory.

4.2.2 *Experimental Design.* In [Berstein et al. 2008], the authors first proposed using nonlinear matroid optimization to solve the statistical experimental design problem (also see [Fries and Hunter 1980]). The experimental design problem can be briefly described as attempting to learn an unknown system whose output $\mathbf{y}$ is an unknown function Φ with input $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{R}^k$. To learn the system, experiments are performed using input $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,k})$ and the output $\mathbf{y}_i = \Phi(\mathbf{p}_i)$ is measured. Then, based on the experiments we wish to *fit a model* for the system, that is, determine an estimation $\widehat{\Phi}$ of the function Φ such that it: lies in a prescribed class of functions, is consistent with the outcomes of the experiments, and minimizes the *aberration* (some suitable criteria) among models in the class.

A deterministic polynomial time algorithm was established in [Berstein et al. 2008] which solves nonlinear matroid optimization over arbitrary matroids (presented by an independence oracle) when the number of weightings d is fixed and the weights, though binary encoded input, take on only a fixed number p of distinct values. For matroids on n elements, their algorithm uses the ordinary matroid intersection algorithm n^{p^d} times, which is polynomial in the input.

For vectorial matroids, they provide a different algorithm, which has polynomial complexity when the weights are encoded in unary. If ω is the size of the maximum unary encoded element of the weights, then this algorithm requires solving a $(rank(\mathcal{M})\omega+1)^d \times (rank(\mathcal{M})\omega+1)^d$ Vandemonde system of linear equations. This nonlinear vectorial-matroid optimization algorithm has been implemented on IBM's Blue Gene/P supercomputer (see [Gunnels et al. 2008]). We present now two

instances P_{20} and P_{28} which were solved using the deterministic vectorial-matroid algorithm on said supercomputer. Each is an instance of an experimental design problem encoded as a nonlinear vectorial-matroid optimization program. Both their exact algorithm and our heuristics (Heuristic 6 and Heuristic 3) do not optimize a particular objective function, but instead list the projected bases (note that our heuristic is not guaranteed to find all projected bases).

Table XV shows the size of the instances P_{20} and P_{28}, the true number of projected bases and the number of seconds required by the exact algorithm (on the Blue Gene/P supercomputer) to find the projected bases. We also show two runs of Heuristic 6 ($DFBFS$) using Fuzzy with the number of projected bases found and seconds required. What is noteworthy is that Heuristic 6 found all the projected bases in at least one of the two runs. Moreover, the other runs of Heuristic 6 found nearly all the projected bases. This demonstrates that our heuristics are useful when solving these difficult problems, especially in light of the fact that they were computed using a modest computer system. An additional point is that the deterministic vectorial-matroid algorithm requires extremely-high precision in order to solve the large Vandemonde systems derived from the matroid optimization problem. On the other hand, our heuristics only depend numerically on solving at most $rank(\mathcal{M}) \times rank(\mathcal{M})$ linear systems. In fact, our software has an option to use exact arithmetic, albeit with a slow down in solution times. We emphasize the drastic difference in running times and computational power between Blue Gene/P and Fuzzy.

In Table XVI we show the result of Heuristic 3 (PT) on P_{20}. We divided the region of feasible projected bases into twelve disjoint regions and ran twelve instances of Heuristic 3 (PT) concurrently. We report the times and the number of projected bases found for each region. We also give the total number of projected bases found, which is the exact number of projected bases. For the search in Heuristic 3 (PT) we used Heuristic 2 (TS) 10 times on each point with a retry limit of 20. We performed this experiment to exhibit the effectiveness and ease by which Heuristic 3 can be distributed.

4.3 Discussion

Nonlinear matroid optimization problems are very difficult. They are NP-complete in general and provably exponential in some cases (see [Berstein et al. 2008]). Our goal was to present and explore the practicality of new heuristics and algorithms for solving these problems. The effectiveness of our new techniques rely on two important properties: (i) the nice local adjacency structure of matroids, and that (ii) although a matroid may have exponentially many bases, under suitable assumptions on the encoding of the weightings, the number of projected bases can be manageable.

The fact that many of our heuristics are not guaranteed to find the optimal solution or list all the projected bases is countered by the fact that they are not overly complex and very fast. For instance, Heuristic 6 ($DFBFS$) is not guaranteed to find all of the projected bases, but in many of the tests it found a very large portion of them. In practice, as seen in Subsection 4.2.2, because Heuristic 6 ($DFBFS$) finds a very large portion of the projected bases, simply running the heuristic several times and unioning the results is extremely effective.

Most all of our heuristics and algorithms complement each other. For example, we can start by finding the vertices using Algorithm 4 (PB) then enumerate as many projected bases as possible using Heuristic 6 ($DFBFS$). Next we can run Heuristic 3 (PT) on any regions we suspect there to be more projected bases.

Although we do not formally establish it, our heuristics and algorithms use minimal memory. In particular, all of our heuristics and algorithms enumerate neighboring bases, which is quadratic in the number of elements of the matroid. The biggest memory usage is Heuristic 6 ($DFBFS$), which at worst stores all the projected bases found thus far.

This leads to the ease by which Heuristic 3 (PT) could be distributed. In fact, the only overhead would be combining the solutions, which would be a simple union. Ideally, if we had X machines with Y processors, we could first find a box containing the potential projected bases using $2n$ linear programs via Heuristic 1 (LS). Next, we could partition the region of potential projected bases into XY regions and run an instance of Heuristic 3 (PT) for each.

4.4 Tables

Table III. Calibration test set: Exact #spanning trees vs. #projected spanning trees in 2, 3, and 5 criteria.

