Approximate Multi-Objective Search Under Rulebooks

Omar Muhammetkulyyev[1], Oren Salzman[2], and Tichakorn Wongpiromsarn[1]

[1] Iowa State University, Ames IA 50010, USA
omar99@iastate.edu, nok@iastate
[2] Technion - Israel Institute of Technology, Haifa 3200003, Israel
osalzman@cs.technion.ac.il

Abstract. Robotic planning often involves multiple objectives with complex priority relationships, such as safety, efficiency, and regulatory compliance. Rulebooks formalize these relationships, allowing partial ordering of objectives that generalizes both Pareto and lexicographic dominance. Computing the full set of rulebook-optimal solutions, however, is computationally expensive. To address this challenge, we introduce the concept of ε-rule-dominance, a principled notion of approximate dominance under rulebooks, and propose RA*pex, a best-first search algorithm that efficiently computes a compact set of ε-approximate rulebook-optimal solutions. RA*pex leverages dimensionality reduction, a technique used to speed up existing multi-objective search algorithms, while respecting rule hierarchies by maintaining separate closed sets and performing dominance checks over truncated and residual rule sets. We provide a formal analysis of RA*pex, proving that every rulebook-optimal solution is ε-rule-dominated (a generalization of approximate dominance we introduce) by at least one solution in the returned set. Empirical results demonstrate that our approach achieves computation times over two orders of magnitude faster than existing methods.

Keywords: Multi-objective search · Motion and path planning.

1 Introduction

Robotic planning and navigation problems often require reasoning over multiple, potentially conflicting objectives, including safety, efficiency, and task-related constraints. For example, an autonomous vehicle must avoid collisions, respect traffic rules, and minimize travel time and energy consumption. These requirements naturally lead to multi-objective search (MOS), in which planning is performed on graphs whose edge weights are represented by vectors of multiple cost components, each representing a distinct objective.

In classical MOS, solution paths are compared using Pareto dominance, i.e., a path π dominates another path π' if every cost component of π is no larger than the corresponding component of π' and at least one component is strictly smaller. Instead of returning a single optimal solution, MOS aims to compute the complete set of non-dominated solution paths. While this formulation captures trade-offs among objectives without requiring manual weighting, it is computationally demanding as the number of non-dominated paths grows exponentially with the size of the graph [3, 8]. Moreover, unlike a single objective search where

there is only one optimal cost, the number of optimal cost vectors in MOS are also exponential in terms of the graph size.

The inherent complexity of MOS has motivated the development of approximate algorithms [9, 20]. Instead of computing the complete set of non-dominated solutions, these methods aim to compute a reduced representative subset that approximates the Pareto frontier within a specified tolerance.

Many robotic applications, however, impose additional structure on objective relationships that is not captured by Pareto dominance alone [14]. In particular, safety-critical objectives are often strictly more important than efficiency-related objectives. This observation has led to lexicographic formulations that impose a strict priority ordering among objectives [18]. Lexicographic search can substantially reduce the solution space, but it requires a strict total ordering and cannot handle partial or mixed priority relationships.

In practice, robotic planning problems frequently involve a combination of hierarchical and non-comparable objectives. For example, collision avoidance may be strictly prioritized over regulatory compliance, while objectives such as minimizing energy consumption and travel time may remain mutually non-comparable. We adopt the *rulebooks* formalism [6] to formalize complex relationships among objectives. Rulebooks capture partial priority structures that subsume both Pareto dominance and lexicographic ordering. However, computing the set of non-dominated solutions under a rulebook remains computationally expensive and exhibits exponential worst-case complexity [19].

This paper addresses the computational challenges of MOS under rulebooks. Our contributions are threefold. First, we introduce the notion of $\boldsymbol{\varepsilon}$-dominance with respect to rulebooks, which provides a principled approximation of the rulebook non-dominated solution set. Second, we propose an algorithm that computes a set of approximate solutions such that every solution path is $\boldsymbol{\varepsilon}$-rule-dominated by at least one path in the returned set. Finally, we present extensive experimental results demonstrating that our approach achieves significant computational improvements (up to two orders of magnitude faster) over existing algorithms that compute the complete set of rulebook non-dominated solutions, while preserving the intended objective priorities.

2 Preliminaries

This section presents formal definitions and concepts of MOS [17] and rulebooks [6, 19]. Throughout the paper, we let $\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{>0}$ and $\mathbb{N}$ denote the set of real, non-negative real, positive real, and natural numbers, respectively. Boldface symbols denote vectors or vector-valued functions and v_i denotes the i-th component of a vector or vector function $\boldsymbol{v}$.

Let $\mathbf{p}$ and $\mathbf{q}$ be N-dimensional vectors. We write $\mathbf{p}+\mathbf{q}$ to denote component-wise addition. For a minimization problem, we say that $\mathbf{p}$ *weakly dominates* $\mathbf{q}$, denoted $\mathbf{p} \preceq \mathbf{q}$, if $p_i \leq q_i$ for all i. We say that $\mathbf{p}$ *dominates* $\mathbf{q}$, denoted $\mathbf{p} \prec \mathbf{q}$, if $\mathbf{p} \preceq \mathbf{q}$ and $p_j < q_j$ for at least one index j. Finally, let $\boldsymbol{\varepsilon} \in \mathbb{R}^N_{\geq 0}$ be another non-negative N-dimensional vector. We say that $\mathbf{p}$ *approximately dominates* $\mathbf{q}$ with an *approximation factor* $\boldsymbol{\varepsilon}$, denoted $\mathbf{p} \preceq_{\boldsymbol{\varepsilon}} \mathbf{q}$, if $p_i \leq (1+\varepsilon_i)q_i$ for all i.

