COMBINATORIAL OPTIMIZATION AND DECISION-MAKING WITH APPLICATIONS IN COMPUTATIONAL SUSTAINABILITY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Qinru Shi

August 2022

COMBINATORIAL OPTIMIZATION AND DECISION-MAKING WITH APPLICATIONS IN COMPUTATIONAL SUSTAINABILITY

Qinru Shi, Ph.D.

Cornell University 2022

Combinatorial optimization and decision-making problems are critical in many real-world computational sustainability problems. The main goals for these projects are often to provide decision-support tools for various groups and institutions to help solve complex computation problems encountered in sustainable planning and development. This thesis mainly focuses on two real-world applications of combinatorial optimization and decision-making in computational sustainability. The first is a multiobjective optimization problem inspired by the real-world problem of placing hydropower dams in the Amazon basin. We propose a fully polynomial-time approximation scheme based on Dynamic Programming (DP) for computing the Pareto frontier within an arbitrarily small error margin $\epsilon > 0$ on tree-structured networks. We also developed a complementary mixed integer programming (MIP) approach for approximating the Pareto frontier and methods for approximating high-dimensional Pareto frontiers. The second is an online matching problem coordinating citizen scientists for invasive species survey efforts. We developed a learning-augmented matching algorithm that can utilize partial information and provides good performance and approximation guarantees. For both applications, we provide not only practical solutions to real-world problems but also novel computational algorithms and techniques.

BIOGRAPHICAL SKETCH

Qinru Shi earned her Ph.D. in applied mathematics at Cornell University in 2022. Her research interests lie in operations research and artificial intelligence, with a focus on problems in combinatorial optimization. Her works revolve around real-world computational sustainability problems that directly impact society and provide unique computational challenges. Before coming to Cornell, she received a B.Sc. in mathematics from the Massachusetts Institute of Technology in 2015. Qinru is originally from Beijing, China.

ACKNOWLEDGEMENTS

I want to thank my Ph.D. advisor, Carla P. Gomes, for her guidance and passion, as well as her patience and support. She connected me with many incredible researchers from different fields and provided invaluable insights into my research directions. None of the works in this thesis would have been possible without her. I would also like to thank my other committee members, David Bindel and Damek Davis, for their helpful comments and suggestions.

I would like to thank Alexander S. Flecker, who started the Amazon Dam Project with Carla and led the group through many difficulties and challenges over the years. I would also like to thank Rafael M. Almeida, Roosevelt Garcia-Villacorta, and Suresh A. Sethi, who aided me greatly in understanding various aspects of the impacts of dams. I share many fond memories with the Amazon Dam working group and have learned much from our weekly discussions.

I would also like to thank my fellow students and collaborators Xiaojian Wu, Jonathan M. Gomes-Selman, Johan Bjorck, Yiwei Bai, Marc Grimson, Yexiang Xue, Héctor Angarita, Jennifer Price Tack, Carrie Brown-Lima, Jennifer Dean, and Angela Fuller.

Lastly, I would like to thank my family and friends for their support and encouragement over the years. Thank you for being there for me when I needed it the most.

My research was funded in part 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.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Combinatorial optimization and decision-making problems play a critical role in many real-world computational sustainability problems. The main goals for these projects are often to provide decision-support tools for various groups and institutions to help solve complex computation problems encountered in sustainable planning and development. More formally, these combinatorial optimization problems can be defined by a set of discrete or continuous variables representing the decisions that need to be made. The goal is often to find solutions that maximize or minimize one or more objective functions. Regarding computational complexity, the total number of feasible solutions to combinatorial optimization problems is often exponential in the number of variables. Moreover, many combinatorial optimization problems are NP-complete or even NP-hard, making it impossible to solve these problems exactly through conventional methods as the problem sizes go up. Thus, it is crucial to find suitable approximation methods that can solve these problems within a reasonable time and error range. which is often achieved by exploiting the special structures of the specific problems.

This thesis will mainly focus on two real-world applications of combi-

natorial optimization and decision-making in computational sustainability. The first is a multiobjective optimization problem regarding the placement of hydropower dams in the Amazon basin. We developed several methods for approximating the Pareto frontier of this problem, each with its own strengths. Most notably, the DP-based approximation algorithm is a fully polynomial time approximation scheme (FPTAS) and is considered a state-of-art algorithm in the field of hydropower planning. The second is an online matching problem of coordinating citizen scientists for invasive species survey efforts. We developed a learning-augmented matching algorithm that can utilize partial information and provides good performance and approximation guarantees. For both applications, we provide not only practical solutions to real-world problems but also novel computational algorithms and techniques.

1.1 Efficiently approximating the Pareto frontier for tree-structured networks for applications in hydropower planning

Multi-objective optimization is critical in the field of Computational Sustainability [Gomes, 2009], for real-world sustainability problems often involve balancing multiple environmental, economic, and societal objectives. While we can convert these problems into single-objective optimization problems by combining the objectives into a weighted sum, such simplifications might not fully characterize the relationships between different objectives: optimizing one objective often sacrifices the values of other objectives. Therefore,

for multiobjective optimization problems, the goal is often not to find a single optimal solution but a set of solutions that characterize the trade-offs between several often competing criteria, i.e., the so-called **Pareto frontier**. The Pareto frontier is the set of all **Pareto-optimal** solutions, where a solution is considered Pareto-optimal if its vector of objective values is not dominated by the corresponding vector of any other feasible solution. Often the Pareto frontier is of exponential size, making it challenging to compute. This thesis will explore various methods for efficiently approximating the Pareto frontier.

Hydropower dam placement in the Amazon Basin

Our work is motivated by a challenging real-world computational sustainability problem concerning the placement of hydropower dams in a river network. In recent years, there has been a significant proliferation of hydropower dams in the Amazon basin: at least 158 dams are already operating or under construction, and another 351 dams are proposed to be built in the next decade. These hydropower dams will heavily impact many ecosystem services such as energy production, navigation, biodiversity, sediment and nutrient production, and fresh-water fisheries. Therefore, it is imperative to integrate ecosystem service trade-offs into decision-making models for the placement of hydropower dams in the Amazon Basin. To this end, we model the hydropower dam placement problem as a multiobjective optimization problem, where we aim to determine the optimal subset of proposed dams to be built based on several social, economic, and environmental criteria. More specifically, we address a set of **six objectives** to characterize the

Amazon hydropower siting problems: hydropower generation, the main benefit provided by dams; river connectivity index, an indicator of the amount of habitat accessible to migratory fish; sediment transportation, which is the amount of sediment and nutrients transported by the river to the main stem and is essential for flood plain agriculture; biodiversity impact, which indicates the overall impact of dams on local biodiversity; degree of regulation, which represents the change of river flow regimes caused by dams and has a lasting influence on fish population; and greenhouse gases emissions, which predicts the total amount of greenhouse gases, such as methane, emitted due the decomposition of organic matter from areas flooded by dams. We aim to find methods that can efficiently and reliably approximate the six-criteria Pareto frontier.

Dynamic Programming Based Algorithm

The Amazon hydropower dam placement problem represents a multiobjective optimization problem naturally captured by a tree-structured network. By exploiting this special tree structure, we develop a **fully polynomial time approximation scheme (FPTAS)** based on dynamic programming (DP). The algorithm finds a polynomially succinct set of solutions that can approximate the actual Pareto frontier within an arbitrarily small error range $\epsilon > 0$ and runs in time that is polynomial in the size of the problem and $1/\epsilon$. The algorithm recursively computes the approximate Pareto frontier above each node of the tree-structured network, propagating from leaves to root. The critical insight of the algorithm is that for most of our

objectives, we only need to keep the Pareto-optimal partial solutions at each node in the tree, which allows us to prune most of the suboptimal solutions early. Some criteria such as RCIP need to be further decomposed to be included as an objective. In practice, the algorithm can compute the exact Pareto frontier ($\epsilon = 0$) for two criteria within minutes and the approximate Pareto frontier ($\epsilon = 0.25$) for five criteria within a week. We also observe that the solutions are generally very close to the actual Pareto frontier even when the error margin ϵ is relatively large. The DP-based algorithm is now considered a state-of-the-art algorithm in the field of hydropower planning.

Mixed Integer Programming Based Algorithm

While the DP-based algorithm can approximate the whole Pareto frontier efficiently, one flaw of the algorithm is that it produces all the solutions simultaneously, which may be excessive if we are only interested in a small section of the solution space. To complement the DP algorithm, we propose a MIP formulation of the hydropower dam placement problem and a scheme for approximating the Pareto-frontier using MIP based on ideas from [Papadimitriou and Yannakakis, 2000a]. 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 can solve the MIP to 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 then forms an ϵ-approximate Pareto-

frontier. In practice, the MIP approximation scheme runs slower than the DP algorithm and produces fewer solutions. However, the MIP approach provides more flexibility when considering additional constraints and can be used to search in a small specific solution space.

Use Expansion and Compression methods to approximate high-dimensional Pareto Frontier

One major problem we face when approximating the Pareto frontier of the hydropower dam placement problem is that although the runtime of the dynamic-programming-based algorithm is polynomial for the number of dams, it is still exponential with respect to the number of objectives, which means that both the runtime and the number of solutions greatly increase as the number of objectives goes up. For large river networks such as the Amazon basin, while the algorithm is able to solve three or four-objective optimization problems efficiently and with a small approximation factor, its performance drops off dramatically once we reach five objectives. However, to encompass the complexity of balancing hydropower generation with ecosystem service impacts in the Amazon, all six objectives (hydropower generation, River connectivity index, sediment transportation, biodiversity impact, degree of regulation, and greenhouse gas emissions) need be considered.

To tackle this issue, we assume that, for multi-objective optimization problems on river networks, the six-objective Pareto frontier might approximately lie on a lower-dimensional manifold. Given that the DP-based algo-

rithm is able to solve this type of MO problem with three or four objectives efficiently and with a guaranteed approximation factor and that these solutions are very likely to be on the Pareto frontier for more objectives, we conjecture that the aggregated solutions from Pareto frontiers optimized for all combinations of three or four-element subsets of the six objectives may form a good approximation of various local regions of the six-objective Pareto frontier. We call this approach the 'expansion' method since it expands the approximate Pareto frontiers of fewer objectives into an approximation of a high-dimensional Pareto frontier. Experiments show that the expansion method produces very good approximations of high-dimensional Pareto frontiers for multi-objective optimization problems on river networks.

1.2 Augmented online metric matching for applications in invasive species management

With the success of machine learning models for prediction, integrating such learned models into real-world systems has become a critical challenge. An emerging paradigm for integrating machine learning into optimization algorithms, while giving theoretical guarantees, is to consider *learning augmented* algorithms [Lykouris and Vassilvitskii, 2018], where a machine learning model gives incomplete or partly incorrect predictions. We construct an algorithm based on the predictions that not only provably performs well when the machine learning advice is sound, but also still gives guarantees when predictions are inaccurate. This approach applies to many problems

in computational sustainability where data are abundant yet noisy, and we still want to provide provable guarantees for the efficiency of our decisions [Dilkina and Gomes, 2010, Bondi et al., 2018].

Coordinating citizen scientist for invasive species management

The primary motivation for this project is to help coordinate the efforts of citizen scientists for invasive species management as part of an ongoing collaboration with the New York Natural Heritage Program, which contributes to the state invasive species database through the online mapping system iMapInvasives [NatureServe, 2020 (Accessed 2020-07-01]. Data from citizen scientists and paid managers are combined in the database to help the state of New York assess and monitor the spread of over four hundred invasive species across the state. A pervasive problem in these settings is that citizen scientists are opportunistic and possibly have misaligned incentives [Xue et al., 2013], and the engagement of such citizen scientists is not known ahead of time. In addition to these volunteers, the database collaborates with institutes with paid employees whose schedules can be decided ahead of time. Our goal is then to coordinate the effort of opportunistic citizen scientists with predictably scheduled paid employees, which can be formulated as an online matching problem with partial a priori information.

Learning augmented matching algorithm

We model the problem as a semi-online matching problem, where we want to find the minimum cost matching in an online fashion given partial predictions of the graph, e.g., citizen scientists that become available one by one and must irrevocably be matched to survey locations as they show up. We develop an algorithm that fully utilizes prior knowledge and provably improves upon pessimistic algorithms in the learning augmented setting, with an approximation bound that depends upon the amount of knowledge available. The basic idea of this algorithm is to always maintain a 'best possible' matching given current knowledge, and every new decision is made based on both past decisions and the current 'best possible' matching. The algorithm is evaluated on two large real-world datasets, the taxi dataset of the NYC Taxi and Limousine commission and invasive species records from the database. We find that our algorithm consistently outperforms baselines. We also prove approximation guarantees for the algorithm that depends upon how much knowledge we have access to.

Summary

In summary, this thesis focuses on two major real-world applications of combinatorial optimization and decision making and presents novel algorithms and techniques inspired by the complexity of these real-world problems. In Chapter 2, we formulate the Amazon hydropower placement problem as a multiobjective optimization problem and provide formal proofs and

experimental results for various algorithms and techniques we develop for approximating the Pareto frontier. The FPTAS DP-based algorithm is now considered a state-of-the-art algorithm in the field of hydropower planning. We also discuss how to best showcase these results and help inform the actual decision-makers. In Chapter 3, we represent the problem of coordinating citizen scientists as an online matching problem with partial information and provide a new algorithm that outperforms baseline methods and provides approximation guarantees. Our work contributes to both fronts of Computational Sustainability, providing novel computational algorithms and techniques and practical solutions to real-world sustainability problems.

List of Publications

Most of the material in this thesis has been reported in the following peer-reviewed publications:

- Efficiently Approximating the Pareto Frontier: Hydropower Dam Placement in the Amazon Basin.

  AAAI 2018

  Xiaojian Wu, Jonathan M. Gomes-Selman, Qinru Shi, Yexiang Xue, Roosevelt Garcia-Villacorta, Suresh Sethi, Scott Steinschneider, Alexander Flecker, and Carla P. Gomes.

- Boosting Efficiency for Computing the Pareto Frontier on Tree Structured Networks.

  CPAIOR 2018

Jonathan M. Gomes-Selman, Qinru Shi*, Yexiang Xue, Roosevelt Garcia-Villacorta, Alexander S. Flecker and Carla P. Gomes.

- Efficiently Optimizing for Dendritic Connectivity on Tree-Structured Networks in a Multi-Objective Framework

  ACM COMPASS 2018

  Jonathan M. Gomes-Selman*, Qinru Shi*, Roosevelt García-Villacorta, Suresh Sethi, Alexander S. Flecker, and Carla P. Gomes.

- Reducing greenhouse gas emissions of Amazon hydropower with optimal dam planning.

  Nature Communications 10 (4281)

  Rafael M. Almeida, Qinru Shi, Jonathan M. Gomes-Selman, Carla P. Gomes, Alexander S. Flecker, et al.

- Reducing adverse impacts of Amazon hydropower expansion

  Science vol. 375, 2022

  Alexander S. Flecker, Qinru Shi, Rafael M. Almeida, Héctor Angarita, Carla P. Gomes, et al.

- Learning Augmented Methods for Matching: Improving Invasive Species Management and Urban Mobility

  AAAI 2020

  Johan Bjorck, Qinru Shi, Carrie Brown-Lima, Jennifer Dean, Angela Fuller, and Carla P. Gomes.

Chapter 2

Efficiently approximating the Pareto frontier for tree-structured networks for applications in hydropower planning

As a renewable resource, hydropower is a major component of the current and future energy portfolio worldwide. While the trend of building large-scale hydropower projects has largely abated and coordinated dam removals are being considered in many developed countries [Kuby et al., 2005, Roy et al., 2018], construction of large dams is steadily expanding in many countries with emerging economies[Winemiller et al., 2016a, Zarfl et al., 2014]. From a socio-environmental perspective, hydropower proliferation is an especially acute issue in tropical river basins such as the Amazon. Currently, at least 158 dams with installed capacities over 1MW are operating or under construction in the Amazon basin, and another 351 dams are being proposed (See Figure 1). These dams have the potential to dramatically affect a variety of ecosystem services such as biodiversity, nutrient cycling, and sediment production, as well as services such as energy production, navigation, and freshwater fisheries. [Finer and Jenkins, 2012, Kareiva, 2012, Winemiller et al., 2016b, Zarfl et al., 2015, Ziv et al., 2012]

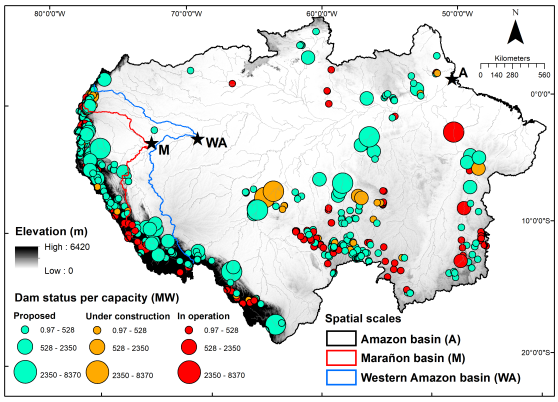


Figure 1: The 158 existing dams and 351 proposed dams in the Amazon basin. The sizes of the dots reflect the capacities of the respective dams. The map also shows the range of the whole Amazon basin, the Western Amazon basin, and the Maranon basin, which will be used in our experiments.

To better protect the world's few remaining unfragmented river basins, tools for strategic dam planning are urgently needed to help minimize the total environmental impacts of dams at the basin scale. From a computational perspective, the Amazon hydropower dam placement problem can be modeled as a multiobjective optimization problem naturally captured by a tree-structured network, where we aim to determine the optimal subset of proposed dams to be built based on several social, economic, and environmental objectives. More specifically, we selected **six objectives** deemed most representative based on expert opinion: hydropower generation, river connectivity index, sediment transportation, biodiversity impact, degree of regulation, and greenhouse gas emissions. Our goal is to find the **Pareto frontier**, which 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.

This multiobjective optimization problem is highly computationally challenging due to the size and the number of objectives. Existing approaches for multi-objective optimization problems are mostly heuristic, based on local search or evolutionary algorithm, do not provide theoretical guarantees, and do not exploit the tree structure. (see e.g., Yukish and Simpson [2004], Wiecek et al. [2008], Ehrgott and Gandibleux [2000], Qian et al. [2016, 2013, 2015], Neumann [2007], Deb et al. [2002], Sheng et al. [2012], Terra-Neves et al. [2017].) Therefore, our goal is to develop new approaches that exploit the special structure of the problem and provide approximation guarantees. To this end, we develop a DP-based fully polynomial time approximation scheme (FPTAS) for approximating the Pareto frontier on tree-structured networks based on ideas from [Wu et al., 2014a,b] and a Mixed Integer Programming (MIP) based approximation scheme following the scheme proposed in [Papadimitriou and Yannakakis, 2000a]. We also propose new techniques for scaling up to higher-dimension multiobjective optimization problems.

We aim to support policymakers in the decision-making process of hydropower planning. To facilitate a better understanding of the optimization solutions, we develop a visualization tool for showcasing and exploring the high-dimensional Pareto-frontier, which will allow policymakers to comprehend the trade-offs between the objectives and make better-informed decisions concerning the trade-offs of socio-economic and environmental impacts of hydropower projects.

The majority of the work presented in this chapter has been reported in the following publications: Flecker et al. [2022], Wu et al. [2018], Gomes Selman et al. [2018], Shi et al. [2018], Almeida et al. [2019].

In Section 2.1, we review relevant definitions and formally define the problem of hydropower dam placement. Section 2.2 presents a rigorous description of the DP-based algorithm and proofs of the algorithm's runtime and error bound. In Section 2.3, we show the MIP formulation of the problem and the approximation scheme adapted from [Papadimitriou and Yannakakis, 2000a]. In Section 2.4, we showcase the experimental results of the DP-based and the MIP-based methods. Section 2.5 discusses incorporating the dendritic connectivity index, a crucial objective with a special structure, into the DP-based and MIP-based methods. In Section 2.6, we present a simple yet effective method, the 'Expansion' method, for approximating high-dimensional Pareto Frontiers. Lastly, Section 2.7 discusses how to best present our results to decision-makers.

2.1 Problem Definition

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

2.1.1 Hydropower Dam Placement Problem:

The real-world problem we aim to solve can be summarized as follows: we are given a set of planned dams and need to decide the optimal subset of dams to build with respect to various objectives. We refer to this problem as the hydropower dam placement problem. We first point out that a river network is essentially a directed tree-structure network and that we don't need to consider every river segment explicitly for the purposes of our hydropower dam placement problem. Hence, our first step for simplifying the problem is to abstract the river network and potential dam locations into a more compact directed rooted tree that captures the critical problem information. Each contiguous section of the river network uninterrupted by existing or potential dam locations is represented by a hypernode; each 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. Note that this conversion step greatly reduces the size of the problem, as the original river network file of the Amazon basin contains over **4 million** river segments, whereas the resulting converted tree only contains 510 nodes.