Name	Dimension	Weight Range	#Projected Trees	Percentage of # Projected Trees out of # Spanning Trees
gn10e22d2w0w20	2	$0-20$	$3,957$	07.42%
gn10e28d2w0w20	2	$0-20$	$7,131$	00.89%
gn10e33d2w0w20	2	$0-20$	$10,833$	00.30%
gn11e41d2w0w20	2	$0-20$	$16,457$	00.02%
gn13e39d2w0w20	2	$0-20$	$18,468$	00.01%
gn10e22d2w0w100	2	$0-100$	$37,204$	69.73%
gn10e28d2w0w100	2	$0-100$	$101,334$	12.65%
gn10e33d2w0w100	2	$0-100$	$166,427$	04.64%
gn11e41d2w0w100	2	$0-100$	$309,961$	00.34%
gn13e39d2w0w100	2	$0-100$	$315,881$	00.24%
gn10e22d2w0w1000	2	$0-1000$	$52,990$	99.31%
gn10e28d2w0w1000	2	$0-1000$	$756,013$	94.39%
gn10e33d2w0w1000	2	$0-1000$	$2,726,287$	76.07%
gn11e41d2w0w1000	2	$0-1000$	$13,884,793$	15.27%
gn13e39d2w0w1000	2	$0-1000$	$15,037,589$	11.41%
gn10e22d3w0w20	3	$0-20$	$42,887$	80.38%
gn10e28d3w0w20	3	$0-20$	$238,529$	29.78%
gn10e33d3w0w20	3	$0-20$	$411,730$	11.49%
gn11e41d3w0w20	3	$0-20$	$959,469$	01.06%
gn13e39d3w0w20	3	$0-20$	$930,322$	00.71%
gn10e22d3w0w100	3	$0-100$	$53,289$	99.87%
gn10e28d3w0w100	3	$0-100$	$786,781$	98.23%
gn10e33d3w0w100	3	$0-100$	$3,351,096$	93.01%
gn11e41d3w0w100	3	$0-100$	$35,943,327$	39.53%
gn13e39d3w0w100	3	$0-100$	$44,757,592$	33.96%
gn10e22d3w0w1000	3	$0-1000$	$53,357$	100.00%
gn10e28d3w0w1000	3	$0-1000$	$800,946$	99.99%
gn10e33d3w0w1000	3	$0-1000$	$3,583,757$	99.99%
gn11e41d3w0w1000	3	$0-1000$	$90,699,181$	99.75%
gn13e39d3w0w1000	3	$0-1000$	$131,464,478$	99.74%
gn10e22d5w0w1	5	$0-1$	$4,746$	8.89%
gn10e28d5w0w1	5	$0-1$	$10,898$	1.36%
gn10e33d5w0w1	5	$0-1$	$15,482$	0.43%
gn11e41d5w0w1	5	$0-1$	$36,847$	0.04%
gn13e39d5w0w1	5	$0-1$	$34,392$	0.03%
gn10e22d5w0w2	5	$0-2$	$14,623$	27.41%
gn10e28d5w0w2	5	$0-2$	$66,190$	8.26%
gn10e33d5w0w2	5	$0-2$	$105,309$	2.94%
gn11e41d5w0w2	5	$0-2$	$290,555$	0.32%
gn13e39d5w0w2	5	$0-2$	$348,703$	0.26%
gn10e22d5w0w5	5	$0-5$	$49,463$	92.70%
gn10e28d5w0w5	5	$0-5$	$493,565$	61.62%
gn10e33d5w0w5	5	$0-5$	$1,294,875$	36.13%
gn11e41d5w0w5	5	$0-5$	$4,711,354$	5.18%
gn13e39d5w0w5	5	$0-5$	$7,737,684$	5.87%

Table IV. Calibration test set: Heuristic 6 ($DFBFS$) vs. exact # of projected spanning trees in 2, 3, and 5 criteria. We tried 100 searches, depth 4, boundary retry limit 100 and interior retry limit of 10000.

Name	Dimension	Weight Range	Computer	Seconds	#Trees Found by DFBFS	Percentage of # Trees Found out of # Projected Trees
gn10e22d2w0w20	2	0 − 20	Fuzzy	0	3,852	97.34%
gn10e28d2w0w20	2	0 − 20	Fuzzy	2	7,074	99.20%
gn10e33d2w0w20	2	0 − 20	Fuzzy	2	10,734	99.08%
gn11e41d2w0w20	2	0 − 20	Fuzzy	4	16,262	98.81%
gn13e39d2w0w20	2	0 − 20	Fuzzy	5	18,330	99.25%
gn10e22d2w0w100	2	0 − 100	Fuzzy	3	35,581	95.63%
gn10e28d2w0w100	2	0 − 100	Fuzzy	8	95,313	94.05%
gn10e33d2w0w100	2	0 − 100	Fuzzy	14	157,685	94.74%
gn11e41d2w0w100	2	0 − 100	Fuzzy	32	298,150	96.18%
gn13e39d2w0w100	2	0 − 100	Fuzzy	37	301,567	95.46%
gn10e22d2w0w1000	2	0 − 1000	Fuzzy	5	51,782	97.72%
gn10e28d2w0w1000	2	0 − 1000	Fuzzy	46	659,761	87.26%
gn10e33d2w0w1000	2	0 − 1000	Fuzzy	90	1,628,981	59.75%
gn11e41d2w0w1000	2	0 − 1000	Fuzzy	455	7092,823	51.08%
gn13e39d2w0w1000	2	0 − 1000	Fuzzy	825	10,260,810	68.23%
gn10e22d3w0w20	3	0 − 20	Fuzzy	4	41,649	97.11%
gn10e28d3w0w20	3	0 − 20	Fuzzy	19	217,444	91.16%
gn10e33d3w0w20	3	0 − 20	Fuzzy	36	374,110	90.86%
gn11e41d3w0w20	3	0 − 20	Fuzzy	97	876,488	91.35%
gn13e39d3w0w20	3	0 − 20	Fuzzy	106	872,397	93.77%
gn10e22d3w0w100	3	0 − 100	Fuzzy	4	52,296	98.13%
gn10e28d3w0w100	3	0 − 100	Fuzzy	50	699,399	88.89%
gn10e33d3w0w100	3	0 − 100	Fuzzy	133	2,095,143	66.66%
gn11e41d3w0w100	3	0 − 100	Fuzzy	801	13,527,453	37.14%
gn13e39d3w0w100	3	0 − 100	Fuzzy	1777	23,502,664	52.27%
gn10e22d3w0w1000	3	0 − 1000	Fuzzy	4	51,592	96.69%
gn10e28d3w0w1000	3	0 − 1000	Fuzzy	52	721,587	90.09%
gn10e33d3w0w1000	3	0 − 1000	Fuzzy	135	2,313,881	64.56%
gn11e41d3w0w1000	3	0 − 1000	Fuzzy	1021	16,919,561	18.65%
gn13e39d3w0w1000	3	0 − 1000	Fuzzy	2315	34,603,989	26.32%
gn10e22d5w0w1	5	0 − 1	Fuzzy	1	4,704	99.12%
gn10e28d5w0w1	5	0 − 1	Fuzzy	3	10,655	97.77%
gn10e33d5w0w1	5	0 − 1	Fuzzy	4	15,235	98.40%
gn11e41d5w0w1	5	0 − 1	Fuzzy	14	36,457	98.94%
gn13e39d5w0w1	5	0 − 1	Fuzzy	16	34,076	99.08%
gn10e22d5w0w2	5	0 − 2	Fuzzy	2	14,205	97.14%
gn10e28d5w0w2	5	0 − 2	Fuzzy	8	63,992	96.68%
gn10e33d5w0w2	5	0 − 2	Fuzzy	16	103,221	98.02%
gn11e41d5w0w2	5	0 − 2	Fuzzy	49	281,519	96.89%
gn13e39d5w0w2	5	0 − 2	Fuzzy	64	339,179	97.27%
gn10e22d5w0w5	5	0 − 5	Fuzzy	5	48,569	98.19%
gn10e28d5w0w5	5	0 − 5	Fuzzy	44	460,744	93.35%
gn10e33d5w0w5	5	0 − 5	Fuzzy	114	1,145,396	88.46%
gn11e41d5w0w5	5	0 − 5	Fuzzy	561	4,247,218	90.15%
gn13e39d5w0w5	5	0 − 5	Fuzzy	907	6,800,052	87.88%

