Configuration Support for Lexicographic Multi-Objective Scheduling: A Framework Applied to Electric Vehicle Charging

Laurenz Tomandl[1], Günther Raidl[1],
Tobias Rodemann[2], and Steffen Limmer[2]

[1] TU Wien, Vienna, Austria
[2] Honda Research Institute Europe GmbH, Offenbach/Main, Germany
{ltomandl|raidl}@ac.tuwien.ac.at,
{tobias.rodemann|steffen.limmer}@honda-ri.de

Abstract. Real-world scheduling problems frequently involve many, often conflicting objectives. Lexicographic optimization is a widely used approach for such many-objective problems, expressing the objectives' relative importance through a fixed priority order. However, as the number of objectives grows, configuring an effective order becomes challenging: optimizing certain objectives before others restricts the search space and the still obtainable values for remaining objectives in complex ways, and some objectives may dominate others or induce costly trade-offs. In this work, we present a data-driven framework for analyzing and supporting the configuration of lexicographic optimization in many-objective scheduling problems formulated as linear programs. We consider the space of all objective function permutations and compile an objective order graph that captures how the solution space and diverse quantitative measures evolve as objectives are fixed sequentially. Merging and pruning strategies recognize and remove symmetries, substantially reducing the graph. We demonstrate the framework on a smart electric vehicle charging scheduling problem with seven objectives deployed in a real-world setting. Results reveal strong asymmetries between objectives and show that only a small fraction of all possible objective orders lead to distinct outcomes. We further present an interactive tool that allows users to explore objective orderings and their consequences.

Keywords: Lexicographic optimization, many-objective scheduling, electric vehicle charging, decision support, objective ordering

1 Introduction

Scheduling problems in practice frequently involve multiple, often conflicting objectives that must be balanced simultaneously. In domains such as transportation, energy management, and timetabling, decision makers face the challenge of prioritizing objectives whose interactions are difficult to foresee. Computing the entire Pareto front is often computationally infeasible or unnecessary. A common alternative is to combine multiple objectives into a single objective, e.g.,

by means of a weighted sum, but this requires the careful selection of suitable weights, which is often difficult and unintuitive, and can easily produce misleading outcomes [12].

Lexicographic optimization provides a more intuitive alternative: objectives are optimized sequentially according to a fixed priority order, and once an objective has been optimized, its value is fixed by adding a constraint before proceeding to the next. Formally, let X be a feasible solution space and $f(x) = (f_1(x), \ldots, f_k(x))$, $x \in X$, a vector of objective functions. For $u, v \in \mathbb{R}^k$, define $u \prec v$ iff there exists an index $j = \min\{i : u_i \neq v_i\}$ such that $u_j < v_j$. The lexicographic optimization problem

$$\text{lexmin}_{x \in X} \ (f_1(x), \ldots, f_k(x))$$

asks for an $x^* \in X$ such that for any $x \in X$, $f(x^*) \prec f(x)$ or $f(x^*) = f(x)$ [5].

In many scheduling applications, such an ordering is easier to specify than numerical weights. However, when more than a few objectives are involved, configuring an effective order is far from trivial. Objectives considered early can drastically restrict the remaining feasible solution space, potentially rendering later objectives irrelevant or inducing severe trade-offs.

In this work, we propose a data-driven framework for analyzing objective orderings in lexicographic optimization of (mixed integer) linear programs (MILPs/LPs). We introduce an *objective order graph* (OOG), which compactly encodes restricted search spaces and achievable objective values for any partial objective order, and derive statistics characterizing objective interactions and order sensitivity. We demonstrate the framework on a smart electric vehicle (EV) charging scheduling problem with seven objectives, showing that only a small fraction of all orderings lead to distinct outcomes.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the OOG and its properties. The EV charging case study is described in Section 4, followed by analysis results in Section 5. Section 6 concludes the paper.

