Scaling Up the Pareto Frontier for Tree Structures with Affine Transformations

Marc Grimson, [1] Qinru Shi, [2] Yiwei Bai, [1] Alexander Flecker, [3] Carla P. Gomes [1]

[1]Department of Computer Science, Cornell University
[2]Center for Applied Mathematics, Cornell University
[3]Department of Ecology and Evolutionary Biology, Cornell University
{mg2425, qs63, yb263, asf3}@cornell.edu, gomes@cs.cornell.edu

Abstract

Working with real-world problems, particularly those in the realm of sustainability, often requires considering multiple competing objectives. It is crucial to provide stakeholders with portfolios of solutions that capture the trade-offs between the different objectives, known as the Pareto frontier. Our work is motivated by the problem of selecting a subset of proposed hydropower dams in the Amazon basin to maximize the hydroelectric power generation while minimizing the adverse effects of dams on the river ecosystem services. We improve the current state-of-the-art dynamic-programming (DP) based algorithm for exact and approximate Pareto frontier on tree-structured problems, with two key enhancements: **(1)** We compute the Pareto dominance of the underlying affine transformations one node induces on the other when merging two sub-trees of the DP approach; and **(2)** we provide an effective heuristic for determining the order in which sub-trees are merged. Our experimental results show that both methods improve the runtime, in some cases by over an order of magnitude, allowing us to better approximate the Pareto frontiers, with respect to energy and a variety of critical ecosystem services, for the Amazon basin, the largest and most biodiverse river basin in the world.

Introduction

Computational Sustainability (Gomes et al. 2019) is a field within Computer Science that aims to utilize Artificial Intelligence to work towards a sustainable future. Such problems often require finding a balance between conflicting concerns, as captured e.g., in the Sustainable Development Goals (SDGs) (United Nations General Assembly 2015): how do we ensure economic and social equity across all strata of society while meeting the needs of a healthy environment? Trade-offs such as these between multiple conflicting objectives naturally translate to a Multi-Objective Optimization Problem (MOOP), problems that have received considerable attention (Wiecek et al. 2008; Ehrgott and Gandibleux 2000). Here, our goal is to understand these trade-offs and to provide stakeholders with the ability to make informed decisions based on their preferences and constraints, which may not be known a priori.

Our work is motivated by the proliferation of hydropower dams and the importance of identifying portfolios of which dams to build in order to maximize economic, social, and environmental benefits, in particular in the Amazon basin (Flecker et al. 2022). As of 2014, there were over 3700 large dams (>1MW energy generation) proposed, planned, or under construction across the world (Zarfl et al. 2014), and over 350 in the Amazon basin alone. In addition to the power that dams provide, they may also negatively impact the surrounding ecosystem, such as forest clearing and flooding of the dam site, which causes the decomposition of organic matter and release of methane, harming biodiversity, and blocking sediment flow and river connectivity. Because of the adverse effects of dams, it is important to carefully navigate the selection of dams to be built to account for different trade-offs among competing objectives (Almeida et al. 2019; Finer and Jenkins 2012; Kareiva 2012; Winemiller et al. 2016; Ziv et al. 2012).

Related work. Previous work framed the problem of selecting which dams to build in the Amazon river basin as a tree-structured MOOP, where the tree structure arose from the underlying river network (Wu et al. 2018; Gomes-Selman et al. 2018). The goal is to find the Pareto frontier, which is the set of all solutions such that no solution is dominated by any other feasible solution. In other words, for a given solution, there is no way to simultaneously improve upon some objective without sacrificing another. Due to this structure, Dynamic Programming (DP) may be used to compute the exact Pareto frontier (Wu et al. 2018). In addition, a rounding technique applied to the exact DP algorithm provides a fully polynomial-time approximation scheme (FPTAS) (Wu et al. 2018; Wu, Sheldon, and Zilberstein 2014a). 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 $1/\epsilon$. The DP approach finds the Pareto frontier of a node by considering the Pareto frontiers of each child and combining the results to form the new frontier. This greatly reduces the search space of the problem as we need only consider the Cartesian product of optimal solutions from each of the children. However, even with the FPTAS, the number of solutions that need to be considered may be too large to process.

For unstructured general multi-objective optimization problems, genetic algorithms have been utilized for approximating the Pareto frontiers, with Non-dominated Sorting Genetic Algorithm and its iterations (NSGA, NSGA-II, and NSGA-III) (Srinivas and Deb 1994; Deb et al. 2002; Deb and Jain 2013) and Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) (Zhang and Li 2007) being among the most popular methods. However, because these algorithms don't consider the underlying structure of the problem, they are unable to provide the same guarantees as the tree-structured DP algorithm.

Another method for identifying the Pareto frontier involves using ray-based techniques (Nowak and Küfer 2020; Lin et al. 2019; Mahapatra and Rajan 2021; Ma, Du, and Matusik 2020) to identify Pareto optimal solutions by finding solutions that match a given ray as a preference vector. These methods, however, depend upon identifying single solutions at a time that match a single ray as a preference vector and generally require a large number of ray vectors to identify a good coverage of the Pareto frontier but often make no guarantees about the coverage.

Other methods use Decision Diagrams (Bergman and Cire 2016) or Propositional Logic (Soh et al. 2017) to identify points along the Pareto frontier, but similar to the ray-based methods, they are unable to scale to systems with exponentially large numbers of solutions on the Pareto frontier.