Table V. Calibration test set: Heuristic 3 (Pivot Test) using Heuristic 1 (Local Search), 10 searches per point.

Name	Dimension	Weight Range	Computer	Seconds	#Trees Found by PT	Percentage of # Trees Found out of # Projected Trees
gn10e22d2w0w20	2	$0-20$	mocr01	101	3,318	83.85%
gn10e28d2w0w20	2	$0-20$	mocr01	207	6,163	86.43%
gn10e33d2w0w20	2	$0-20$	mocr01	356	8,842	81.62%
gn11e41d2w0w20	2	$0-20$	mocr01	806	14,747	89.61%
gn13e39d2w0w20	2	$0-20$	mocr01	1,160	15,082	81.67%
gn10e22d2w0w100	2	$0-100$	mocr02	4,315	5,249	14.11%
gn10e28d2w0w100	2	$0-100$	mocr02	6,224	8,236	08.13%
gn10e33d2w0w100	2	$0-100$	mocr02	11,925	19,247	11.56%
gn11e41d2w0w100	2	$0-100$	mocr02	26,144	29,373	09.48%
gn13e39d2w0w100	2	$0-100$	mocr02	32,958	38,530	12.20%
gn10e22d3w0w20	3	$0-20$	mocr01	18,141	8,564	19.97%
gn10e28d3w0w20	3	$0-20$	mocr01	50,429	39,819	16.69%
gn10e33d3w0w20	3	$0-20$	mocr01	76,051	47,606	11.56%
gn11e41d3w0w20	3	$0-20$	mocr01	216,987	120,142	12.52%
gn13e39d3w0w20	3	$0-20$	mocr01	232,975	126,044	13.55%

Table VI. Calibration test set: Heuristic 3 (Pivot Test) using Heuristic 2 (Tabu Search), 10 searches per point, 20 search limit.

Name	Dimension	Weight Range	Computer	Seconds	#Trees Found by DFBFS	Percentage of # Trees Found out of # Projected Trees
gn10e22d2w0w20	2	$0-20$	mocr04	394	3,878	98.00%
gn10e28d2w0w20	2	$0-20$	mocr04	712	7,085	99.35%
gn10e33d2w0w20	2	$0-20$	mocr04	1,146	10,739	99.13%
gn11e41d2w0w20	2	$0-20$	mocr04	2,484	16,379	99.53%
gn13e39d2w0w20	2	$0-20$	mocr04	3,233	18,316	99.18%
gn10e22d2w0w100	2	$0-100$	mocr04	25,007	20,976	56.38%
gn10e28d2w0w100	2	$0-100$	mocr04	33,342	50,722	50.05%
gn10e33d2w0w100	2	$0-100$	mocr04	56,517	82,275	49.44%
gn11e41d2w0w100	2	$0-100$	mocr04	120,393	188,619	60.85%
gn13e39d2w0w100	2	$0-100$	mocr04	138,832	197,263	62.45%

Table VII. Calibration test set: Minimizing non-convex functions f,g,h over all feasible points, and points found by Heuristic 6 (DFBFS), Heuristic 3 (Pivot Test) using Local Search, and Heuristic 3 (Pivot Test) using Tabu Search. Presented are the optimal values (when data is available).