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

Hydropower (E): For a given 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

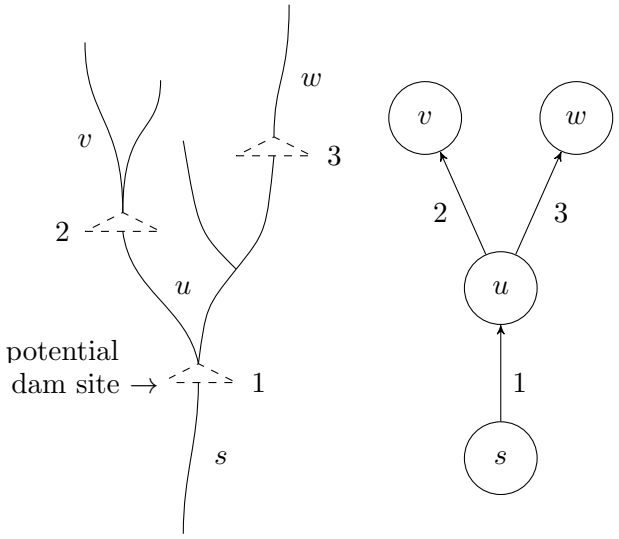


Figure 2: Converting a river network (left) into a more compact directed rooted tree (right). Each contiguous region of the river network (labeled by different letters) is converted into a node, also referred to in this thesis as a hypernode (labeled with the corresponding letter) in the tree network. Each potential dam location (shown as triangles on the left) is represented as an edge in the directed rooted tree on the right.

generation (installed capacity) of the dam represented by edge e. This is an objective to be maximized.

Longitudinal Connectivity (C): For a given solution π, the connectivity of a river network is measured by the total length of the un-obstructed stream segments that one can travel starting from the root (river mouth) without passing any dam site in π. This objective is to be maximized.

Sediment (S): For a given solution π, this objective represents the total amount of sediment that will be transported to the river mouth (the root

of the directed tree) per year. We assume that each node produces a fixed amount of sediment per year and that each dam traps a certain percentage of total sediment from upstreams. This objective is to be maximized.

Fish Biodiversity Impact (B): For a given solution π, this objective measures the overall impact on the fish population caused by dam construction. We use b_e to denote the fish-weighted endemism index of the dam represented by edge e. Then, $B(\pi) = \sum_{e \in \pi} b_e$. This objective and all objectives below are to be minimized.

Degree of Regulation (DOR): For a given solution π, this objective measures the total degree of flow regime alteration caused by dam construction. We define $C_{DOR}(e)$ as the total cumulative downstream flow impact of the dam represented by edge e. (The values of C_{DOR} are pre-computed.) Then, $DOR(\pi) = \sum_{e \in \pi} C_{DOR}(e)$.

Greenhouse Gas Emissions (GHG): For a given solution π, this objective measures the total greenhouse gas emissions caused by dam construction. We use g_e to denote the greenhouse gas emissions caused by the dam represented by edge e. Then, $GHG(\pi) = \sum_{e \in \pi} g_e$.

Seismic Risk (S_s): For a given solution π, this objective measures the total amount of seismic risk at the dam locations. We use ss_e to denote the greenhouse gas emissions caused by the dam represented by edge e. Then, $S_s(\pi) = \sum_{e \in \pi} ss_e$. This objective is not included in the final paper because we don't have very reliable data. However, this objective is still used in many experiments.

2.1.2 General Formulation

In general, a multi-objective optimization problem is to optimize a given multi-objective function: $(z^1(\pi), z^2(\pi), ..., z^d(\pi))$, 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 solution is one that is not dominated by any other policies. The Pareto frontier is the set containing all Pareto optimal solutions.

An Example: Consider a multi-objective function (z^1, z^2, z^3) and policies π_1, π_2, and π_3. Assume $(z^1(\pi_1), z^2(\pi_1), z^3(\pi_1))$ = (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. 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 formally define the multi-objective optimization problem on a tree-structured network. Every node v in the tree is associated with node rewards for each objective $r_v^1, \ldots, r_v^d$. For a leaf node v, all objective functions defined on v are its

corresponding rewards, i.e., $z_v^i(\pi) = r_v^i$ for all is. Each edge (u, v) has two states (representing whether the corresponding dam is built or not). If edge (u, v) is in its first state (the dam at (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 an indicator variable $I_{uv}(\pi)$ denoting whether the corresponding edge is in π or not. The objective function on a non-leaf node u is defined recursively:

$$z_u^i(\pi) = r_u^i + \sum_{v \in ch(u)} \left(I_{uv}(\pi) p_{uv}^i + (1 - I_{uv}(\pi)) q_{uv}^i \right) z_v^i(\pi). \tag{2.1}$$

Here, $ch(u)$ 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., $z^i(\pi) = z_s^i(\pi)$.

Some Examples: When we are computing connectivity, we can set r_u^i to be the total length 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). The encoding of objectives like hydropower is slightly more complicated. For any dam site (u, v) that can produce hydropower h_{uv}, we set $p_{uv}^i = q_{uv}^i = 1$. We move h_{uv} to its parent u. In other words, the reward at node u, $r_u^i = \sum_{v \in ch(u)} I_{uv}(\pi) h_{uv}$. Notice that r_u^i depends on whether each edge (u, v) is in policy π.

The value function defined in this way is general and captures many interesting families of functions. For example, the value function of a finite binary-state Markov Decision Process can be encoded in this way by unfolding the decision process into a chain, which is in a special tree form. The value function can also be equivalently written in a more concise way. Let $u \rightsquigarrow v$ denote the path from u to v. Define the probability that u is

connected to v as

$$p^i_{u \rightsquigarrow v}(\pi) = \prod_{\text{e on } u \rightsquigarrow v} \left(I_e(\pi) p^i_e + (1 - I_e(\pi)) q^i_e \right),$$

that is, the product of all passage probabilities of edges on the path from u to v depending on their states. Then,

$$z^i(\pi) = \sum_{v \in V} p^i_{s \rightsquigarrow v}(\pi) r_v. \tag{2.2}$$

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.

Theorem 1. *Multi-objective optimization problem on a tree-structure network is NP-complete, even for just two objectives.*

Proof. We reduce the PARTITION problem (pp. 47 [Garey and Johnson, 1979])) to a multi-objective optimization problem. The PARTITION problem is: given a finite set $A = \{a_1, \cdots, a_n\}$, decide if there is $A' \subseteq A$, such that $\sum_{a_i \in A'} a_i = \sum_{a_i \in A \setminus A'} a_i$.

To encode the PARTITION problem, we design a star-shaped tree with $n + 1$ nodes. The root node 0 is connected to leaf nodes $1, \cdots, n$. The node reward are chosen to be $r^1_i = r^2_i = a_i, p^1_{0i} = q^2_{0i} = 1$, and $p^2_{0i} = q^1_{0i} = 0$ for all $i = 1, \cdots, n$. It is easy to verify that the existence of A' satisfying the condition of PARTITION is equivalent to checking if there exists a policy π on the star-shaped tree such that $z^1(\pi) \geq \frac{1}{2} \sum_{a \in A} a$ and $z^2(\pi) \geq \frac{1}{2} \sum_{a \in A} a$. □

2.2 The Dynamic Programming Based Algorithm

In this section, we will first present a dynamic programming algorithm that computes the Pareto frontier exactly for multi-objective optimization problems on a tree-structured network. We shall prove that the algorithm can find all Pareto optimal solutions. However, in practice, the number of Pareto optimal policies may be exponential even for a fixed number of objectives. Motivated by ideas from Wu et al. [2014a,b], we apply a rounding technique to the dynamic programming algorithm and provide a fully polynomial-time approximation scheme (FPTAS). The DP-based algorithm with rounding finds a solution set of polynomial size that approximates the Pareto frontier within an arbitrarily small ϵ and runs in time polynomial in the size of the instance and $1/\epsilon$.

Definition 1. *We say that a policy set S ϵ-approximates the Pareto frontier if and only if for any policy π in the Pareto frontier, there exists a policy π' from set S, such that $z^i(\pi') \geq (1-\epsilon)z^i(\pi)$ for all $i \in \{1, 2, \cdots, d\}$.*

Definition 2. *We say an algorithm is a fully polynomial-time approximation scheme (FPTAS) for a multi-objective optimization problem if the algorithm finds a policy set S that ϵ-approximates the Pareto frontier and runs in time polynomial in the size of the instance and $1/\epsilon$.*

Suppose there are universal constants c and C such that $c \leq r_u^i \leq C$ for all $u \in V$ and all $i \in \{1, 2, \cdots, d\}$. Our DP-based approximation algorithm is an FPTAS for the multiple objective optimization problems defined on a tree-structured network. The size of the solution set found by the algorithm

is bounded by $O\left(\left(\frac{n}{\epsilon}\right)^d\right)$, and the running time is bounded by $O\left(\left(\frac{n}{\epsilon}\right)^{2d}\right)$, where n is the number of nodes in $\mathcal{T}$ and d is the number of objectives.

While the approximation algorithm theoretically runs in polynomial time, its performance is still far from ideal when the number of objectives is larger than three, as the runtime of the algorithm is still exponential in terms of the number of objectives. To further improve the performance of the approximation algorithm, we also introduce techniques such as batching, divide-and-conquer, and parallelization. Currently, for multi-objective optimization problems on the Amazon basin, we are able to solve problems with two objectives exactly within seconds, solve problems with three objectives within a few minutes to an hour for $\epsilon = 0.05$ or $\epsilon = 0.1$, and solve problems with four objectives within a few hours for $\epsilon = 0.1$ or $\epsilon = 0.25$.

2.2.1 The Exact Dynamic Programming Algorithm

We first present a dynamic programming (DP) based algorithm which computes the Pareto frontier **exactly** for multi-objective optimization problems on a tree-structured network. Before we present the algorithm, we want to transform the original directed tree into an equivalent directed binary tree.

Transforming the original tree into a binary tree: To improve the efficiency of the DP algorithm, we first convert the original directed rooted tree into an equivalent binary tree. Given a directed rooted tree, we transform each node u with more than two children into a binary equivalent subtree by creating additional intermediary nodes and linking the children of u to these intermediary nodes (and u). The newly added edges between the u and intermediate nodes are treated as special non-dam edges. The

algorithm never builds a dam on these edges. We also propose a scheme to maintain the objective value information of the original tree. Node rewards such as connectivity and sediments are split evenly among the original root u and intermediary nodes. Edge rewards such as energy and fish biodiversity impact are moved to their parent node in the binary tree. Overall, our construction is such that dam placements and resulting objective values are in one-to-one correspondence with those of the original tree. Transforming to binary tree representation allows us to apply early pruning and scale to large instances. If we apply the DP approach to the original non-binary tree, the calculation can quickly become infeasible. Consider a node u with k children, where each child has l Pareto optimal partial policies. In the worst case, when enumerating each group of partial policies, we would have to consider $l^k 2^k$ new partial policies when considering whether or not to build each dam. For example, the root node in the full Amazon basin has $k = 44$; therefore, running DP becomes infeasible even for very small values of l. Converting the original tree into a binary tree ensures that at each node, the partial policies of at most two children are considered for pruning. As a result, the binary tree leads to a dramatic increase in efficiency by allowing for earlier and more frequent pruning of partial solutions, which is critical for ensuring that the set of Pareto optimal partial policies remains of reasonable size throughout the computation.

The Exact DP-based Algorithm: The critical insight that motivated our algorithm is that at a given node u, we only need to keep the Pareto optimal partial solutions. Let Φ_u denote the set of all edges in the *subtree* rooted at u and π_u be a partial policy on u that only includes edges in Φ_u;

in other words, $\pi_u \subseteq \Phi_u$. A partial policy is Pareto optimal if and only if it is not dominated by any other partial policies on the same node. Our dynamic programming algorithm is based on the following theorem, which states that all Pareto optimal partial policies on a node can be computed based on Pareto optimal partial policies from its children.

Theorem 2. *Let l and r be the children of u. Any Pareto-optimal partial policy at u can be constructed by combining one Pareto-optimal partial policy from node l, one from node r, and four different joint states of edge (u, l) and (u, r).*

Proof. Here we use proof by contradiction. Suppose π_u is a Pareto-optimal partial policy for node u. l and r are u's left and right child. π_l is the partial policy π_u mapped on l; i.e., it contains all edges of π_u that are in the subtree Φ_l. π_r is the partial policy π_u mapped on r. Suppose Theorem 7 is false, then at least one of π_l and π_r are not Pareto optimal. Without loss of generality, suppose π_l is not Pareto optimal and is dominated by π'_l. This implies that $z_l^i(\pi'_l) \geq z_l^i(\pi_l)$ for all i and at least one strict inequality holds. Take $\pi'_u = (\pi_u \setminus \pi_l) \cup \pi'_l$. We can see that π'_u dominates π_u, which contradicts our assumption. □

Motivated by Theorem 2, we design a DP-based algorithm to recursively compute the Pareto optimal partial policies from leaf nodes to the root. Instead of keeping all partial policies, we only keep the Pareto-optimal partial policies at each node. The dynamic programming algorithm is shown in Algorithm 1.

The recursion of computing the Pareto optimal partial policy for node

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^d)};
3  else
4      l ← u.left_child;
5      r ← u.right_child;
6      Pleft ← Pareto_T(l);
7      Pright ← Pareto_T(r);
8      P ← ∅;
9      foreach (z_l^1, ..., z_l^d) ∈ Pleft do
10         foreach (z_r^1, ..., z_r^d) ∈ Pright do
11             foreach I(ul ∈ π) ∈ {0, 1} do
12                 foreach I(ur ∈ π) ∈ {0, 1} do
13                     Compute (z_u^1, ..., z_u^d) ;
14                     Remove from P all points dominated by
                        (z_u^1, ..., z_u^d);
15                     if (z_u^1, ..., z_u^d) is not dominated by any point in P
                        then
16                         P ← P ∪ {(z_u^1, ..., z_u^d)};
17     return P;
```

u is as follows. Suppose l and r are the two children of node u. We first compute the Pareto optimal partial policies for l and r (lines 6 and 7). Then we enumerate each pair of partial policies from node l and r (lines 9 and 10) and consider four different combinations of whether to include edge (u, l) and (u, r) (lines 11 and 12). For each combination, we compute the objective values and add them to the policy set P. Finally, we remove all dominated policies (lines 14-17). Based on Theorem 2, the remaining policies are Pareto optimal for node u. Note that it is trivial to adapt this algorithm to include the case when node u only has one child.

2.2.2 Approximation with Rounding

In practice, the optimal Pareto frontier may have an exponential number of points. In this case, we propose `Pareto_approx` based on the rounding technique that can give an FPTAS to the multi-objective optimization problem.

We first explain how rounding works for connectivity and sediment. The rounding scheme for additive objectives such as energy and biodiversity impact, which will be detailed later, is very similar to the rounding scheme presented here, except for an additional pre-rounding step to take into account the fact that the node rewards vary according to policy.

We introduce a hyperparameter K_v^i for each node and each objective. Define a rounded objective value $\hat{z}_u^i(\pi)$, which is calculated by the following recursion, $\hat{z}_u^i(\pi) =$

$$r_u^i + \left\lfloor \frac{\sum_{v \in ch(u)} \left(I_{uv}(\pi) p_{uv}^i + (1 - I_{uv}(\pi)) q_{uv}^i \right) \hat{z}_v^i(\pi)}{K_u^i} \right\rfloor K_u^i. \tag{2.3}$$

In Equation. 2.3, the fraction less than K_u^i is removed so that the number of different rounded objective values is reduced. This idea is similar to rounding different fractional numbers to the same integer value.

The rounded dynamic programming algorithm `Pareto_approx` is similar to the exact DP algorithm, except that we use Equation 2.3 to calculate objective values in line 13 of Algorithm 1.

Proposition 1. *$z^i(\pi) \geq \hat{z}^i(\pi)$ for any hyperparameter K_v^i and any policy π.*

This proposition is a direct consequence of taking the floor operator in Equation 2.3.

Proposition 2. *If we set $K_u^i = \epsilon r_u^i$, for any policy π we have*

$$z^i(\pi) - \hat{z}^i(\pi) \leq \epsilon z^i(\pi).$$

To see why this proposition is true, at each node, we at most lose a value of K_u^i. Based on Equation 2.2, the total value we lost due to the floor operation in Equation 2.3 is

$$z^i(\pi) - \hat{z}^i(\pi) \leq \sum_{v \in V} p_{s \rightsquigarrow v}(\pi) K_u^i = \epsilon z^i(\pi). \tag{2.4}$$

Proposition 3. *The approximate algorithm computes all Pareto optimal points for the rounded objective.*

The approximate algorithm `Pareto_approx` is very similar to the exact algorithm `Pareto`, except for replacing the exact objective function with a rounded one. The correctness of `Pareto` does not depend on a particular objective function. Therefore, this proposition is true.

Theorem 3. *Let $P(s)$ be the set of (partial) Pareto optimal policies for a node s. Let $\overline{P}(s)$ be the set of approximate (partial) Pareto optimal policies computed via* `Pareto_approx` *with rounded objective function (Equation 2.3). We must have $\overline{P}(s)$ ϵ-approximates $P(s)$.*

Proof. Suppose π is in the exact Pareto frontier $P(s)$. If π is in $\overline{P}(s)$, we are done. Otherwise, because of Proposition 3, there must be at least one $\pi' \in \overline{P}(s)$, such that π' dominates π in terms of the rounded objective; i.e.,

$$\hat{z}^i(\pi') \geq \hat{z}^i(\pi), \tag{2.5}$$

for all i. Because of Proposition 1, we have

$$z^i(\pi') \geq \hat{z}^i(\pi'). \tag{2.6}$$

From Proposition 2, we have

$$\hat{z}^i(\pi) \geq (1-\epsilon) z^i(\pi). \tag{2.7}$$

Combining Equation (2.5) (2.6) and (2.7), we have

$$z^i(\pi') \geq (1-\epsilon) z^i(\pi), \tag{2.8}$$

which concludes the proof.

□

We can show that `Pareto_approx` runs in polynomial time by bounding the number of partial policies in $\overline{P}(u)$ for all node u. We first make an assumption as mentioned in the paper by Wu et al. [2014a].

Assumption 1. *There are universal constants c and C such that $c \leq r_u^i \leq C$ for all $u \in V$ and all i.*

Theorem 4. *With Assumption 1, the number of different rounded tuples in* $\overline{P}(u)$ *is* $O\left(\left(\frac{n_u}{\epsilon}\right)^d\right)$*, where* n_u *is the number of nodes in the subtree rooted at* u*.*

Proof. The upper bound of the i^{th} objective value of u is $UB_u^i = \sum_{v \in \mathcal{T}_u} r_v^i$, with $\mathcal{T}_u$ denoting the induced subtree rooted at u. Due to Assumption 1, the number of different tuples at u is bounded by

$$\prod_{i=1}^{d} \frac{UB_u^i}{K_u^i} \leq \prod_{i=1}^{d} \frac{n_u C}{\epsilon c} = O\left(\left(\frac{n_u}{\epsilon}\right)^d\right). \tag{2.9}$$

□

Theorem 5. *The runtime of the rounded dynamic programming algorithm is* $O\left(\left(\frac{n}{\epsilon}\right)^{2d}\right)$*.*

Proof. As we proved, the number of different rounded tuples $\overline{P}(u)$ at node u is $O\left(\left(\frac{n_u}{\epsilon}\right)^d\right)$. Let T_u be the runtime to compute all these tuples at u. We have the recursion

$$T_u = O(\overline{P}_l \overline{P}_r) + T_l + T_r. \tag{2.10}$$

Solving the recursion, we have $T_u = O\left(\left(\frac{n_u}{\epsilon}\right)^{2d}\right)$. Therefore, the total runtime is $T_s = O\left(\left(\frac{n}{\epsilon}\right)^{2d}\right)$, where n is the number of nodes in the tree. □

Rounding for additive objectives: We apply a two-step rounding method for additive objectives. Notice the rounding method described below can also be applied to connectivity and sediments to obtain the same approximation guarantee. For presentation purposes, we use hydropower as an example. In the first step, we round upfront the energy value h_{uv} to

$$\lfloor \frac{h_{uv}}{K_1} \rfloor K_1$$

, where $K_1 = \frac{\epsilon}{2} h_{min}$ and h_{min} is the minimal energy value among all dams. This rounding is similar to the one used in the FPTAS to solve the knapsack problem (see page 67 of [Williamson and Shmoys, 2011]). One can prove that we already get an FPTAS if we only apply the first-step rounding. (The proof is similar to that of the knapsack rounding.) However, the rounding in the first step is often too conservative. To obtain additional pruning, we apply a second rounding step in our DP approach when combining partial solutions from the children of the parent node. The rounding scheme looks exactly like the one we use for connectivity and sediments in 2.3, where K_u^i is set to $\lfloor \frac{r_u^i}{h_{min}} \rfloor \frac{\epsilon}{2} h_{min}$, except that we do not round when $r_u^i(\pi) = 0$. The second rounding is more aggressive. As more partial solutions are rounded to the same value, the algorithm is able to prune more solutions, therefore scaling to larger instances.

To prove the approximation guarantee of the two-step rounding, we show that the first and the second rounding each incurs at most a $\epsilon/2$-approximation. Therefore, the entire algorithm has an ϵ-approximation guarantee. Let $z_u^i(\pi)$ be the true objective function without any rounding; $\tilde{z}_u^i(\pi)$ is the objective value after applying the first rounding; $\hat{z}_u^i(\pi)$ is the objective value after applying both the first and the second rounding. We are able to prove that

$$z_u^i \pi - \tilde{z}_u^i(\pi) \leq \frac{\epsilon}{2} z_u^i(\pi) \tag{2.11}$$

$$\tilde{z}_u^i(\pi) - \hat{z}_u^i(\pi) \leq \frac{\epsilon}{2} \tilde{z}_u^i(\pi) \tag{2.12}$$

Adding these two inequalities together, we get Proposition 2, on which the proof of the approximation guarantee mainly depends. The proof to

Equation 2.11 is similar to that for the knapsack algorithm and the proof to Equation 2.12 is similar to the proof of Proposition 2 with connectivity and sediments as objectives. To prove that the DP algorithm runs in polynomial time, we notice that during the execution of the algorithm, all intermediary energy values are in the form of kK_1, where k is an integer and K_1 is the rounding factor of the first step. As a consequence, the maximal number of different energy values at node u are bounded by

$$\frac{n_u h_{max}}{K_1} = \frac{2n_u h_{max}}{\epsilon h_{min}} = O(\frac{n_u}{\epsilon})$$

where n_u is the number of nodes of the induced subtree rooted at u and h_{max} is the maximal energy value.

Theorem 6. `Pareto_approx` *is an FPTAS for multi-objective optimization problems on tree-structured networks.*

Proof. This is a direct consequence of Theorem 3 and Theorem 5. □

2.2.3 Improving the Performance of the DP-based algorithm

While the approximation algorithm theoretically runs in polynomial time, its performance is still far from ideal when the number of objectives is larger than three. To further speed up the approximation algorithm, we introduced some new techniques when implementing the DP algorithm.

Using Divide-and-Conquer for Identifying Dominated Solutions

The major runtime bottleneck of Algorithm 1 is pruning the dominated solutions (lines 9-16). 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 boosts 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 $O(n \log n^{d-1})$ time if we use a comparison-based sort or $O(n \log n^{d-2})$ time if the data is stored in the Lattice Latin Hypercube (LLH) form [Yukish, 2004]. This algorithm is inspired by an approach proposed in [Yukish, 2004].

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

When the number of objectives is two, we use Algorithm 2, which sorts the solutions based on the first objective (good ones first). The first solution must be Pareto optimal. Then, we go through the list of solutions sequentially and look for solutions with a 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

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 P[−1] then
5         Append s to P;
6 return P;
```