Our contributions. In this paper, we propose two major enhancements to the DP algorithm proposed in (Gomes-Selman et al. 2018) while maintaining the same optimality guarantees of the exact Pareto frontier algorithm and the FPTAS. **(1)** We formulate the problem of joining the sub-frontiers of a node as an ***affine transformation*** acting on the sub-frontier of its children that maintains Pareto optimality of the transformed sub-set, allowing us to efficiently check for dominance in the transformations first. **(2)** We *dynamically select the children* to join pair-wise based on different ranking heuristics of the children. This is in contrast to the previous approach that follows a static order. **(3)** We show experimentally *a reduction in the number of solutions considered and the runtime of the algorithm* by over an order of magnitude or more in the majority of cases, as well as *better approximate Pareto frontiers* in practice, when dealing with extremely large problem instances, as is the case with six objectives for the entire Amazon basin and sub-basins.

Preliminaries and Main Algorithm

In the hydropower dam portfolio selection problem, our goal is to determine the subset of the proposed dams to build to jointly optimize a set of criteria. Note, we cannot optimize each criteria independently, as maximizing one criterion may require sacrificing the other criteria.

Preliminaries

Tree-Structured Network. In the hydropower dam portfolio selection problem, we define a solution π as the subset of dams that are selected to be built. In the Amazon basin, there are 158 already built hydropower dams and 351 proposed hydropower dams that may be chosen to be built or not. To search over this large space (2^{351} feasible solutions), we take advantage of the underlying structure of the river basin (Wu et al. 2018; Gomes-Selman et al. 2018). Firstly we abstract out the river network and dam locations into a directed connected tree, motivated by the work in (Wu, Sheldon, and Zilberstein 2014b,a), as shown in Figure 1. A node in the tree is a contiguous portion of the river network undisturbed by potential dams, which typically consists of multiple river segments. We associate with node u rewards corresponding to the different d river ecosystem services $\{r_u^1, ..., r_u^d\}$. A directed edge between nodes u and v in the network corresponds to a dam site, connecting downstream segments to upstream segments. A decision variable, to build or not build a dam associated with edge (u, v), is denoted by $\pi_{uv} \in \{0, 1\}$. We associate with each edge a set of values $\{s_{uv}^{(1)}, s_{uv}^{(2)}, ..., s_{uv}^{(d)}\}$, denoting the different d river ecosystem services. Energy is an example of an ecosystem service that is only associated with an edge as it is produced by the dam it denotes. Dams partially trap sediment, and block river connectivity for e.g., transportation, so we also associate passage probabilities for the different ecosystem services $\{p_{uv}^{(1)}, p_{uv}^{(2)}, ..., p_{uv}^{(d)}\}$ (in case the associated dam is built) and $\{q_{uv}^{(1)}, q_{uv}^{(2)}, ..., q_{uv}^{(d)}\}$ (in case the dam is not built), with a given edge (u, v). The root of the tree is the contiguous river network, undisturbed by potential dams, starting at the mouth of the river.

Pareto Dominance. For a given solution π, $z(\pi) = (z^1(\pi), ..., z^d(\pi))$ is the set of values of the d objectives. A solution π dominates another solution π' - written as $z(\pi) \succ z(\pi')$ - if and only if the following two properties hold: (1) for all $1 \leq i \leq d, z^i(\pi) \geq z^i(\pi')$; and (2) there exists at least one strict inequality, $1 \leq j \leq d$ such that $z^j(\pi) > z^j(\pi')$.

Pareto Frontier. Given multiple competing objectives, we aim to find the Pareto frontier, i.e., the set of all solutions that are not dominated by any other feasible solution in the set. Let $\mathcal{P}$ be the set of all feasible solutions, we define the Pareto set as $\{\pi \in \mathcal{P} | z(\pi) \nprec z(\pi'), \forall \pi' \in \mathcal{P}\}$. For example, consider three solutions (π_1, π_2, π_3), with objective values $z(\pi_1) = (10, 4, 3)$, $z(\pi_2) = (9, 4, 2)$, and $z(\pi_3) = (8, 5, 3)$. Solution π_1 dominates π_2 as it has strictly greater values for its first and third objectives, and equal values for the second. However, π_1 does not dominate π_3, and vice versa, as π_1 has a greater value in its first objective than π_3, but π_3 has a greater value in its second objective. Additionally, π_2 and π_3 do not dominate each other. Since π_1 and π_3 are not dominated by any other solution, the Pareto set is $\{\pi_1, \pi_3\}$.

ϵ-approximate Pareto Frontier. Given an exact Pareto frontier P, an approximate Pareto frontier P' is said to ϵ approximate P if and only if for every $\pi \in P$, there exists a solution $\pi' \in P'$ such that $z^i(\pi') \geq (1 - \epsilon) z^i(\pi)$ for all criteria i.

Problem Formulation *Input:* We are given a directed connected tree $T = (V, E)$, where each node $u \in V$ has a set of rewards $\{r_u^1, ..., r_u^d\}$, and each edge $(u, v) \in E$ is associated with a set of values $\{s_{uv}^{(1)}, s_{uv}^{(2)}, ..., s_{uv}^{(d)}\}$, and passage probabilities $\{p_{uv}^{(1)}, p_{uv}^{(2)}, ..., p_{uv}^{(d)}\}$ and $\{q_{uv}^{(1)}, q_{uv}^{(2)}, ..., q_{uv}^{(d)}\}$. Each