Name	Function	True Optimum	Heuristic 6 (DFBFS)	Heuristic 3 (Pivot Test, LS)	Heuristic 3 (Pivot Test, TS)
gn10e22d2w0w20	f	1.82892×10^9	1.82892×10^9	1.82892×10^9	1.82892×10^9
gn10e28d2w0w20	f	2.56901×10^7	2.56901×10^7	2.56901×10^7	2.56901×10^7
gn10e33d2w0w20	f	2.47646×10^8	2.47646×10^8	2.47646×10^8	2.47646×10^8
gn11e41d2w0w20	f	9.11141×10^7	9.11141×10^7	9.11141×10^7	9.11141×10^7
gn13e39d2w0w20	f	1.67772×10^7	1.67772×10^7	1.67772×10^7	1.67772×10^7
gn10e22d2w0w100	f	2.35746×10^{12}	2.35746×10^{12}	2.35746×10^{12}	2.35746×10^{12}
gn10e28d2w0w100	f	2.63506×10^{12}	2.63506×10^{12}	2.63506×10^{12}	2.63506×10^{12}
gn10e33d2w0w100	f	3.98419×10^{11}	3.98419×10^{11}	3.98419×10^{11}	3.98419×10^{11}
gn11e41d2w0w100	f	1.00562×10^{11}	1.00562×10^{11}	1.00562×10^{11}	1.00562×10^{11}
gn13e39d2w0w100	f	1.69141×10^{12}	1.69141×10^{12}	1.69141×10^{12}	1.69141×10^{12}
gn10e22d2w0w1000	f	8.07213×10^{16}	8.07213×10^{16}		
gn10e28d2w0w1000	f	1.18485×10^{16}	1.18485×10^{16}		
gn10e33d2w0w1000	f	3.0741×10^{16}	3.0741×10^{16}		
gn11e41d2w0w1000	f	6.42209×10^{16}	6.42209×10^{16}		
gn13e39d2w0w1000	f	1.71265×10^{17}	1.71265×10^{17}		
gn10e22d3w0w20	g	4091.69	4091.69	4091.69	
gn10e28d3w0w20	g	3860.38	3860.38	3860.38	
gn10e33d3w0w20	g	1817.41	1817.41	1817.41	
gn11e41d3w0w20	g	2277.52	2277.52	2277.52	
gn13e39d3w0w20	g	1179.44	1179.44	1179.44	
gn10e22d3w0w100	g	40100.8	40100.8		
gn10e28d3w0w100	g	30284.8	30284.8		
gn10e33d3w0w100	g	50521.1	50521.1		
gn11e41d3w0w100	g	38975.5	38975.5		
gn13e39d3w0w100	g	87875.4	98396.4		
gn10e22d3w0w1000	g	4.04314×10^6	4.04314×10^6		
gn10e28d3w0w1000	g	5.63403×10^6	5.63403×10^6		
gn10e33d3w0w1000	g	4.32731×10^6	4.32731×10^6		
gn11e41d3w0w1000	g	6.93761×10^6	7.17085×10^6		
gn13e39d3w0w1000	g	8.96618×10^6	8.96618×10^6		
gn10e22d5w0w1	h	-12783.4	-12783.4		
gn10e28d5w0w1	h	-46634.3	-46634.3		
gn10e33d5w0w1	h	-41585.1	-41585.1		
gn11e41d5w0w1	h	-36117.7	-36117.7		
gn13e39d5w0w1	h	-49281	-49281		
gn10e22d5w0w2	h	-562722	-562722		
gn10e28d5w0w2	h	-618422	-618422		
gn10e33d5w0w2	h	-693793	-693793		
gn11e41d5w0w2	h	-2.59324×10^6	-2.59324×10^6		
gn13e39d5w0w2	h	-3.45193×10^6	-3.45193×10^6		
gn10e22d5w0w5	h	-2.47039×10^7	-2.47039×10^7		
gn10e28d5w0w5	h	-4.65803×10^7	-4.65803×10^7		
gn10e33d5w0w5	h	-4.56097×10^7	-4.56097×10^7		
gn11e41d5w0w5	h	-1.07001×10^8	-1.07001×10^8		
gn13e39d5w0w5	h	-1.95109×10^8	-1.95109×10^8		

Table VIII. Calibration test set: Algorithm 4 (Boundary Calculation) with 2 criteria. All times under 1 second.

Name	Weight Range	Computer	Extremal Points
gn10e22d2w0w20	$0-20$	mocr02	35
gn10e28d2w0w20	$0-20$	mocr02	33
gn10e33d2w0w20	$0-20$	mocr02	10
gn11e41d2w0w20	$0-20$	mocr02	38
gn13e39d2w0w20	$0-20$	mocr02	52
gn10e22d2w0w100	$0-100$	mocr02	26
gn10e28d2w0w100	$0-100$	mocr02	31
gn10e33d2w0w100	$0-100$	mocr02	37
gn11e41d2w0w100	$0-100$	mocr02	41
gn13e39d2w0w100	$0-100$	mocr02	42
gn10e22d2w0w1000	$0-1000$	mocr02	37
gn10e28d2w0w1000	$0-1000$	mocr02	9
gn10e33d2w0w1000	$0-1000$	mocr02	21
gn11e41d2w0w1000	$0-1000$	mocr02	31
gn13e39d2w0w1000	$0-1000$	mocr02	26

Table IX. Calibration test set: Heuristic 5 (Boundary and Triangular Region Pareto Test) with 2 criteria. Internally, Heuristic 3 (PT) used 10 searches per point, 100 pivot limit.

Name	Weight Range	Computer	Seconds	# Pareto Optimum	#Computed Pareto Optimum	Percent
gn10e22d2w0w20	$0-20$	Fuzzy	29	19	19	100.00%
gn10e28d2w0w20	$0-20$	Fuzzy	9	10	10	100.00%
gn10e33d2w0w20	$0-20$	Fuzzy	59	21	21	100.00%
gn11e41d2w0w20	$0-20$	Fuzzy	96	17	17	100.00%
gn13e39d2w0w20	$0-20$	Fuzzy	33	19	19	100.00%
gn10e22d2w0w100	$0-100$	Fuzzy	1,286	21	21	100.00%
gn10e28d2w0w100	$0-100$	Fuzzy	1,251	28	28	100.00%
gn10e33d2w0w100	$0-100$	Fuzzy	1,625	23	23	100.00%
gn11e41d2w0w100	$0-100$	Fuzzy	2,472	28	28	100.00%
gn13e39d2w0w100	$0-100$	Fuzzy	3,328	33	33	100.00%

Table X. Calibration test set: Local Search Heuristic 1 minimizing $(x - \widehat{x})^2$, where for each instance, $\widehat{x}$ is an interior integer point that is the image of some base. Here $d = 2, 3,$ and 5 and all tests run on Fuzzy. Shown are the number of spanning trees and projected bases of each instance and the number of successes out of 1000 Local Searches and seconds to perform the 1000 searches.

Name	Dimension	Weight Range	#Successes out of 1000	Seconds
gn10e22d2w0w20	2	$0-20$	316	1
gn10e28d2w0w20	2	$0-20$	252	2
gn10e33d2w0w20	2	$0-20$	172	1
gn11e41d2w0w20	2	$0-20$	190	5
gn13e39d2w0w20	2	$0-20$	244	4
gn10e22d2w0w100	2	$0-100$	5	1
gn10e28d2w0w100	2	$0-100$	0	2
gn10e33d2w0w100	2	$0-100$	12	3
gn11e41d2w0w100	2	$0-100$	6	4
gn13e39d2w0w100	2	$0-100$	13	5
gn10e22d2w0w1000	2	$0-1000$	5	1
gn10e28d2w0w1000	2	$0-1000$	0	1
gn10e33d2w0w1000	2	$0-1000$	0	2
gn11e41d2w0w1000	2	$0-1000$	0	2
gn13e39d2w0w1000	2	$0-1000$	0	3
gn10e22d3w0w20	3	$0-20$	0	2
gn10e28d3w0w20	3	$0-20$	0	3
gn10e33d3w0w20	3	$0-20$	0	3
gn11e41d3w0w20	3	$0-20$	8	5
gn13e39d3w0w20	3	$0-20$	7	4
gn10e22d3w0w100	3	$0-100$	29	1
gn10e28d3w0w100	3	$0-100$	0	2
gn10e33d3w0w100	3	$0-100$	0	3
gn11e41d3w0w100	3	$0-100$	0	4
gn13e39d3w0w100	3	$0-100$	0	6
gn10e22d3w0w1000	3	$0-1000$	12	1
gn10e28d3w0w1000	3	$0-1000$	0	2
gn10e33d3w0w1000	3	$0-1000$	0	2
gn11e41d3w0w1000	3	$0-1000$	0	3
gn13e39d3w0w1000	3	$0-1000$	0	3
gn10e22d5w0w1	5	$0-1$	197	2
gn10e28d5w0w1	5	$0-1$	122	3
gn10e33d5w0w1	5	$0-1$	110	4
gn11e41d5w0w1	5	$0-1$	195	6
gn13e39d5w0w1	5	$0-1$	203	6
gn10e22d5w0w2	5	$0-2$	284	2
gn10e28d5w0w2	5	$0-2$	112	2
gn10e33d5w0w2	5	$0-2$	53	4
gn11e41d5w0w2	5	$0-2$	394	4
gn13e39d5w0w2	5	$0-2$	352	4
gn10e22d5w0w5	5	$0-5$	94	2
gn10e28d5w0w5	5	$0-5$	10	2
gn10e33d5w0w5	5	$0-5$	16	3
gn11e41d5w0w5	5	$0-5$	0	7
gn13e39d5w0w5	5	$0-5$	0	6