2 Related Work

Existing work in lexicographic optimization focuses mostly on algorithmic refinements or specific applications. Cococcioni et al. [4] embed lexicographic priorities into single-objective formulations using the Grossone methodology, while Lai et al. [10,11] study mixed Pareto-lexicographic problems. In metaheuristic optimization, Branke [1] and Gutiérrez et al. [2,3] use lexicographic orderings as preference information to guide search. For a broader perspective, we refer to the survey by Halffmann et al. [8] and to scalarization methods [14] as an alternative.

Closest to our work, Khorram et al. [9] perform a sensitivity analysis on objective priority in lexicographic multi-objective linear programs, showing how to efficiently re-solve the problem when the ordering changes. Their work focuses on adapting solutions to a given new ordering; in contrast, our framework

systematically analyzes and compares all orderings simultaneously to support configuration decisions. Related to this work is that of Eskelinen and Miettinen [7], who analyze local trade-offs in the neighborhood of a selected Pareto-optimal solution in scalarization-based multi-objective optimization, but they do not consider lexicographic orderings of objectives.

In industrial settings, lexicographic optimization is applied for its simplicity and interpretability, including electric vehicle charging [12] and production scheduling [13]. As a concrete test case, we use the hierarchical smart EV charging scheduler of Limmer et al. [12], deployed in a real-world charging management system [6,15].

3 The Objective Order Graph

Consider a many-objective (mixed integer) linear optimization problem solved via lexicographic optimization. We aim to capture how the search space and obtainable objective function values evolve across all meaningful permutations of objectives. To this end, we introduce a data structure we call the *objective order graph* (OOG). As a precursor, we first derive the *objective order tree* (OOT) before compressing and refining it into the OOG. In principle, the OOT is obtained by recursively considering all relevant permutations of the objectives. Each node in the tree represents a subspace of the original search space of the optimization problem, defined by constraints that fix a subset of objectives to the optimal values obtained at earlier stages of the lexicographic optimization process, with the root node corresponding to the original (unrestricted) feasible search space X of the optimization problem, i.e., before any objective has been fixed.

Each edge in the tree represents the act of optimizing a not yet considered objective within the search space of the source node and fixing the objective to the obtained optimal value in the target node. A complete OOT for k objectives thus has $k!$ leaves and $\sum_{i=0}^{k} k!/(k-i)!$ nodes in total (e.g., for $k = 7$: 5040 leaves and 13 700 nodes), since every prefix of every permutation corresponds to a distinct node. Consequently, each root-to-leaf path corresponds to one complete permutation of the objectives. The corresponding leaf represents the feasible search space remaining after all objectives have been fixed in that order.

We now formalize this in a more precise way. Let k be the number of objective functions, vector x be the decision variables of the MILP or LP by which the optimization problem is modeled, and $f_i(x)$ be the i-th objective function for $i = 1, \ldots, k$ in the natural (arbitrary) ordering. W.l.o.g., we assume that all objective functions are to be minimized. Let $a = (u, v)$ be an arc in the OOT from node u to node v. Let u be a node of the OOT. We associate the following data/functions with u:

- Let $d(u)$ be the depth of node u in the OOT, i.e., the number of arcs and therefore fixed objectives along the path from the root to u.
- Let $S(u)$ be the unordered set of indices of objectives along the path from the root to u, i.e., the index set of all already fixed objectives.

- Vector $f_{\min}(u) = (f_{\min}(u)_i)_{i=1,\dots,k}$, where $f_{\min}(u)_i$ is the smallest value attainable by the objective function f_i when it is minimized over the search space represented by node u.
- Vector $f_{\mathrm{ub}}(u) = (f_{\mathrm{ub}}(u)_i)_{i=1,\dots,k}$, where $f_{\mathrm{ub}}(u)_i$ is the largest value attainable by the objective function f_i when it is maximized independently over the search space represented by node u. These values serve as upper bounds and may not be attainable depending on the problem formulation, as meaningful upper bounds do not always exist.
- Vector $f_{\max}(u) = (f_{\max}(u)_i)_{i=1,\dots,k}$, where $f_{\max}(u)_i$ is the largest value that objective f_i actually takes across all final solutions reachable from node u, i.e., the component-wise maximum over the $f_{\min}$-vectors of all leaf nodes in the subtree rooted at u. Unlike $f_{\mathrm{ub}}(u)_i$, it reflects values actually occurring in solutions produced by the lexicographic process. This vector is computed bottom-up once the OOT has been fully constructed. For any leaf node u, all objectives are fixed and $f_{\min}(u)_i = f_{\max}(u)_i = f_{\mathrm{ub}}(u)_i$ for all $i = 1, \dots, k$. For inner nodes u, the bounds satisfy $f_{\min}(u)_i \leq f_{\max}(u)_i \leq f_{\mathrm{ub}}(u)_i$.

