

TETREES: Trade-off Evaluation Through Refined Exact Epsilon-Constraint Solver

Federica de Trizio, Giancarlo Sciddurlo, Giuseppe Piro, Gennaro Boggia, Dept. of Electrical and Information Engineering - Politecnico di Bari, Bari, Italy, CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Email: f.detrizio@phd.poliba.it, {name.surname}@poliba.it

Abstract

TETREES is an open-source Python framework for multi-objective optimization that implements the exact ε -constraint method to compute complete and deterministic Pareto-optimal sets. Its modular architecture decouples objective definition, constraint modeling, and solver configuration, enabling reproducible workflows and seamless integration of new metrics or optimization modules. Built on Gurobi, TETREES supports fine-grained control over branching, heuristics, and search strategies, facilitating scalable exploration of complex decision spaces. A service-to-resource assignment case study illustrates how the framework exposes performance-value trade-offs and enables systematic evaluation of alternative solver configurations, providing a robust software environment for optimization research and decision-support tool development.

Keywords

Multi-objective Optimization, Solver Configuration, Decision Support Systems, Network Orchestration, Sustainability Aware Resource Allocation

Code metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	<i>V1.0</i>
C2	Permanent link to code/repository used for this code version	https://github.com/fdrctrz/TETREES
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	<i>GNU General Public License (GPL) 3.0</i>
C5	Code versioning system used	<i>git, GitHub</i>
C6	Software code languages, tools, and services used	<i>Python, MATLAB, Gurobi</i>
C7	Compilation requirements, operating environments & dependencies	<i>Python\geq3.9, matplotlib\geq3.9.4, pandas\geq2.3.3, gurobipy\geq12.0.3, numpy\geq2.0.2, MATLAB R2023b or later</i>
C8	If available Link to developer documentation/manual	https://github.com/fdrctrz/TETREES
C9	Support email for questions	<i>f.detrizio@phd.poliba.it</i>

1. Introduction

Multi-objective optimization is a cornerstone of multi-criteria decision-making, enabling the systematic improvement of systems and processes across domains such as economics, finance, engineering, and public policy. By simultaneously addressing conflicting objectives, it supports decision-makers in identifying solutions that balance trade-offs among competing criteria [1]. Traditional approaches typically either (i) reformulate a Multi-objective Optimization Problem (MOP) into a single-objective problem via scalarization (most commonly weighted-sum aggregation) or (ii) rely on meta-heuristic and evolutionary search, alongside solver-native lexicographic multi-objective functionalities. While widely used, these approaches often target approximate trade-offs or a single compromise solution rather than enabling systematic, reusable exact Pareto-front enumeration. In particular, weighted aggregation requires empirically chosen weights—introducing subjectivity and additional computational effort that may not always be feasible—whereas meta-heuristics typically perform largely uninformed stochastic exploration, can underuse available domain knowledge, and frequently require extra hyperparameter tuning; hybrid global–local schemes can mitigate some of these issues, but their effectiveness remains strongly problem-dependent [2, 3].

This paper introduces Trade-off Evaluation Through Refined Exact Epsilon-Constraint Solver (TETREES), a Python-based optimization framework that integrates the Gurobi solver and implements the exact ε -constraint method to address the main constraints of scalarization- and heuristic-based techniques. Unlike existing multi-objective toolchains and Gurobi’s native multi-objective features, the proposed software provides a plug-and-play framework for exact Pareto-front enumeration and visualization via MATLAB, where users only specify objectives and constraints while the entire ε -constraint orchestration and result management are fully automated. In contrast to aggregation methods that obscure the nature of inter-objective trade-offs, TETREES explicitly constructs a set of feasible, non-dominated solutions reflecting different prioritizations among objectives. The approach optimizes one objective while converting the remaining ones into inequality constraints bounded by adjustable ε -values [4]. These bounds are systematically swept over ranges defined by the ideal and nadir points, representing the best and worst attainable scores on each objective across the Pareto set. The resulting solution set provides a structured and interpretable approximation of the Pareto frontier. A final dominance-filtering stage ensures that only Pareto-optimal configurations are retained. From a software engineering perspective, TETREES adopts a modular and extensible architecture that separates problem specification, solver configuration, and result analysis into independent, reusable components. This design facilitates adaptation to new problem classes and supports reproducible computational experiments. Objectives, constraints, and solver

parameters can be introduced or modified without altering the core system. Integration with established optimization backends, such as Gurobi, ensures computational efficiency and numerical consistency, while the Python interface promotes accessibility for both research prototyping and applied deployments. In this work, TETREES is applied to the case study introduced in our technical paper [5], which focuses on optimization for next-generation network orchestration. This application demonstrates that the framework supports a structured exploration of complex trade-offs and confirms the validity and practicality of the methodology and results.