Table XI. Calibration test set: Tabu Search Heuristic 2 minimizing $(x - \widehat{x})^2$, where for each instance, $\widehat{x}$ is an interior integer point that is the image of some base. Here $d = 2, 3,$ and 5 and all tests run on Fuzzy. Shown are the number of successes out of 1000 Tabu Searches for Tabu Search limits of $1, 5, 20, 100$ and seconds (in parenthesis) to perform the 1000 searches.

Name	Dimension	Weight Range	Tabu Limit 1	Tabu Limit 5	Tabu Limit 20	Tabu Limit 100
gn10e22d2w0w20	2	$0-20$	56(0)	503(2)	800(6)	915(27)
gn10e28d2w0w20	2	$0-20$	4(0)	331(3)	717(9)	1000(39)
gn10e33d2w0w20	2	$0-20$	18(1)	396(2)	765(10)	986(42)
gn11e41d2w0w20	2	$0-20$	0(1)	368(5)	807(17)	1000(62)
gn13e39d2w0w20	2	$0-20$	5(1)	456(4)	850(16)	1000(66)
gn10e22d2w0w100	2	$0-100$	0(0)	7(2)	13(8)	19(38)
gn10e28d2w0w100	2	$0-100$	0(1)	2(3)	18(12)	37(53)
gn10e33d2w0w100	2	$0-100$	0(0)	30(4)	130(10)	417(52)
gn11e41d2w0w100	2	$0-100$	0(0)	0(5)	44(17)	47(74)
gn13e39d2w0w100	2	$0-100$	0(1)	29(6)	113(19)	368(94)
gn10e22d2w0w1000	2	$0-1000$	0(0)	35(2)	68(10)	58(44)
gn10e28d2w0w1000	2	$0-1000$	0(0)	0(3)	8(11)	23(51)
gn10e33d2w0w1000	2	$0-1000$	0(1)	0(2)	0(13)	2(67)
gn11e41d2w0w1000	2	$0-1000$	0(1)	0(4)	2(17)	15(87)
gn13e39d2w0w1000	2	$0-1000$	0(0)	0(5)	0(19)	64(98)
gn10e22d3w0w20	3	$0-20$	0(0)	4(3)	19(9)	141(39)
gn10e28d3w0w20	3	$0-20$	0(0)	17(4)	44(13)	185(56)
gn10e33d3w0w20	3	$0-20$	0(1)	0(4)	2(13)	24(48)
gn11e41d3w0w20	3	$0-20$	0(1)	11(6)	43(20)	77(85)
gn13e39d3w0w20	3	$0-20$	0(1)	22(6)	89(20)	352(84)
gn10e22d3w0w100	3	$0-100$	0(1)	113(2)	272(9)	984(40)
gn10e28d3w0w100	3	$0-100$	0(0)	0(3)	0(12)	0(51)
gn10e33d3w0w100	3	$0-100$	0(0)	0(4)	29(15)	44(66)
gn11e41d3w0w100	3	$0-100$	0(0)	0(5)	0(21)	0(108)
gn13e39d3w0w100	3	$0-100$	0(0)	5(8)	79(24)	235(110)
gn10e22d3w0w1000	3	$0-1000$	0(0)	42(2)	122(9)	125(36)
gn10e28d3w0w1000	3	$0-1000$	0(0)	0(3)	0(12)	0(62)
gn10e33d3w0w1000	3	$0-1000$	0(0)	0(4)	2(15)	6(65)
gn11e41d3w0w1000	3	$0-1000$	0(0)	0(5)	0(22)	0(103)
gn13e39d3w0w1000	3	$0-1000$	0(0)	0(5)	0(21)	0(108)
gn10e22d5w0w1	5	$0-1$	0(1)	668(3)	1000(8)	1000(32)
gn10e28d5w0w1	5	$0-1$	0(0)	309(3)	756(11)	1000(47)
gn10e33d5w0w1	5	$0-1$	0(0)	401(4)	938(13)	995(49)
gn11e41d5w0w1	5	$0-1$	0(1)	528(7)	873(20)	1000(79)
gn13e39d5w0w1	5	$0-1$	0(1)	461(6)	676(19)	831(84)
gn10e22d5w0w2	5	$0-2$	1(0)	432(3)	923(9)	1000(35)
gn10e28d5w0w2	5	$0-2$	0(1)	356(4)	688(11)	735(43)
gn10e33d5w0w2	5	$0-2$	0(1)	399(4)	407(12)	414(51)
gn11e41d5w0w2	5	$0-2$	0(0)	555(5)	828(18)	890(76)
gn13e39d5w0w2	5	$0-2$	2(0)	627(6)	966(19)	979(79)
gn10e22d5w0w5	5	$0-5$	0(1)	212(3)	563(9)	757(36)
gn10e28d5w0w5	5	$0-5$	0(0)	30(4)	89(12)	415(56)
gn10e33d5w0w5	5	$0-5$	0(0)	21(4)	83(15)	919(79)
gn11e41d5w0w5	5	$0-5$	0(1)	127(7)	229(23)	398(96)
gn13e39d5w0w5	5	$0-5$	0(0)	14(8)	65(23)	318(106)

Table XII. Calibration test set: Local Search Heuristic 1 minimizing $(x - \widehat{x})^2$, where for each instance, $\widehat{x}$ is an interior rational non-integer point. Here $d = 2, 3,$ and 5 and all tests run on Fuzzy. Shown are the number of spanning trees and projected bases of each instance and the number of successes out of 1000 Local Searches and seconds to perform the 1000 searches.