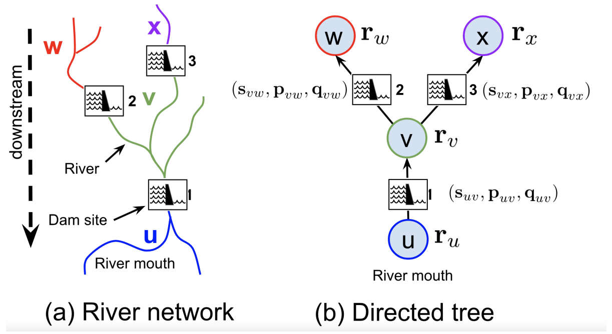
w
x
3
2
v
downstream
River
Dam site
1
u
River mouth
$\mathbf{r}_w$
$\mathbf{r}_x$
$(\mathbf{s}_{vw}, \mathbf{p}_{vw}, \mathbf{q}_{vw})$
$(\mathbf{s}_{vx}, \mathbf{p}_{vx}, \mathbf{q}_{vx})$
$\mathbf{r}_v$
$(\mathbf{s}_{uv}, \mathbf{p}_{uv}, \mathbf{q}_{uv})$
$\mathbf{r}_u$
River mouth
(a) River network
(b) Directed tree

Figure 1: Converting a river network (a) into a directed tree (b). Potential dams, represented by numbers, become tree directed edges, with associated river ecosystem services rewards (s) and passage probabilities (p, if dam is built; q otherwise). The river segments forming a sub-tree, undisturbed by potential dams, represented by different colored segments and letters, become nodes, with associated river ecosystem services rewards (r). The mouth-of-the-river undisturbed downstream river segments, u, become the tree root.

edge $(u, v) \in E$ is also associated with a decision variable $\pi_{uv} \in \{0, 1\}$. A solution π is the vector comprised of the values for all the decision variables, one for each edge. For a given solution π, and tree T_u, a tree rooted at node u, we can compute the value of d objectives $z(\pi, T_u) = (z^1(\pi, T_u), ..., z^d(\pi, T_u))$. Let P$(u, v)$ be the set of edges in the path from node u to node v.

$$z^i(\pi, T_u) = r_u^i + \sum_{(v,w)\in E} r_w^i \prod_{(x,y)\in \mathrm{P}(u,w)} (\pi_{xy} p_{xy}^i + (1 - \pi_{xy}) q_{xy}^i) + s_{vw}^i \prod_{(x,y)\in \mathrm{P}(u,v)} (\pi_{xy} p_{xy}^i + (1 - \pi_{xy}) q_{xy}^i) \quad (1)$$

Output: The Pareto frontier with respect to the d objectives within the set of all feasible solutions $\mathcal{P} = \{\pi | \pi_{uv} \in \{0, 1\}, \forall (u, v) \in E\}$.

Main Algorithm

One can take advantage of the tree structure of the river network and compute the Pareto frontier at each node recursively, since the criteria we care about share the characteristic that an optimal solution on T is also optimal on a subtree T_u rooted at node $u \in V$, i.e. the subtree induced on the set of all nodes above u including u.

From the problem formulation, we recursively define the $z^i(\pi, T_u)$ values for a tree T rooted at node u and solution π:

$$z^i(\pi, T_u) = r_u^i + \sum_{(u,v)\in E} \pi_{uv} s_{uv}^i + \left(\pi_{uv} p_{uv}^i + (1 - \pi_{uv}) q_{uv}^i\right) z^i(\pi, T_v) \quad (2)$$

Algorithm 1: PARETO(u) calculates the Pareto frontier of a tree T_u rooted at node u. The functions NON_DOMINATED and NON_DOMINATED_TRANSFORMS take a set of solutions and transforms and return the set of non-dominated solutions and transforms respectively. Our contributions are in italics.

```
Input: Tree $T_u$ rooted at node u
Output: Pareto frontier for u
  while u.num_children $> 2$ do
    u.rank_children()
    v = new Node()
    v.push(u.children.pop())
    v.push(u.children.pop())
    u.push(v)
    PARETO(v)
  end while
  if u.num_children $== 0$ then
    u.frontier = $\{(r_u^1, r_u^2, ..., r_u^d)\}$
  end if
  if u.num_children $== 1$ then
    v = u.children.pop()
    PARETO(v)
    S = the set of solutions at u by taking all solutions from
    v and all decisions for $(u, v)$
    u.frontier = NON_DOMINATED(S)
  end if
  if u.num_children $== 2$ then
    w = u.children.pop(), v = u.children.pop()
    PARETO(w)
    PARETO(v)
    T = the set of transforms by considering all solutions
    from w and all decisions for $(u, w)$ and $(u, v)$
    T = NON_DOMINATED_TRANSFORMS(T)
    S = the set of solutions at u by taking all solutions from
    v and transforming them by each transform in T
    u.frontier = NON_DOMINATED(S)
  end if
```

(Gomes-Selman et al. 2018; Wu et al. 2018) propose a DP algorithm that takes, as input, a directed tree, and outputs the set of non-dominated solutions at the root of the tree. The algorithm, which is shown with our contributions in Algorithm 1, recursively generates the Pareto frontier of all the children of a node, and joins the sub-frontiers of each child together to produce the Pareto frontier of the parent. In practice, the size of the frontier grows exponentially in the number of objectives, making it infeasible to calculate the exact Pareto frontier for any substantial number of objectives. To alleviate this issue, (Wu et al. 2018) propose a rounding scheme that ϵ-approximates the Pareto frontier in polynomial time. The steps performed on a node u can be broken into four cases.

Case 1 - Leaf node. The Pareto frontier for a leaf node u is simply defined as a single solution which has objective values equal to the rewards $(r_u^1, r_u^2, ..., r_u^d)$ of the node u and an empty set of dams that are to be built.

