Boosting Efficiency for Computing the Pareto Frontier on Tree Structured Networks

Jonathan M. Gomes-Selman[1], Qinru Shi[2], Yexiang Xue[3], Roosevelt García-Villacorta[4], Alexander S. Flecker[4], and Carla P. Gomes[3(✉)]

[1] Department of Computer Science, Stanford University, Stanford, USA
jgs8@stanford.edu
[2] Center for Applied Mathematics, Cornell University, Ithaca, USA
qs63@cornell.edu
[3] Department of Computer Science, Cornell University, Ithaca, USA
yx247@cornell.edu, gomes@cs.cornell.edu
[4] Department of Ecology and Evolutionary Biology, Cornell University, Ithaca, USA
rg676@cornell.edu, asf3@cornell.edu

Abstract. Multi-objective optimization plays a key role in the study of real-world problems, as they often involve multiple criteria. In multi-objective optimization it is important to identify the so-called Pareto frontier, which characterizes the trade-offs between the objectives of different solutions. We show how a divide-and-conquer approach, combined with batched processing and pruning, significantly boosts the performance of an exact and approximation dynamic programming (DP) algorithm for computing the Pareto frontier on tree-structured networks, proposed in [18]. We also show how exploiting restarts and a new instance selection strategy boosts the performance and accuracy of a mixed integer programming (MIP) approach for approximating the Pareto frontier. We provide empirical results demonstrating that our DP and MIP approaches have complementary strengths and outperform previous algorithms in efficiency and accuracy. Our work is motivated by a problem in computational sustainability concerning the evaluation of trade-offs in ecosystem services due to the proliferation of hydropower dams throughout the Amazon basin. Our approaches are general and can be applied to computing the Pareto frontier of a variety of multi-objective problems on tree-structured networks.

Keywords: Multi-objective optimization · Pareto frontier
Approximation algorithms · Dynamic programming
Mixed-integer programming

J.M. Gomes-Selman and Q. Shi—These authors are contributed Equally.

W.-J. van Hoeve (Ed.): CPAIOR 2018, LNCS 10848, pp. 263–279, 2018.
https://doi.org/10.1007/978-3-319-93031-2_19

1 Introduction

In recent years there has been a rapid proliferation of hydropower dams throughout the Amazon basin, which dramatically affects a variety of ecosystem services provided by the river network such as biodiversity, nutrient and sediment transport, freshwater fisheries, navigation, and energy production [4,17,21,22] (see Fig. 1). Hydropower dam placement is a good example of a challenging real-world problem in computational sustainability [6], which often involves multiple objective problems concerning the balancing of environmental, economic, and societal needs. More concretely, hydropower dam placement is a multi-objective optimization problem concerning the placement of dams throughout a river network, which is naturally a tree-structured network, trading-off various ecological, social, and economic goals. In multi-objective optimization, the so-called *Pareto frontier* captures the trade-offs among multiple objectives. The Pareto frontier is the set of all *Pareto optimal solutions*; a solution is considered Pareto optimal if its vector of objective values is not *dominated* by any other feasible solution. See Fig. 1 for an example of a 2-dimensional Pareto frontier.

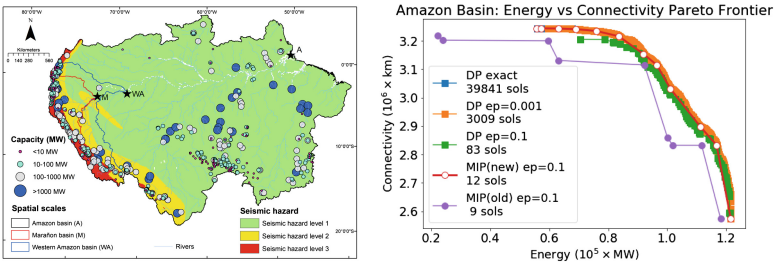


Fig. 1. Left panel: Amazon basin: around 300 hydropower dams are proposed or planned. Three objectives are depicted: (1) Energy (dot sizes denote dam capacity in MW); (2) Longitudinal connectivity (continuous unobstructed river segments from the root of the basins, which are marked with stars); and (3) Seismic risk (map background colors). **Right panel:** The approximate DP ($\epsilon = 0.001$) overlaps the exact DP. The new MIP approach is substantially better than the previous MIP approach ($\epsilon = 0.1$) and its solutions are on the exact Pareto curve, slightly better than the DP approximation for the same $\epsilon = 0.1$. For a given ϵ, DP produces substantially more Pareto solutions than MIP.

Recently there has been considerable interest in the study of multi-objective optimization problems (see e.g., [2,3,8,10,11,13,14,16,20]). Existing approaches are primarily heuristic, based on local search or evolutionary algorithms, without theoretical guarantees, and do not exploit the tree structure. [5] provides a constraint programming exact algorithm which is extended by [12] with large neighborhood search. Both algorithms are designed for general problems. [1] provides data structures to store Pareto optimal policies in an exact algorithm.

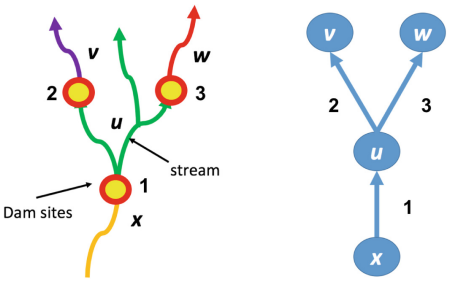
This paper focuses on *computing the Pareto frontier, both exact and with approximation guarantees, on tree-structured networks*. In [18] we proposed a dynamic programming algorithm for trees which computes the exact Pareto frontier, as well as a rounding technique applied to the exact dynamic programming algorithm that provides a fully polynomial-time approximation scheme (FPTAS). The FPTAS finds a solution set of polynomial size, which approximates the Pareto frontier within an arbitrary small ϵ factor and runs in time that is polynomial in the size of the instance and $1/\epsilon$. We also formulated the problem of optimizing the placement of dams as a mixed integer programming problem (MIP) and used it to approximate the Pareto frontier. While the results in [18] are encouraging, there is room for improvement.