Name	Dimension	Weight Range	#Successes out of 1000	Seconds
gn10e22d2w0w20	2	$0-20$	203	2
gn10e28d2w0w20	2	$0-20$	364	2
gn10e33d2w0w20	2	$0-20$	216	2
gn11e41d2w0w20	2	$0-20$	318	2
gn13e39d2w0w20	2	$0-20$	232	3
gn10e22d2w0w100	2	$0-100$	47	2
gn10e28d2w0w100	2	$0-100$	0	2
gn10e33d2w0w100	2	$0-100$	16	2
gn11e41d2w0w100	2	$0-100$	21	3
gn13e39d2w0w100	2	$0-100$	34	3
gn10e22d2w0w1000	2	$0-1000$	0	1
gn10e28d2w0w1000	2	$0-1000$	0	2
gn10e33d2w0w1000	2	$0-1000$	0	2
gn11e41d2w0w1000	2	$0-1000$	0	2
gn13e39d2w0w1000	2	$0-1000$	0	3
gn10e22d3w0w20	3	$0-20$	18	2
gn10e28d3w0w20	3	$0-20$	0	2
gn10e33d3w0w20	3	$0-20$	0	3
gn11e41d3w0w20	3	$0-20$	14	2
gn13e39d3w0w20	3	$0-20$	14	3
gn10e22d3w0w100	3	$0-100$	2	1
gn10e28d3w0w100	3	$0-100$	0	2
gn10e33d3w0w100	3	$0-100$	0	2
gn11e41d3w0w100	3	$0-100$	0	3
gn13e39d3w0w100	3	$0-100$	0	4
gn10e22d3w0w1000	3	$0-1000$	10	1
gn10e28d3w0w1000	3	$0-1000$	0	2
gn10e33d3w0w1000	3	$0-1000$	0	2
gn11e41d3w0w1000	3	$0-1000$	0	3
gn13e39d3w0w1000	3	$0-1000$	0	3
gn10e22d5w0w1	5	$0-1$	693	2
gn10e28d5w0w1	5	$0-1$	774	2
gn10e33d5w0w1	5	$0-1$	956	3
gn11e41d5w0w1	5	$0-1$	953	3
gn13e39d5w0w1	5	$0-1$	623	5
gn10e22d5w0w2	5	$0-2$	97	2
gn10e28d5w0w2	5	$0-2$	268	2
gn10e33d5w0w2	5	$0-2$	189	3
gn11e41d5w0w2	5	$0-2$	430	4
gn13e39d5w0w2	5	$0-2$	361	4
gn10e22d5w0w5	5	$0-5$	0	1
gn10e28d5w0w5	5	$0-5$	9	2
gn10e33d5w0w5	5	$0-5$	1	3
gn11e41d5w0w5	5	$0-5$	33	4
gn13e39d5w0w5	5	$0-5$	16	5

Table XIII. Calibration test set: Tabu Search Heuristic 2 minimizing $(x - \widehat{x})^2$, where for each instance, $\widehat{x}$ is an interior rational non-integer point that is the image of some base. Here $d = 2, 3,$ and 5 and all tests run on Fuzzy. Shown are the number of successes out of 1000 Tabu Searches for Tabu Search limits of $1, 5, 20, 100$ and seconds (in parenthesis) to perform the 1000 searches.

Name	Dimension	Weight Range	Tabu Limit 1	Tabu Limit 5	Tabu Limit 20	Tabu Limit 100
gn10e22d2w0w20	2	$0-20$	16(1)	491(2)	711(7)	734(27)
gn10e28d2w0w20	2	$0-20$	45(0)	680(3)	958(9)	1000(36)
gn10e33d2w0w20	2	$0-20$	17(0)	449(2)	934(11)	998(40)
gn11e41d2w0w20	2	$0-20$	15(1)	619(3)	970(13)	1000(55)
gn13e39d2w0w20	2	$0-20$	20(1)	478(4)	897(16)	1000(65)
gn10e22d2w0w100	2	$0-100$	0(0)	50(3)	43(8)	52(34)
gn10e28d2w0w100	2	$0-100$	0(0)	13(3)	128(11)	250(52)
gn10e33d2w0w100	2	$0-100$	2(1)	6(3)	58(11)	198(51)
gn11e41d2w0w100	2	$0-100$	14(0)	34(4)	107(17)	334(92)
gn13e39d2w0w100	2	$0-100$	1(1)	22(4)	124(20)	483(97)
gn10e22d2w0w1000	2	$0-1000$	0(1)	0(2)	3(9)	98(45)
gn10e28d2w0w1000	2	$0-1000$	0(1)	0(3)	0(11)	90(59)
gn10e33d2w0w1000	2	$0-1000$	0(0)	0(3)	0(12)	0(61)
gn11e41d2w0w1000	2	$0-1000$	0(1)	0(4)	0(18)	16(90)
gn13e39d2w0w1000	2	$0-1000$	0(1)	0(5)	7(19)	3(104)
gn10e22d3w0w20	3	$0-20$	0(0)	32(2)	54(11)	205(49)
gn10e28d3w0w20	3	$0-20$	0(0)	23(4)	34(13)	381(62)
gn10e33d3w0w20	3	$0-20$	0(1)	3(3)	47(13)	125(49)
gn11e41d3w0w20	3	$0-20$	0(1)	25(5)	213(19)	645(94)
gn13e39d3w0w20	3	$0-20$	0(0)	37(5)	148(21)	554(106)
gn10e22d3w0w100	3	$0-100$	0(1)	22(2)	25(10)	233(46)
gn10e28d3w0w100	3	$0-100$	0(0)	1(3)	12(11)	32(49)
gn10e33d3w0w100	3	$0-100$	0(0)	0(4)	0(16)	0(79)
gn11e41d3w0w100	3	$0-100$	0(0)	0(5)	0(21)	2(112)
gn13e39d3w0w100	3	$0-100$	0(0)	0(5)	0(21)	0(85)
gn10e22d3w0w1000	3	$0-1000$	1(0)	12(3)	77(9)	134(34)
gn10e28d3w0w1000	3	$0-1000$	0(0)	0(3)	0(13)	0(70)
gn10e33d3w0w1000	3	$0-1000$	0(1)	0(3)	0(15)	10(71)
gn11e41d3w0w1000	3	$0-1000$	0(0)	0(5)	0(20)	0(103)
gn13e39d3w0w1000	3	$0-1000$	0(1)	0(5)	0(23)	0(106)
gn10e22d5w0w1	5	$0-1$	91(1)	949(2)	1000(8)	1000(33)
gn10e28d5w0w1	5	$0-1$	125(0)	877(3)	1000(9)	1000(43)
gn10e33d5w0w1	5	$0-1$	70(0)	1000(3)	1000(10)	1000(47)
gn11e41d5w0w1	5	$0-1$	40(0)	994(4)	1000(14)	1000(70)
gn13e39d5w0w1	5	$0-1$	19(1)	734(6)	942(18)	1000(80)
gn10e22d5w0w2	5	$0-2$	1(0)	157(2)	283(8)	371(33)
gn10e28d5w0w2	5	$0-2$	20(0)	492(3)	851(11)	992(45)
gn10e33d5w0w2	5	$0-2$	6(1)	629(5)	879(12)	896(48)
gn11e41d5w0w2	5	$0-2$	7(0)	643(6)	924(18)	956(74)
gn13e39d5w0w2	5	$0-2$	3(1)	616(6)	896(20)	981(83)
gn10e22d5w0w5	5	$0-5$	0(0)	0(3)	70(10)	143(37)
gn10e28d5w0w5	5	$0-5$	0(1)	39(4)	118(16)	643(80)
gn10e33d5w0w5	5	$0-5$	0(0)	20(4)	115(17)	566(78)
gn11e41d5w0w5	5	$0-5$	0(0)	47(6)	144(25)	555(127)
gn13e39d5w0w5	5	$0-5$	0(1)	45(7)	180(27)	567(118)