the set of solutions S into two sets, A and B, of approximately the same size based on the last objective so that solutions in A have a better last objective than solutions in B (line 8). The splitting procedure is shown in Algorithm 4. Then, we recursively identify the non-dominated solutions from A and B (lines 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 objectives of solutions in A' are better than the dth objectives of solutions in B', so we only need to consider the first $d-1$ objectives. 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

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;          // this is the base case of recursion.
5  else if d = 2 then
6  |  return Non_Dominated_2D(S);     // this is just a special
   |   case
7  else
8  |  A, B ← Split(S, d);
9  |  A' ← Non_Dominated(A) ;  // all non-dominated solutions
   |   in A must be non-dominated in S.
10 |  B' ← Non_Dominated(B);
11 |  return A' ∪ Marry(A', B', d − 1);
```

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 on the d'th objective so that solutions in X have better d'th criterion than solutions in Y (line 7). Next, we consider the four disjoint subsets $X \cap A$, $X \cap B$, $Y \cap A$, and $Y \cap B$. Note that they cover all solutions in $A \cup B$, and $|(X \cap A) \cup (X \cap B)| \approx n/2 \approx$

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 element of each solution.

```
1 Find the median v_d of the dth element of solutions is S;
2 if we aim to maximize the dth element then
3 |   A ← {s ∈ S | s[d] ≥ v_d};
4 else
5 |   A ← {s ∈ S | s[d] ≤ v_d};
6 B ← S − A ;     // A and B are disjoint and solutions in A
    have better dth elements.
7 return A, B;
```

$|(Y \cap A) \cup (Y \cap B)|$. In lines 8 and 9, we recursively call $Marry$ on half-sized problems. Similarly, we know that the solutions in B_x are non-dominated in $A \cup B$. Still, we must 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 recursively call $Marry$ on $X \cap A$ and B_y. Note that solutions in $X \cap A$ have better d'th objective than solutions in B_y, so we only need to consider the first $d' - 1$ objectives. Hence, we return $B_x \cup B'_y$.

For the runtime analysis, we assume that we use a comparison-based sorting algorithm. The time complexity of `Non_Dominated_2D`(S) is $O(n(\log n)^{d-1})$.

Proposition 4. *Given a set S of n 2-dimensional solutions,* `Non_Dominated_2D`(S) *runs in $O(n \log n)$ time.*

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);        // base case of
         recursion
4  else if n = 1 then
5      return A ∪ 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');
9      B_y ← Marry(Y ∩ A, Y ∩ B, d');
10     B'_y ← Marry(X ∩ A, B_y, d' − 1);
11     return B_x ∪ B'_y
```

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

Proposition 5. *Given a set S containing n solutions,* `Split`(S, d) *runs in $O(n \log n)$ time.*

This is also a result of the sorting step taking $O(n \log n)$ time.

Proposition 6. *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, $\texttt{Marry}(A, B, d')$ *runs in* $O(n \log n^{d'-1})$ *time.*

Proof. We denote the runtime of $\texttt{Marry}(A, B, d')$ as $t(n, d')$. For the two base cases $d' = 2$ and $n = 1$, the proposition obviously holds. Assume the proposition holds for $d' < d'_0$. Now we consider cases where $n = 2^k$ for some positive integer k and $d' = d'_0$. We have

$$\begin{aligned}
t(2^k, d'_0) &= O(n \log n) + 2 \cdot t(n/2, d'_0) + t(n, d'_0 - 1) \\
&= 2 \cdot t(2^{k-1}, d'_0) + O(n(\log n)^{d'_0-2}) \\
&= 2 \cdot t(2^{k-1}, d'_0) + O(2^k \cdot k^{d'_0-2}) \\
&= 4 \cdot t(2^{k-2}, d'_0 - 1) + O(2^k \cdot k^{d'_0-2}) + 2 \cdot O(2^{k-1} \cdot (k-1)^{d'_0-2}) \\
&= \cdots\cdots \\
&= 2^k \cdot t(1, d'_0 - 1) + O(2^k \cdot \sum_{i=1}^{k} i^{d'_0-2}) \\
&= O(n(\log n)^{d'_0-1}).
\end{aligned}$$

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

Proposition 7. *Given a set* S *containing* n d*-dimensional solutions* ($d \geq 3$)*,* $\texttt{Non\_Dominated}(S)$ *runs in* $O(n(\log n)^{d-1})$ *time.*

Proof. We denote the runtime as $T(n, d)$. For the base case $n = 1$, the proposition obviously holds. We know that the runtime of the sorting step is $O(n \log n)$ and the runtime of `Marry` is $O(n(\log n)^{d'-1})$. Now we consider

cases where $n = 2^k$ for some positive integer k. Then,

$$\begin{aligned}
T(n,d) &= O(n \log n) + 2 \cdot T(n/2, d) + t(n, d-1) \\
&= 2 \cdot T(2^{k-1}, d) + O(n(\log n)^{d-2}) \\
&= 2 \cdot T(2^{k-1}, d) + O(2^k \cdot k^{d-2}) \\
&= 4 \cdot T(2^{k-2}, d) + O(2^k \cdot k^{d-2}) + 2 \cdot O(2^{k-1} \cdot (k-1)^{d-2}) \\
&= \cdots\cdots \\
&= 2^k \cdot T(1, d) + O(2^k \cdot \sum_{i=1}^{k} i^{d-2}) \\
&= O(n(\log n)^{d-1}).
\end{aligned}$$

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

Other 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 speed up the DP algorithm and allow us to solve problems on much larger networks, with higher precision, and with more criteria. However, the number of solutions to evaluate grows exponentially with the number of criteria 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 speeds up the DP algorithm.

2.3 The Mixed Integer Programming Based Algorithm

While the DP-based algorithm can efficiently approximate the whole Pareto frontier, one flaw of the algorithm is that it produces all the solutions simultaneously, which may be excessive if we are only interested in a small section of the solution space. To complement the DP algorithm, we propose a Mixed Integer Programming (MIP) formulation of the hydropower dam placement problem and a scheme for approximating the Pareto-frontier using MIP based on ideas from [Papadimitriou and Yannakakis, 2000a]. The key idea is to divide the space of objectives into small hyper-rectangles and query whether a **feasible** solution exists in each hyper-rectangle. Then, in each feasible cell, 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 rectangular cell is $(1+\epsilon)$ of the lower bound, the set of non-dominated solutions from S forms an ϵ-approximate Pareto-frontier [Papadimitriou and Yannakakis, 2000a]. This scheme clearly has room for improvement. Namely only the non-dominated solutions are needed for the Pareto frontier, which means that the computation for the dominated solutions is wasted.

In this section, we first show the MIP formulation of the problem and the naive approximation scheme based on ideas from [Papadimitriou and Yannakakis, 2000a]. Then, we introduce an improved scheme to reduce the number of MIPs to solve and exploit restarts to improve the runtimes of MIPs.

2.3.1 Mixed Integer Programming Formulation

As explained in Section 2.1, we use a number of objectives to characterize the Amazon hydropower siting problems: Hydropower (E), Longitudinal Connectivity (C), Sediment (S), Fish Biodiversity Impact (B), Degree of Regulation (DOR), Greenhouse Gas Emissions (GHG), and Seismic Risk (S_s). We first construct a MIP formulation of the problem. We will be using the following notations:

- V: the set of all hypernodes.
- D: the set of all edges (dams).
- s: the root of the tree (also the mouth of the river network).
- $e = (u, v)$: u is the hypernode downstream of e and v is the hypernode upstream of e.
- c_v: the total connectivity of river segments in a hypernode v.
- s_v: the total sediment produced in a hypernode v.
- h_e: hydropower generation (installed capacity) of a dam e.
- b_e: fish weighted endemism index of dam e.
- $C_{DOR}(e)$: cumulative downstream flow impact of dam e. Values of $C_{DOR}(e)$ are pre-computed.
- g_e: greenhouse gas emissions caused by dam e.
- ss_e: seismic risk at the site of dam e.

- p_e: percentage of sediment trapped by dam e if built.
- π: a set of dams we plan to build
- π_e: indicator variable for $e \in \pi$.
- n_v: indicator variable for v can be reached from the root uninterrupted.
- y_v: the percentage of the sediment produced at the hypernode v not trapped by dams.

Next, we show our encoding of the objectives:

Longitudinal Connectivity (C): To define connectivity we assign a binary variable $n_v \in \{0, 1\}$ to each node v, which represents whether a node can be reached uninterrupted.

$$C = \sum_{v \in V} n_v c_v.$$

For each hypernode $v \neq s$, $n_v = 1$ if and only if, for the downstream hypernode u, $n_u = 1$ and the dam $e = (u, v)$ is not built ($\pi_e = 0$). These relationships can be encoded using the following constraints:

$$n_s = 1; n_u \geq n_v, \forall (u, v) \in D$$
$$n_v \leq 1 - \pi_{u,v}, \forall (u, v) \in D$$
$$n_v \geq n_u - \pi_{u,v}, \forall (u, v) \in D$$

Sediment (S): For each hypernode v, we define a continuous variable $y_v \in [0, 1]$ that represents the percentage of the sediment produced at the hypernode not trapped by dams downstream. Obviously, at source s, $y_s = 1$.

Also, $y_v = \begin{cases} y_u & \text{if } \pi(i) = 0, \\ (1 - p_e)y_u & \text{otherwise.} \end{cases}$ This condition can be linearized using the big M method. Since $y_v \in [0, 1]$ for all $v \in V$, we can replace the condition with

$$y_v \leq y_u \text{ and } y_v \geq (1 - p_e)y_u, \forall (u, v) \in D$$

$$y_v \leq (1 - p_e)y_u + (1 - \pi_e), \forall e = (u, v) \in D$$

$$y_v \geq y_u - \pi_e, \forall e = (u, v) \in D.$$

MIP solvers like CPLEX can also linearize the constraint automatically. Then, $S = \sum_{v \in V} y_v s_v$.

Hydropower (E): Naturally, $E = \sum \pi_e \cdot h_e$.

Fish Biodiversity (B), Degree of Regulation (DOR), Greenhouse Gas Emissions (GHG), and Seismic Risk (S_s): Very similar to Hydropower.

Note that Mixed Integer Programming is a single objective method. To solve this multiobjective problem, we apply the scheme proposed by [Papadimitriou and Yannakakis, 2000a] for approximately constructing the Pareto frontier. The scheme can be summarized as follows: for a multiobjective optimization problem, we divide the space of objectives into small rectangular cells of certain sizes, and, for each cell, query whether there exists a solution. For each cell with solutions inside, we pick one solution. The set of dominating solutions within all the solutions we picked form a ϵ-approximate Pareto-frontier.

Specifically, the rectangular cells should satisfy the condition that, for each dimension, the upper bound should be $(1 + \epsilon)$ of the lower bound

Minimize: x

subject to: $x = 0$ (x is a dummy variable)

$C \in [\hat{C}, (1+\epsilon)\hat{C}];\ S \in [\hat{S}, (1+\epsilon)\hat{S}]$

$E \in [\hat{H}, (1+\epsilon)\hat{E}]$

$$C = \sum_{v \in V} n_v c_v;\ S = \sum_{v \in V} y_v s_v$$

$$E = \sum_{e \in D} \pi_e h_e$$

$n_v \in \{0,1\}, \forall v \in V; \pi_e \in \{0,1\}, \forall e \in E$

$n_s = 1; n_u \geq n_v, \forall (u,v) \in D$

$n_v \leq 1 - \pi_{u,v}, \forall (u,v) \in D$

$n_v \geq n_u - \pi_{u,v}, \forall (u,v) \in D$

$$S = \sum_{v \in V} s_v y_v, y_s = 1$$

$y_v \leq y_u$ and $y_v \geq (1-p_e) y_u, \forall (u,v) \in D$

$y_v \leq (1-p_e) y_u + (1-\pi_e), \forall e = (u,v) \in D$

$y_v \geq y_u - \pi_e, \forall e = (u,v) \in D$

Figure 3: An example of the MIP encoding of the problem for three objectives E, C, and S

(assuming the objectives are always positive values and are to be maximized). For instance, assume we are considering three objectives: C (longitudinal connectivity), S (Sediment), and E (Hydropower). Let $C_{\min}$

and $C_{\max}$ denote the minimum and maximum possible connectivity values for all possible solutions. We let $\hat{C}$ range over $\{C_{\min}(1+\epsilon)^k \mid k = 0, 1, 2, ...$ and $C_{\min}(1+\epsilon)^k \leq C_{\max}\}$ and define $\hat{E}$ and $\hat{S}$ similarly. Then, each rectangular cell should be of the form $[\hat{C}, (1+\epsilon)\hat{C}]\times[\hat{S}, (1+\epsilon)\hat{S}]\times[\hat{E}, (1+\epsilon)\hat{E}]$. We will then use MIP to check if there is a feasible solution in each rectangular cell. An example of the MIP encoding for three objectives is shown in Figure 3.

2.3.2 Improving the Performance of the MIP Approach

To improve upon the naive scheme, we start by **optimizing** for one of the objectives and dividing 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 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 objectives, cycling through every objective as the target objective to get better coverage and a more exact approximation. 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 and $\bar{P}$ be the set of non-dominated solutions from S. Then, $\bar{P}$ ϵ-approximates

P.

Proof. Without loss of generality, assume objectives $z^1, \cdots, z^d$ are all to be maximized, and suppose we optimize for z^1. For any $\pi \in P$, assume $(z^1(\pi), \cdots, z^d(\pi))$ lies in the rectangular cell $[\hat{Z}_1, (1+\epsilon)\hat{Z}_1] \times \cdots \times [\hat{Z}_d, (1+\epsilon)\hat{Z}_d]$. Since there is already a solution π in the rectangular cell, MIP can find a solution π' in $[\hat{Z}_2, (1+\epsilon)\hat{Z}_2] \times \cdots \times [\hat{Z}_d, (1+\epsilon)\hat{Z}_d]$ that maximizes z^1, which means that $z^1(\pi') \geq z^1(\pi)$. Hence, we may conclude that$(1+\epsilon)z^i(\pi') \geq z^i(\pi)$ for all $i = 1, \cdots, d$, which means that 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.

□

We also observe fat and heavy-tailed behavior in the MIP runtime distributions [Gomes et al., 2000]. Hence, we employ a geometric restart strategy, doubling the cutoff time in every run to improve the performance of the MIP [Walsh, 1999, Gomes et al., 2000]. Our experiments show that the restart strategy significantly boosts performance.

2.4 Experimental Results

Data: For our experiments, we use real-world data from the Amazon basin. In particular, we use the Amazon river network for the tree-structured graph and corresponding values for the relevant objectives described in Section 2.1. We generate a directed rooted tree that collapses contiguous regions of the network without dams into a node and associates with each node and edges with the corresponding values of each objective. To test the performance

of the DP-based methods at different scales, we used three sets of data of varying sizes in our experiments: the Marañon Basin (95 nodes), Western Amazon (173 nodes), and the entire Amazon Basin (510 nodes). The locations of these basins are shown in Figure 1. The entire Amazon basin has more than four million (4083059) river segments, whereas the western Amazon and the Marañon basin have 455156 and 128801 river segments, respectively. We compare the performance between the unmodified DP-based algorithm, the modified DP-based algorithm, the naive MIP implementation, and the modified MIP implementation. See Fig.4 and Table 1 for a summary of the results. Note that we are currently unable to run the DP-based algorithm for five or more objectives with reasonable time and error bound for the whole Amazon basin, so here we will only show results for two to four objectives. We shall explain our methods of dealing with higher-dimensional Pareto-frontiers in Section 2.6.

Exact Pareto frontier: The DP approach can compute the exact Pareto Frontier (time permitting), which is infeasible for the MIP approach. We see that both the original DP approach and the modified DP approach can compute the exact Pareto frontier, for connectivity and hydropower, for the entire Amazon basin, which contains 39841 solutions, in around 5 hours (17170 secs) and 6 minutes. (See figure 5.)