While the definitions above assume the complete OOT, this is not required in practice: large parts can be omitted by exploiting problem structure. Note that $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\mathrm{ub}}(u)$ naturally extend to matrix representations when multiple problem instances are considered simultaneously, with each row corresponding to one instance.

Irrelevant Objectives and Nodes. We call an objective f_i *irrelevant* at node u when its fixation does not reduce the search space, i.e., $f_{\min}(u)_i = f_{\mathrm{ub}}(u)_i$. A node u is *irrelevant* if all unfixed objectives are irrelevant. Such nodes can be identified early during construction, avoiding unnecessary further exploration. Since the search space only shrinks along any root-to-leaf path, $f_{\min}(u)_i$ is non-decreasing and $f_{\mathrm{ub}}(u)_i$ non-increasing with depth. Consequently, once an objective is irrelevant at u, it remains irrelevant throughout the entire subtree rooted at u and cannot become relevant again deeper in the tree.

Problem Dependent Knowledge. Problem-specific or expert knowledge can be incorporated as precedence constraints on objectives, restricting which permutations are explored during OOT construction.

Subtree Merging. Different partial objective orders may lead to identical restricted search spaces. Two nodes u and v are *congruent* if they have fixed the same set of objectives ($S(u) = S(v)$) to the same optimal values ($f_{\min}(u)_i = f_{\min}(v)_i$ for all $i \in S(u)$), meaning that the search spaces represented by both nodes are identical and so are their entire subtrees. Note that $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\mathrm{ub}}(u)$ depend only on this search space, not on the order in which the objectives in $S(u)$ were fixed. Merging congruent nodes by redirecting all incoming arcs of one to the other compresses the OOT into a directed acyclic graph, the *objective order graph* (OOG), which preserves all relevant information. In practice, congruent nodes are identified efficiently during construction via hashing on $S(\cdot)$ and the corresponding fixed objective values.

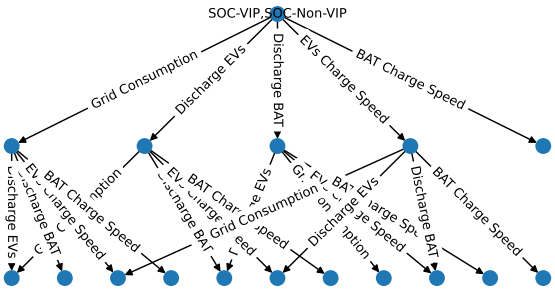

Fig. 1: Partial OOG rooted at the search space after fixing `SOC-VIP` and `SOC-Non-VIP`, shown to depth three. Nodes represent restricted search spaces; edges represent objective fixations. Multiple parents indicate that different orderings lead to the same search space (merging). The node reached by fixing `BAT-Charge-Speed` has no children because this fixation completely determines all remaining objectives.

Pruning of Order-Independent Postfixes. As an optional simplification, we remove *order-independent postfixes*: at a node u where $f_{\min}(u) = f_{\max}(u)$, the attainable minimum and maximum objective values coincide for all remaining objectives, meaning that fixing them in any order yields the same final solution. Consequently, all descendants of u can be removed, and u becomes a leaf. After this pruning step, the leaf nodes represent all decision-relevant prefixes of objective permutations, with $f_{\min}(u) = f_{\max}(u)$ directly giving the final objective values. This criterion differs from irrelevance: irrelevance is defined via the independent upper bound f_{ub}, whereas an order-independent postfix is defined via $f_{\max}$, the maximum actually attained across all solutions reachable from u, and $f_{\max}(u)_i \leq f_{\text{ub}}(u)_i$ in general. A node may therefore admit an order-independent postfix without any remaining objective being irrelevant. This step, however, discards structural information: in particular, details about how the search space is further restricted by the remaining objectives and how upper bounds (f_{ub}) evolve are lost. Since this information may be useful for diagnostic analyses, we treat the pruning as optional.