Case 2 - One Edge. If a node u has only a single edge

(u, v), it takes each of the solutions π_i in the Pareto frontier at node v and each of the decisions that may be made on edge (u, v) and generates a new solution π for each combination, and calculates the non-dominated solutions of the new set. If there were m solutions in the frontier at node v, then we end up with at most $2m$ new solutions.

Case 3 - Two Edges. If a node u has two edges (u, l) and (u, r), then it takes each solution π in the frontier at node l, solution π' in the frontier at node r, and each decision that may be made on edges (u, l) and (u, r), and generates a new solution π'' for each combination. If l has m solutions and r has n solutions on their respective frontiers, then the new node has at most $4mn$ solutions. Here we introduce our first contribution, by generating a smaller set of transformations defined by the decisions on each edge and each solution on the frontier of node l, which consists of at most $4m$ transformations, and checking for dominance within the transformation set first, producing a new set of transformations $m' \leq 4m$. We then use these transformations to generate the $m'n \leq 4mn$ solutions that get checked for dominance. This is discussed and analyzed in Section .

Case 4 - Greater Than Two Edges. Finally, if a node u has $k > 2$ number of edges, it selects and removes two edges (u, v_1) and (u, v_2) from u and generates an intermediate node u' with edges (u', v_1) and (u', v_2), executes the two edge case on u', and adds an edge (u, u') to u, resulting in u now having $k - 1$ edges. It repeats until u has only two edges. In the original algorithm, the edges are selected in the order in which they are listed. Our contributions add a ranking step to determine which two edges to select at a time. This is discussed and analyzed in Section .

Sub-frontier Transformations

When joining two children together, any combination of left child solution and decisions made on the left and right dams defines an affine transformation that is applied to the entirety of the right child solution set. More concretely, for a given node u, children v and w, let π_v[1] and π_w be some specific solutions from the Pareto frontiers associated with the trees T_v and T_w respectively. Let π_u be a new solution containing the solutions π_v and π_w and let $\pi_{ux} \in \{0, 1\}$ be a decision variable for building the dam (u, x) for some node $x \in \{w, v\}$. Consider Equation 2, expanding out the sum for the two children and using the values defined above:

$$\begin{aligned} z^i(\pi_u, T_u) = r_u^i &+ \pi_{uw} s_{uw}^i \\ &+ (\pi_{uw} p_{uw}^i + (1 - \pi_{uw}) q_{uw}^i) z^i(\pi_w, T_w) \\ &+ \pi_{uv} s_{uv}^i + (\pi_{uv} p_{uv}^i + (1 - \pi_{uv}) q_{uv}^i) z^i(\pi_v, T_v) \end{aligned} \tag{3}$$

By fixing the π_w, π_{uv} and π_{uw}, we see that the $r_u^i + \pi_{uw} s_{uw}^i + (\pi_{uw} p_{uw}^i + (1 - \pi_{uw}) q_{uw}^i) z^i(\pi_w) + \pi_{uv} s_{uv}^i$ forms a constant term, and the $(\pi_{uv} p_{uv}^i + (1 - \pi_{uv}) q_{uv}^i) z^i(\pi_v, T_v)$ is a scalar multiplied by the value $z^i(\pi_v, T_v)$ for all solutions π_v associated with T_v. Thus we obtain an affine transformation $a^i z^i(\pi_v) + b^i$ for each i, where

$$\begin{aligned} b^i(\pi_{uw}, \pi_{uv}, \pi_w) = r_u^i &+ \pi_{uw} s_{uw}^i + \pi_{uv} s_{uv}^i \\ &+ (\pi_{uw} p_{uw}^i + (1 - \pi_{uw}) q_{uw}^i) z^i(\pi_w, T_w) \end{aligned} \tag{4}$$

$$a^i(\pi_{uv}) = \pi_{uv} p_{uv}^i + (1 - \pi_{uv}) q_{uv}^i \tag{5}$$

Because the scalar a^i is dependent only on the decision variable π_{uv}, for a fixed π_{uv}, for all possible decisions π_{uw} and solutions π_w associated with tree T_w, the objective values from $z^i(\pi_v, T_v)$ are scaled by the same amount, and the difference in the resulting position of the transformed solution is dependent only on the shift caused by b^i. Therefore, for a given π_{uw}, if a shift $b(\pi_{uw}, \pi_{uv}, \pi_w)$ induced by decision π_{uw} and solution π_w is dominated by another shift $b'(\pi'_{uw}, \pi_{uv}, \pi'_w)$ induced by decision π'_{uw} and solution π'_w, then every point shifted by b must be dominated by at least one point shifted by b'.

Lemma 1. *Let $A \in \mathbb{R}^{d \times d}$ be a diagonal matrix, B be a set of vectors in $\mathbb{R}^d$, and X be a set of non-dominated points in $\mathbb{R}^d$. If a vector $b \in B \succ b' \in B$, that is b dominates b', then every point in $\{Ax + b' | x \in X\}$ is dominated by at least one point in $\{Ax + b | x \in X\}$.*

Proof. Let x be a point in X. Let $b \in B$ be a vector that dominates a vector $b' \in B$. This means that $b \geq b'$, where $\geq$ is defined element-wise, and $\exists i, b_i > b'_i$. We can trivially show that $Ax + b$ dominates $Ax + b'$ for both sets of inequalities.

$$\begin{gathered} b \geq b' \\ \Rightarrow Ax + b \geq Ax + b' \end{gathered}$$

and for the strict inequality on criterion i

$$\begin{gathered} b_i > b'_i \\ \Rightarrow A_{i,i} x_i + b_i > A_{i,i} x_i + b'_i \end{gathered}$$

Therefore the entire set of X translated by b dominates the set translated by b'. □

This result allows us to check for dominance in the set of transformations first, and remove any transformations that are dominated. If we have two children w and v with m and n solutions respectively, and the decisions to build or not build each dam, without considering transform domination we must consider $4mn$ solutions. If we instead first consider the set of transformations that are non-dominated, of which there are $m' \leq 4m$, we first must consider $4m$ transformations and $m'n$ solutions. So long as the resulting number of transformations $m' \leq 4m\frac{n-1}{n}$, where n may be quite large, then we will consider fewer solutions and transformations combined. Figure 2 shows an example of how a dominated transform results in the entire frontier being dominated.