2. Software Description

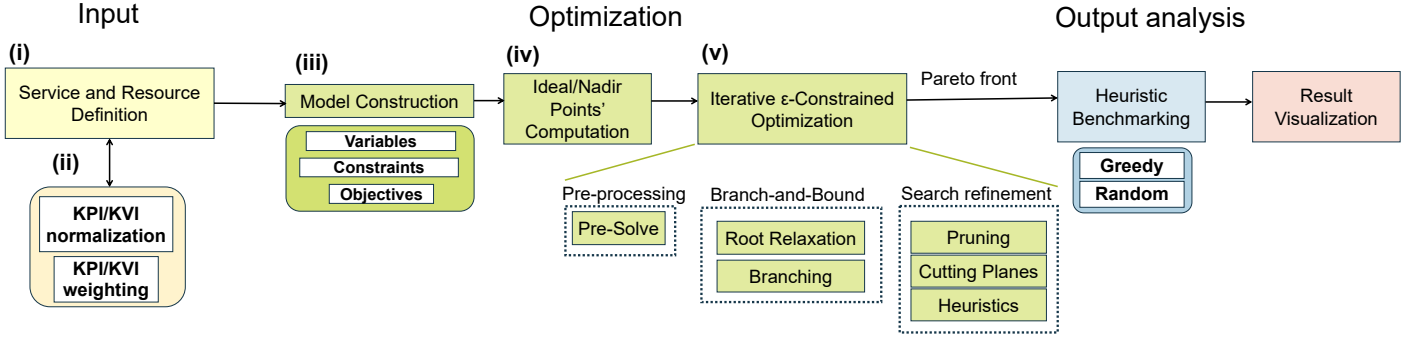


Figure 1: Overview of the TETREES framework architecture.

TETREES is an open-source Python framework that provides decision-support capabilities for multi-objective optimization, with a focus on demand-to-resource assignment problems. The underlying model is formulated as a Binary Integer Linear Programming (BILP) where each service request must be assigned to exactly one resource. The framework jointly optimizes heterogeneous performance and value indicators, Key Performance Indicators (KPIs) (e.g., latency, packet loss, throughput) and Key Value Indicators (KVIs) (e.g., sustainability, trustworthiness, inclusiveness), thus enabling decision-makers to balance operational efficiency with ethical, environmental, and societal factors. The modular architecture of TETREES (Fig. 1) is organized into five core components: (i) definition of service and resource entities; (ii) normalization and weighting of KPI/KVI vectors; (iii) objective-specific optimization routines; (iv) computation of ideal and nadir points for ϵ -constraint initialization; and (v) iterative ϵ -constrained optimization for Pareto-front construction. Custom weighting schemes allow users to encode explicit priorities across indicators. Services specify demand profiles, quality thresholds, and target KPI/KVI values, while resources capture capacity, reliability, energy efficiency, and computational performance. The resulting optimization problem is constrained by service-level requirements, assignment uniqueness, capacity limitations, and ϵ -constraints enforcing Pareto-efficient trade-offs. The ϵ range is derived from the ideal and nadir values of the constrained objective and discretized using a user-defined step size. Exact optimization is executed via the Gurobi Mixed-Integer Programming (MIP) solver, which addresses integrality and linearity constraints using a branch-and-bound algorithm enriched by presolve, pruning, and cutting-plane techniques. The solver relaxes integrality constraints at the root node, solves the corresponding linear relaxation, and recursively branches on fractional variables. Subproblems that cannot improve the incumbent solution are pruned, while cutting planes tighten feasible regions by injecting valid inequalities. During presolve, redundant constraints are eliminated and structurally similar model components are aggregated. Gurobi also employs heuristic routines, such as rounding, local search, probing, and infeasibility repair, to accelerate the discovery of high-quality feasible solutions. Solver behavior can be finely tuned through several parameters. *MIPGap* sets the acceptable optimality gap between the incumbent and the bound; *MIPFocus* biases the search toward feasibility, bound tightening, or optimality proof; *VarBranch* defines the branching strategy (e.g., reduced-cost or strong branching); and *Heuristics* controls the effort allocated to heuristic procedures. These parameters give users control over convergence precision and computational cost. Alongside exact optimization, TETREES offers heuristic benchmarking modules implementing greedy and randomized allocation strategies under capacity constraints. These provide fast approximations and enable comparative assessment against optimal solutions. Results are exported as structured CSV files containing Pareto-optimal

allocations with dominated solutions removed through filtering, ε -constrained solutions, and heuristic benchmarks. MATLAB scripts are included for post-processing and visualization, including Pareto-front plotting and trade-off analysis between KPI and KVI indicators.