Quality of the Pareto Frontier approximation: We observe that, for the same $\epsilon > 0$, the accuracy of the DP approximation is, in general, better than the MIP approximation methods. More specifically, we notice that, for a relatively larger theoretically guarantee of $\epsilon = 0.1$, the actual approximate Pareto frontier computed by DP methods is still very close to

B	Criteria	ϵ	DP-Orig (secs)	DP-New (secs)	MIP-Naive (secs)	MIP-New (secs)	DP#Sol	MIP#Sol
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	exact	2077	64	N/A	N/A	25732	N/A
M	E, S	0.001	4	2	1day+	1day+	318	—
WA	E, S	exact	***46291***	***924***	N/A	N/A	***58808***	N/A
WA	E, S	0.001	30	13	1day+	2187	2668	1671
A	E, S	exact	mem	15153	N/A	N/A	177490	N/A
A	E, S	0.001	2368	226	1day+	1day+	7973	—
A	E, S_s	exact	54581	335	N/A	N/A	72591	N/A
A	E, S_s	0.001	2471	83	***35050***	***31***	8737	1558
M	E, C, S	exact	***2day+***	***526***	N/A	N/A	***283898***	N/A
M	E, C, S	0.001	630	32	1day+	1day+	5563	—
WA	E, C, S	0.001	mem	1120	1day+	1day+	88710	—
WA	E, C, S	0.005	254	69	1day+	65638	12655	4129
A	E, C, S	0.05	1680	58	1day+	1day+	12866	—
A	E, C, S	0.1	40	6	1day+	4503	4724	62
A	E, C, S_s	0.005	***mem***	***88246***	***1day+***	***4809***	***2274168***	***40981***
A	E, C, S_s	0.05	109910	2121	238	51	47978	581
M	E, C, S, S_s	0.001	mem	53620	1day+	1day+	1479660	—
M	E, C, S, S_s	0.1	886	28	1day+	15484	1406	773
WA	E, C, S, S_s	0.01	mem	695153	1day+	1day+	3540829	—
WA	E, C, S, S_s	0.1	47704	1154	1day+	1day+	107087	—
A	E, C, S, S_s	0.1	mem	19510	1day+	1day+	491578	—
A	E, C, S, S_s	0.25	11358	410	1day+	1505	23019	95

Table 1: Samples of runtimes and numbers of solutions for the different methods ran on a single core. Notations: A (Amazon, 510 nodes); WA (Western Amazon; 173 nodes) and M (Marañon; 95 nodes) (E hydropower; C connectivity; S sediment; S_s Seismic Risk). *mem* denotes memory limit. N/A denotes MIP cannot produce the exact Pareto frontier. We bold several entries to highlight the performance improvements.

the exact Pareto frontier. (See Figure 4) This observation suggests that the accuracy of the DP algorithm in practice is much better than the theoretical guarantee. The solutions of the modified MIP methods are almost always located on the exact Pareto frontier but much sparser than DP approaches;

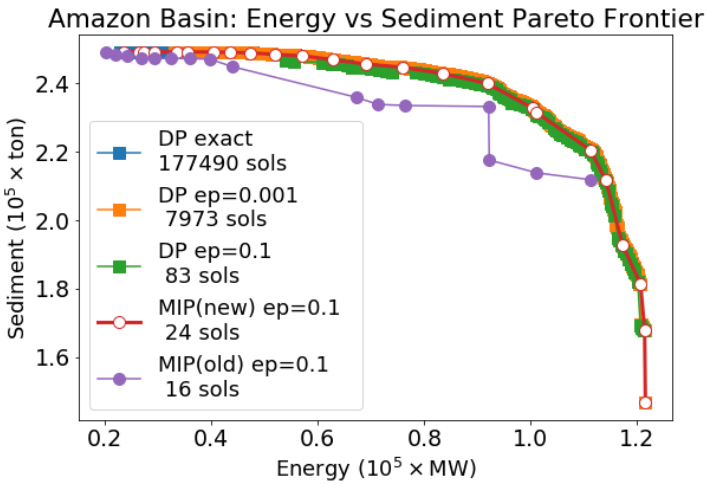


Figure 4: Example of the exact and approximate two-dimensional Pareto-frontier computed by the DP-based and MIP-based methods for energy and sediment. Note that the approximate DP ($\epsilon = 0.001$) almost overlaps the exact DP, except at the beginning of the Pareto curve. Also, we notice that, for a relatively larger theoretically guarantee of $\epsilon = 0.1$, the actual approximate Pareto frontier is still very close to the exact Pareto frontier. This suggests that the accuracy of the DP algorithm in practice is much better than the theoretical guarantee. We also observe significant improvements in accuracy from the modified MIP approach.

the solutions of the naive MIP methods are further away from the Pareto frontier and fewer but still within error bounds.

Approximation Coverage of the Pareto Frontier: The DP approximation has substantially better coverage of the exact Pareto frontier compared to the MIP approximation, which is reflected in the larger number of solutions it produces. See Figures 4 and 5.

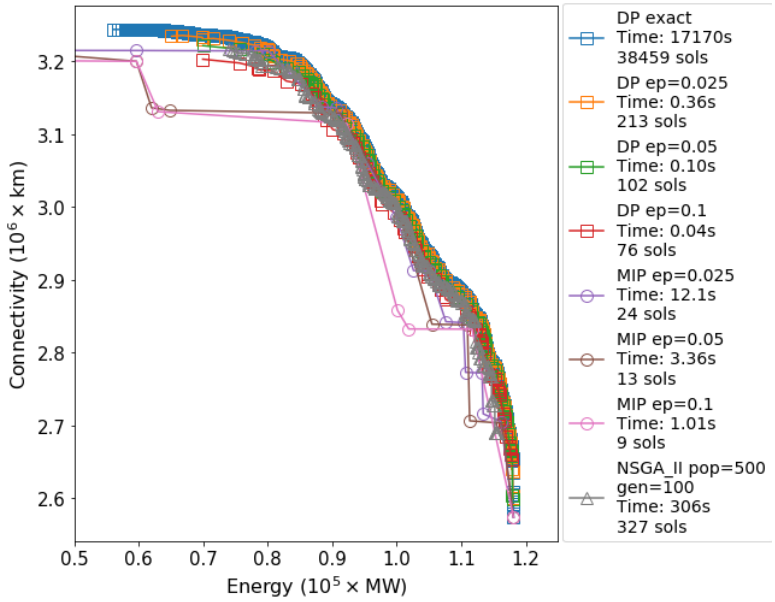


Figure 5: Approximation quality comparison of the DP methods, naive MIP method, and NSGA-II: The DP approximations all nearly overlap the exact Pareto curve (blue); NSGA-II performs slightly worse than DP methods but better than naive MIP methods; The modified MIP solutions (omitted here for the sake of clarity) will be on the exact Pareto frontier but with fewer solutions.

Evolutionary algorithms have been widely used to approximate (without approximation guarantees) Pareto frontiers (see e.g., [Neumann, 2007],[Qian et al., 2016], [Wiecek et al., 2008], [Märtens and Izzo, 2013]). Here we include a comparison to the Non-dominated Sorting Genetic Algorithm (NSGA-II). From Figure 5, we see that NSGA-II can achieve similar levels of accuracy as the DP approximations with $\epsilon = 0.1$ but takes longer to compute (and without approximation guarantees).

Speed: Our experiments show that the modified DP approach is up to three orders of magnitude faster than the original DP approach and scales to significantly larger instances and more objectives. The batching technique also solves the issue of hitting the memory limit when computing for three or more objectives. The new MIP approach is generally faster and can now solve larger problems.

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

2.5 Optimizing for Dendritic Connectivity Index

A critical condition for applying the DP-based algorithm is that an optimal solution for a specific objective must still be optimal on the induced subtrees. While most of the objectives we care about satisfy this condition, there are still a few exceptions, the most important of which is the Dendritic Connectivity Index for potadromous fish, i.e., $\mathrm{DCI_P}$.

Almost all fish species move during the course of their life cycle, transiting between habitats to breed, forage, or escape predators [Lucas and Baras, 2001]. Thus, fisheries ecosystem services are particularly vulnerable to river connectivity loss. Planning to restore connectivity in impacted systems, or to pro-actively site infrastructure problems to minimize connectivity impacts, is a top challenge in maintaining sustainable ecosystem

service production from rivers.

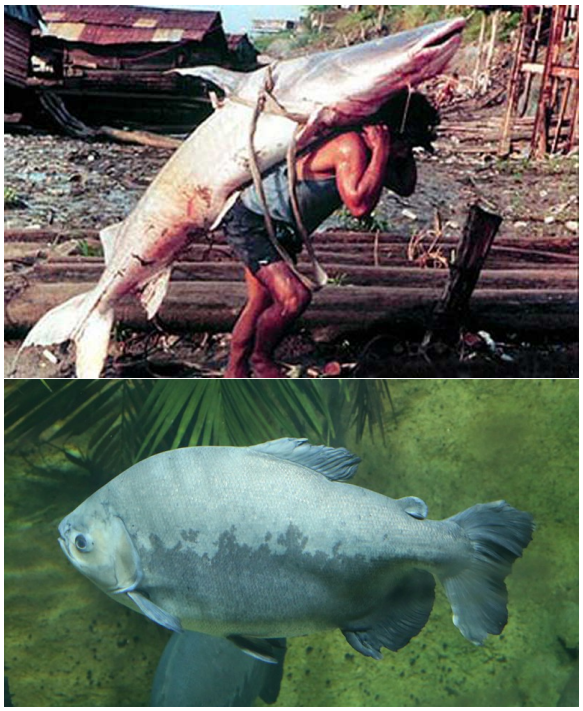


Figure 6: Amazonian fish species. (Top) Migratory goliath catfish (*Brachyplatystoma*): species in this genus travel throughout much of the Amazon basin during their life cycle and make the longest migrations known for freshwater fish, including measured distances between spawning and nursery areas exceeding 5700 km [Barthem et al., 2017]. (Photo credit: USGS). (b) Short-distant migrant (*Colossoma macropomum*): a commercially important fruit-eating fish that migrates from rivers to flooded Amazon forests during periods of high water. These fish species play vital roles in food security, recreation, and cultural identity, throughout the Amazon basin.(Photo credit: Wikipedia)

The migrations of fish species span a continuum of distances and are often divided into two major groups: (1) long-distance migrators that utilize the full river network length, such as *diadromous* species that move between the oceans to headwaters of rivers (1000s of km) (Fig.6, top panel) and (2) species that move within more localized regions of river networks (e.g., 10s-100s of km), such as *potadromous* species which exclusively utilize freshwater habitats throughout their life cycle. Cote et al. [2009] developed the **Dendritic Connectivity Index (DCI)** to describe the degree of river network connectedness, which can be mapped onto fish life histories. One version, DCI for longitudinal connectivity, describes the probability that the path between a root reference location and any other point in the network is unimpeded. This connectivity index is suited to describe how long-distance migrators use river networks, such as Pacific salmon (*Oncorhynchus spp.*) in North America or Amazonian catfish (*Brachyplatystoma spp.*) (Fig.6, top panel), which migrate over 5000 km from river mouth estuaries to headwater spawning habitats. As such, this connectivity metric is labeled $\mathrm{DCI_D}$ for *diadromy*. Alternatively, $\mathrm{DCI_P}$, for *potadromy*, quantifies the probability that the path between any two locations in the river network is unimpeded by a barrier. The $\mathrm{DCI_P}$ metric focuses on localized river connectivity and thus better describes the life history needs of fish that move short to middle distances (Fig.6, bottom panel)

In the previous sections, we consider **only the longitudinal connectivity (i.e., $\mathbf{DCI_D}$)**. In this section, we discuss how to include the localized connectivity (i.e., $\mathrm{DCI_P}$) in our optimization methods, which requires different encodings when using the DP-based and MIP-based methods.

2.5.1 Formal Definition of $\mathbf{DCI_P}$

The formal definition of the Dendritic Connectivity Index (DCI) proposed in [Cote et al., 2009] is the probability that a fish can successfully migrate from its starting point to its destination in a river network. When considering diadromous fish which migrate between marine and freshwater environments, we only need to compute the probability that a fish can successfully migrate from the mouth of the river to its destination in the river network. Let l_1 be the length of the connected section from the mouth of the river and L be the length of the whole river network. Under the assumption that the possible locations of the destination are distributed evenly across the entire network, the DCI for diadromous fish is defined as

$$\mathrm{DCI}_D = \frac{l_1}{L} \times 100,$$

which is essentially the longitudinal connectivity objective.

For potadromous fish which migrate within the freshwater system, we make the assumption that the fish migrate between two randomly chosen locations in the river network. Let $s_1, s_2, \cdots, s_k$ be the contiguous sections in the river network uninterrupted by dams or other barriers, and let $l_1, l_2, \cdots, l_k$ be the lengths of these sections, respectively. Let L be the length of the whole river network. Let p_{ij} be the passability coefficient between s_i and s_j, i.e., the probability that a fish can move between s_i and s_j. Then,

the DCI for potadromous fish is defined as

$$\mathrm{DCI_P} = \sum_{1 \le i,j \le k} \Pr[\text{move from } s_i \text{ to } s_j] \cdot p_{ij} \times 100 \tag{2.13}$$

$$= \sum_{1 \le i,j \le k} \frac{l_i}{L} \frac{l_j}{L} p_{ij} \times 100 \tag{2.14}$$

In this thesis, we only consider the case where the barriers that divide the river network are absolutely non-passable, which is valid for most fish species. This assumption implies that

$$p_{ij} = \begin{cases} 0, & \forall i \neq j, \\ 1, & \text{otherwise.} \end{cases}$$

Then, we can simplify DCI_P to

$$\mathrm{DCI_P} = \sum_{i=1}^{k} \frac{l_i^2}{L^2} \times 100. \tag{2.15}$$

The new formula of $\mathrm{DCI_P}$ in (2.15) makes it possible for us to apply a dynamic programming algorithm. For the MIP formulation, we still use the definition of $\mathrm{DCI_P}$ in (2.14).

To incorporate $\mathrm{DCI_P}$ into our methods, we assume that V is the set of all hypernodes and c_v is the total length of river segments in node $v \in V$. We use L to denote the full length of the entire river network. When we use the definition of $\mathrm{DCI_P}$ in (2.14), let x_{uv} be the indicator variable of whether u and v are connected. Then, we have

$$\mathrm{DCI_P}(\pi) = \left(\sum_{u,v \in V} c_u c_v x_{uv}(\pi) \right) / L^2 \times 100. \tag{2.16}$$

When we use the definition of $\mathrm{DCI_P}$ in (2.15), we assume that $|\pi| = k-1$ and these dams divide V into k subset $s_1, s_2, \cdots, s_k$. Thus,

$$\mathrm{DCI_P}(\pi) = \left(\sum_{i=1}^{k} (\sum_{v \in s_i} c_v)^2 \right) / L^2 \times 100. \tag{2.17}$$

For instance, in Figure 7, we decide to build dams (s, u), (u, x), and (v, y). The $\mathrm{DCI_P}$ value of this solution is then

$$\mathrm{DCI_P}(\pi) = \frac{(c_s + c_v + c_z)^2 + (c_u + c_w)^2 + c_x^2 + c_y^2}{L^2} \times 100.$$

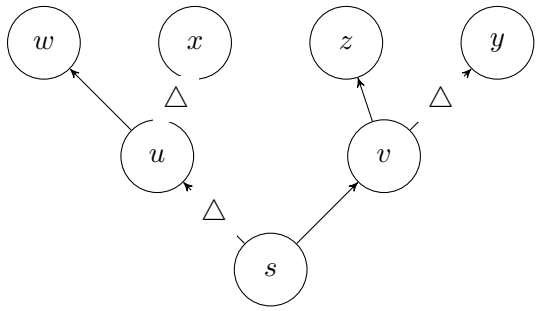


Figure 7: Example of a solution: The triangle means a dam is built at the location of the edge

2.5.2 Using the DP Algorithm to optimize for $\mathrm{DCI_P}$

The validity of the DP algorithm depends on the following theorem:

Theorem 7. *Let l and r be the children of u. Any Pareto-optimal partial solution at u can be constructed by combining one Pareto-optimal partial solution from node l, one from node r and four different joint states of edge (u, l) and (u, r).*

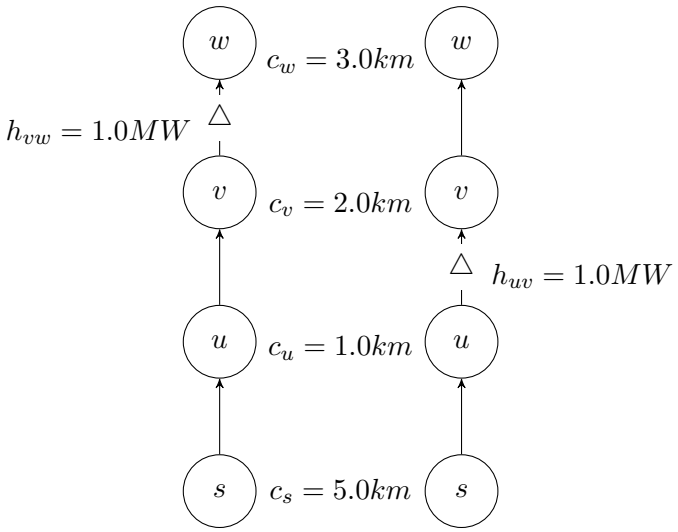


Figure 8: Counterexample of Theorem 7 when the objectives are DCI_P and hydropower. s is the root (river mouth) of the river network. Triangles represent dams we decide to build. Solution 1 provides better DCI_P than solution 2. However, when only considering the part of the network above u, solution 1 is dominated by solution 2.

However, this theorem does not hold when one of the objectives is DCI_P. A simple counterexample is shown in Fig. 8. If we directly apply the DP algorithm to optimize for energy and DCI_P on the river network shown in Fig. 8, we will miss a Pareto-optimal solution.

We observe that the central conflict between DCI_P and the DP algorithm is that when the DP algorithm is computing the Pareto-frontier at a node, it only considers the part of the network above the current node. However, the length of the section connected to the current node can still be expanded.

To address this problem, we need to keep track of two numbers: (1) the sum of squares of the lengths of all closed sections (sections that are separated from the current node by dams) above the current node, which we will call Q; (2) the total length of upstream river segments connected to the current node, which we call C. (Notice that C is essentially the numerator of $\mathrm{DCI_D}$ or the longitudinal connectivity, so when we are computing $\mathrm{DCI_P}$ using the DP algorithm, we also compute longitudinal connectivity as a byproduct.)

For instance, in Fig. 7, at node s, we have

$$Q_s = (c_u + c_w)^2 + c_x^2 + c_y^2,$$

and

$$C_s = c_s + c_v + c_z.$$

Combination Step: Assume the tree is binary. For a non-leaf node u, denote its children as l and r. Given fixed partial schemes on l and r, we now show how to combine these partial schemes at u for computing $\mathrm{DCI_P}$. If we decide not to build (u, l) and (u, r), then $Q_u = Q_l + Q_r$ and $C_u = C_l + C_r + c_u$. If we decide to build (u, l) but not (u, r), then $Q_u = Q_l + Q_r + C_l^2$ and $C_u = C_r + c_u$. If we decide to build both d_{ur} and d_{ul}, then $Q_u = Q_r + Q_l + C_r^2 + C_l^2$ and $C_u = c_u$.

At the root node s, we have

$$DCI_P = \frac{Q_s + C_s^2}{L^2} \times 100.$$

Pruning Step: At the pruning step, we treat C and Q as two separate objectives. For instance, assume we are optimizing for Hydropower and

DCI_P. Then, at a node u, a partial scheme dominates another partial scheme if and only if it has larger H, C, and Q values. We have the following theorem.

Theorem 8. *Assume π is a solution on the Pareto-frontier of the hydropower dam placement problem using DCI_P and other objectives that satisfy Theorem 7. π will not be eliminated in the pruning steps if we keep track of both C and Q.*

Proof. We use proof by contradiction. We use π_u to represent the part of the solution π on the subtree rooted at u. Assume for contradiction that, at a node u, π_u is dominated by π'_u. Then, we can modify π by replacing the part π_u with π'_u and get a new solution π'. Note that π' is superior to π in terms of DCI_P and all other objectives, i.e., π' dominates π, which contradicts previous assumptions. □

The above theorem ensures that, by splitting the DCI_P objective into C and Q, we can now apply the DP algorithm to successfully obtain the entire Pareto-frontier when one of the objectives is DCI_P.

Rounding Scheme for DCI_P: Recall that the main idea for the rounding scheme in the DP algorithm is to bound the number of solutions in every partial Pareto-frontier by rounding the objective values in all the partial solutions to a multiple of K, where the value of K depends on the objective and the current node. Here, we show that we can also apply this rounding technique to DCI_P-related problems and obtain an FPTAS.

Notice that C is essentially the same objective as longitudinal connectivity, so we will apply a similar rounding scheme as longitudinal connectivity.

At node u, assume that the total length of river segments in node u is c_u, then the longitudinal connectivity value is rounded to

$$\lfloor \frac{C}{\frac{\epsilon}{4}c_u} \rfloor \cdot \frac{\epsilon}{4} c_u.$$

This rounding scheme incurs at most $\frac{\epsilon}{4}C$ error.

Q is similar to the Hydropower objective since the increase at each node can be 0. Fortunately, the minimum value of Q at the root node s is

$$Q_{\min} = \sum_{v \in V, v \neq s} c_v^2,$$

which is reasonably large. Hence, here we can apply a fixed error rounding scheme. Note that the total number of rounding steps is at most n. Hence, let

$$K = \frac{\epsilon}{2n} Q_{\min}.$$

After each combination step, we round Q to

$$\lfloor \frac{Q}{K} \rfloor \cdot K.$$

The error incurred by the rounding of Q is at most $\frac{\epsilon}{2}Q_{\min}$.