[1]We are slightly overloading the usage of π. When used with a single subscript, such as π_u it refers to a specific node solution which is a vector of decision variables, one for each edge in the tree rooted at node u. When used with two subscripts π_{uv}, it refers to the specific decision variable associated with edge (u, v).

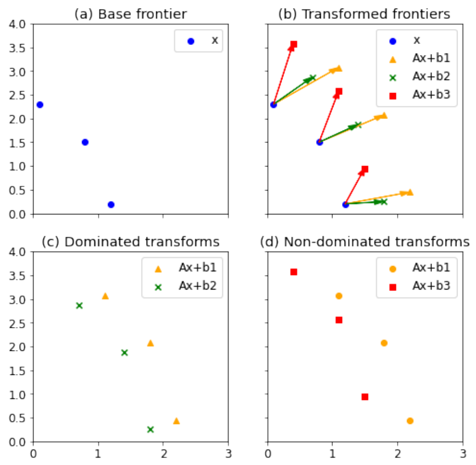

Figure 2: Example of how transformation dominance results in an entire frontier being dominated. In (a) we have the initial points of a sub-frontier. In (b) we show a series of affine transformations $Ax + b$ for different b values. In this example, our A matrix is a 2×2 diagonal matrix with $(1, 1.25)$ on the diagonal, $b_1 = [1, 0.2]^T, b_2 = [0.6, 0]^T, b_3 = [0.3, 0.7]^T$. In (c) we show just the two sub-frontiers $Ax + b_1$ and $Ax + b_2$ where $b_1 \succ b_2$. Each point in $Ax + b_2$ is dominated by its respective point in $Ax + b_1$. In (d) we look at $Ax + b_1$ and $Ax + b_3$, where neither b_1 nor b_3 are dominated, with a mixture of dominated and non-dominated points.

Node Size	# Considered	# Pruned	% Pruned
1	1,279,556	628,936	49.15%
2	2,559,112	1,825,397	63.94%
3	5,118,224	4,287,876	83.78%
4	8,637,003	7,880,444	90.23%
6	12,795,560	11,890,912	92.81%
7	17,913,784	17,256,410	96.33%
8	15,354,672	14,362,079	93.54%
10	25,591,120	24,668,982	96.40&
28	57,580,020	56,555,761	98.22%
35	74,214,248	71,994,683	97.01%
36	56,300,464	55,350,589	98.31%
58	23,032,008	22,125,136	96.06%
89	140,751,160	139,658,474	99.22%
127	213,685,852	211,477,450	98.97%
488	793,324,720	791,872,434	99.82%
946	1,444,618,724	1,441,058,719	99.75%
989	1,356,314,520	1,351,523,177	99.65%
2969	3,799,001,764	3,777,806,664	99.44%

Table 1: Effect of joining the largest node on different size siblings. The left node in the join is fixed to be the largest, in this case with a frontier size of 639,778 solutions, generated with the criteria: Energy, Sediment and Connectivity. The top 5 joins for pruning the highest solution percentage were in the top 6 largest nodes.

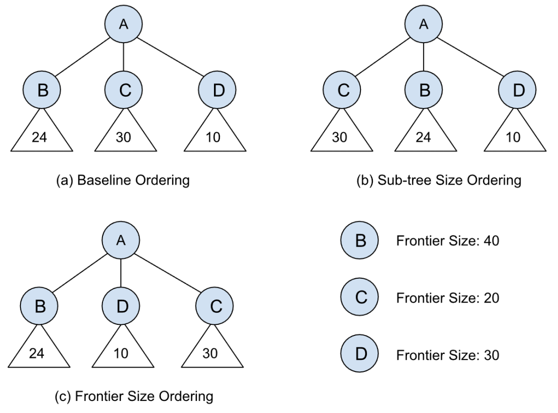

Figure 3: Example ranking strategies. The tree has a root node A with three children B, C, and D. The algorithm selects two of the children to join first, in this case we will always select the left two children first and produce an intermediate node to be joined with the last child. Each of the nodes B, C, and D have a sub-tree containing some number of nodes, as well as a size of their Pareto frontier. In (a) we see the baseline ordering which sorts by listed order. In (b) we see the ordering that ranks by the size of the sub-trees. Finally, in (c) we see the ordering that ranks by the size of the Pareto frontier.