Our Contributions: **(1)** A key component of our DP algorithm is the pruning of dominated solutions. We provide a **divide-and-conquer approach** that **significantly improves the efficiency of the pruning of dominated solutions** and outperforms the previous approach, leading to **speed-ups of two to three orders of magnitude**, in practice; **(2)** To cope with the large memory requirements of a multi-objective Pareto frontier, we propose **batching to identify and prune dominated solutions incrementally**, scaling up to much larger problems; **(3)** We also propose a **new MIP based approximation scheme** that exploits restarts and a new instance selection strategy, which boosts the performance and accuracy of the previous MIP approach for approximating the Pareto frontier. **(4)** We design a visualization tool of our results intended for decision makers. **(5)** We provide empirical results showing that **our proposed algorithms significantly outperform previous approaches.**

Preview of Results: Our DP and MIP Pareto frontier algorithms are complementary and scale up to much larger real-world instances than previous algorithms: **the DP can now approximate the Pareto frontier for the entire Amazon basin**, when optimizing for energy, connectivity (a proxy for e.g., unimpeded fish migrations and transportation), seismic risk, and sediment, in around 5 days, with a coverage of $2,193,314$ non-dominated solutions, with the guarantee that the solutions are within at most 5% of the true optimum ($\epsilon = 0.05$); in less than 6 h, the DP provides a coverage of $491,578$ non-dominated solutions ($\epsilon = 0.1$); in around 6.5 min, the DP provides a coverage of $23,019$ non-dominated solutions ($\epsilon = 0.25$); for the same $\epsilon = 0.25$, the MIP approach approximates the Pareto frontier in around 25 min, with a smaller coverage of 95 non-dominated solutions, but the MIP approach provides more flexibility when considering additional constraints and, in practice, its solutions tend to be closer to the exact Pareto frontier for a given ϵ. Our overall goal is to enable more informed decisions concerning the trade-offs of multiple objectives of optimization problems.

2 Problem Formulation

In this section, we first introduce the hydropower dam placement problem as an example of a multi-objective optimization problem on a tree structured network. Then, we show the general formulation of such problems.



(a) River Network (b) Directed Rooted Tree

Fig. 2. Converting a river network (left) into a more compact directed rooted tree (right: x is the root). Each contiguous region of the river network (represented by different colors, and labeled x, u, v, w) is converted into a node, also referred to as a hypernode (labeled with the corresponding letter, x, u, v, w) in the tree network. Each potential dam site (represented by a red-yellow circle) is represented by an edge in the directed rooted tree. (Color figure online)

2.1 Hydropower Dam Placement Problem

We are given a set of planned dams and need to decide the optimal subset of dams to build. We refer to this problem as the hydropower dam placement problem. We first point out that a river network is a directed tree-structure network and that, for the purposes of our hydropower dam placement problem, we don't need to explicitly consider every river segment. So, we first abstract the river network and potential dam locations into a more compact directed rooted tree that captures the key problem information. Each contiguous section of the river network uninterrupted by existing or potential dam locations is represented by a node (we also call it hypernode to emphasize that it encapsulates a river sub network). Each existing or potential dam location is represented by a directed edge pointing from downstream to upstream. See Fig. 2 for an example of our conversion of a river network into a more compact directed rooted tree.

A policy (or solution) π is a subset of potential dam sites to be built. We can encode many environmental and economical objectives as a function of π. In this paper, we focus on the following four objectives:

Energy (E)**:** Given a solution π, the total hydropower produced by the selected dams is $E(\pi) = \sum_{e \in \pi} h_e$, where h_e is the hydropower of the dam represented by edge e. We want to maximize this objective.

Longitudinal Connectivity (C)**:** For a given solution π, the connectivity of a river network is measured by the total length of the unobstructed stream segments that one can travel starting from the root (river mouth) without passing any dam site in π. We want to maximize this objective.

Sediment (S_d)**:** For a given solution π, this objective represents the total amount of sediment transported to the river mouth (the ocean in the case of the entire Amazon basin). We assume that each node produces a fixed amount of sediment and each dam traps a certain percentage of the sediment from upstreams. We want to maximize this objective.

Seismic Risk (S_e)**:** Each dam is associated with a seismic risk factor computed based on its location and its capacity. Given a solution π, this objective is just the sum of seismic risk factors of all dams in π. We want to minimize this objective.

The goal of the hydropower dam placement problem is then to optimize the objective function: $(E(\pi), C(\pi), S_d(\pi), S_e(\pi))$. Here we are not just looking for a single solution. For every possible solution, we say that the solution is optimal as long as there does not exist another solution that is superior in every aspect.

2.2 General Formulation

In general, a multi-objective optimization problem is to optimize a given multi-objective function: $(z^1(\pi), z^2(\pi), \ldots, z^d(\pi))$, where the value of each function z^i depends on a common *solution* π, also referred to as a *policy*. Without loss of generality, we only consider the problem of *maximizing* objective functions. Minimizing objective functions can be treated in a similar fashion.