Figure 1 illustrates a partial OOG for our case study (see next Section 4), showing how different objective fixation orders can lead to the same restricted search space (merged nodes with multiple parents), and how a single objective fixation can completely determine all remaining objectives (leaf without children).

4 Case Study: Smart EV Charging Scheduling

As a case study, we consider the smart EV charging scheduler of Limmer et al. [12], deployed in a real-world charging management system at a company

building in Germany. The scheduler assigns charging power to vehicles and stationary batteries across discrete time slots, considering a photovoltaic system, two-way charging, and grid load constraints. The problem is formulated as a many-objective linear program with the following seven objectives to be minimized:

1. `SOC-VIP`: dissatisfaction of target state-of-charge (SOC) requirements of VIP EVs
2. `SOC-Non-VIP`: dissatisfaction of target SOC requirements of non-VIP EVs
3. `Grid-Consumption`: grid energy consumption
4. `Discharge-EV`: overall discharging of EV batteries
5. `Discharge-BAT`: overall discharging of stationary batteries
6. `EV-Charge-Speed`: negative objective prioritizing early charging and fully charging EVs
7. `BAT-Charge-Speed`: negative objective prioritizing early charging and fully charging batteries

Hard constraints ensure compliance with grid load limits, battery capacity bounds, and minimum SOC requirements. All decision variables are continuous. We treat the scheduler as a black box and refer to Limmer et al. [12] for the full model. The scheduler can be re-invoked with any (partial) objective order and reports the optimal value of each objective as it is fixed. We apply the scheduler to each subproblem associated with each OOG node and thereby compile the graph while imposing the precedence constraint that `SOC-VIP` must always precede `SOC-Non-VIP`.

5 Statistical Analysis

All experiments are conducted on three classes of smart EV charging scheduling instances with increasing problem size, containing 50, 100, and 200 electric vehicles, respectively. For each class, we generated 100 independent artificial instances that roughly mimic realistic scenarios, resulting in a total of 300 problem instances. These instances are available on our website[1]. We combine all 300 instances into a single aggregated OOG: the bound vectors $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\text{ub}}(u)$ turn into matrices with one row per instance, and nodes are merged or pruned only if the respective relationship holds across all rows. As the framework targets offline configuration support, construction cost is non-critical. It is dominated by the underlying LP solve time and the number of explored permutations, which merging and pruning reduce substantially.

Based on this OOG, we derive descriptive statistics that quantify objective interactions, hierarchy sensitivity, and structural complexity. The goal is not to benchmark against alternative multi-objective approaches, but to characterize how objective orderings influence the solution space and final outcomes. We organize the results into node- and edge-level, objective-level, and graph statistics.

[1] `https://www.ac.tuwien.ac.at/files/resources/instances/HRI-charging_problem/instances_HRI_charging_problem_case_study.zip`

Table 1: Average bounding box size by objective and node depth across all test instances relative to the bounding box at depth 0.

	Depth					
Objective (i)	0 (root)	1	2	3	4	5
`SOC-VIP`	1.00	0.73	0.34	0.09	0.02	0.00
`SOC-Non-VIP`	1.00	0.63	0.31	0.16	0.06	0.00
`Grid-Consumption`	1.00	0.78	0.48	0.29	0.13	0.01
`Discharge-EV`	1.00	0.57	0.25	0.11	0.04	0.00
`Discharge-BAT`	1.00	0.60	0.22	0.09	0.04	0.00
`EV-Charge-Speed`	1.00	0.74	0.42	0.18	0.06	0.00
`BAT-Charge-Speed`	1.00	0.74	0.41	0.21	0.12	0.05

5.1 Node- and Edge-Related Statistics

At the level of individual nodes and arcs of the OOG, we analyze how the fixing of respective objectives restricts the remaining search space. For a node u, we consider the normalized bounding box size $\Delta(u)_i = (f_{\max}(u)_i - f_{\min}(u)_i)/(f_{\max}(r)_i - f_{\min}(r)_i)$ for $i = 1, \ldots, k$, where r denotes the root node. This measures the remaining variability of each objective relative to the full range at the root, yielding values in $[0, 1]$.