Child Ranking

Our second contribution concerns the ordering in which children are joined when there are more than two children to decide from. In the original algorithm (Gomes-Selman et al. 2018), children were joined using the input order. Our work looks at two different children orderings: one sorting by the sub-tree size of the nodes and the other sorting by the size of the Pareto frontier of the nodes. See Figure 3. We find that it is best to rank the children in descending order in both cases, choosing to join the children with either the largest sub-trees or the largest frontiers first. Consider an example with a node with three children: u, v, and w, with 4, 8, and 10 solutions respectively. First consider the case where we keep all solutions on each step. If we join the smallest two first, u and v, we produce and keep all $4 \cdot 8 = 32$ solutions. Next we join these 32 solutions with the 10 solutions in w, producing $10 \cdot 32 = 320$. In all, we had to search through $320 + 32 = 352$ solutions. Instead, if we join the two largest first, v and w, we have to search through 400 solutions. However, if we were to keep a different percentage of values from each of the joins, it's possible that joining the largest first is better. For example, if joining u and v keeps 50%, but v and w only keeps 10%, if we join the smallest first, we generate $4 \cdot 8 = 32$ solutions as before, but keep 16. Next we join $16 \cdot 10$ solutions to produce 160 solutions, thus needing to consider $32 + 160 = 192$ solutions in all. If we instead join the largest two first, we only consider 112 solutions.

In practice this is the behavior that we see, where a much smaller percentage of solutions is kept when joining larger frontiers or sub-trees, as shown in Table 1.

Basin	Criteria	ϵ	Baseline Order		Node Size Order		Frontier Size Order	
			No Transforms	Transforms	No Transforms	Transforms	No Transforms	Transforms
A	ECB	0.01	>86400	>86400	3317	**2599**	4002	2813
		0.05	5530	5001	162	132	123	**86**
	ECD	0.01	>86400	>86400	6157	**4490**	7000	4871
		0.05	17187	8248	370	303	362	**278**
	ECG	0.025	25567	12199	449	**369**	561	421
	ECS	0.01	18787	9708	992	762	926	**653**
	EBD	0.1	>86400	>86400	18689	**14807**	21605	18213
	EBG	0.05	>86400	>86400	14362	**12406**	15558	13663
	EBS	0.1	21510	17246	7374	6288	6207	**5826**
	EDG	0.05	>86400	>86400	5323	**4782**	8350	7497
	EDS	0.05	27229	22834	3181	2775	2615	**2260**
	EGS	0.05	37276	33858	3729	**3173**	7265	6348
M	EBCDG	0.01	>86400	>86400	26445	**22103**	26454	22621
		0.025	36556	29270	5059	4766	4410	**4123**
	EBCDS	0.025	>86400	>86400	8942	**8609**	9887	8959
	EBCGS	0.01	>86400	>86400	22908	19764	23003	**18551**
		0.025	36046	26730	3344	**3130**	3569	3321
	EBDGS	0.025	>86400	>86400	7903	7738	8168	**7464**
	ECDGS	0.025	13841	11714	2254	**2015**	2253	2076
		0.01	>86400	82353	15267	14070	15782	**13982**
	All	0.05	>86400	>86400	53372	50387	46016	**43236**

Table 2: Running time in seconds for different combinations of criteria and epsilon values, considering different child orderings and the inclusion of the affine transform dominance pruning. Three criteria were used against the full Amazon river basin (A) and five or six criteria were used against the Marañón sub-basin (M), a sub-basin of the Amazon. Each run was given 86,400 seconds (or 24 hours) to complete, jobs that did not complete within that time are listed as taking over 86,400 seconds. The criteria are Energy (E), Connectivity (C), Sediment (S), Degrees of Regulation (D), Biodiversity (B) and Greenhouse Gases (G). Epsilon values used range from 0.01 to 0.1. The fastest running time for each set of parameters is written in bold. We see that in all cases, some combination of either node size or frontier size sorting and the use of transforms outperforms all other cases. The baseline consists of no ordering and no transforms.

Lemma 2. *Given a set of d-dimensional vectors B, a $d \times d$ diagonal matrix A, the set of all feasible solutions F, and the set of all non-dominated solutions $X = \{x \in F | x \not\prec x', \forall x' \in F\}$, that is X is the set of all solutions in F that are not dominated by any other solution, adding a new vector to the set B increases the probability that any solution is dominated by some other solution.*

Proof. Consider a point $x' \in X$ and a vector $b' \in B$. We show first the probability that the new solution $Ax' + b'$ is **not** dominated by any other solution, since the probability that this solution is dominated by any solution point is just 1 minus the probability it is not dominated by any solution.

$$P(\cap_{x \in X, b \in B} Ax' + b' \not\prec Ax + b)$$

If we add a new vector $b'' \notin B$, this can only decrease the probability that the solution $Ax' + b'$ is not dominated. Let E be the event $\cap_{x \in X, b \in B} Ax' + b' \not\prec Ax + b$, then with the new vector b'', we have the following probability

$$\begin{aligned} &P(E \cap_{x \in X} Ax' + b' \not\prec Ax + b'') \\ &= P(E) \cdot P(\cap_{x \in X} Ax' + b' \not\prec Ax + b'' | E) \\ &\leq P(E) \end{aligned}$$

Because this new b'' vector decreases the probability that the point is not dominated, it therefore increases the probability that it is dominated.

$\square$

Lemma 3. *Given a node in the tree u with two children v and w, the number of solutions generated at u before the pruning step grows quadratically.*

Proof. Consider a node u with two children v and w. Let u_F be the frontier at u prior to removing dominated solutions, and let $v_{F'}$ and $w_{F'}$ be the frontiers at nodes v and w respectively, with only non-dominated solutions.

The size of u_F is the cartesian product of all dam decisions, of which there are 2 decisions to make on each of v and w, and the entire frontiers $v_{F'}$ and $w_{F'}$, thus $|u_F| = 4|v_{F'}||w_{F'}| \geq 4 \min\{|v_{F'}|, |w_{F'}|\}^2$, and thus the size of u_F grows quadratically.