Table XIV. Solids, Heuristic 6 ($DFBFS$). 100 searches, depth 4, boundary retry limit 100 and interior retry limit of 10000. Note that Matsui's algorithm was only able to enumerate the spanning trees of the first six graphs.

Name	Nodes	Edges	Weight Range	Computer	Seconds	#Spanning Trees	#Projected Trees	#Trees Found by DFBFS	Percent
Solid-tetrahedron-d2w0w20	4	6	$0-20$	mocr04	0	16	16	16	100.00
Solid-cube-d2w0w20	4	6	$0-20$	mocr04	0	384	321	320	99.69
Solid-ocahedron-d2w0w20	6	12	$0-20$	mocr04	0	384	296	292	98.65
Solid-truncatedtetrahedron-d2w0w20	12	18	$0-20$	mocr04	0	6,000	1,947	1,931	99.18
Solid-cuboctahedron-d2w0w20	12	24	$0-20$	mocr04	2	331,776	6,291	6,209	98.70
Solid-icosahedron-d2w0w20	12	30	$0-20$	mocr04	3	5,184,000	12,190	11,911	97.71
Solid-dodecahedron-d2w0w20	20	30	$0-20$	mocr04	4	5,184,000	11,708	11,625	99.29
Solid-truncatedoctahedron-d2w0w20	24	36	$0-20$	mocr04	11	101,154,816		16,882	
Solid-srhombicuboctahedron-d2w0w20	24	48	$0-20$	mocr04	31	301×10^{9}		35,132	
Solid-snubcube-d2w0w20	24	60	$0-20$	mocr04	56	89×10^{12}		56,992	
Solid-icosidodecahedron-d2w0w20	30	60	$0-20$	mocr04	78	208×10^{12}		61,926	
Solid-grhombicuboctahedron-d2w0w20	48	72	$0-20$	mocr04	201	12×10^{15}		73,166	
Solid-truncatedicosahedron-d2w0w20	60	90	$0-20$	mocr04	623	375×10^{18}		112,292	
Solid-rhomicosidodecahedron-d2w0w20	60	120	$0-20$	mocr04	1,266	201×10^{27}		166,622	
Solid-snubdodecahedron-d2w0w20	60	150	$0-20$	mocr04	1,955	438×10^{33}		199,764	
Solid-tetrahedron-d2w0w100	4	6	$0-100$	mocr04	1	16	16	16	100.00
Solid-octahedron-d2w0w100	6	12	$0-100$	mocr04	0	384	383	380	99.22
Solid-cube-d2w0w100	8	12	$0-100$	mocr04	0	384	383	383	100.00
Solid-truncatedtetrahedron-d2w0w100	12	18	$0-100$	mocr04	1	6,000	5,812	5,782	99.48
Solid-cuboctahedron-d2w0w100	12	24	$0-100$	mocr04	13	331,776	93,982	90,788	96.60
Solid-icosahedron-d2w0w100	12	30	$0-100$	mocr04	27	5,184,000	207,447	200,641	96.72
Solid-dodecahedron-d2w0w100	20	30	$0-100$	mocr04	49	5,184,000	217,882	211,169	96.92
Solid-truncatedoctahedron-d2w0w100	24	36	$0-100$	mocr04	102	101,154,816	353,262	339,369	96.07
Solid-srhombicuboctahedron-d2w0w100	24	48	$0-100$	mocr04	233	301×10^{9}		793,894	
Solid-snubcube-d2w0w100	24	60	$0-100$	mocr04	322	89×10^{12}		1,017,998	
Solid-icosidodecahedron-d2w0w100	30	60	$0-100$	mocr04	518	208×10^{12}		1,335,581	
Solid-grhombicuboctahedron-d2w0w100	48	72	$0-100$	mocr04	1409	12×10^{15}		1,554,008	
Solid-truncatedicosahedron-d2w0w100	60	90	$0-100$	mocr04	3005	375×10^{18}		2,440,020	
Solid-rhombicosidodecahedron-d2w0w100	60	120	$0-100$	mocr04	5772	201×10^{27}		4,327,586	
Solid-snubdodecahedron-d2w0w100	60	150	$0-100$	mocr04	7878	438×10^{33}		5,278,828	
Solid-tetrahedron-d2w0w1000	4	6	$0-1000$	mocr04	0	16	16	16	100.00
Solid-octahedron-d2w0w1000	6	12	$0-1000$	mocr04	1	384	384	376	97.92
Solid-cube-d2w0w1000	8	12	$0-1000$	mocr04	0	384	384	383	99.74
Solid-truncatedtetrahedron-d2w0w1000	12	18	$0-1000$	mocr04	1	6,000	6,000	5,662	94.37
Solid-cuboctahedron-d2w0w1000	12	24	$0-1000$	mocr04	34	331,776	331,776	318,156	95, 89
Solid-icosahedron-d2w0w1000	12	30	$0-1000$	mocr04	328	5,184,000	5,184,000	3,263,800	62, 96
Solid-dodecahedron-d2w0w1000	20	30	$0-1000$	mocr04	628	5,184,000	5,184,000	3,606,194	69.56
Solid-truncatedoctahedron-d2w0w1000	24	36	$0-1000$	mocr04	2,637	101,154,816	101,154,816	13,024,774	12.88
Solid-srhombicuboctahedron-d2w0w1000	24	48	$0-1000$	mocr04	9,354	301×10^{9}		48,407,588	
Solid-snubcube-d2w0w1000	24	60	$0-1000$	mocr04	32,481	89×10^{12}		83,600,261	