One complication of the above rounding scheme is that Q is computed based on C. During the recursion process, whenever we decide to build a dam, we add the square of the C value of the cutoff branch to Q, so the rounding error of C will be carried over to Q. Let $\hat{C}$ represent the rounded value of C. Let $\tilde{Q}_s$ denote the value of Q at the root computed with rounding for C but no rounding for Q. Also, let $\hat{Q}_s$ denote the value of Q at the root computed with rounding for both C and Q. We know that

$$\tilde{Q}_s - \frac{\epsilon}{2} \sum_{v \in V, v \neq s} c_v^2 \leq \hat{Q}_s \leq Q_s.$$

Assume that D is the set of nodes under which the dams are built. Then, we have

$$Q_s = \sum_{v \in D} C_v^2$$

and

$$\tilde{Q}_s = \sum_{v \in D} \hat{C}_v^2 \geq \sum_{v \in D} [(1 - \epsilon/4)C_v]^2 \geq (1 - \epsilon/2) \sum_{v \in D} \hat{C}_v^2.$$

Let C_s denote the longitudinal connectivity value at the root node s. Then,

$$\begin{aligned}
\hat{Q}_s + \hat{C}_s^2 &\geq \tilde{Q}_s - \frac{\epsilon}{2} \sum_{v \in V, v \neq s} c_v^2 + [(1 - \epsilon/4)C_s]^2 \\
&\geq (1 - \epsilon/2) \sum_{v \in D} \hat{C}_v^2 - \frac{\epsilon}{2} \sum_{v \in V, v \neq s} c_v^2 + (1 - \epsilon/2)C_s^2 \\
&\geq (1 - \epsilon/2)(Q_s + C_s^2) - \frac{\epsilon}{2}(Q_s + C_s^2) \\
&\geq (1 - \epsilon)(Q_s + C_s^2)
\end{aligned}$$

Hence, we may conclude that the DCI_P computed by the rounding scheme is within ϵ of the actual DCI_P value. Following the same proof idea in Section 2.2, we have the following theorem:

Theorem 9. *The new DP algorithm with rounding is an FPTAS for multi-objective optimization problems on tree-structured networks with DCI_P as one of the objectives.*

2.5.3 Mixed Integer Programming Formulation for DCI_P

Encoding DCI_P into the MIP method is a much easier task. Note that we only consider the case of binary passability and assume that a dam will block all fish. The notations we used are as follows:

- V: the set of all nodes.
- E: the set of all edges (dams).
- x_{uv}: the indicator variable of whether fish can move between u and v. ($x_{vv} = 1$ for all $v \in V$.)
- π: the set of dams we plan to build.
- π_e: indicator variable of of $e \in \pi$.
- D_{uv}: the set of all edges on the unique path between u and v.
- H: total hydropower output of the dams.
- $\hat{H}$: the bound for hydropower. In practice, $\hat{H}$ will range over $\{(1+\epsilon)^i H_{\min} : i \geq 0, (1+\epsilon)^i H_{\min} \leq H_{\max}\}$.
- h_e: hydropower output of dam e.

Following the idea in Equation 2.16, we obtain the MIP formulation of the problem as shown in Fig. 9

2.5.4 Experimental Results

To evaluate the performance of the new methods at different scales, we used the same three datasets as in previous sections: the Marañon, Western Amazon, and Amazon basins, with 95, 173, and 510 hypernodes, respectively. We considered three criteria: hydropower generation, longitudinal connectivity (the numerator of $\mathrm{DCI_D}$), and $\mathrm{DCI_P}$. Note that the DP algorithm computes the longitudinal connectivity as a byproduct when computing for

$$
\begin{aligned}
&\text{maximize:} && \mathrm{DCI_P} \\
&\text{subject to :} && \mathrm{DCI_P} = \sum_{u \neq v \in V} \frac{c_u c_v}{L^2} \cdot x_{uv} + \frac{\sum_{v \in V} c_v^2}{L^2} \\
& && H = \sum_{e \in E} \pi_e h_e \\
& && H \geq \hat{H} \\
& && \pi_e \in \{0,1\}, \quad \forall\, e \in E \\
& && x_{uv} \in \{0,1\}, \quad \forall\, u \neq v \in V \\
& && x_{uv} \leq 1 - \pi_e, \quad \forall\, e \in D_{uv},\ \forall\, u \neq v \in V \\
& && x_{uv} \geq 1 - \sum_{e \in D_{uv}} \pi_e, \quad \forall u \neq v \in V.
\end{aligned}
$$

Figure 9: MIP formulation for optimizing $\mathrm{DCI_P}$ with constraint on H. We ignore the $\times 100$ term in $\mathrm{DCI_P}$.

$\mathrm{DCI_P}$. For a better comparison of the performances, we tested against two versions of the MIP formulation: The first one is the 2-dimensional MIP approximation scheme that optimizes for hydropower generation and $\mathrm{DCI_P}$; the second one is a 3-dimensional MIP approximation scheme that optimizes for hydropower generation, longitudinal connectivity, and $\mathrm{DCI_P}$. (See Section 2.3 for the MIP formulation of longitudinal connectivity and the MIP approximation scheme for higher dimensions). The performance of the three approaches is shown in Table 2. We observe that when we consider only two objectives: hydropower and $\mathrm{DCI_P}$, the MIP method is generally faster, especially in larger networks. When we also consider longitudinal connectivity, the DP method performs much better in terms of speed. In both cases, the DP approach provides better coverage of the Pareto-frontier. (See Figure 10.)

We also compared the results of optimizing for $\mathrm{DCI_P}$ with the results of optimizing for longitudinal connectivity (under the same energy production

B	ϵ	DP(sec) H,DCI_P, DCI_D	MIP (sec) H,DCI_P	MIP (sec) H,DCI_P, DCI_D	DP #Sol H,DCI_P, DCI_D	MIP #Sol H,DCI_P	MIP #Sol H,DCI_P, DCI_D
M	0	1076	–	–	55364	–	–
M	0.001	566	2627	1day+	37998	2055	1day+
M	0.005	202	599	1day+	10377	518	1day+
M	0.01	80	297	23323	4422	280	868
M	0.05	4	61	972	317	63	114
W	0.005	1day+	5052	1day+	1day+	530	1day+
W	0.05	6287	477	14761	1385	61	72
W	0.1	588	275	4285	511	34	41
A	0.025	1day+	5140	1day+	1day+	89	1day+
A	0.1	6616	1510	14742	2547	23	34

Table 2: Examples of runtimes and numbers of solutions for the objectives H (Hydropower), DCI_P and DCI_D for Marañon (M; 109 hypernodes), Western Amazon (W; 219 hypernodes) and Amazon (A; 468 hypernodes) showing the trade-offs between the different methods. Note that DP computes simultaneously for energy, DCI_P, and DCI_D, so we consider two versions for MIP: one optimizing for energy and DCI_P and the other one optimizing for energy, DCI_P, and DCI_D, so the time comparison is fairer.

constraint). See Figure 11 for an example. We observe that when optimizing for longitudinal connectivity, the optimal solutions tend to build every dam

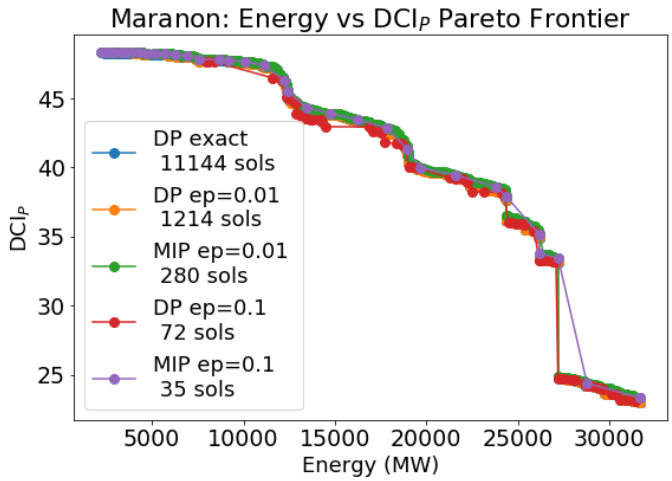
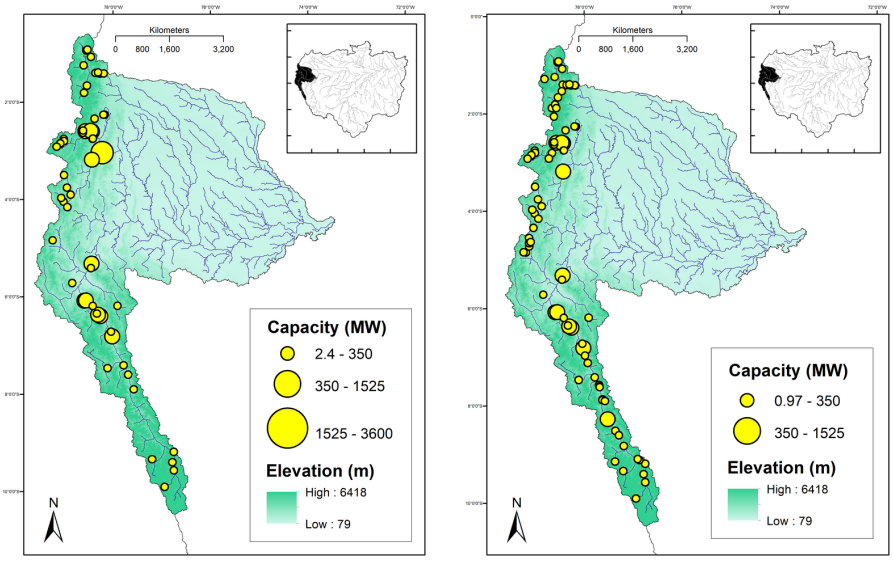


Figure 10: The Pareto frontier for hydropower generation vs DCI_P. The MIP and DP approximations for $\epsilon = 0.01$ nearly overlap the exact Pareto curve (blue). The DP approximations for $\epsilon = 0.01$ (red) lie slightly farther from the exact Pareto-Frontier than the corresponding MIP approximation (purple). The DP approximations have much better coverage of the exact Pareto frontier in the number of solutions. (Note: the number of solutions in the DP results is different from Table 2 because we eliminated the solutions that have better longitudinal connectivity but worse DCI_P.)

upstream of existing dams; when optimizing for DCI_P, the optimal solutions tend to build larger dams in order to not fracture the upstream area.



(a) Optimized for DCI_P (b) Optimized for DCI_D

Figure 11: The left and right panels depict two Pareto-optimal solutions for the Marañon basin, with similar total hydropower generation (around 14000MW): the left panel shows a solution using localized connectivity (DCI_P) as the second objective, while the bottom panel shows a solution using longitudinal connectivity (DCI_D) as the second objective. The solution optimized for DCI_D tries to build all dams upstream of existing dams (84 dams in total), while the solution optimized for DCI_P replaced many small dams with a larger one (3600MW) downstream (56 dams in total).

2.6 Using Expansion method to approximate high-dimensional Pareto Frontier

One major problem we face when approximating the Pareto frontier of the hydropower dam placement problem is that although the runtime of

the dynamic-programming-based algorithm is polynomial for the number of dams, it is still exponential with respect to the number of objectives, which means that both the runtime and the number of solutions greatly increase as the number of objectives goes up. For large river networks such as the Amazon basin, while the algorithm is able to solve three or four-objective optimization problems efficiently and with a small approximation factor, its performance drops off dramatically once we reach five objectives. However, to encompass the complexity of balancing hydropower generation with ecosystem service impacts in the Amazon, all six objectives (hydropower generation, River connectivity index, sediment transportation, biodiversity impact, degree of regulation, and greenhouse gas emissions) need be considered.

High-dimensional real-world data are often shown to dwell on lower-dimensional manifolds [Li et al., 2015, Huang et al., 2009]. Similarly, we make the assumption that, for multi-objective optimization problems on river networks, the six-objective Pareto frontier might approximately lie on a lower-dimensional manifold. Given that the DP-based algorithm is able to solve this type of MO problem with three or four objectives efficiently and with a guaranteed approximation factor and that these solutions are very likely to be on the Pareto frontier for more objectives, we conjecture that the aggregated solutions from Pareto frontiers optimized for all combinations of three or four-element subsets of the six objectives may form a good approximation of various local regions of the six-objective Pareto frontier. More generally speaking, here we consider two questions. We want to know if, for a specific n-objective optimization problem, there exists an in-

teger $k < n$ such that we can closely approximate the n-dimensional Pareto frontier by aggregating solutions from all possible k-objective Pareto frontiers. We call this approach the "expansion" method since it expands the k-objective Pareto frontiers into an n-objective Pareto frontier (optimized only w.r.t to k criteria).

2.6.1 The Expansion Method

For any solution π, we define the value vector of π to be

$$v(\pi) = (z^1(\pi), \cdots, z^n(\pi)).$$

We denote the actual n-objective Pareto frontier as P_n and define $V_n = \{v(\pi)|\pi \in P_n\}$. Given a positive integer $k < n$, for all $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, we compute an ϵ-approximate Pareto frontier $\tilde{P}_{i_1,\cdots,i_k}$ w.r.t. $z^{i_1}, z^{i_2}, \cdots, z^{i_k}$. We define the union for these k-objective Pareto frontiers to be

$$\tilde{P}_k = \bigcup_{1 \leq i_1 < \cdots < i_k \leq n} \tilde{P}_{i_1,\cdots,i_k},$$

and

$$\tilde{V}_k = \{v(\pi)|\pi \in \tilde{P}_k\}.$$

$\tilde{P}_k$ is the output of the **Expansion method**. In this paper, we study the following proposition:

Proposition 8. *For some real-world problems, $\tilde{V}_k$ forms a sufficiently good coverage of V_n with appropriate choices of k and ϵ.*

The **Expansion method** might look counter-intuitive at first because when we consider the smallest possible cases, two-criteria optimization so-

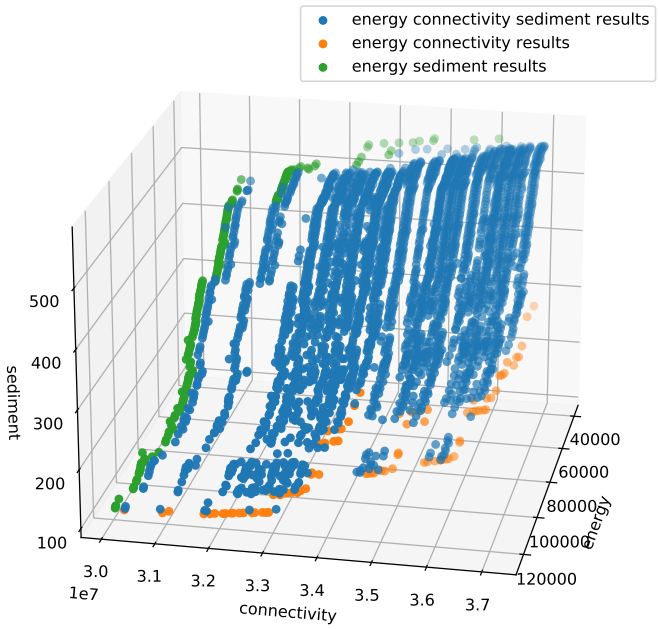


Figure 12: An example of trying to approximate a three-criteria Pareto frontier with two-criteria optimization results. Each dot represents the values of the three criteria of one solution. For the two criteria solutions, we compute the value of the remaining criterion based on the dams built in that solution. We can observe that the two-criteria solutions only cover the edges of the hyperplane formed by the three-criteria optimization results.

lutions usually only cover the edges of a three-criteria Pareto frontier (see Fig. 12). However, once we start approximating higher dimensional Pareto-frontiers with three criteria optimization results, we can see that the idea might actually work in practice. The first difficulty we face, however, is that it is hard to visualize Pareto frontiers of dimensions higher than 3. Thus, for the sake of clear representation, we use the t-Distributed Stochastic

Neighbor Embedding (t-SNE) [van der Maaten and Hinton, 2008] method to project the high-dimensional Pareto-frontier onto a two-dimensional map while preserving the general proximity relationships between the values of solutions. We merge the solutions generated by directly computing the six-objective Pareto frontier of the Marañón basin ($\epsilon = 0.1$) using the DP-based method and the expansion methods and use t-SNE to visualize these solutions. Fig. 13 illustrates how our method can cover almost every solution of the six criteria Pareto frontier computed by the Tree-DP algorithm (approximation factor 0.1) for the Marañón sub-basin.

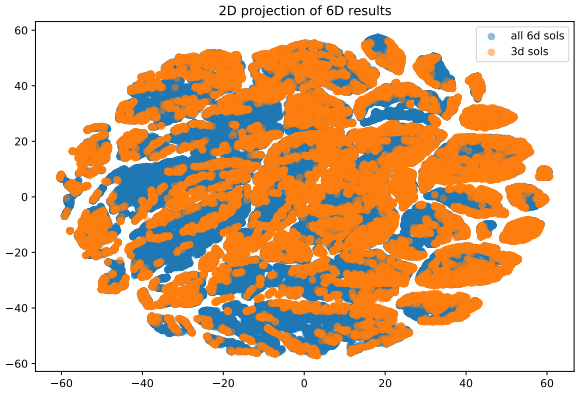


Figure 13: t-SNE projection of the approximate six-objective Pareto frontier of Marañón basin ($\epsilon = 0.1$) and the Expansion approximation results. We can observe that our method covers most solutions of the DP algorithm.

2.6.2 Experimental Results and Conclusions

Evaluation Method

To compare the optimization results of the various methods, we need a met-

ric that can evaluate both the optimality and the coverage of the approximate Pareto frontiers. Note that the exact Pareto frontier we are trying to approximate can be non-convex (see Fig. 14). So, not only do we care about the overall shape of the Pareto frontier, but we also evaluate the individual solutions. Therefore, we propose an evaluation method that divides the solution space into ϵ hypercubes following Papadimitriou and Yannakakis [2000b]'s approach.

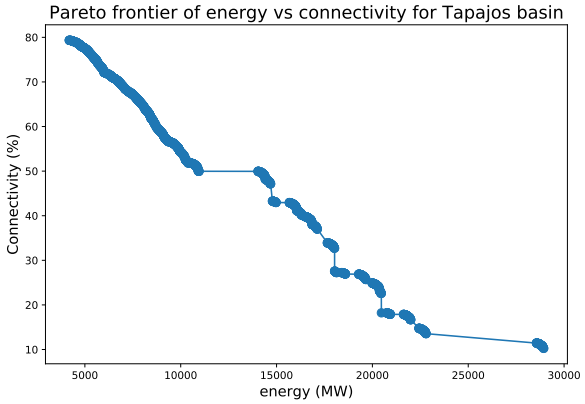


Figure 14: Exact non-convex Pareto frontier for two objectives of the Tapajos basin.

More specifically, for an n-objective optimization problem where, without loss of generality, every objective is to be maximized, and the objective values are always positive, we define the solution space to be an n-dimensional space where each axis represents the value of one objective. We also make the assumption that the minimum possible value of each objective is positive. For a given error bound $\epsilon > 0$, we divide the solution space into hypercubes where the upper bound is $1 + \epsilon$ the lower bound on each axis, with the smallest value of the lower bounds being the minimum

possible value of the corresponding objective. Similar to the definition of Pareto-dominance, for two different hypercubes, if for each axis, the upper bound of the first hypercube is greater than or equal to the upper bound of the second hypercube, we say that the first hypercube dominates the second hypercube.

To compare two or more sets of solutions, we first identify all of the ϵ-hypercubes that are occupied by at least one solution in any of the solution sets. We then find the set of occupied hypercubes that are not dominated by any other occupied hypercube. Finally, we compute for each set of solutions the number of non-dominated hypercubes they occupy. If one set of solutions covers more non-dominated ϵ-hypercubes than the other set, we say that the first solution set has better ϵ-coverage than the second solution set. Notice that since the set of occupied hypercubes will change depending on the sets of solutions compared, the number of non-dominated hypercubes covered by one set of solutions may change depending on the solution sets compared to, so the number of non-dominated hypercubes covered cannot be used as a universal metric of the quality of approximate Pareto frontiers. However, when comparing fixed sets of solutions, the metric provides a good comparison of the accuracy and coverage of the solution sets.

Experiment Setup

To show the general ability of our methods and provide scalability insights, we considered two smaller sub-basins of the Amazon basin: Marañón and Ucayali, for which we can directly compute the six-objective approximate Pareto frontier with an error bound of $\epsilon = 0.1$. (We can only optimize for six

objectives using $\epsilon = 1.0$ or 1.5 and the quality of the solutions is very poor.) We compute the Pareto frontiers with respect to the six important criteria introduced in the introduction: hydropower generation, river connectivity index, sediment transportation, biodiversity impact, the degree of river regulation, and greenhouse gas emissions. To speed up the computation, we use a parallelized version of the modified DP algorithm shown in Section 2.2 that computes the exact or approximate Pareto frontier. Our baseline is to directly optimize for all the six criteria with the minimal approximation factor the runtime constraints allow.

When setting up the expansion method, we always include hydropower generation as a single objective. Otherwise the optimal solution would be the trivial solution of building no dams. Our goal is approximate the six-objective Pareto frontier of the whole Amazon basin, where the DP algorithm can only compute or approximate the Pareto frontier within a reasonable amount of time for no more than 4 criteria. Thus, for the **Expansion method**, we consider all the possible three-objective combinations (referred to as **Expansion-3**) and four-objective combinations (referred to as **Expansion-4**), and they both consider $\binom{5}{2} = 10$ combinations.

Experimental Results

We use the number of non-dominated hypercubes occupied by the solutions computed by a given method as a metric of performance. For the comparison of solution sets, from both fine-grain and coarse perspectives, we used two hypercube error bounds, $\epsilon = 0.01$ and $\epsilon = 0.05$. The results are summarized in Table 3.