Pareto Dominance: Given two policies π and π', we say that π dominates π' if the following two conditions hold: (1) for all i, $z^i(\pi) \geq z^i(\pi')$; (2) at least one strict inequality holds for some i.

Pareto Frontier: A Pareto optimal policy is the one that is not dominated by any other policies. The Pareto frontier is the set containing all Pareto optimal policies.

An Example: Consider a multi-objective function (z^1, z^2, z^3) and policies π_1, π_2, and π_3, leading to: $(z^1(\pi_1), z^2(\pi_1), z^3(\pi_1)) = (5, 7, 10)$, $(z^1(\pi_2), z^2(\pi_2), z^3(\pi_2)) = (4, 7, 9)$, and $(z^1(\pi_3), z^2(\pi_3), z^3(\pi_3)) = (6, 6, 9)$. π_1 dominates π_2 because it has higher or equal values in all objectives, with some objectives with higher values. π_1 does not dominate π_3 because of the first objective. π_3 does not dominate π_1 because of the second and third objectives. π_1 and π_3 are Pareto optimal and form the Pareto frontier.

Multi-objective Function on a Tree: We now give a formal definition of the multi-objective optimization problem on a tree structured network: the hydropower dam placement problem is a particular example. Given a *tree structured network* (such as a river network), the objective function z^i is defined recursively. Node rewards $r_v^1, \ldots, r_v^d$ are associated with each node v in the tree. The objective function defined on a leaf node v is its corresponding reward, i.e.,

$z_v^i(\pi) = r_v^i$. Each edge is associated with a transfer coefficient that is affected by whether the corresponding dam is built or not. If the dam represented by (u, v) is built, then (u, v) has a transfer coefficient of p_{uv}^i; otherwise, q_{uv}^i. Also associated with each edge (u, v) is a reward s_{uv}^i and an indicator variable denoting whether the corresponding edge is in π or not. The objective function on a non-leaf node u is defined recursively:

$$z_u^i(\pi) = r_u^i + \sum_{v \in ch(u)} I(uv \in \pi) s_{uv}^i + \sum_{v \in ch(u)} \left(I(uv \in \pi) p_{uv}^i + I(uv \notin \pi) q_{uv}^i \right) z_v^i(\pi). \quad (1)$$

Here, $I(\cdot)$ is an indicator function. $ch(u)$ is the child set of u. The objective function for the entire tree network $\mathcal{T}$ is the function at the root node s, i.e., $z^i(\pi) = z_s^i(\pi)$. Given a multi-objective function defined on a tree network $\mathcal{T}$, our **multi-objective optimization problem on a tree structured network** is to find the Pareto frontier consisting of all non-dominated policies, which is NP-hard even though it is defined on a tree. See [18] for further details.

Application to the Hydropower Dam Placement Problem: In the hydropower dam placement problem, when modeling **connectivity** (i.e., $i =$ connectivity), we set r_u^i to be the total lengths of all stream segments in the region represented by node u. We set $p_{uv}^i = 0$ and $q_{uv}^i = 1$; that is, we either acquire all upstream segments (when the dam corresponding to edge (u, v) is not built) or lose all of them (when the dam is built). We set $s_{uv}^i = 0$. When modeling energy (i.e., $i =$ energy), we set $s_{uv}^i = h_{uv}$, in which h_{uv} is the hydropower produced by the dam site (u, v). r_u^i is set to 0, p_{uv}^i and q_{uv}^i are both set to 1.

3 DP-Based Pareto Frontier

In [18] we proposed a dynamic programming (DP) algorithm (shown in Algorithm 1) that recursively computes the Pareto optimal partial solutions from leaf nodes up to the root. The key insight is that at a given node u we only need to keep the Pareto optimal partial solutions [18]. To increase incremental pruning, we convert the original tree into an equivalent binary tree. Given a binary tree, we first compute Pareto optimal solutions for the two children of u (line 6 and 7), enumerate the partial policies from the children and consider four different combinations of whether to include each of the edges from the children, computing the objective values based on Eq. 1, and adding them to the policy set P (line 8). We then remove all dominated policies (line 9). So, the remaining policies are Pareto optimal for the parent node.

In [18] we also proposed a rounding scheme applied to the exact DP algorithm, which provides a fully polynomial-time approximation scheme (FPTAS) that finds a polynomially succinct policy set, which approximates the Pareto frontier within an arbitrary small ϵ factor and runs in time polynomial in the size of the instance and $1/\epsilon$. The key idea is to project objective values that are ϵ-close into one, which decreases the number of Pareto solutions.

Algorithm 1. $\texttt{Pareto}_{\mathcal{T}}(u)$**:** compute the Pareto frontier for the value function defined on the subtree of $\mathcal{T}$ rooted at node u.

```
1  if is_leaf(u) then
2  |  return {(r_u^1, ..., r_u^m)};
3  else
4  |  l ← u.left_child;
5  |  r ← u.right_child;
6  |  Pleft ← Pareto_T(l);
7  |  Pright ← Pareto_T(r);
8  |  S_u ← the set of all possible partial solutions at u obtained by combining
   |  solutions from Pleft and Pright and possible policies on (u, l) and (u, r);
9  |  return Non_Dominated(S_u);
10 end
```

3.1 Divide-and-Conquer for Identifying Dominated Solutions

The major runtime bottleneck of Algorithm 1 is pruning the dominated solutions (line 9). Let d be the number of objectives. Assume we generate n partial solutions and get m non-dominated partial solutions, then the naive pruning step takes $O(mnd)$ time. Here we describe a strategy that *significantly boost the efficiency of the overall DP algorithms for computing the Pareto frontier*, which leverages the dimensionality of solutions to efficiently identify the subset of non-dominated solutions from a set of candidate solutions. The new divide-and-conquer based algorithm for finding the non-dominated solutions runs in $O(n(\log n)^{d-1})$ if we use comparison-based sort or $O(n(\log n)^{d-2})$ if the data is stored in the Lattice Latin Hypercube (LLH) form [19]. Here d is the number of criteria we are considering. This algorithm is inspired by an approach proposed in [19].