$\square$

Conjecture 1. *From Lemmas 2 and 3 we conjecture that larger sub-trees and frontier sizes are likely to provide a higher percentage of dominated solutions.*

From Lemma 3, we see that the density of the affine transformations increases quadratically prior to checking for dominance. From Lemma 2, more transformations increases the percentage of dominated solutions. Thus we expect to see that larger frontiers and node sizes are likely to have a higher percentage of dominated points.

Experiments

The goal of these methods is to scale up calculations of the exact or approximate Pareto frontier and to enable us to compute better approximations. Each of the runs always includes energy as an objective, as it is the only objective that is maximized by building a dam, whereas all other objectives are maximized by not building - thus any run without energy will have a single trivial solution of building no dams.

For the first experiment, we take different combinations of three criteria from Energy, Connectivity, Sediment Transport, Biodiversity Threat, Greenhouse Gas Emissions (GHG), and Degree of River Regulation (DOR) and find the Pareto frontier for the entire Amazon basin. Additionally, we take different combinations of five criteria from the same list, and one run with all six criteria, on the Marañón, a sub-basin of the Amazon. We ran each combination with ϵ values between 0.01 and 0.1 and a maximum running time of 24 hours. We execute each of these experiments using 12 threads running on Intel® Xeon® 3.47 GHz CPUs with 100 GB of memory. We see the results on the full Amazon river basin in Table 2. In all cases, some combination of using a child ordering - either frontier size or node size - and the transformation dominance resulted in the fastest running time. In many cases, the algorithm did not complete within a 24 hour allotted period when running with no child ordering. We see that, while the introduction of the transforms does improve the running time, by up to 1.5 times faster over a similar ordering without transforms, the child ordering has the largest impact, resulting in an order of magnitude improvement over the baseline.

For the second experiment, we calculate the Pareto frontier for the entire Amazon and all six criteria with different ϵ values ranging from 1.0 to 1.5 using the transform dominance and node size ranking. We run each experiment for up to 10 days on 24 threads running on Intel® Xeon® Skylake 6154 3 GHz CPUs with 1.5 TB of memory, with ϵ values between 1.0 and 1.5 to determine the best approximation we can get within the time limit. We also provide a baseline run at the largest ϵ value to show the time improvements are maintained on more criteria. (See Table 3). In the baseline, with a 10-day allotted time period, we were able to calculate an approximation with an ϵ of 1.5, taking a total of 98,306 seconds and generating only 97 solutions at the final node, where much of the work was done in parsing through over 10^{11} solutions at one node of the tree, only to keep a tiny fraction of those solutions after the approximation. Conversely, at a similar ϵ, when using the transform dominance and node size ordering, we only needed to consider 8×10^8 solutions, ran in only 4,489 seconds, and ended up with 5,108 solutions. Due to the approximate nature, it is not surprising to have the final solution size differ from one run to the next with the same ϵ values. Most importantly though, the algorithm using transform dominance and node size ordering was able to calculate the Pareto frontier with an ϵ value as low as 1.0 within 10 days, generating 300,547 solutions after 720,912 seconds.

Scenario	ϵ	Solutions Considered	Final Solutions	Run Time (s)
Baseline	1.5	1.5×10^{10}	97	98,306
Node Size	1.5	8.4×10^8	5,108	4,489
	1.25	1.1×10^{10}	94,523	200,683
	1.0	2.8×10^{10}	300,547	720,912

Table 3: Results of running six criteria for the entire Amazon Basin using node size and transform dominance, for different ϵ values. Each run was given 10 days. The baseline (no ordering, no transforms) took 98,306 seconds for $\epsilon = 1.5$, producing only 97 solutions, and did not terminate in 10 days for other values of ϵ.

Conclusion

Previous work took advantage of the tree-structure that arises from river networks, and was able to theoretically calculate the exact and approximate Pareto frontiers for the selection of which hydropower dams to build in the Amazon from among 2^{351} feasible solutions, though in practice large ϵ values were needed for more than 3 criteria. We show that by utilizing the underlying affine transformations at each step of the DP algorithm and taking into account their expected impacts on which children to join first, we are able to improve the running time of the algorithm by over an order of magnitude in some cases, and as a consequence we are able to calculate better approximate frontiers. The Pareto frontier captures the solutions' tradeoffs with respect to different criteria, without an explicit determination of the relative importance of one criterion versus another ahead of time, which is left to the decision makers. Our contributions provide stakeholders better Pareto approximations in less time than previous algorithms, lending greater confidence to the decision making process for deciding which dams to construct in the Amazon.

References

Almeida, R. M.; Shi, Q.; Gomes-Selman, J. M.; Wu, X.; Xue, Y.; Angarita, H.; Barros, N.; Forsberg, B. R.; García-Villacorta, R.; Hamilton, S. K.; et al. 2019. Reducing greenhouse gas emissions of Amazon hydropower with strategic dam planning. *Nature Communications*, 10(1): 1–9.

Bergman, D.; and Cire, A. A. 2016. Multiobjective optimization by decision diagrams. In *International Conference on Principles and Practice of Constraint Programming*, 86–95. Springer.

Deb, K.; and Jain, H. 2013. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4): 577–601.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2): 182–197.

Ehrgott, M.; and Gandibleux, X. 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-spektrum*, 22(4): 425–460.

Finer, M.; and Jenkins, C. N. 2012. Proliferation of Hydroelectric Dams in the Andean Amazon and Implications for Andes-Amazon Connectivity. *PLOS ONE*, 7(4): 1–9.