Basin	hypercube ϵ	Baseline (6D)($\epsilon = 0.1$)	Expansion-3 ($\epsilon = 0$)	Expansion-4($\epsilon = 0.01$)	Expansion-3 and 4	Total
Marañón	0.1	33	175	125	234	250
Marañón	0.05	135	580	472	930	1044
Ucayali	0.1	141	116	106	153	201
Ucayali	0.05	861	456	499	736	1252

Table 3: Non-dominated hypercubes occupied by the different methods. Note that the occupied non-dominated hypercubes are computed through merging and comparing DP solutions and Expansion solutions. The number in the parentheses is the number of criteria that are considered by the DP algorithm. The epsilon is used in the hypercube computation. Expansion is a good approximation for all the basins, outperforming the baseline DP, which provides a theoretical approximation guarantee.

We find the set of hypercubes occupied by the solutions of each method. We then combine these hypercubes and remove the dominated ones to form the hypercube Pareto frontier. Since the number of solutions for each sub-experiment is quite large, and there are many solutions that are highly similar, we sort the solutions with respect to the number of dams they build and sample 3000 solutions uniformly from each sub-experiment of the Expansion method. For the baseline methods, we select all of their solutions. For each method, we count how many grids of its solution set belong to the hypercube Pareto frontier. Table 3 shows that even for the smaller sub-basins like Marañón and Ucayali, for which the DP-based algorithm can compute the six-objective Pareto frontier with a 0.1 approximation factor, the expansion method can get an as good or better approximation of the Pareto frontier.

In conclusion, the Expansion method provides a good Pareto frontier

approximation for two smaller Amazon sub-basins. Furthermore, we expect that for the whole Amazon basin for six criteria, the expansion method will also provide a high-quality approximation of the Pareto frontier.

2.7 Visualizing the Pareto frontier

Visualizing the Pareto frontier on a two-dimensional space is straightforward when two objectives are considered. However, simultaneous consideration of three or more objectives adds considerable complexity to interpreting optimization outcomes, given the inherent limitations of human cognition in visualizing high-dimensional spaces. This complexity is further compounded by our methods having the power to generate the Pareto frontier for many objectives for a small ϵ, which translates into a very large number of Pareto optimal solutions. In order to support policy-maker in the decision-making process of hydropower planning and facilitate a better understanding of the optimization solutions, we developed an interactive visualization tool: **Amazon EcoVistas**, for showcasing and exploring the high-dimensional Pareto frontier.

Amazon EcoVistas provides three types of visualizations. The first one is an interactive map (A) showing the locations of dams contained in a given solution selected by clicking a specific point (yellow star) in the 2-D scatter plots (B-F). The 2-D scatter plots (B-F) show criterion-specific outcomes for each solution in the 6-D Pareto frontier, illustrating the tradeoffs for each pair of energy and environmental criteria when all criteria are considered. The magenta diamonds in B-F are solutions selected using the parallel coor-

dinate plot (shown as colored lines in G); the orange dots are the remaining unselected solutions in the 6-D Pareto frontier. In the parallel coordinate plot in G, each coordinate corresponds to the value of a criterion, and each zigzag line connecting different values on the coordinates corresponds to a single solution. Constraints can be added to each objective (shown as pink lines on the coordinates), and only the solutions that satisfy the constraints will be shown in color. We believe that the parallel coordinate plot will greatly aid decision makers in removing unwanted solutions and that the scatter plots will better illustrate the trade-offs between different objectives.

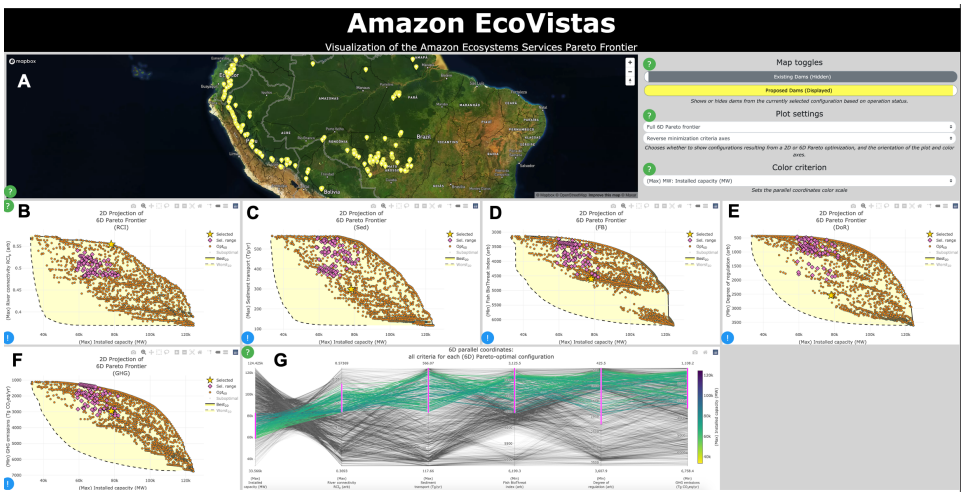


Figure 15: A Screenshot of the webpage of Amazon EcoVistas, (https://www.cs.cornell.edu/gomes/udiscoverit/amazon-ecovistas/), an interactive tool for visualizing Pareto frontiers.

2.8 Discussion

Motivated by the pressing real-world problem of hydropower dam placement in the Amazon basin, this work introduces novel methods for approximating the Pareto frontier for multi-objective optimization problems on tree-structured networks. We present two new methods, the DP-based method and the MIP-based method, that provide approximation guarantees and run efficiently for the whole Amazon basin for no more than four objectives. More specifically, the DP-based method is an FPTAS and can compute the exact Pareto frontier for two objectives for the whole Amazon basin. We also discussed how to incorporate DCI_P, a critical objective previously incompatible with the DP method, into the DP-based and MIP-based methods. Then, we address the issue that the runtime of the DP-based algorithm is still exponential in the number of objectives and introduce the Expansion method, which is shown to be able to produce good approximations of the Pareto frontier in practice. Lastly, we discuss how we should communicate our results with the decision-makers and present an interactive visualization tool: Amazon EcoVistas, for showcasing and exploring the high-dimensional Pareto frontier.

Going forward, we aim to further improve the speed and accuracy of the DP-based algorithm. We discover that the solutions generated by combining Pareto frontiers from child nodes have a lattice-like structure, which could be exploited to reduce the time needed for pruning sub-optimal solutions. Another direction we are considering right now is to produce more robust solutions such that even if a few dams in one solution cannot be built in the

end, we can still find a similar feasible solution.

We believe this research provides ecologists and decision-makers with efficient optimization tools for hydropower planning. We hope our work will lead to more awareness of the potential impacts of hydropower dams and promote strategic planning of hydropower dams on larger scales.

Chapter 3

Augmented online metric matching for applications in invasive species management

With the success of machine learning models for prediction, integrating such learned models into real-world systems has become a critical challenge. While the models are typically evaluated with aggregate statistics, e.g., accuracy on a test set, for many applications, such averages might not necessarily imply efficiency. For example, in combinatorial optimization, predicting most of the choices correctly might nonetheless lead to poor solutions. An emerging paradigm for integrating machine learning into optimization algorithms is to consider *learning augmented* algorithms [Lykouris and Vassilvitskii, 2018] while giving theoretical guarantees. We assume that we have a machine learning model that gives incomplete or partly incorrect predictions or just an oracle with incomplete knowledge. Then, given access to the model or oracle, we wish to construct an algorithm that provably performs well as long as we obtain enough correct information, but still gives guarantees for inaccurate predictions. This approach applies to many problems in computational sustainability where data is abundant yet noisy, and we still want to provide guarantees to ensure that resources are spent

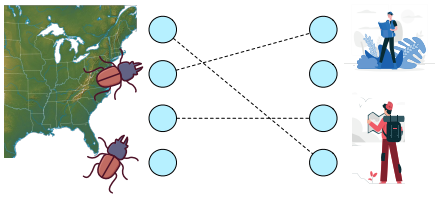


Figure 16: In invasive species management, we want to match observers to spatial locations in a landscape that might contain invasive species. Observers might be employees whose efforts can be planned or opportunistic citizen scientists whose engagement is unknown at the planning time.

efficiently [Dilkina and Gomes, 2010, Bondi et al., 2018].

In this chapter, we consider two matching problems subject to partial information: matching citizen scientists to observation locations for invasive species management and matching riders to taxis in urban mobility. In both settings, some parts of the matching might be known a priori, while other parts might be unknown. We model these problems as a semi-online matching problem, where we want to find the minimum cost matching in an online fashion given partial predictions of the graph, e.g., workers become available one by one and must irrevocably be matched to jobs as they become available.

A primary motivation for this project is coordinating the efforts of citizen scientists for invasive species management as part of an ongoing collaboration with the New York Natural Heritage Program, which contributes to the state invasive species database through the online mapping system iMap-Invasives [NatureServe, 2020 (Accessed 2020-07-01]. Citizen scientists is a broad term for engaging volunteer citizens in science projects [Bonney et al.,

2009, Cox et al., 2015], and data from citizen scientists and paid managers are combined in the database to help the state of New York assess and monitor the spread of over four hundred invasive species across the state. Coordinating the effort of opportunistic citizen scientists with predictably scheduled paid employees proves an important challenge in computational sustainability. A pervasive problem in these settings is that citizen scientists are opportunistic and possibly have misaligned incentives [Xue et al., 2013]. Like many similar projects [Sullivan et al., 2009, Follett and Strezov, 2015], the database uses citizen scientists to monitor invasive species, but the engagement of such citizen scientists is not known ahead of time. In addition to these volunteers, the database collaborates with paid employees whose schedules can be decided ahead of time. Another inspiration for this work is from a competition on using data from the NYC Taxi and Limousine Commission (TLC) and developing a machine learning model predicting the duration of taxi rides. Matching taxi drives to passengers and adjusting the matching between customers and taxis for model errors could also be modeled as an online-matching problem with partial information.

In this chapter, we model the problems shown above as learning-augmented min-cost matching problems and present an algorithm that provably improves upon pessimistic algorithms in the learning augmented setting, with an approximation bound that depends upon the amount of knowledge available. The majority of the work presented in this chapter has been reported in Bjorck et al. [2021].

3.1 Background

Min-Cost Matching. Consider a weighted bipartite graph $G = (V, E)$ with weights w_e for edge e. The two sides of the graph might represent, e.g., taxis and potential customers. We will refer to the two sides of the graphs as the jobs S and workers R. A perfect matching E is a subset $M \subseteq E$ such that each node is incident to exactly one edge in E. The weight of a matching E is $w(E) = \sum_{e \in E} w_e$. We will assume that there are n workers and jobs; a perfect matching is then of size n. In our applications, we primarily consider matching over spatial domains, e.g., matching taxis and customers; we thus assume that the graph G is *metric*. Let $d(i, j)$ denote the distance between nodes i, j, the metric property then implies

$$d(i, j) + d(j, k) \leq d(i, k) \qquad \forall i, j, k \in V \tag{3.1}$$

The min-cost perfect matching problems entail finding a matching M that minimizes $w(M)$, which can be done in polynomial time [Kuhn, 1955]. Solutions often rely on so-called *augmenting paths* P (augmenting w.r.t. some matching M), which are paths in G whose ending and starting nodes are unmatched in M such that every other edge $\in P$ is also $\in M$. Given such a path, one can expand the matching M by setting

$$M \leftarrow M \otimes P. \tag{3.2}$$

Here $\otimes$ is the symmetric difference between the two sets, returning elements found in exactly one of the operand sets.

Online Matching. Online algorithms model scenarios where parts of a problem are revealed one by one, but one needs to make irreversible choices before all information is revealed. Online algorithms are typically evaluated based on their *competitive ratio*, which measures the expected cost $\mathbb{E}[w(M)]$ of the obtained solution M and the optimal solution M_* that can be obtained if all information was known a priori. The competitive ratio c is defined as

$$c = \mathbb{E}\left[\frac{w(M)}{w(M_*)}\right] \tag{3.3}$$

Learning Augmented Metric Matching. As machine learning methods have become increasingly successful, researchers are now interested in the so-called *learning augmented* algorithm design, which utilizes incomplete or noisy advice from a machine learning model or other possible sources [Purohit et al., 2018, Lykouris and Vassilvitskii, 2018]. Here we assume the information comes from some oracle (which need not be a machine-learning model).

We will consider an online matching problem where parts of the network are known ahead of time. Our goal is to construct an algorithm that can use such side information and improve upon pessimistic online algorithms. We will assume that the workers arrive in random order and formally define our problem as

Definition 3. *The learning augmented metric matching problem consists of*

- *a metric bipartite graph G with jobs S and workers R. We assume $|R| = |S| = n$*

· *A set R_p of predicted workers containing $n - k$ workers known ahead of time.*

workers from R are revealed one by one in random order and must be irrevocably matched to a job upon arrival. An algorithm must output a perfect matching E.

3.2 Related Work

Matching problems are ubiquitous in sustainability applications. Examples include health interventions [Wilder et al., 2018], organ donor matching [Roth et al., 2004], fair division of goods [Aleksandrov et al., 2015], and supply-demand matching in energy storage [Pickard et al., 2009]. The idea of learning augmented algorithms goes back at least to [Lykouris and Vassilvitskii, 2018], which studies the problem in the context of machine-learned advice for caching policies. Other applications include frequency estimation in data streams [Hsu et al., 2018], low-rank estimation [Indyk et al., 2019], and scheduling [Lattanzi et al., 2020, Purohit et al., 2018]. Within matching, Kumar et al. [2018] introduces semi-supervised maximum matching, online matching where a certain subset of the nodes is known before. In this work, we consider min-cost perfect matching, which introduces additional complications as the externalities of a single bad choice can grow substantially. Pure online algorithms have a long history, see, e.g., [Karp et al., 1990], but is still an active area of research [Buchbinder et al., 2019, 2020, Devanur and Huang, 2017]. Both applications we consider here have extensive literature owing to their practical importance. In the context of invasive

species management, citizen science has proven to be a promising strategy for conservation work [Crall et al., 2015, Rutledge et al., 2013, McKinley et al., 2017]. On the computational side, considered methods include reinforcement learning [Taleghan et al., 2015], mixed integer programming solvers [Büyüktahtakın et al., 2014], stochastic dynamic programming, and other methods [Shea and Possingham, 2000]. From a theoretical perspective Bjorck et al. [2018] considers a predator-prey model for biocontrol, Gupta et al. [2018] uses Hawkes processes for modeling, and Spencer [2012] considers an extension of the firefighter problem. Within the domain of urban mobility [Lowalekar et al., 2018, Freund et al., 2019], various dimensions of the problem have been considered. Examples include pricing [Qiu et al., 2017], finding the minimum fleet size [Vazifeh et al., 2018], and allocating multiple passengers to the same ride [Santi et al., 2014]. Again there is a heterogeneous set of strategies considered, from traditional combinatorial optimization methods [Nair and Miller-Hooks, 2011] to reinforcement learning [Schultz and Sokolov, 2018].

3.3 An Algorithm for Learning Augmented Matching

We now present a method for solving problems of the type given in Definition 3 by incorporating machine learning advice into classical online algorithms [Raghvendra, 2016], a high-level illustration of the algorithm can be found in Figure 17. We will first present the base algorithm and then discuss how to improve it with local search.

Algorithm 6:

input : Graph $G = (E, N)$, jobs $S \subseteq N$, predicted workers $R_p \subseteq N$, a random permutation R.

output: A matching M

$M \leftarrow \{\}$ // current matching

$G_0 \leftarrow$ induced subgraph of G on $R_p \cup S$

$M_* \leftarrow$ MinMatch (G_0)

$M_f \leftarrow M_*$ // planned matching for predicted workers not revealed

$\phi \leftarrow w(M*)$

for $r \in R$ **do**

  if $r \in R_p$ **then**

    $M \leftarrow M \cup M_f(r)$

  else

    $P \leftarrow \text{aug}(r, M_*)$ minimizing $\Delta\phi$ 3.4

    $\phi \leftarrow \phi + \Delta\phi(P, M_*)$ via 3.4

    $M_* \leftarrow M_* \bigoplus P$

    $M \leftarrow M \cup (r, s)$ where (r, s) are endpoints of P

  end

end

Output M

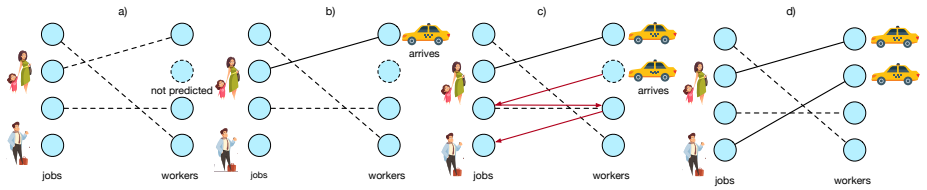


Figure 17: The high-level idea of our algorithm. a) For all predicted workers, we first calculate an optimal matching M_*. b) When a predicted worker arrives, we match it according to M_*. c) When an unpredicted worker arrives, we calculate an augmenting path relative to M_* and d) match the endpoints. We improve this general strategy by adding a local search procedure.

3.3.1 Algorithm

The algorithm will maintain a matching M, which it outputs at the end, and auxiliary matchings M_f and M_*, the latter of which is updated throughout the procedure. At a high level, our strategy is to first find a suitable set of jobs for the predicted nodes R_p and then update our matching M as workers are revealed one by one via augmenting paths with respect to M_*. Throughout the algorithm, we will keep track of the variable ϕ, which will be useful for the analysis. At the very start, the algorithm initializes M to the empty set and M_* to the min-cost matching from all predicted workers, using, e.g., the Hungarian method [Kuhn, 1955]. When predicted workers arrive, they are assigned as per M_f. When an adversarial node $r \in R_{\mathcal{A}}$ is revealed, we find an augmenting path P from r to an empty job s that minimizes $\Delta\phi$, defined as

$$\Delta\phi(P, M_*) = \sum_{e \in P \setminus M_*} w(e) \tag{3.4}$$

after this, we update $M_* \leftarrow M_* \bigoplus P$ and assign the worker to the endpoint of P. See Algorithm 6.

3.3.2 Formal Analysis

It is straightforward to verify that Algorithm 6 gives a perfect matching, the reason being that we always connect unpredicted workers with endpoints of paths that are augmenting w.r.t M_*. Since all predicted workers are matched in M_* at the start, such paths must terminate in unmatched jobs.

Proposition 9. *Algorithm 6 outputs a perfect matching.*

Proof. Let S_0 be the set of jobs matched in MinMatch (R_p). When a node $r \notin R_p$ is revealed at time t and successfully matched to a job s we update $S_t \leftarrow S_{t-1} \cup \{s\}$. We now claim that all jobs $s \in S_t$ are matched in M_* after the tth iteration. This statement holds at the start of the algorithm; furthermore, it holds inductively as 1) when a worker $r \in R_p$ is revealed, S_t and M_* are unchanged; 2) when a worker $r \notin R_p$ is revealed and matched to s, M_* is updated via an augmenting path P, after which all jobs in S_{t-1} will remain matched in M_*, and s will be matched in M_* as it is the endpoint of P.

When a worker $r \in R_p$ is revealed, we can show that job $s = M_f(r)$ is yet unmatched in M. Firstly, all revealed adversarial workers $r' \notin R_p$ are matched to jobs s' unmatched in M_*, as endpoints of augmenting paths cannot be matched in M_*, and thus $s' \notin S_t$. Secondly, for any job s matched

in M_f, $s \in S_t$ for all t since $s \in S_0$ and S_t grows monotonically. Now, since workers $r \in R_p$ are matched to jobs $s \in S_t$, and workers $r' \notin R_p$ are matched to jobs $s' \notin S_t$, the only way in which a worker $r' \notin R_p$ could not be matched to a job s' would be if another worker $r'' \notin R_p$ was already matched to s'. This condition, however, would imply that two augmenting paths P, P' would both end in s', which is a contradiction as augmenting paths must end in unmatched jobs. □

Of more interest are the approximation guarantees, which will depend upon k the number of unpredicted workers as follows.

Theorem 10. *Algorithm 6 has competitive ratio* $\mathcal{O}(1 + \log k)$.

As is common in the analysis of learning augmented algorithms [Purohit et al., 2018, Lykouris and Vassilvitskii, 2018], our algorithm will interpolate between known methods at the extremes of perfect or unavailable machine learning advice. The algorithm reduces to the Hungarian method or the online method of [Raghvendra, 2016] at such extremes, borrowing from the analysis of these methods but improving upon the $\mathcal{O}(\log n)$ guarantee the latter provides in the case of partial information. The algorithm can be analyzed by bounding the value of ϕ. We first give a lower bound of ϕ.