To simplify the description of our algorithms, we assume that the values of each criterion never repeat. In practice, it is fairly trivial to consider the corner case. Specifically, when splitting the set S based on the dth criterion, we implemented a modified sorting routine that sorts the solutions in lexicographic order, based on the d^{th}criterion. If two solutions have the same d^{th} criterion, we sort them based on the $(d-1)^{th}$ criterion, etc. Note that if two solutions are equal for all criteria then their ordering does not matter.

When the number of objectives is two, we use the method as shown in Algorithm 2. The idea is to sort the solutions based on the first criterion (good ones first). The first solution must be Pareto optimal. Then, we go through the list of solutions sequentially and look for solutions with better second objective than the last non-dominated solution.

When the number of objectives $d \geq 3$, we use our divide-and-conquer based recursive algorithm shown in Algorithm 3. The first step is to split the set of solutions S into two sets A and B of approximately the same size based on the last criterion, so that solutions in A have better last criterion than solutions in B (line 8). The splitting procedure is shown in Algorithm 4. Then, we recursively

Algorithm 2. `Non_Dominated_2D`(S)**:** given a set S of 2-dimensional partial solutions, find the set of non-dominated solutions in S.

```
1  Sort solutions in S by their first element, in descending order if we aim to
   maximize the element, in ascending order otherwise;
2  P ← {S[1]};
3  foreach s ∈ S[2 :] do
4  |  if s is not dominated by the last element of P then
5  |  |  Append s to P;
6  |  end
7  end
8  return P;
```

Algorithm 3. `Non_Dominated`(S)**:** given a set S of d-dimensional partial solutions ($d \geq 2$), find the set of non-dominated solutions in S.

```
1  d ← dimensionality of solutions in S;
2  n ← number of solutions in S;
3  if n = 1 then
4  |  return S
5  else if d = 2 then
6  |  return Non_Dominated_2D(S)
7  else
8  |  A, B ← Split(S, d);
9  |  A' ← Non_Dominated(A);   // solutions in A' are non-dominated in S.
10 |  B' ← Non_Dominated(B);
11 |  return A' ∪ Marry(A', B', d − 1);
12 end
```

identify the non-dominated solutions from A and B (line 9 and 10). We know that the non-dominated solutions from set A' are also non-dominated in S, but the same statement may not be true for non-dominated solutions from B'. Thus, the last step is to find the solutions from set B' that are not dominated by solutions from A' (line 11). Note that we already know that the dth objective of solutions in A' are better than the dth criterion of solutions in B', so we only need to consider the first $d-1$ criteria. To find non-dominated solutions in B', we introduce a slightly modified divide-and-conquer procedure shown in Algorithm 5.

The algorithm `Marry`(A, B, d') shown in Algorithm 5 returns the set of all solutions in B that are not dominated by any solution in A considering only the first d' criteria. The inputs A and B must be disjoint and no two solutions from the same set dominate one another. Let n be the total number of solutions in $A \cup B$. We split the set of solutions $A \cup B$ into two sets X and Y of approximately the same size based the d'th criterion, so that solutions in X have better d'th criterion than solutions in Y (line 7). Next, we consider the four disjoint subsets $X \cap A$, $X \cap B$, $Y \cap A$, $Y \cap B$. Note that they cover all solutions in $A \cup B$, and

Algorithm 4. `Split`(S, d)**:** given a set S of partial solutions, split S into two disjoint sets of roughly the same size based on the dth criterion of each solution.

```
1 Sort solutions in S by their dth criterion, in descending order if we aim to
  maximize the criterion, in ascending order otherwise;
2 A ← S[1 : ⌊n/2⌋];
3 B ← S − A ;
4 return A, B;     // A and B are disjoint and solutions in A have better
  dth criterion.
```

Algorithm 5. `Marry`(A, B, d')**:** consider only the first d' elements in each solution, return the set of all solutions in B that are not dominated by any solutions in A. A and B must be disjoint and no two solutions from the same set dominate one another.

```
1  n ← number of solutions in A ∪ B;
2  if d' = 2 then
3  |  return B ∩ Non_Dominated_2D(A ∪ B);        // a base case of recursion
4  else if A = ∅ or B = ∅ then
5  |  return B ;                          // also a base case of recursion
6  else
7  |  X, Y ← Split(A ∪ B, d');
8  |  B_x ← Marry(X ∩ A, X ∩ B, d');                    // n reduce in half
9  |  B_y ← Marry(Y ∩ A, Y ∩ B, d');                    // n reduce in half
10 |  B'_y ← Marry(X ∩ A, B_y, d' − 1);                  // d' reduce by 1
11 |  return B_x ∪ B'_y
12 end
```

$|(X \cap A) \cup (X \cap B)| \approx n/2 \approx |(Y \cap A) \cup (Y \cap B)|$. In line 8 and 9, we recursively call `Marry` on half-sized problems. Similarly as before, we know that the solutions in B_x are non-dominated in $A \cup B$, but we need to figure out which solutions in B_y are non-dominated in $A \cup B$. Solutions in B_y can only be dominated by solutions in $X \cap A$ since solutions inside B cannot dominate each other, so we only need to recursive call `Marry` on $X \cap A$ and B_y. Note that solutions in $X \cap A$ have better d'th criterion than solutions in B_y, so we only need to consider the first $d' - 1$ objectives. Finally, we return $B_x \cup B'_y$.