Table XV. Two experimental design problems P_{20} and P_{28} and comparing run times of IBM Blue Gene/P vs Fuzzy to find all projected bases. We used Heuristic 6 ($DFBFS$) with 100 searches, boundary retry limit 100, interior retry limit 10,000 and truncation depth 8.

Name	P_{20}	P_{28}
Columns(#matroid elements)	100	100
Rows(matroid rank)	20	28
Projected bases	13,816	19,193
Seconds for IBM Blue Gene/P	23,773	56,763
# Projected bases found by run 1 of Heuristic 6	13,757	19,193
Seconds for run 1 of Heuristic 6	923	2,557
# Projected bases found by run 2 of Heuristic 6	13,816	19,191
Seconds for run 2 of Heuristic 6	945	2,645

Table XVI. Experimental design problem P_{20} and comparing run times of IBM Blue Gene/P vs Fuzzy to find all projected bases. We partitioned the region of feasible projected bases into 12 disjoint regions and ran 12 separate concurrently running instances of Heuristic 3 (PT) on Fuzzy to find as many projected bases as possible. When running Heuristic 3 (PT) we used Heuristic 2 (TS) and tried each point 10 times with a Tabu limit of 20.

Region	#Projected Bases Found	Seconds
Box1	1,044	9,544
Box2	1,122	5,086
Box3	1,122	4,966
Box4	1,135	8,066
Box5	1,122	5,414
Box6	1,122	4,540
Box7	1,122	4,528
Box8	1,188	5,614
Box9	1,147	9,181
Box10	1,124	5,232
Box11	1,124	5,252
Box12	1,249	8,173
TOTAL	13,816	(Longest time)9,544

Table XVII. Machines Used in Experiments.

Name	#CPUS	Clock Speed	Memory
Fuzzy	16 AMD	2400 MHz	64GB
Truth	16 AMD	2400 MHz	64GB
Mathocr01	2 AMD	2000 MHz	4GB
Mathocr02	2 AMD	2000 MHz	4GB
Mathocr03	2 AMD	2000 MHz	4GB
Mathocr04	2 AMD	2000 MHz	12GB

REFERENCES

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. 1999. *LAPACK Users' Guide*, Third ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Barvinok, A. and Samorodnitsky, A. 2007. Random weighting, asymptotic counting, and inverse isoperimetry. *Israel Journal of Mathematics 158*, 159–191.

Berstein, Y., Lee, J., Maruri-Aguilar, H., Onn, S., Riccomagno, E., Weismantel, R., and Wynn, H. 2008. Nonlinear matroid optimization and experimental design. *SIAM Journal on Discrete Mathematics 22*, 901–919.

Berstein, Y., Lee, J., Onn, S., and Weismantel, R. 2009. Parametric nonlinear discrete optimization over well-described sets and matroid intersections. *Mathematical Programming (to appear)*.

Berstein, Y. and Onn, S. 2008. Nonlinear bipartite matching. *Discrete Optimization 5*, 53–65.

Borovik, A. V., Gelfand, I. M., and White, N. 2007. Coxeter matroid polytopes. *Annals of Combinatorics 1,* 1 (December), 123–134.

De Loera, J., Haws, D. C., O'Hair, A., and Lee, J. 2009. Matroid Optimization: Combinatorial Heuristics and Algorithms. Available from URL `http://www.coin-or.org/projects/MOCHA.xml`.

Ehrgott, M. 1996. On matroids with multiple objectives. *Optimization 38,* 1, 73–84. Multicriteria optimization and decision theory (Holzhau, 1994).

Ehrgott, M. and Gandibleux, X. 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum 22,* 4, 425–460.

Fries, A. and Hunter, W. 1980. Minimum aberration 2^{k-p} designs. *Techno.* 22, 601–608.

Garey, M. and Johnson, D. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, San Francisco.

Gelfand, I. M., Goresky, M., MacPherson, R. D., and Serganova, V. V. 1987. Combinatorial geometries, convex polyhedra, and Schubert cells. *Adv. Math. 63*, 301–316.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research 13*, 533–549.

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics* 5, 287–326.

Granlund, T. and et al. 2009. GNU multiple precision arithmetic library. `http://gmplib.org/`.

Gunnels, J., Lee, J., and Margulies, S. 2008. Efficient high-precision dense matrix algebra on parallel architectures for nonlinear discrete optimization. *IBM Research Report RC24682 (October, 2008)*.

Holtzmann, C. and Harary, F. 1972. On the tree graph of a matroid. *SIAM Journal of Applied Math 22,* 2, 187–193.

Knowles, J. D. and Corne, D. W. 2002. Enumeration of Pareto optimal multi-criteria spanning trees - a proof of the incorrectness of Zhou and Gen's proposed algorithm. *European Journal of Operational Research 143,* 3, 543–547.

Lee, J. 2004. *A first course in combinatorial optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press.

Matsui, T. 1997. A flexible algorithm for generating all the spanning trees in undirected graphs. *Algorithmica 18,* 4, 530–543.

Okamoto, Y. and Uno, T. 2007. A polynomial-time-delay polynomial-space algorithm for enumeration problems in multi-criteria optimization. *Lecture Notes in Computer Science 4835/2007*, 609–620.

Onn, S. 2003. Convex matroid optimization. *SIAM Journal on Discrete Mathematics 17*, 249–253.

Schrijver, A. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

Warburton, A. 1985. Worse case analysis of greedy and related heuristics for some min-max combinatorial optimization problems. *Mathematical Programming 33,* 2 (November), 234–241.