Table 1 shows average Δ-values over all nodes at each depth, across all instances. We observe a substantial decrease with increasing depth, confirming that earlier objectives are more impactful. The rate varies across objectives: some drop below 10% of their root value already at depth three, while others retain more flexibility.

To quantify the impact that fixing an individual objective along an arc $a = (u, v)$ has, we consider normalized arc-level reduction measures. The *relative range reduction* $\rho(a)_i$ captures the fraction of the remaining relative bounding box of an objective i, assuming $\Delta(u)_i > 0$. In addition, the *relative minimum increase* $\rho_{\min}(a)_i$ measures how strongly fixing the objective associated with arc a increases the minimum attainable value of objective i, i.e., actually worsens the best possible value of f_i.

$$\rho(a)_i = \frac{\Delta(u)_i - \Delta(v)_i}{\Delta(u)_i} \qquad \text{and} \qquad \rho_{\min}(a)_i = \frac{f_{\min}(v)_i - f_{\min}(u)_i}{f_{\max}(u)_i - f_{\min}(u)_i}$$

A value of $\rho(a)_i = 1$ means that f_i is completely fixed (dominated), while $\rho_{\min}(a)_i = 0$ means the attainable minimum is unaffected. We aggregate these into scalar indicators for arc a:

$$R(a) = \frac{1}{k - |S(v)|} \sum_{i \notin S(v)} \rho(a)_i, \qquad R_{\min}(a) = \frac{1}{k - |S(v)|} \sum_{i \notin S(v)} \rho_{\min}(a)_i$$

Tables 2 and 3 report average values of $R(a)$ and $R_{\min}(a)$ across depths. Values close to zero indicate little effect; values close to one indicate near-complete

Table 2: Average search space reduction indicator $R(a)$, averaged over all arcs a that fix the given objective at the given depth.

	Depth					
Objective of arc a	1	2	3	4	5	6
`SOC-VIP`	0.04	0.18	0.41	0.48	0.52	-
`SOC-Non-VIP`	-	0.72	0.58	0.61	0.64	1.00
`Grid-Consumption`	0.31	0.51	0.57	0.60	0.78	1.00
`Discharge-EV`	0.01	0.09	0.43	0.66	0.82	1.00
`Discharge-BAT`	0.23	0.18	0.15	0.15	0.23	1.00
`EV-Charge-Speed`	0.94	0.89	0.78	0.83	0.90	1.00
`BAT-Charge-Speed`	0.38	0.49	0.59	0.71	0.91	1.00

Table 3: Average minimum value increase $R_{\min}(a)$, averaged over all arcs a that fix the given objective at the given depth.

	Depth					
Objective of arc a	1	2	3	4	5	6
`SOC-VIP`	0.01	0.11	0.35	0.46	0.52	-
`SOC-Non-VIP`	-	0.35	0.27	0.36	0.45	1.00
`Grid-Consumption`	0.30	0.50	0.56	0.60	0.78	1.00
`Discharge-EV`	0.00	0.07	0.29	0.50	0.74	1.00
`Discharge-BAT`	0.04	0.06	0.09	0.09	0.19	1.00
`EV-Charge-Speed`	0.35	0.34	0.29	0.39	0.61	1.00
`BAT-Charge-Speed`	0.09	0.24	0.45	0.64	0.89	1.00

elimination of flexibility. A large gap between $R(a)$ and $R_{\min}(a)$ indicates a harmless restriction (search space shrinks but best values are preserved), while similar values of $R(a)$ and $R_{\min}(a)$ indicate a costly fixation.