3. Illustrative Example

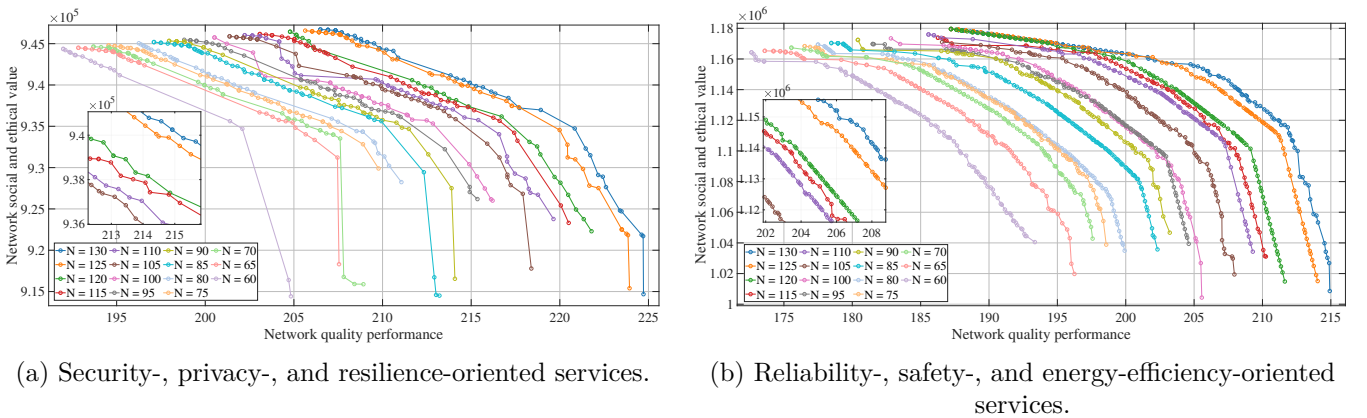
This section demonstrates the application of the TETREES framework to a representative use case inspired by the network orchestration scenario presented in [5]. In this setting, a network orchestrator dynamically assigns incoming service requests to the available resources of one or more providers, selecting the allocation that best satisfies Service Level Agreements (SLAs) while balancing potentially conflicting KPI and KVI objectives.

For the KPI objective, we consider the network performance metric defined in [5], computed as a weighted and normalized aggregation of the data rate, latency, and packet loss associated with serving each request. For the KVI objective, we adopt the social and ethical value introduced in the same reference, which combines trustworthiness, inclusiveness, and environmental sustainability into a single composite measure. The examples rely on the parameters defined in the *main.py* script, namely *num_resources* (number of providers), *num_services_list* (number of service requests), and the corresponding service and resource characteristics. In all experiments, the number of resources varies from 60 to 130, while the number of service requests is fixed at 300.

Figure 2 reports the resulting Pareto fronts. Each curve corresponds to the set of non-dominated allocations obtained for a particular number of providers, and each point represents a feasible service-resource assignment that achieves an optimal trade-off between performance and social/ethical value. As the number of providers increases, the solution space expands and becomes denser, generating progressively richer and more diverse sets of non-dominated solutions.

In the first scenario (Fig. 2a), service requests prioritize stringent security, privacy, and resilience requirements. The extreme points of each frontier correspond to the maximum attainable value of each individual objective, obtained by optimizing solely the social/ethical value or solely the performance metric. Intermediate points represent trade-offs between them, with the central region of each curve identifying the most balanced compromise solutions. The zoomed-in region highlights that even small increases in resource availability yield measurable improvements in the attainable trade-offs, enabling stricter SLAs compliance and more efficient service allocation.

In the second scenario (Fig. 2b), service requests emphasize reliability, physical safety, and energy efficiency. The resulting fronts are more uniformly populated and exhibit smoother progressions compared to the previous case, indicating that this distribution of service requirements induces a more regular and homogeneous solution space. These observations are particularly relevant for infrastructure planning, as they help operators assess the resource availability needed to concurrently meet demanding performance constraints and stringent value-driven objectives.



(a) Security-, privacy-, and resilience-oriented services.

(b) Reliability-, safety-, and energy-efficiency-oriented services.

Figure 2: Pareto fronts for increasing numbers of service providers (N) considering 300 service requests.