Flecker, A. S.; Shi, Q.; Almeida, R. M.; Angarita, H.; Gomes-Selman, J. M.; García-Villacorta, R.; Sethi, S. A.; Thomas, S. A.; Poff, N. L.; Forsberg, B. R.; Heilpern, S. A.; Hamilton, S. K.; Abad, J. D.; Anderson, E. P.; Barros, N.; Bernal, I. C.; Bernstein, R.; Cañas, C. M.; Dangles, O.; Encalada, A. C.; Fleischmann, A. S.; Goulding, M.; Higgins, J.; Jézéquel, C.; Larson, E. I.; McIntyre, P. B.; Melack, J. M.; Montoya, M.; Oberdorff, T.; Paiva, R.; Perez, G.; Rappazzo, B. H.; Steinschneider, S.; Torres, S.; Varese, M.; Walter, M. T.; Wu, X.; Xue, Y.; Zapata-Ríos, X. E.; and Gomes, C. P. 2022. Reducing adverse impacts of Amazon hydropower expansion. *Science*, 375(6582): 753–760.

Gomes, C.; Dietterich, T.; Barrett, C.; Conrad, J.; Dilkina, B.; Ermon, S.; Fang, F.; Farnsworth, A.; Fern, A.; Fern, X.; Fink, D.; Fisher, D.; Flecker, A.; Freund, D.; Fuller, A.; Gregoire, J.; Hopcroft, J.; Kelling, S.; Kolter, Z.; Powell, W.; Sintov, N.; Selker, J.; Selman, B.; Sheldon, D.; Shmoys, D.; Tambe, M.; Wong, W.-K.; Wood, C.; Wu, X.; Xue, Y.; Yadav, A.; Yakubu, A.-A.; and Zeeman, M. L. 2019. Computational Sustainability: Computing for a Better World and a Sustainable Future. *Commun. ACM*, 62(9): 56–65.

Gomes-Selman, J. M.; Shi, Q.; Xue, Y.; Garcia-Villacorta, R.; Flecker, A. S.; and Gomes, C. P. 2018. Boosting efficiency for computing the Pareto frontier on tree structured networks. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 263–279. Springer.

Kareiva, P. M. 2012. Dam choices: Analyses for multiple needs. *Proceedings of the National Academy of Sciences*, 109(15): 5553–5554.

Lin, X.; Zhen, H.-L.; Li, Z.; Zhang, Q.; and Kwong, S. 2019. Pareto Multi-Task Learning.

Ma, P.; Du, T.; and Matusik, W. 2020. Efficient Continuous Pareto Exploration in Multi-Task Learning.

Mahapatra, D.; and Rajan, V. 2021. Exact Pareto Optimal Search for Multi-Task Learning: Touring the Pareto Front.

Nowak, D.; and Küfer, K.-H. 2020. A Ray Tracing Technique for the Navigation on a Non-convex Pareto Front.

Soh, T.; Banbara, M.; Tamura, N.; and Le Berre, D. 2017. Solving multiobjective discrete optimization problems with propositional minimal model generation. In *International Conference on Principles and Practice of Constraint Programming*, 596–614. Springer.

Srinivas, N.; and Deb, K. 1994. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3): 221–248.

United Nations General Assembly. 2015. Transforming our world: the 2030 Agenda for Sustainable Development. Accessed: 2022-11-23.

Wiecek, M. M.; Ehrgott, M.; Fadel, G.; and Figueira, J. R. 2008. Multiple criteria decision making for engineering. *Omega*, 36(3): 337–340.

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.; Stiassny, M. L. J.; Silvano, R. A. M.; Fitzgerald, D. B.; Pelicice, F. M.; Agostinho, A. A.; Gomes, L. C.; Albert, J. S.; Baran, E.; Petrere, M.; Zarfl, C.; Mulligan, M.; Sullivan, J. P.; Arantes, C. C.; Sousa, L. M.; Koning, A. A.; Hoeinghaus, D. J.; Sabaj, M.; Lundberg, J. G.; Armbruster, J.; Thieme, M. L.; Petry, P.; Zuanon, J.; Vilara, G. T.; Snoeks, J.; Ou, C.; Rainboth, W.; Pavanelli, C. S.; Akama, A.; van Soesbergen, A.; and Sáenz, L. 2016. Balancing hydropower and biodiversity in the Amazon, Congo, and Mekong. *Science*, 351(6269): 128–129.

Wu, X.; Gomes-Selman, J.; Shi, Q.; Xue, Y.; Garcia-Villacorta, R.; Anderson, E.; Sethi, S.; Steinschneider, S.; Flecker, A.; and Gomes, C. 2018. Efficiently Approximating the Pareto Frontier: Hydropower Dam Placement in the Amazon Basin. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Wu, X.; Sheldon, D.; and Zilberstein, S. 2014a. Rounded dynamic programming for tree-structured stochastic network design. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 479–485.

Wu, X.; Sheldon, D.; and Zilberstein, S. 2014b. Stochastic network design in bidirected trees. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 1*, 882–890.

Zarfl, C.; Lumsdon, A.; Berlekamp, J.; Tydecks, L.; and Tockner, K. 2014. A Global Boom in Hydropower dam Construction. *Aquatic Sciences*, 77: 161–170.

Zhang, Q.; and Li, H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6): 712–731.

Ziv, G.; Baran, E.; Nam, S.; Rodríguez-Iturbe, I.; and Levin, S. A. 2012. Trading-off fish biodiversity, food security, and hydropower in the Mekong River Basin. *Proceedings of the National Academy of Sciences*, 109(15): 5609–5614.