The objective `EV-Charge-Speed` consistently attains large $R(a)$ values, meaning it almost completely collapses the remaining search space, yet its $R_{\min}(a)$ values are substantially lower, indicating that the remaining solutions are not necessarily bad for other objectives. In contrast, the `Grid-Consumption` objective shows comparable $R(a)$ and $R_{\min}(a)$ values, indicating that its fixation directly worsens the attainable values of remaining objectives. The empty entries for `SOC-Non-VIP` at depth one and `SOC-VIP` at depth six result from the precedence constraint and pruning, respectively.

5.2 Objective-Related Statistics

Figure 2 summarizes the average arc-level reduction measures $\rho(a)_i$ and $\rho_{\min}(a)_i$ over all arcs at any depth and all 300 instances (columns: fixed objective; rows: affected objective). The left plot reveals strong asymmetries: fixing `BAT-Charge-Speed` or `EV-Charge-Speed` leads to strong range reductions in other objectives. Most notably, `BAT-Charge-Speed` always dominates `Discharge-BAT` (complete range collapse), and `EV-Charge-Speed` exerts a similar influence on `BAT-Charge-Speed`. The right plot shows that ef-

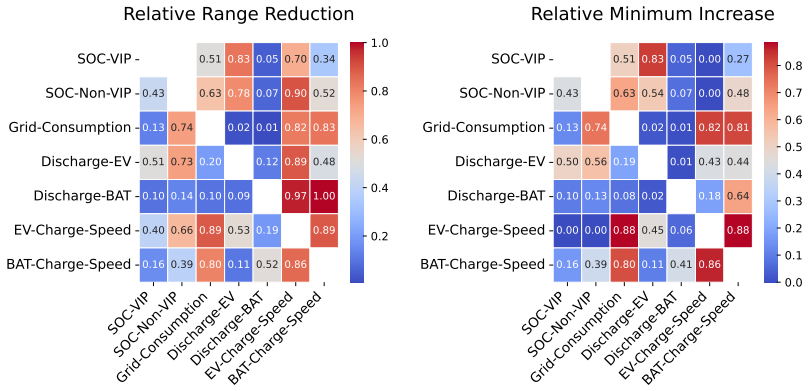

Fig. 2: Average *relative range reduction* and average *relative minimum increase* indicators when the objective in the respective column is optimized and fixed before the objective in the respective row.

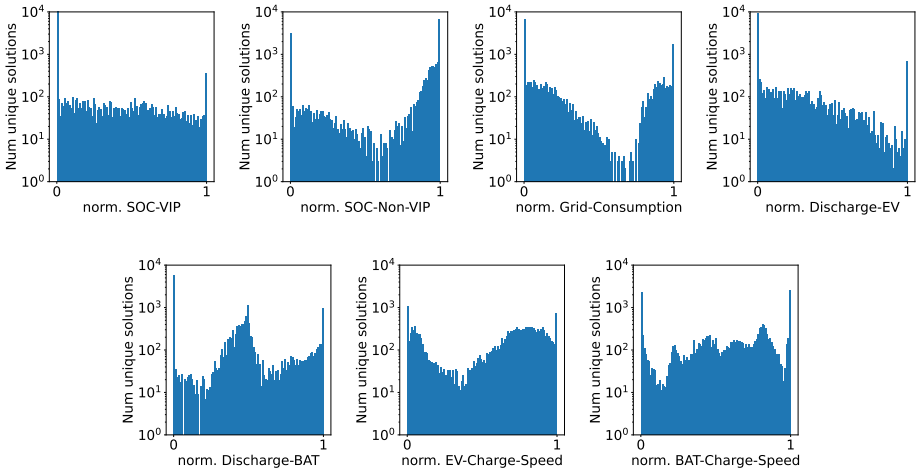

Fig. 3: Distributions of objective function values over all distinct final objective vectors in the OOGs, normalized per instance to $[0, 1]$, displayed on a logarithmic scale.

fects on ranges and on best attainable values can differ substantially: fixing `EV-Charge-Speed` does not worsen the minimum of `SOC-VIP` despite cutting its range considerably, whereas fixing `Grid-Consumption` reduces both range and minimum equally.