Beyond adjusting the number of providers, TETREES also supports the reconfiguration of the solver's internal search strategy within the same operational context. By tuning parameters that control heuristic

intensity, branching depth, and global search priorities, users can obtain Pareto fronts with comparable densities but significantly different computational costs. To illustrate this, Table 1 compares five solver configurations for 110 providers and 300 service requests: Adaptive, Cut-and-Solve, Strong Branching, Sub-Optimal, and Infeasibility-Driven. These configurations modify the Gurobi parameters (e.g., *VarBranch*, *MIPFocus*, and *Heuristics*) introduced in Section 2.

While all configurations eventually produce high-quality Pareto sets, the density and shape of the frontiers differ. The Adaptive and Cut-and-Solve strategies achieve similar runtimes (221–237 s) and produce comparable numbers of Pareto points, though the latter relies on a significantly larger heuristic budget (80%). Strong Branching, despite its computationally intensive branching rule, yields fewer Pareto points (36), confirming that aggressive bound tightening reduces search breadth. The Infeasibility-Driven configuration generates the largest number of non-dominated points (94), but at a runtime more than twice that of the other strategies, illustrating the trade-off between frontier granularity and computational effort. These results show that solver-parameter choices affect not only runtime but also the structure, density, and granularity of the resulting Pareto front. Selecting an appropriate configuration therefore becomes an application-aware design decision, particularly important when timely computation is required for next-generation network services.

Table 1: Simulation results considering 300 service requests and 110 service providers under different solver configurations.

Approach	Branch Strategy	Global Search Strategy	Heuristics (%)	Run time (s)	Pareto Points (#)
Adaptive	Pseudo-Reduced Cost	Balanced	10	221.98	58
Cut-and-Solve	Pseudo-Reduced Cost	Feasible Solutions	80	237.05	59
Strong-Branching	Strong	Best Bound	0	248.18	36
Sub-Optimal	Pseudo-Shadow Price	Best and Fast Bound	5	280.38	36
Infeasibility-Driven	Max. Infeasibility	Balanced	10	624.79	94

4. Impact Overview

This work introduced TETREES, a modular Python-based optimization framework that employs the exact ϵ -constraint method to address conflicting objectives in multi-criteria decision-making. Its architecture cleanly separates objectives, constraints, and solver configurations, ensuring reproducibility and enabling the straightforward integration of new performance and value indicators. The framework has already been used by the authors to support the experimental analysis presented in [5], demonstrating its practical applicability in a peer-reviewed research context and was furthermore evaluated in a representative service-to-resource assignment scenario inspired by real-world network orchestration, where service requests are dynamically allocated to provider resources. The examples illustrate how TETREES supports systematic exploration of allocation strategies that jointly balance network performance with social/ethical value, thus making trade-offs between conflicting objectives transparent and interpretable. By adjusting solver configurations and tailoring search strategies, network operators can identify solutions that maximize performance while upholding sustainability- and trust-oriented criteria, all within operational constraints such as runtime or computational budget. These insights can guide infrastructure planning, strengthen service-level compliance, and support the implementation of value-aware decision policies, ultimately improving both operational efficiency and alignment with long-term sustainability goals. While the experimental evaluation focuses on a representative assignment problem for clarity and reproducibility, the framework itself is not tied to this specific application domain. Beyond the illustrative network orchestration case study, TETREES is applicable to a broad class of binary bi-objective optimization problems, including set covering and partitioning, assignment and scheduling problems, and shortest-path formulations. Future developments will incorporate a full-fledged multi-objective extension, moving beyond aggregation into two macro-categories of objectives, and automated indicator selection and integration through Large Language Model (LLM)-based reasoning, enabling the framework to adapt to evolving requirements. These enhancements aim to advance TETREES toward self-optimizing network systems that are simultaneously efficient, sustainable, and responsive to societal expectations.

References

- [1] N. Gunantara, “A review of multi-objective optimization: Methods and its applications,” *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018.
- [2] D. Whitley, “An overview of evolutionary algorithms: practical issues and common pitfalls,” *Information and Software Technology*, vol. 43, no. 14, pp. 817–831, 2001. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584901001884>
- [3] M. Ehrgott, M. Köksalan, M. Kadziński, and K. Deb, “Fifty years of multi-objective optimization and decision-making: From mathematical programming to evolutionary computation,” *European Journal of Operational Research*, vol. 330, no. 1, pp. 1–25, 2026. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221725004849>
- [4] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin, “An exact ε -constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits,” *European Journal of Operational Research*, vol. 194, no. 1, pp. 39–50, 2009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221707012106>
- [5] G. Sciddurlo, F. de Trizio, G. Piro, and G. Boggia, “A value-driven system design framework for sustainable 6g networks,” *Computer Networks*, vol. 269, p. 111477, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S138912862500444X>