Proposition 10. *Let* M_t *be the matching at iteration* t. *For any* t *we then have*

$$w(M_t) + w(M_t^*) + w(M_f \setminus M_t) \leq 2\phi \tag{3.5}$$

Proof. We use proof by induction. The statement holds at $t = 0$ trivially. Assume that the statement holds for t. We now consider the case of $t + 1$

and the incoming node r. If $r \in R_p$, the RHS of 3.5 is unchanged as per Algorithm 6. r will be matched to $s = M_f(r)$, and thus $w(M_t)$ will increase by $w(e_{(r,s)})$ whereas $w(M_f \setminus M_t)$ will decrease by the same amount. The LHS of 3.5 can also be proved similarly. If $r \notin R_p$. We then have

$$w(M_i^*) - w(M_{i-1}^*) = \sum_{e \in P \setminus M_{i-1}^*} w(e) - \sum_{e \in P \cap M_{i-1}^*} w(e)$$

$$= \Delta\phi(P_i) - \left(\frac{1}{2} \sum_{e \in P \cap M_{i-1}^*} w(e) + \frac{1}{2} \sum_{e \in P \cap M_{i-1}^*} w(e) \right)$$

we add and subtract $\frac{1}{2} \sum_{e \in P \setminus M_{i-1}^*} w(e)$ in the parenthesis

$$= \Delta\phi(P_i) - \Bigg(\underbrace{\frac{1}{2} \sum_{e \in P \cap M_{i-1}^*} w(e) + \frac{1}{2} \sum_{e \in P \setminus M_{i-1}^*} w(e)}_{= \frac{1}{2}\ell(P_i)} +$$

$$\underbrace{\frac{1}{2} \sum_{e \in P \cap M_{i-1}^*} w(e) - \frac{1}{2} \sum_{e \in P \setminus M_{i-1}^*} w(e)}_{= -\frac{1}{2}(w(M_i^*) - w(M_{i-1}^*))} \Bigg).$$

Thus

$$\frac{1}{2}\big(w(M_i^*) - w(M_{i-1}^*)\big) = \Delta\phi(P_i) - \frac{1}{2}\ell(P_i). \tag{3.6}$$

Using the metric property we have $\ell(P) \geq d(r, s)$, and we note that $d(r, s)$ is $\Delta w(M)$. This gives us

$$\Delta w(M_*) + \Delta w(M) \leq 2\Delta\phi. \tag{3.7}$$

As in 9 we know that s is unmatched in M_f, and since M_f is unchanged $w(M_f \setminus M)$ is also unchanged. Thus the inductive statement still holds. $\square$

Next, we utilize the random permutation to give an upper bound to the expectation of ϕ. Here we use the fact that not all workers are adversarial to improve upon pessimistic adversarial bounds [Raghvendra, 2016].

Proposition 11. $\mathbb{E}[\phi] \leq (1 + H_k)w(M_{opt})$.

Proof. At $t = 0$, we have $\phi \leq w(M_{\text{opt}})$. We thus only need to bound the change $\Delta\phi$ from when we add $r \notin R_p$. Let us consider the augmenting paths from M_i^* to M_{opt}. Since M_{opt} is a perfect matching, there are $n - i$ vertex disjoint augmenting paths – one for each node that has not yet arrived. We let P'_j be the set of these augmenting paths. Then,

$$\sum_{j=1}^{n-i} \Delta\phi(P'_j) = \sum_{j=1}^{n-i} \Bigg(\sum_{(s,r \in P'_j \notin M_i^*)} d(s,r) \Bigg) \tag{3.8}$$

$$= \sum_{j=1}^{n-i} \Bigg(\sum_{(s,r \in P'_j \cap M_{\text{opt}})} d(s,r) \Bigg) \leq w(M_{\text{opt}})$$

Here we use the fact that the paths are vertex disjoint. We now relabel the workers such that worker r_i is r'_j. Since we always chose the path with the smallest $\Delta\phi$, we have

$$\Delta\phi(P_i) \leq \Delta\phi(P'_j)$$

In the random arrival model, all adversarial nodes are equally likely to arrive at any time. Let us assume that k' adversarial nodes have already arrived. Using equation 3.8 we then have

$$\mathbb{E}[\Delta\phi(P_i)|r_i \in \mathcal{A}] \leq \frac{1}{k - k'} \sum_{q \in \mathcal{A}} \Delta\phi(P_q)$$

$$\leq \frac{1}{k-k'} \sum_q \Delta\phi(P_q) \leq \frac{w(M_{\text{opt}})}{k-k'}$$

In total, we then have

$$\mathbb{E}[\sum_q \Delta\phi(P_q)] = \sum_q \mathbb{E}[\mathbf{1}_{r_q \in \mathcal{A}}]\mathbb{E}[\Delta\phi(P_i)|r_i \notin R_p\mathcal{A}]] \tag{3.9}$$

$$\leq w(M_{\text{opt}}) \sum_{k'=1}^{k} \frac{1}{k'}$$

adding (3.9) to the fact that $\phi \leq w(M_{\text{opt}})$ at $t = 0$ yields the claim. □

Proof of Theorem 10. Combining Proposition 10 for $t = n$ and Proposition 11 gives us

$$\frac{1}{2}\mathbb{E}[w(M)] \leq \frac{1}{2}\mathbb{E}\big[\big(w(M) + w(M_f \setminus M) + w(M_*)\big)\big]$$

$$\leq \mathbb{E}[\phi] \leq (1 + H_k)w(M_{\text{opt}}).$$

inspecting the ends of this inequality yields the result.

□

3.3.3 Improvement via Local Search

It is possible to perform a local search before assigning a worker to a job to improve the solution further. When an adversarial worker r is revealed, we find a minimum weight augmenting path P from r to empty job s and use P to update M_*. Then, we define N_f to be the set of nodes that are matched in M_f but not matched in M, and H to be a subgraph of G

induced on $N_f \cup \{r, s\}$. We observe that if we update M_f and assign r based on the minimum weight matching on H, we can reduce the weight of the final matching without disturbing the overall structure of the algorithm. See Algorithm 7 for a complete description of the local search routine. If we replace line 13 in Algorithm 6 with the local search routine, we can show that M_* will remain the same in every iteration. It is straightforward to prove that this procedure can only improve Algorithm 6.

Algorithm 7: LocalSearch(r, s)

1 $N_f \leftarrow$ nodes that are matched in M_f but not matched in M

2 $H \leftarrow$ induced subgraph of G on $N_f \cup \{r, s\}$.

3 $M_f \leftarrow$ MinMatch(H)

4 $M \leftarrow M \cup M_f(r)$

Proposition 12. *The Local Improvement Algorithm always outputs a perfect matching at least as good as Algorithm 6.*

Lemma 1. *Given the same input arguments, the matching M^* in the Local Improvement Algorithm remains the same as the M^* in Algorithm 6 in every iteration.*

Proof. The lemma holds initially. In each iteration, if r is predicted, M^* remains unchanged; if r is unpredicted, under the condition that M^* in both algorithms are the same after the previous iteration, the minimum augmenting path P and subsequently M^* should also be the same in both algorithms. By induction, we prove that M^* is the same across both algorithms in every iteration. □

Lemma 2. *After every iteration, the set of jobs matched in either M_f or M is the same as the set of jobs matched in M^*. Also, if a node is matched in both M and M_f, it should be matched to the same node in both matchings. M and M_f are always valid matchings.*

Proof. The lemma is true at the start. Assume that the lemma holds after the previous iteration. If r is predicted, both sets are unchanged, and r is matched to the same node in M_f and M, so the lemma still holds. If r is unpredicted, s should be unmatched in M^* because it is the endpoint of an augmenting path of M^*, and therefore should also be unmatched in M_f and M based on our induction assumption. After updating M^* with P, s is added to the set of jobs matched in M^*. Next, N_f should have the same number of jobs and workers since the overlapping nodes of M and M_f have a perfect matching. Note that r is unpredicted, so r and s are not in M_f, M or N_f. The induced subgraph H on $N_f \cup \{r, s\}$ should then have the same number of jobs and workers, and all of them are unmatched in M. Thus, the new M_f is a perfect matching on H where s and all jobs in N_f are matched. Note that all jobs removed from M_f are already matched in M, so the set of jobs matched in either M_f or M is still the same as the set of jobs matched in M^*. Finally, we update M to be $M \cup M_f(r)$. Both nodes from in $M_f(r)$ are unmatched in M before the step, so the new M is still a matching. Also, nodes in $M_f(r)$ are the only overlapping nodes between M_f and M, so the second part of the lemma still holds. By induction, we prove that the lemma holds after every iteration. □

Proof of Proposition 12. By Lemma 2, we know that the output M is a

matching. Since all workers are matched in M, and we have the same number of workers and jobs, we know that M is a perfect matching. Also, by Lemma 2, we know that $M \cup M_f$ and $M_f \setminus M$ are valid matching. We consider the value of $w(M \cup M_f)$ in both algorithms, which in the end will be the weight of the output matching. In the beginning, the two values are equal. In every iteration, if r is predicted, the value is unchanged in both algorithms; if r is unpredicted, $w(M \cup M_f)$ is increased by $w(r, s)$ in Algorithm 6. In the Local Improvement Algorithm, before we update M_f, we know that

$$\begin{aligned} w(M \cup M_f) + w(r, s) &= w(M) + w(M_f \setminus M) + w(r, s) \\ &\geq w(M) + w(\text{MinMatch}(H)). \end{aligned}$$

Thus, the increase of $w(M \cup M_f)$ is less than or equal to $w(r, s)$ after the iteration. Hence, the increase of $w(M \cup M_f)$ in the Local Improvement Algorithm is less than or equal to that of Algorithm 6 in every iteration. In conclusion, the Local Improvement Algorithm always outputs a perfect matching at least as good as Algorithm 6. □

3.4 Experimental Results

To evaluate our algorithms, we perform experiments using data from two real-world datasets from invasive species management and taxi-cab matching. We consider the problem of finding the minimum cost perfect matching in the setting of Definition 3. We compare our algorithms with the following baselines:

method	2/21	2/22	2/23	2/24	2/25	2/26	2/27	3/21	3/22	3/23	3/24	3/25	3/26	3/27
opt	132	179	87	129	92	143	256	249	110	77	192	137	117	118
bipartite 0.5	163	235	133	176	131	196	337	319	164	104	253	218	180	178
bipartite 1.0	192	271	142	195	143	222	401	381	180	107	304	228	198	190
bipartite 2.0	257	354	179	242	193	279	520	501	215	137	387	301	245	238
online	156	227	111	170	130	191	333	294	143	94	235	201	174	181
hybrid	156	227	112	171	129	178	333	294	143	94	234	201	172	181
greedy	158	238	136	177	135	204	353	299	173	104	241	210	177	181
local	**154**	**203**	**99**	**156**	**118**	**177**	**282**	**262**	**133**	**90**	**218**	**180**	**167**	**154**
adversarial	158	238	137	177	135	204	352	300	176	104	242	212	181	183
method	4/21	4/22	4/23	4/24	4/25	4/26	4/27	5/21	5/22	5/23	5/24	5/25	5/26	5/27
opt	89	168	106	160	184	135	115	260	161	134	116	184	196	149
bipartite 0.5	137	236	138	228	256	206	160	339	230	187	176	259	257	204
bipartite 1.0	142	277	159	250	277	215	180	380	266	200	184	292	291	223
bipartite 2.0	185	363	200	328	341	260	227	476	332	255	257	370	379	267
online	130	228	130	221	237	186	156	315	229	163	166	244	235	191
hybrid	130	228	129	217	237	186	154	315	227	163	167	242	235	191
greedy	140	241	133	222	239	210	160	322	219	190	178	243	242	199
local	**116**	**200**	**124**	**201**	**230**	**152**	**144**	**301**	**201**	**151**	**154**	**232**	**220**	**173**
adversarial	142	242	134	225	239	211	161	323	224	189	179	246	243	202

Table 4: Total distance (km) traveled for various methods on the taxi matching problem for dates (given as month/day) in 2016. Less is better. We see that our local-search method consistently outperforms alternatives.

- A greedy algorithm that myopically assigns a worker r to the closest unmatched job s. We refer to it as greedy.

- A naive matching algorithm that first predicts the min-cost matching of the predicted workers and reserves the workers not matched to this set to be greedily matched to unpredicted workers. This comes with no guarantees, and we refer to it as hybrid.

- The semi-supervised matching algorithm of [Kumar et al., 2018], which reserves a set of jobs for unpredicted workers with low "externality". This algorithm was developed for unweighted maximum matching; we adopt it by considering edges present if their distance is less than some cutoff proportional to the average distance per edge in the optimal solution. For constant of proportionality p, we refer to these as bipartite-p. Nodes that cannot be matched with edges below the threshold are matched greedily.

- The online algorithm of [Raghvendra, 2016] which gives pessimistic performance guarantees and does not use any additional information, and we refer to it as adversarial.

We will refer to our algorithm, with and without the local search routing, as online and local, respectively.

3.4.1 Taxi-Cab Matching

We first conduct experiments on matching taxi drivers to customers. Traffic conditions and other issues are typically hard to predict. To this end, NYC

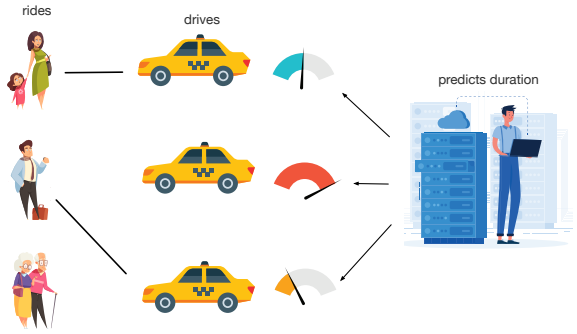


Figure 18: In urban mobility, matching customers to drivers is challenging as, e.g., changing traffic conditions make it hard to estimate the travel time. We use a machine learning model to predict the trip duration and evaluate our algorithm's ability to generate matchings when facing prediction errors.

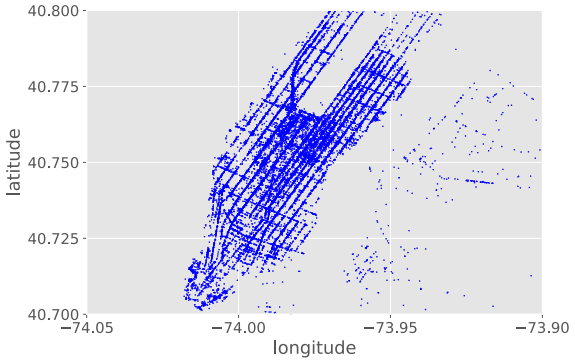


Figure 19: The area for the taxi dataset, each dot corresponds to a ride. Note the outline of Manhattan and Central park.

method	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
bipartite 0.5	4777	4896	5310	4587	3723	4783	5645	6897	5873	4312	6723	3246	4003	5410
bipartite 1.0	5346	5494	5519	4880	3944	4925	5964	7103	6224	4560	6995	3360	4165	5812
bipartite 2.0	5444	5685	6132	5466	4837	6158	7395	8806	7209	5455	8003	4601	5109	6478
online	4316	4899	5005	4305	**3094**	4469	5042	6665	5006	3836	5967	2821	3897	4689
hybrid	4316	4899	4963	4268	**3094**	4469	5042	6665	4898	3814	5967	2851	3753	4690
greedy	4792	4835	5075	4387	3616	4689	5719	6734	5738	4349	6543	3441	4027	5314
local	**4085**	**4583**	**4706**	**4000**	**3094**	**4250**	**4692**	**6204**	**4881**	**3497**	**5695**	**2700**	**3229**	**4417**
adversarial	5079	5006	5202	4441	3801	4737	5758	7144	5966	4352	6462	3550	3944	5952
opt	4083	4443	4701	3973	3092	4158	4659	6145	4862	3415	5635	2575	3079	4399

method	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
bipartite 0.5	4272	4656	3324	4550	4214	4439	3978	5675	1705	**1971**	1983	1856	2043	1921
bipartite 1.0	4309	5152	3840	5124	4278	4780	4249	6155	1798	2310	2206	1933	2171	2071
bipartite 2.0	5023	5825	4402	5622	5553	6003	5250	8199	2594	2587	2831	2556	2372	2438
online	3531	4624	3108	3883	3797	4861	4261	5897	1898	2129	2224	1849	1842	1836
hybrid	3530	4652	3063	3885	3903	4889	4468	6107	1915	2155	2249	2044	1970	1899
greedy	3990	4493	3394	4478	4224	4342	3920	**5415**	1746	2029	2174	1855	1779	1902
local	**3363**	**4061**	**2819**	**3659**	**3556**	**3999**	3658	5468	**1703**	1982	**1935**	**1699**	**1726**	**1759**
adversarial	4211	4544	3335	4619	4280	4433	**3654**	5779	1906	2173	2062	2025	1999	1968
opt	3278	3723	2602	3617	3272	3552	3066	4819	1362	1490	1477	1372	1413	1389

Table 5: Total distance (km) traveled for various methods on the invasive species management problem by year. Less is better. We see that our method local typically outperforms alternatives, only being beaten by a small margin for three years.

Taxi and Limousine Commission (TLC) released a dataset of roughly two million taxi rides in the greater NYC area and organized a competition to predict the duration of taxi rides [TLC, 2016]. The dataset contains the location and time of pickup/dropoff of customers and various other data (e.g., number of passengers, taxi company), but not the time of the cab order. See Figure 19 for an illustration over the geographic area. We consider the problem of matching taxis (**workers**) to riders (**jobs**). To construct the

graph, we consider all n taxi cabs that become available in some time interval $[T, T + t_d]$ (i.e., they drop off a customer in this interval) and try to match them to the next n customers. We aim to minimize the Manhattan distances from the initial dropoff to the next pickup, and the edge weights correspond to this quantity. In the spirit of the original competition, we construct a machine learning model that predicts the duration of taxi rides. (The machine learning model is implemented using random forests via XGBoost [Chen and Guestrin, 2016]. Features used in the prediction model include distance, pickup and dropoff area, and the time of the day.) Taxis that are predicted to be free for new customers after $T + t_d$, but arrive before the time point due to beneficial traffic conditions, miscalibrated prediction, or other factors are treated as unpredicted workers. The other taxis that arrive as predicted are treated as predicted workers. In practical applications, one might also want to optimize for waiting time. As such information is not available for this dataset, we defer such studies to future work. We consider seven days for each of four months in 2016, taking $T = 3$ pm and t_d as 30 minutes. Results are given in Table 4, where we see that our method with local search consistently gives the best matchings.

3.4.2 Invasive Species Management

We now consider the problem of invasive species management as part of a collaboration with the New York Natural Heritage Program. The dataset upon which the experiments are conducted comes from the iMapInvasives project [NatureServe, 2020 (Accessed 2020-07-01]. Its database currently consists of more than 200,000 observations and 408 species, spread over

more than 30 years and 2,200 observers. Of central importance is the heterogeneous agents collaborating on this initiative; the state is divided into eight regions administered by individual organizations that make use of both employees and individual citizen scientists. For each year, we will aim to match volunteers/employees (**workers**) to sites deemed necessary for investigation (**jobs**). For this task, we do not assume that the side advice comes from a machine learning model, but instead that volunteers are unpredicted due to their opportunistic engagement, whereas employees are known ahead of time. Observations are typically strongly correlated at the small scale, representing observations conducted the same day a couple of meters apart. We divide the landscape into patches corresponding to 0.2 degrees latitudes/longitudes and subsample the set of observations so that each patch has at most one observation per year, approximately corresponding to a day's work. Then, for each year, we construct the matching graph where the jobs correspond to the observations conducted that year, and the workers correspond to the observers. As we have not been able to obtain employment data, observers that only conduct one observation for a given year and have no previous observations are treated as citizen scientists (modeled as unpredicted workers), others as employees (predicted workers). We weight an edge by the distance from the centroid of the employee's/citizen scientist's observations to the observation location. The results are shown in Table 5, where we see that our method outperforms the alternatives. We also consider an ablation experiment to evaluate the impact of the number of predicted observers. Instead of treating the observers as unknown based on historical data, we randomly pick the observer to be predicted or not.

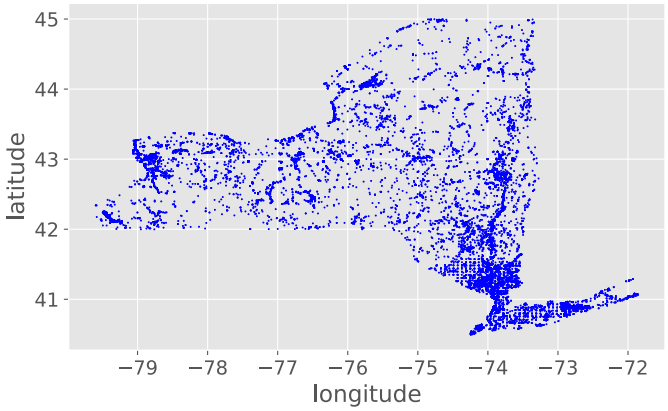


Figure 20: The geographical area that the invasive species dataset covers, each dot corresponds to an observation, on the ground, of an invasive species. Data are collected over 31 years.