Figure 3 shows the distributions of the objective values of the final solutions over all objective orders, counting only distinct objective vectors, i.e., distinct leaf

Table 4: OOG remaining size after subtree merging and pruning, and number of distinct objective vectors.

Setting	Nodes after merging [%]	Leaves after merging [%]	Nodes after pruning [%]	Leaves after pruning [%]	Avg. (stddev.) distinct vectors
50 EVs	4.4	1.1	1.8	3.0	54.49 (11.49)
100 EVs	4.3	1.1	1.9	3.1	56.04 (12.79)
200 EVs	4.3	1.1	1.9	3.1	57.10 (11.67)
Multi	8.9	3.6	4.4	7.2	16763

$f_{\min}$ values after merging. Note that lexicographic optimization assigns a well-defined optimal objective vector to each order, even though the underlying LP optimum need not be a unique point in decision space. Values are normalized per instance using the root bounds $f_{\min}(r)$ and $f_{\max}(r)$, and the y-axis is logarithmic.

Most distributions exhibit a bimodal structure: solutions cluster near the best or worst attainable bounds, with few intermediate values, which is characteristic of lexicographic optimization where early objectives dominate. Notable exceptions are `Discharge-BAT`, `EV-Charge-Speed`, and `BAT-Charge-Speed`, which show broader spreads with substantial intermediate values.

5.3 Graph Statistics

Table 4 summarizes the OOG size after merging and pruning, relative to the full OOT (13 700 nodes, 5040 leaves). The first three rows average over the individual per-instance OOGs, while the "Multi" row is the single OOG combining all 300 instances. Subtree merging reduces the graph to about 4.4% of nodes, and postfix pruning further to 1.9%. On average, only about 56 distinct objective vectors exist per instance, a number remarkably stable across instance sizes. Note that the number of leaf nodes can increase after pruning, as removing a merged leaf with multiple predecessors exposes multiple new leaves at shallower depths. When combining all instances (row "Multi"), the OOG roughly doubles in size, indicating that orders prunable for some instances remain relevant for others.

5.4 Interactive Analysis Tool

To support practical configuration decisions, we developed a prototype tool that allows users to interactively explore the OOG by successively fixing objectives; cf. Figure 4. In this example, `Discharge-BAT` and `Grid-Consumption` have been fixed. The right panel shows histograms of objective value distributions for the current (blue) and preceding (red) search space, allowing immediate assessment of each decision's impact. The upper left panel shows the remaining graph structure, and the lower left panel reports $R(a)$ and $R_{\min}(a)$ values to guide the selection of the next objective.

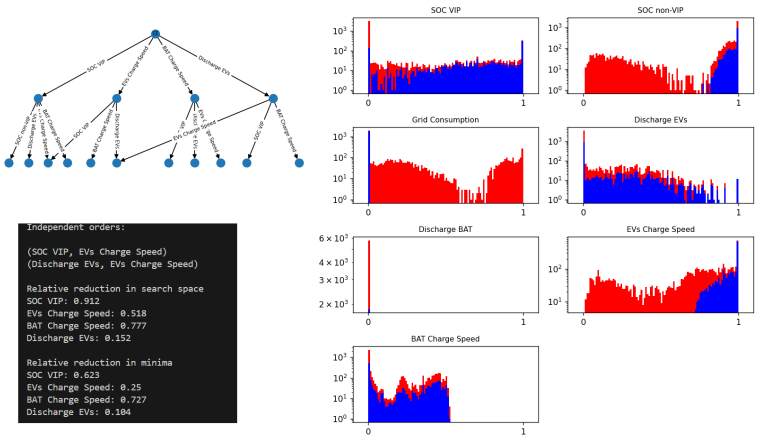

Fig. 4: User interface of prototypical interactive exploration tool.