3.2 Runtime Analysis

For the runtime analysis, we assume that we use a sorting algorithm based on comparison, so the time complexity of `Non_Dominated`(S) is $O(n(\log n)^{d-1})$.

Proposition 1. *Given a set S of n 2-dimensional solutions,* `Non_Dominated_2D` (S) *runs in $O(n \log n)$ time.*

This is because the sorting step takes $O(n \log n)$ time and the for-loop takes $O(n)$ time.

Proposition 2. *Given a set S containing n solutions,* `Split`(S, d) *runs in $O(n \log n)$ time.*

This is also because the sorting step takes $O(n \log n)$ time.

Proposition 3. *Given two disjoint sets A and B such that no two solutions from the same set dominate one another and that $A \cup B$ contains n solutions,* `Marry`(A, B, d') *runs in $O(n(\log n)^{d'-1})$ time.*

Proof: We denote the runtime of `Marry`(A, B, d') as $t(n, d')$. For the base case $d' = 2$, the proposition obviously holds. For $d'_0 \geq 3$, assume the proposition holds for $d' < d'_0$, which means that $t(n, d'_0 - 1) = O(n(\log n)^{d'_0 - 1 - 1}) = O(n(\log n)^{d'_0 - 2})$ for any positive integer n. Now we consider cases where $n = 2^k$ for some positive integer k and $d' = d'_0$. The major components of `Marry` are: a `Split` step ($O(n \log n)$ time), two half sized `Marry` steps ($2\ t(n/2, d'_0)$ time), and a `Marry` step with dimension reduced by one ($t(n, d'_0 - 1)$ time). With induction, we know that $t(n, d'_0 - 1) = O(n(\log n)^{d'_0 - 2})$. For any positive integer k and $n = 2^k$, we have

$$\begin{aligned} t(2^k, d'_0) &= O(2^k \log(2^k)) + 2 \cdot t((2^k)/2, d'_0) + t(2^k, d'_0 - 1) \\ &= 2 \cdot t(2^{k-1}, d'_0) + O(2^k \cdot k^{d'_0 - 2}). \end{aligned}$$

Then, by induction on k, we can prove the following statement

$$t(2^k, d'_0) = O(n(\log n)^{d'_0 - 1}).$$

Since the runtime of `Marry` increases monotonically with n, the proposition also holds when n is not a power of 2. Hence, `Marry`(A, B, d') runs in $O(n(\log n)^{d'-1})$ time.

Proposition 4. *Given a set S containing n d-dimensional solutions ($d \geq 3$),* `Non_Dominated`(S) *runs in $O(n(\log n)^{d-1})$ time.*

Proof: When $d \leq 2$, the proposition clearly holds. For $d \geq 3$, we denote the runtime as $T(n, d)$. Similarly as in the proof of Proposition 3, we have

$$T(2^k, d) = 2 \cdot T(2^{k-1}, d) + O(2^k \cdot k^{d-2}).$$

Then, by induction on k, we get

$$T(2^k, d) = O(n(\log n)^{d-1}).$$

Since the runtime of `Non_Dominated(S)` increases monotonically with n, the proposition also holds when n is not a power of 2. Hence, `Non_Dominated`(S) runs in $O(n(\log n)^{d-1})$ time.

3.3 Implementation Notes

Split: The split procedure shown in Algorithm 4 can also be implemented using an $O(n)$ find median algorithm. However, the numerous steps of copying arrays and creating new arrays in the $O(n)$ find median algorithm are hard to implement and perform poorly in practice. Hence, we chose to use sorting to work "in-place" on the sets of solutions. Each time we drop a dimension we must create a new array sorted based on that dimension and then in the recursive process we simply keep track of the location within the array that we are working on. We found that in practice sorting and working in place give us much better performance.

Batching: Our new divide-and-conquer algorithm for pruning dominated solutions considerably speeds up the DP algorithm and allow us to solve problems on much larger networks, with higher precision, and with more objectives. However, the number of solutions to evaluate grows exponentially with the number of objectives and memory soon becomes a problem. For example, for the entire Amazon basin, for four criteria, with a precision of $\epsilon = 0.01$, the algorithm has to evaluate $144,823,974,336$ partial solutions at a single node of the tree, which is way beyond the memory available. To circumvent this problem, we introduced a batching process: at each tree node, instead of evaluating all possible solutions at once, we feed them to `Non_Dominated` in smaller batches of size $K = 10^7$. Then, we run `Non_Dominated` on the set of all non-dominated solutions from each batch. In practice, this batching routine actually also speeds up the DP algorithm. In the future we plan to consider different batching strategies and also parallel batching, which can be done in a straightforward way.

4 MIP-Based Pareto Frontier

We also proposed a MIP formulation (see Fig. 3) and a scheme for ϵ-approximating the Pareto-frontier of a multi-objective optimization problem in [18]. The key idea is to divide the space of objectives into small hyper-rectangles and query whether there exists a **feasible** solution in each hyper-rectangle. Then, from each feasible hyper-rectangle, we find one solution and form a set S of all the solutions we find. Under the condition that for each dimension, the upper bound of each hyper-rectangle is $(1 + \epsilon)$ of the lower bound, the set of non-dominated solutions from S forms an ϵ-approximate Pareto-frontier [9].