We then plot the average distance traveled, averaged over all years and five repetitions (with five fixed seeds), as a function of the fraction of known nodes. The results are given in Figure 21, with performance improving as with predicted observers.

3.5 Discussion

Motivated by problems in computational sustainability, this work introduces a novel learning augmented method for matching problems with partially unknown nodes. The proposed algorithm interpolates between a completely known setting and an adversarial online setting, and we provide a theoretical bound that improves with accurate predictions. We evaluate our method on two large-scale datasets covering urban mobility and invasive species management and find that our method consistently outperforms alternatives.

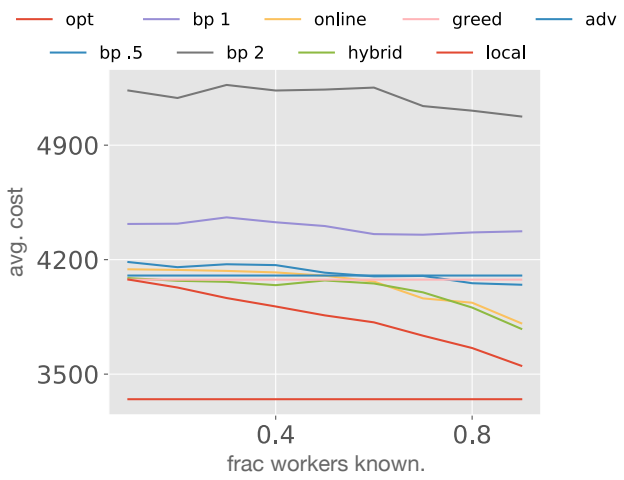


Figure 21: Ablation experiments showing the performance on the invasive species management matching problem of various algorithms, as the fraction of predicted nodes changes. Solution quality is given by the average cost of matchings, measured in kilometers. Our proposed methods improve upon the alternatives and improve as the number of predicted nodes grows.

We believe this research direction is broadly applicable to problems in computational sustainability and hope that it can inspire future work outside of our two applications.

Chapter 4

Conclusions

The main goal of this research is to tackle real-world sustainability problems that are not only impactful to various environmental, social, and economic aspects of the world but also difficult and intriguing from a computational perspective. The works presented in this thesis are all in alignment with this goal.

In Chapter 2, we perform an in-depth study of the Amazon Dam Placement Problem, which has a significant impact on the lives of the local people and the ecosystem of the whole planet due to the importance of the Amazon Basin. The rapid proliferation of hydropower dams in the Amazon basin will heavily impact many ecosystem services provided by river networks, such as energy production, navigation, biodiversity, sediment and nutrient production, and fresh-water fisheries. Therefore, it is imperative to integrate ecosystem service trade-offs into decision-making models for the placement of hydropower dams in the Amazon Basin. To this end, we model the hydropower dam placement problem as a multiobjective optimization problem on a tree-structured network and identify a set of **six objectives** that characterizes the major environmental and socio-economical impacts of the

dams. This multiobjective optimization problem is NP-hard, and no existing method provides runtime and approximation guarantees. To solve this issue, we develop a DP-based algorithm that exploits the tree structure of the problem. The DP-based algorithm provides theoretical guarantees and runs very efficiently for no more than four objectives for the whole Amazon basin. The FPTAS DP-based algorithm is now considered a state-of-the-art algorithm in the field of hydropower planning. We also presented a complementary MIP-based method that provides more flexibility when considering additional constraints and can be used to search in a small specific solution space. Then, we address the issue that the runtime of the DP-based algorithm is still exponential in the number of objectives and introduce the Expansion method, which is shown to produce good approximations of the Pareto frontier in practice. Lastly, we discuss how we should communicate our results with the decision-makers and present an interactive visualization tool: Amazon EcoVistas, for showcasing and exploring the high-dimensional Pareto frontier.

In Chapter 3, we study the problem of coordinating citizen scientists for invasive species planning as part of an ongoing collaboration with the New York Natural Heritage Program, which contributes to the state invasive species database through the online mapping system iMapInvasives. Data from citizen scientists and paid managers are combined in the database to help the state of New York assess and monitor the spread of over four hundred invasive species across the state. We formulate the problem as an online matching problem with partial information, where we want to find the minimum cost matching in an online fashion given partial predictions of

the graph, e.g., citizen scientists that become available one by one and must irrevocably be matched to survey locations as they show up. We develop an algorithm that fully utilizes prior knowledge and provably improves upon pessimistic algorithms in the learning augmented setting, with an approximation bound that depends upon the amount of knowledge available. Our algorithm consistently outperforms baselines. We also prove approximation guarantees for the algorithm, which depends upon how much knowledge we have access to.

Personally, it has been a great honor for me to be able to collaborate with passionate and knowledgeable researchers from several fields and contribute to real-world applications with direct social impact while at the same time developing formal models and algorithms that add to the innovation of computer science. Going forward, the problems and methods presented in this thesis will surely provide a foundation for continued research in more combinatorial optimization problems in computational sustainability.

Bibliography

Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: Analysing a food bank problem. *arXiv preprint arXiv:1502.07571*, 2015.

Rafael M Almeida, Qinru Shi, Jonathan M Gomes-Selman, Xiaojian Wu, Yexiang Xue, Hector Angarita, Nathan Barros, Bruce R Forsberg, Roosevelt García-Villacorta, Stephen K Hamilton, et al. Reducing greenhouse gas emissions of amazon hydropower with strategic dam planning. *Nature Communications*, 10(1):1–9, 2019.

Ronaldo Barthem, Michael Goulding, Rosseval Leite, Carlos Canas, Bruce Forsberg, Eduardo Venticinque, Paulo Petry, Mauro Ribeiro, Junior Chuctaya, and Armando Mercado. Goliath catfish spawning in the far western amazon confirmed by the distribution of mature adults, drifting larvae and migrating juveniles. *Scientific Reports*, 7, 02 2017. doi: 10.1038/srep41784.

Johan Bjorck, Yiwei Bai, Xiaojian Wu, Yexiang Xue, Mark Whitmore, and Carla Gomes. Scalable relaxations of sparse packing constraints: Optimal biocontrol in predator-prey networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Johan Bjorck, Qinru Shi, Carrie Brown-Lima, Jennifer Dean, Angela Fuller, and Carla Gomes. Learning augmented methods for matching: Improving invasive species management and urban mobility. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):14702–14710, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17727`.

Elizabeth Bondi, Fei Fang, Mark Hamilton, Debarun Kar, Donnabell Dmello, Jongmoo Choi, Robert Hannaford, Arvind Iyer, Lucas Joppa, Milind Tambe, et al. Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. In *AAAI*, pages 7741–7746, 2018.

Rick Bonney, Caren B Cooper, Janis Dickinson, Steve Kelling, Tina Phillips, Kenneth V Rosenberg, and Jennifer Shirk. Citizen science: a developing tool for expanding science knowledge and scientific literacy. *BioScience*, 59(11):977–984, 2009.

Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, 81(5): 1781–1799, 2019.

Niv Buchbinder, Moran Feldman, Yuval Filmus, and Mohit Garg. Online submodular maximization: Beating 1/2 made simple. *Mathematical Programming*, pages 1–21, 2020.

İ Esra Büyüktahtakın, Zhuo Feng, and Ferenc Szidarovszky. A multi-objective optimization approach for invasive species control. *Journal of the Operational Research Society*, 65(11):1625–1635, 2014.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

David Cote, Dan G Kehler, Christina Bourne, and Yolanda F Wiersma. A new measure of longitudinal connectivity for stream networks. *Landscape Ecology*, 24(1):101–113, 2009.

Joe Cox, Eun Young Oh, Brooke Simmons, Chris Lintott, Karen Masters, Anita Greenhill, Gary Graham, and Kate Holmes. Defining and measuring success in online citizen science: A case study of zooniverse projects. *Computing in Science & Engineering*, 17(4):28–41, 2015.

Alycia W Crall, Catherine S Jarnevich, Nicholas E Young, Brendon J Panke, Mark Renz, and Thomas J Stohlgren. Citizen science contributes to our knowledge of invasive plant species distributions. *Biological invasions*, 17 (8):2415–2427, 2015.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Nikhil R Devanur and Zhiyi Huang. Primal dual gives almost optimal energy-efficient online algorithms. *ACM Transactions on Algorithms (TALG)*, 14(1):1–30, 2017.

Bistra Dilkina and Carla P Gomes. Solving connected subgraph problems in wildlife conservation. In *International Conference on Integration of*

Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, pages 102–116. Springer, 2010.

Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *Or Spectrum*, 22(4): 425–460, 2000.

Matt Finer and Clinton N Jenkins. Proliferation of hydroelectric dams in the Andean Amazon and implications for Andes-Amazon connectivity. *Plos one*, 7(4):e35126, 2012.

Alexander S. Flecker, Qinru Shi, Rafael M. Almeida, Héctor Angarita, Jonathan M. Gomes-Selman, Roosevelt García-Villacorta, Suresh A. Sethi, Steven A. Thomas, N. LeRoy Poff, Bruce R. Forsberg, Sebastian A. Heilpern, Stephen K. Hamilton, Jorge D. Abad, Elizabeth P. Anderson, Nathan Barros, Isabel Carolina Bernal, Richard Bernstein, Carlos M. Cañas, Olivier Dangles, Andrea C. Encalada, Ayan S. Fleischmann, Michael Goulding, Jonathan Higgins, Céline Jézéquel, Erin I. Larson, Peter B. McIntyre, John M. Melack, Mariana Montoya, Thierry Oberdorff, Rodrigo Paiva, Guillaume Perez, Brendan H. Rappazzo, Scott Steinschneider, Sandra Torres, Mariana Varese, M. Todd Walter, Xiaojian Wu, Yexiang Xue, Xavier E. Zapata-Ríos, and Carla P. Gomes. Reducing adverse impacts of amazon hydropower expansion. *Science*, 375(6582):753–760, 2022. doi: 10.1126/science.abj4017. URL `https://www.science.org/doi/abs/10.1126/science.abj4017`.

Ria Follett and Vladimir Strezov. An analysis of citizen science based research: usage and publication patterns. *PloS one*, 10(11):e0143687, 2015.

Daniel Freund, Shane G Henderson, Eoin O'Mahony, and David B Shmoys. Analytics and bikes: Riding tandem with motivate to improve mobility. *INFORMS Journal on Applied Analytics*, 49(5):310–323, 2019.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

Carla P Gomes. Computational sustainability: Computational methods for a sustainable environment, economy, and society. *The Bridge*, 39(4):5–13, 2009.

Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1):67–100, Feb 2000. ISSN 1573-0670. doi: 10.1023/A:1006314320276. URL `https://doi.org/10.1023/A:1006314320276`.

Jonathan Gomes Selman, Qinru Shi, Yexiang Xue, Roosevelt Garcia-Villacorta, Alexander Flecker, and Carla Gomes. *Boosting Efficiency for Computing the Pareto Frontier on Tree Structured Networks*, pages 263–279. 06 2018.

Amrita Gupta, Mehrdad Farajtabar, Bistra Dilkina, and Hongyuan Zha. Discrete interventions in hawkes processes with applications in invasive species management. In *IJCAI*, pages 3385–3392, 2018.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based

frequency estimation algorithms. In *International Conference on Learning Representations*, 2018.

Dong Huang, Zhang Yi, and Xiaorong Pu. Manifold-based learning and synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):592–606, 2009. doi: 10.1109/TSMCB.2008.2007499.

Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In *Advances in Neural Information Processing Systems*, pages 7402–7412, 2019.

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

Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

Michael Kuby, William Fagan, Charles ReVelle, and William Graf. A multiobjective optimization model for dam removal: An example trading off salmon passage with hydropower and water storage in the willamette basin. *Faculty Publications*, 28, 08 2005. doi: 10.1016/j.advwatres.2004.12.015.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. Semi-online bipartite matching. *arXiv preprint arXiv:1812.00134*, 2018.

Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM, 2020.

Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48 (1), sep 2015. ISSN 0360-0300. doi: 10.1145/2792984. URL `https://doi.org/10.1145/2792984`.

Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, 261:71–112, 2018.

MC Lucas and E Baras. Migration of freshwater fishes blackwell science oxford 420 google scholar. 2001.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *arXiv preprint arXiv:1802.05399*, 2018.

Marcus Märtens and Dario Izzo. The asynchronous island model and nsga-ii: Study of a new migration operator and its performance. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, page 1173–1180, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319638. doi: 10.1145/2463372.2463516. URL `https://doi.org/10.1145/2463372.2463516`.

Duncan C McKinley, Abe J Miller-Rushing, Heidi L Ballard, Rick Bonney,

Hutch Brown, Susan C Cook-Patton, Daniel M Evans, Rebecca A French, Julia K Parrish, Tina B Phillips, et al. Citizen science can improve conservation science, natural resource management, and environmental protection. *Biological Conservation*, 208:15–28, 2017.

Rahul Nair and Elise Miller-Hooks. Fleet management for vehicle sharing operations. *Transportation Science*, 45(4):524–540, 2011.

NatureServe. *iMapInvasives: NatureServe's online data system supporting strategic invasive species management*, 2020 (Accessed 2020-07-01). URL `http://www.imapinvasives.org`.

Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620 – 1629, 2007.

C. H. Papadimitriou and M. Yannakakis. 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 '00, 2000a.

Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st annual symposium on foundations of computer science*, pages 86–92. IEEE, 2000b.

William F Pickard, Amy Q Shen, and Nicholas J Hansing. Parking the power: Strategies and physical limitations for bulk energy storage in supply–demand matching on a grid whose input power is provided by

intermittent sources. *Renewable and Sustainable Energy Reviews*, 13(8): 1934–1945, 2009.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pages 9661–9670, 2018.

Chao Qian, Yang Yu, and Zhi-Hua Zhou. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 204 (Supplement C):99 – 119, 2013.

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Pareto ensemble pruning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2935–2941, 2015.

Chao Qian, Ke Tang, and Zhi-Hua Zhou. Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 835–846. Springer, 2016.

Han Qiu et al. *Dynamic pricing in shared mobility on demand service and its social impacts*. PhD thesis, Massachusetts Institute of Technology, 2017.

Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Kidney exchange. *The Quarterly journal of economics*, 119(2):457–488, 2004.

Samuel G. Roy, Emi Uchida, Simone P. de Souza, Ben Blachly, Emma Fox, Kevin Gardner, Arthur J. Gold, Jessica Jansujwicz, Sharon Klein, Bridie McGreavy, Weiwei Mo, Sean M. C. Smith, Emily Vogler, Karen Wilson, Joseph Zydlewski, and David Hart. A multiscale approach to balance trade-offs among dam infrastructure, river restoration, and cost. *Proceedings of the National Academy of Sciences*, 115(47):12069–12074, 2018. doi: 10.1073/pnas.1807437115. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1807437115`.

Claire Rutledge, Melissa Fierke, Philip Careless, and Thomas Worthley. First detection of agrilus planipennis in connecticut made by monitoring cerceris fumipennis (crabronidae) colonies. *Journal of Hymenoptera Research*, 32:75, 2013.

Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

Laura Schultz and Vadim Sokolov. Deep reinforcement learning for dynamic urban transportation problems. *arXiv preprint arXiv:1806.05310*, 2018.

K Shea and Hugh Philip Possingham. Optimal release strategies for biological control agents: an application of stochastic dynamic programming to population management. *Journal of Applied ecology*, 37(1):77–86, 2000.

Wanxing Sheng, Yongmei Liu, Xiaoli Meng, and Tianshu Zhang. An improved strength Pareto evolutionary algorithm 2 with application to the

optimization of distributed generations. *Computers and Mathematics with Applications*, 64(5):944 – 955, 2012.

Qinru Shi, Jonathan M. Gomes-Selman, Roosevelt García-Villacorta, Suresh Sethi, Alexander S. Flecker, and Carla P. Gomes. Efficiently optimizing for dendritic connectivity on tree-structured networks in a multi-objective framework. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, COMPASS '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450358163. doi: 10.1145/3209811.3209878. URL `https://doi.org/10.1145/3209811.3209878`.

Gwen Spencer. Robust cuts over time: Combatting the spread of invasive species with unreliable biological control. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Brian L Sullivan, Christopher L Wood, Marshall J Iliff, Rick E Bonney, Daniel Fink, and Steve Kelling. ebird: A citizen-based bird observation network in the biological sciences. *Biological conservation*, 142(10):2282–2292, 2009.

Majid Alkaee Taleghan, Thomas G Dietterich, Mark Crowley, Kim Hall, and H Jo Albers. Pac optimal mdp planning with application to invasive species management. *The Journal of Machine Learning Research*, 16(1): 3877–3903, 2015.

Miguel Terra-Neves, Inês Lynce, and Vasco Manquinho. Introducing Pareto

minimal correction subsets. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 195–211. Springer, 2017.

TLC. *New York City Taxi Trip Duration*, 2016. URL `www.kaggle.com/c/nyc-taxi-trip-duration/data`.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

Mohammed M Vazifeh, Paolo Santi, Giovanni Resta, Steven H Strogatz, and Carlo Ratti. Addressing the minimum fleet problem in on-demand urban mobility. *Nature*, 557(7706):534–538, 2018.

Toby Walsh. Search in a small world. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1172–1177, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1624312.1624385`.

Margaret M Wiecek, Matthias Ehrgott, Georges Fadel, and José Rui Figueira. Multiple criteria decision making for engineering, 2008.

Bryan Wilder, Han-Ching Ou, Kayla de la Haye, and Milind Tambe. Optimizing network structure for preventative health. In *AAMAS*, pages 841–849, 2018.

David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011. ISBN 978-0-521-19527-0.

K. O. Winemiller, P. B. McIntyre, L. Castello, E. Fluet-Chouinard, T. Giarrizzo, S. Nam, I. G. Baird, W. Darwall, N. K. Lujan, I. Harrison, M. L. J. Stiassny, R. A. M. Silvano, D. B. Fitzgerald, F. M. Pelicice, A. A. Agostinho, L. C. Gomes, J. S. Albert, E. Baran, M. Petrere, C. Zarfl, M. Mulligan, J. P. Sullivan, C. C. Arantes, L. M. Sousa, A. A. Koning, D. J. Hoeinghaus, M. Sabaj, J. G. Lundberg, J. Armbruster, M. L. Thieme, P. Petry, J. Zuanon, G. Torrente Vilara, J. Snoeks, C. Ou, W. Rainboth, C. S. Pavanelli, A. Akama, A. van Soesbergen, and L. Sáenz. Balancing hydropower and biodiversity in the amazon, congo, and mekong. *Science*, 351(6269):128–129, 2016a. doi: 10.1126/science.aac7082. URL `https://www.science.org/doi/abs/10.1126/science.aac7082`.

KO Winemiller, PB McIntyre, L Castello, E Fluet-Chouinard, T Giarrizzo, S Nam, IG Baird, W Darwall, NK Lujan, I Harrison, et al. Balancing hydropower and biodiversity in the Amazon, Congo, and Mekong. *Science*, 351(6269):128–129, 2016b.

Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Rounded dynamic programming for tree-structured stochastic network design. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 479–485, 2014a.

Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Stochastic network design in bidirected trees. In *Advances in Neural Information Processing Systems*, pages 882–890, 2014b.

Xiaojian Wu, Jonathan Gomes-Selman, Qinru Shi, Yexiang Xue, Roosevelt

García-Villacorta, Elizabeth Anderson, Suresh Sethi, Scott Steinschneider, Alexander Flecker, and Carla P. Gomes. Efficiently approximating the pareto frontier: Hydropower dam placement in the amazon basin. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Yexiang Xue, Bistra Dilkina, Theodoros Damoulas, Daniel Fink, Carla Gomes, and Steve Kelling. Improving your chances: Boosting citizen science discovery. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

Michael Yukish. *Algorithms to identify Pareto points in multi-dimensional data sets*. PhD thesis, 2004.

Michael Yukish and Timothy W Simpson. Analysis of an algorithm for identifying pareto points in multi-dimensional data sets. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4324, 2004.

Christiane Zarfl, Alexander Lumsdon, Jürgen Berlekamp, Laura Tydecks, and Klement Tockner. A global boom in hydropower dam construction. *Aquatic Sciences*, 77:161–170, 10 2014. doi: 10.1007/s00027-014-0377-0.

Christiane Zarfl, Alexander E Lumsdon, Jürgen Berlekamp, Laura Tydecks,

and Klement Tockner. A global boom in hydropower dam construction. *Aquatic Sciences*, 77(1):161–170, 2015.

Guy Ziv, Eric Baran, So Nam, Ignacio Rodríguez-Iturbe, and Simon A Levin. Trading-off fish biodiversity, food security, and hydropower in the Mekong river basin. *Proceedings of the National Academy of Sciences*, 109 (15):5609–5614, 2012.