6 Conclusion and Future Work

We introduced a data-driven framework for analyzing objective orders in lexicographic many-objective optimization. By constructing an *objective order graph* from the full space of objective permutations, merging congruent subtrees and pruning order-independent postfixes, we obtain a compact representation of all decision-relevant orderings. The derived *relative range reduction* and *relative minimum increase* indicators identify dominating objectives and costly trade-offs.

Applied to a real-world smart EV charging scheduling problem with seven objectives, the framework reveals substantial asymmetries between objectives that are difficult to anticipate a priori, highlighting the need for systematic configuration support.

Several directions for future work arise: (a) ranking objective orders by user-defined criteria; (b) relaxing the strict lexicographic order via a tolerance that permits controlled worsening of fixed objectives; (c) scaling to more objectives via more aggressive pruning or sampling; (d) comparing lexicographic and Pareto-based approaches on the same problem; (e) validating the framework on further scheduling and timetabling applications. Although our case study used a linear program, the approach applies to mixed-integer linear programs as well.

References

1. Branke, J.: Consideration of partial user preferences in evolutionary multiobjective optimization. In: Multiobjective Optimization, LNCS, vol. 5252, pp. 157–178. Springer (2008)
2. Castro-Gutiérrez, J., Landa-Silva, D., Moreno-Pérez, J.: Dynamic lexicographic approach for heuristic multi-objective optimization. In: Proceedings of the Work-

shop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009). pp. 153–163. Seville, Spain (2009)
3. Castro-Gutiérrez, J., Landa-Silva, D., Pérez, J.M.: Improved dynamic lexicographic ordering for multi-objective optimisation. In: Parallel Problem Solving from Nature, PPSN XI, LNCS, vol. 6239, pp. 31–40. Springer (2010)
4. Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Towards lexicographic multi-objective linear programming using grossone methodology. AIP Conference Proceedings **1776**(1), 090040 (2016)
5. Ehrgott, M.: Multicriteria Optimization. Springer (2005)
6. Engel, J., Castellani, A., Wollstadt, P., Lanfermann, F., Schmitt, T., Schmitt, S., Fischer, L., Limmer, S., Luttropp, D., Jomrich, F., et al.: A real-world energy management dataset from a smart company building for optimization and machine learning. Scientific Data **12**(1), 1–19 (2025)
7. Eskelinen, P., Miettinen, K.: Trade-off analysis approach for interactive nonlinear multiobjective optimization. OR Spectrum **34**(3), 803–816 (2012)
8. Halffmann, P., Schäfer, L.E., Dächert, K., Klamroth, K., Ruzika, S.: Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. Journal of Multi-Criteria Decision Analysis **29**(5-6), 341–363 (2022)
9. Khorram, E., Zarepisheh, M., Ghaznavi-ghosoni, B.: Sensitivity analysis on the priority of the objective functions in lexicographic multiple objective linear programs. European Journal of Operational Research **207**(3), 1162–1168 (2010)
10. Lai, L., Fiaschi, L., Cococcioni, M.: Solving mixed pareto-lexicographic multi-objective optimization problems: The case of priority chains. Swarm and Evolutionary Computation **55**, 100687 (2020)
11. Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Pure and mixed lexicographic-paretian many-objective optimization: state of the art. Natural Computing **22**(2), 227–242 (2023)
12. Limmer, S., Attig, C., Rodemann, T.: A hierarchical controller as an alternative multi-objective EV charging manager. In: 16th International Conference on Power, Energy, and Electrical Engineering (CPEEE) (2026, in press, accepted)
13. Sawik, T.: Multi-objective master production scheduling in make-to-order manufacturing. International Journal of Production Research **45**(12), 2629–2653 (2007)
14. Sherali, H.D.: Equivalent weights for lexicographic multi-objective programs: Characterizations and computations. European Journal of Operational Research **11**(4), 367–379 (1982)
15. Stadie, M., T., R., Burger, A., Jomrich, F., Limmer, S., Rebhan, S., Saeki, H.: V2B vehicle to building charging manager. In: Proceedings of the EVTeC: 5th International Electric Vehicle Technology Conference. JSAE (2021)