In this paper we exploit restart strategy and introduce a new scheme to reduce the number of MIPs to solve. We first **optimize** for one of the objectives. We divide the space of the remaining objectives into small hyper-rectangles. Specifically, the hyper-rectangles are designed to satisfy the condition that, for each dimension, the upper bound is $(1 + \epsilon)$ of the lower bound (assuming the objectives are always positive values). For each cell, we formulate a MIP to find the solution in that cell that **optimizes** the target objective if a feasible solution exists. We form a set S of all the solutions found by MIP. Under the assumption

$$
\begin{aligned}
&\textbf{Minimize: } d \\
&\textbf{subject to: } d = 0 \ \textbf{(d is a dummy variable)} \\
&\hat{C} \leq C \leq (1+\epsilon)\hat{C};\ \hat{S_d} \leq S_d \leq (1+\epsilon)\hat{S_d} \\
&\hat{E} \leq E \leq (1+\epsilon)\hat{E};\ \hat{S_s} \leq S_s \leq (1+\epsilon)\hat{S_s} \\
&C = \sum_{v \in V} n_v c_v;\ S_s = \sum_{e \in E} \pi_e r_e \\
&E = \sum_{e \in E} \pi_e h_e;\ S_d = \sum_{v \in V} y_v s_v \\
&n_v \in \{0,1\}, \forall v \in V; \pi_e \in \{0,1\}, \forall e \in E \\
&n_s = 1; n_u \geq n_v, \forall (u,v) \in E \\
&n_v \leq 1 - \pi_{u,v}, \forall (u,v) \in E \\
&n_v \geq n_u - \pi_{u,v}, \forall (u,v) \in E \\
&y_s = 1 \\
&y_v \leq y_u \text{ and } y_v \geq (1-p_e)y_u, \forall (u,v) \in E \\
&y_v \leq (1-p_e)y_u + (1-\pi_e), \forall e = (u,v) \in E \\
&y_v \geq y_u - \pi_e, \forall e = (u,v) \in E
\end{aligned}
$$

Fig. 3. MIP formulation of the dam placement problem for four criteria. $\hat{C}, \hat{E}, \hat{S_d}, \hat{S_s}$: bounds on the objectives. V: the set of all nodes; E: the set of all edges (dams); s: the root of the tree; $e = (u, v)$: u is downstream of v; c_v, s_v, r_e, and h_e: connectivity value, sediment production, seismic risk, and hydropower associated with each hypernode or dam, respectively; p_e: percentage of sediment trapped by dam e; π_e: indicator variable of whether the dam will be built; n_v: indicator variable of whether node v can be reached from the river mouth without passing a dam; y_v: continuous variable representing the percentage of the sediment produced at the node v not trapped by dams.

that we solve the MIPs optimally, the set of non-dominated solutions from S forms an ϵ-approximate Pareto-frontier. In practice, we repeat the above scheme as many times as the number of criteria, cycling through every objective as target objective to get better coverage and a more accurate approximation. See details of the MIP formulation in [18]. A key difference in this new scheme is that we always optimize for the target objective, instead of solving decision problems.

Theorem: Let P be the set of all solutions on the Pareto frontier. Let $\bar{P}$ be the set of non-dominated solutions from S. Then, $\bar{P}$ ϵ-approximates P.

Proof: Assume that we are optimizing for k objectives O_1, O_2, O_3, $\cdots$, and O_k where k is greater or equal to 2. Without loss of generality, assume we aim to maximize O_1. For any $\pi \in P$, assume $(O_2(\pi), O_3(\pi), \cdots, O_k(\pi))$ lies in the rectangular cell $[\hat{O_2}, (1+\epsilon)\hat{O_2}] \times [\hat{O_3}, (1+\epsilon)\hat{O_3}] \times \cdots \times [\hat{O_k}, (1+\epsilon)\hat{O_k}]$. Since there is already a solution π in the rectangular cell, MIP can find a solution π' in the same cell that optimizes O_1, which means that $O_1(\pi') \geq O_1(\pi)$ and π' ϵ-dominates π. If $\pi' \notin \bar{P}$, then there exists a $\pi'' \in \bar{P}$ that dominates π' and consequently ϵ-dominates π. Hence, $\bar{P}$ ϵ-approximates P.

Table 1. Sample of runtimes and number of solutions for the different methods. A (Amazon); WA (Western Amazon); and M (Marañon); (E energy; C connectivity; S_d sediment; S_e seismic risk). *mem* denotes memory limit. N/A denotes MIP cannot produce the exact Pareto frontier. We bolded several entries to highlight performance improvements.

B	Criteria	ϵ	DP orig (secs)	DP new (secs)	MIP orig (secs)	MIP new (secs)	DP #Sols	MIP #Sols
A	E, C	exact	***18291***	***254***	N/A	N/A	***39841***	N/A
A	E, C	0.001	72	14	***6432***	***48***	3020	894
M	E, S_d	exact	2077	64	N/A	N/A	25732	N/A
M	E, S_d	0.001	4	2	1day+	1day+	318	—
WA	E, S_d	exact	***46291***	***924***	N/A	N/A	***58808***	N/A
WA	E, S_d	0.001	30	13	1day+	2187	2668	1671
A	E, S_d	exact	mem	15153	N/A	N/A	177490	N/A
A	E, S_d	0.001	2368	226	1day+	1day+	7973	—
A	E, S_d	0.1	0.1	0.1	297	3359	83	24
A	E, S_e	exact	54581	335	N/A	N/A	72591	N/A
A	E, S_e	0.001	2471	83	***35050***	***31***	8737	1558
M	E, C, S_d	exact	***2day+***	***526***	N/A	N/A	***283898***	N/A
M	E, C, S_d	0.001	630	32	1day+	1day+	5563	—
WA	E, C, S_d	exact	mem	90251	N/A	N/A	3267859	N/A
WA	E, C, S_d	0.001	mem	1120	1day+	1day+	88710	—
WA	E, C, S_d	0.0025	1667	269	1day+	1day+	28804	—
WA	E, C, S_d	0.005	254	69	1day+	65638	12655	4129
A	E, C, S_d	0.005	mem	32175	1day+	1day+	758462	—
A	E, C, S_d	0.025	70348	607	1day+	1day+	48381	—
A	E, C, S_d	0.05	1680	58	1day+	1day+	12866	—
A	E, C, S_d	0.1	40	6	1day+	4503	4724	62
A	E, C, S_d	0.15	11	2	1day+	6025	2493	43
A	E, C, S_e	0.005	***mem***	***88246***	***1day+***	***4809***	***2274168***	***40981***
A	E, C, S_e	0.05	109910	2121	238	51	47978	581
M	E, C, S_d, S_e	exact	***mem***	***763150***	N/A	N/A	***23364120***	N/A
M	E, C, S_d, S_e	0.001	mem	53620	1day+	1day+	1479660	—
M	E, C, S_d, S_e	0.02	***278310***	***649***	1day+	1day+	15961	—
M	E, C, S_d, S_e	0.1	886	28	1day+	15484	1406	773
WA	E, C, S_d, S_e	0.01	mem	695153	1day+	1day+	3540829	—
WA	E, C, S_d, S_e	0.1	47704	1154	1day+	1day+	107087	—
WA	E, C, S_d, S_e	0.15	11712	424	1day+	13692	69422	296
A	E, C, S_d, S_e	0.05	mem	***437271***	1day+	1day+	***2193314***	—
A	E, C, S_d, S_e	0.1	mem	19510	1day+	1day+	491578	—
A	E, C, S_d, S_e	0.15	mem	7471	1day+	1day+	198772	—
A	E, C, S_d, S_e	0.2	74940	1333	1day+	1day+	47059	—
A	E, C, S_d, S_e	0.25	11358	410	1day+	1505	23019	95

We observed fat and heavy-tailed behavior in the MIP runtime distributions [7]. To improve performance, we run the MIP solver with a cutoff, using a geometric restart strategy that doubles the cutoff time in every run [7,15]. Our experiments show that the restart strategy significantly boosts performance.

5 Experimental Results

To test the performance of the new methods at different scales, we used three datasets: the Marañon, Western Amazon, and Amazon basins, with 107, 219, and 467 hypernodes, respectively (corresponding to 128801, 455156 and 4083059

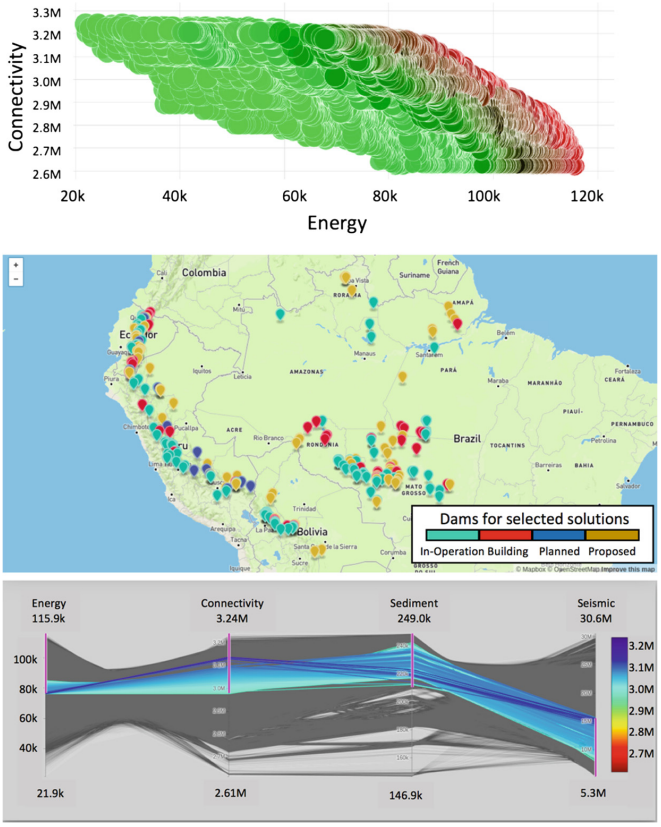


Fig. 4. Top: The visualization of the 4-dimensional Amazon Pareto frontier ($\epsilon = 0.4$). X axis: energy; Y axis: connectivity; Marker size: sediment; Color: seismic risk. **Middle:** The dam placement of a particular Pareto solution. **Bottom:** Parallel coordinate plot for the Amazon Pareto frontier ($\epsilon = 0.4$). The four axes are hydropower, connectivity, sediment and seismic risk. The color of each solution is based on its hydropower output. The plot displays only **1440** solutions due to the bounding of the objectives (pink lines on the axes). (Color figure online)

river segments, respectively). We compare the performance of the new DP and MIP methods with the methods in [18] and see significant improvements in both **speed** and **accuracy**. See Fig. 1 and Table 1 for a summary of results.

Specifically, in terms of accuracy, the new MIP approach is substantially better than the previous MIP approach. As shown in Fig. 1, in the 2-dimensional case, the solutions produced by the new MIP approach are on the exact Pareto frontier, slightly better than the DP approximation for the same $\epsilon = 0.1$.

The new DP approach has the same high level of accuracy as the previous DP approach and still produces more solutions than the MIP approaches.

In terms of speed, our experiments show that the new DP approach is up to three orders of magnitude faster than the original DP and scales to significantly larger instances and more criteria. The batching technique also solves the issue of hitting the memory limit when computing for three or more objectives. The new MIP approach is faster and can now solve larger problems.

The DP and MIP methods are complementary since in practice our new MIP scheme provides solutions closer to the exact Pareto frontier (for a given ϵ) and it provides more flexibility for considering additional constraints for what-if analyses, which is important to decision makers.

We are developing a web-based visualization tool for policy makers to explore the Pareto frontier interactively. For example, Fig. 4 displays: (1) the Pareto frontier for four criteria for the entire Amazon ($\epsilon = 0.4$); (2) the placement of the selected dams for a particular Pareto solution; and (3) a parallel coordinate plots to visualize the solutions, in which each axis represents an objective, and each line across the different axes represents a solution. We can bound each objective (pink lines on the axes) and only show solutions that satisfy the bounds. By bounding each objective appropriately, we notably decrease the number of solutions to consider.

6 Conclusions

We introduced new DP and MIP approaches that significantly boost the efficiency and accuracy of computing the exact Pareto frontier and its approximation with guarantees on tree-structured networks. Our DP and MIP approaches show complementary strengths and are now able to scale up to much larger real-world problems. We are developing interactive tools for what-if analyses and visualizations for policy makers. The overall goal of this project is to assist policy makers in making informed decisions when planning hydropower dams in the Amazon Basin. Our methods are general and can be adapted to other multi-objective optimization problems on tree-structured networks.

Acknowledgments. This work was supported by NSF Expedition awards for Computational Sustainability (CCF-1522054 and CNS-0832782), NSF CRI (CNS-1059284) and Cornell University's Atkinson Center for a Sustainable Future.

References

1. Altwaijry, N., EI Bachir Menai, M.: Data structures in multi-objective evolutionary algorithms. J. Comput. Sci. Technol. **27**(6), 1197–1210 (2012)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
3. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spectrum **22**(4), 425–460 (2000)
4. Finer, M., Jenkins, C.N.: Proliferation of hydroelectric dams in the Andean Amazon and implications for Andes-Amazon connectivity. PLoS One **7**(4), e35126 (2012)
5. Gavanelli, M.: An algorithm for multi-criteria optimization in CSPs. In: Proceedings of the 15th European Conference on Artificial Intelligence, ECAI, pp. 136–140 (2002)
6. Gomes, C.P.: Computational sustainability: computational methods for a sustainable environment, economy, and society. Bridge **39**(4), 5–13 (2009)
7. Gomes, C.P., Selman, B., Crato, N., Kautz, H.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. J. Auto. Reason. **24**(1), 67–100 (2000)
8. Neumann, F.: Expected runtimes of a simple evolutionary algorithm for the multiobjective minimum spanning tree problem. Eur. J. Oper. Res. **181**(3), 1620–1629 (2007)
9. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, FOCS 2000 (2000)
10. Qian, C., Tang, K., Zhou, Z.-H.: Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 835–846. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_78
11. Qian, C., Yu, Y., Zhou, Z.-H.: Pareto ensemble pruning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015, pp. 2935–2941 (2015)
12. Schaus, P., Hartert, R.: Multi-objective large neighborhood search. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 611–627. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40627-0_46
13. Sheng, W., Liu, Y., Meng, X., Zhang, T.: An improved strength pareto evolutionary algorithm 2 with application to the optimization of distributed generations. Comput. Math. Appl. **64**(5), 944–955 (2012)
14. Terra-Neves, M., Lynce, I., Manquinho, V.: Introducing pareto minimal correction subsets. In: Gaspers, S., Walsh, T. (eds.) SAT 2017. LNCS, vol. 10491, pp. 195–211. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66263-3_13
15. Walsh, T.: Search in a small world. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI 1999, San Francisco, CA, USA, vol. 2, pp. 1172–1177. Morgan Kaufmann Publishers Inc. (1999)
16. Wiecek, M.M., Ehrgott, M., Fadel, G., Figueira, J.R.: Multiple criteria decision making for engineering (2008)
17. Winemiller, K.O., McIntyre, P.B., Castello, L., Fluet-Chouinard, E., Giarrizzo, T., Nam, S., Baird, I.G., Darwall, W., Lujan, N.K., Harrison, I., et al.: Balancing hydropower and biodiversity in the Amazon, Congo, and Mekong. Science **351**(6269), 128–129 (2016)

18. Wu, X., Gomes-Selman, J.M., Shi, Q., Xue, Y., Garcia-Villacorta, R., Sethi, S., Steinschneider, S., Flecker, A., Gomes, C.P.: Efficiently approximating the pareto frontier: hydropower dam placement in the Amazon basin. In: AAAI (2018)
19. Yukish, M.: Algorithms to identify Pareto points in multi-dimensional data sets. Ph.D. thesis (2004)
20. Yukish, M., Simpson, T.W.: Analysis of an algorithm for identifying pareto points in multi-dimensional data sets. In: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 4324 (2004)
21. Zarfl, C., Lumsdon, A.E., Berlekamp, J., Tydecks, L., Tockner, K.: A global boom in hydropower dam construction. Aquat. Sci. **77**(1), 161–170 (2015)
22. Ziv, G., Baran, E., Nam, S., Rodríguez-Iturbe, I., Levin, S.A.: Trading-off fish biodiversity, food security, and hydropower in the Mekong River Basin. Proc. Nat. Acad. Sci. **109**(15), 5609–5614 (2012)