2.1 Multi-Objective Search (MOS)

In Multi-Objective Search (MOS), we are given a directed graph $G = (V, E)$ where each edge $e \in E$ has a nonnegative cost vector $\mathbf{c}(e) \in \mathbb{R}^N_{\geq 0}$, where $N > 0$ is the number of objectives. A path is a sequence of vertices $\pi = \langle v_1, \ldots, v_k \rangle$ such that $v_i \in V$ and $(v_i, v_{i+1}) \in E$ for all i. We let Π denote the set of all paths in G. The cost of a path $\pi \in \Pi$ is defined as the component-wise sum of the cost vectors of its edges: $c(\pi) = \sum_i \mathbf{c}(v_i, v_{i+1})$.

Given start and target vertices $s, t \in V$, a path from s to t is called a *solution*. A solution is *Pareto-optimal* iff its cost is not dominated by any other solution. The objective of the basic MOS problem is to compute the set $\Pi^\star \subseteq \Pi$ of Pareto-optimal solutions, also known as the *Pareto front* (PF), for given $s, t \in V$. As the size of $\Pi^\star$ may be exponential in $|V|$, computing the entire PF is often impractical. Thus, we instead seek its bounded approximation. Specifically, given an approximation factor $\boldsymbol{\varepsilon} \in \mathbb{R}^N_{\geq 0}$, the *$\boldsymbol{\varepsilon}$-approximate PF*, denoted $\Pi^\star_{\boldsymbol{\varepsilon}}$, is a set of solutions such that $\forall \pi \in \Pi^\star, \exists \pi' \in \Pi^\star_{\boldsymbol{\varepsilon}}$ *s.t.* $\mathbf{c}(\pi') \preceq_{\boldsymbol{\varepsilon}} \mathbf{c}(\pi)$. Namely, every solution in $\Pi^\star$ is approximately dominated by some solution in $\Pi^*_{\boldsymbol{\varepsilon}}$.

Exact MOS algorithms and dimensionality reduction. Arguably, the most common approach to MOS is using a best-first search approach [2, 4, 5, 12, 13, 16] which generalizes the celebrated A* algorithm [10] to the MOS setting.

A key insight that dramatically improved the efficiency of these algorithms was to order the nodes in the priority queue in increasing lexicographic order and apply the notion of *dimensionality reduction* [15]. More specifically, when the search frontier is ordered lexicographically, the value of the first objective is guaranteed to be non-decreasing throughout the expansion process. This monotonicity implies that any newly generated node will essentially have a first-objective cost that is equal to or greater than that of any previously expanded node, rendering explicit comparisons for this dimension redundant. As a result, the dominance check can be restricted to the remaining objectives, effectively reducing the dimensionality of the problem by one. This was shown (see, e.g., [11, 15]) to have a dramatic impact on the running time of the algorithm. For additional details, see [15, 17].

Approximate MOS algorithms. Arguably, the state-of-the-art algorithm to compute an $\boldsymbol{\varepsilon}$-approximate PF is A*pex [20], which also serves as the algorithmic foundation of our algorithm. Similar to exact MOS heuristic search algorithms, A*pex performs a best-first search over the search space. The efficiency of A*pex stems from the fact that instead of reasoning about single paths, A*pex reasons about *sets* of paths with the same last vertex and similar costs, which results in small numbers of search-node expansions and thus small runtimes.

Specifically, each node of A*pex is represented by an *apex-path pair* $\mathcal{AP} = \langle \boldsymbol{A}, \pi \rangle$, where $\boldsymbol{A} \in \mathbb{R}^N_{\geq 0}$ is called the node's *apex* and $\pi \in \Pi$ is called the node's *representative path*, with the requirement that $\boldsymbol{A} \preceq \mathbf{c}(\pi)$, i.e., the apex weakly dominates the cost of the representative path. The vertex $v(\mathcal{AP})$ of $\mathcal{AP}$ is defined as $v(\mathcal{AP}) := v(\pi)$, where $v(\pi) \in V$ is the last vertex of path π. Each apex-path pair $\mathcal{AP} = \langle \boldsymbol{A}, \pi \rangle$ corresponds to a set of paths $\Pi_{\mathcal{AP}}$ (which includes π) with the same last vertex $v(\mathcal{AP})$. Instead of storing $\Pi_{\mathcal{AP}}$ explicitly, A*pex only

stores the apex $\boldsymbol{A} = \min_{\pi' \in \Pi_{\mathcal{AP}}}\{\mathbf{c}(\pi')\}$, which is the component-wise minimum of (and hence weakly dominates) the costs of all paths in $\Pi_{\mathcal{AP}}$, including the representative path $\pi \in \Pi_{\mathcal{AP}}$. Given a heuristic function $\mathbf{h} : V \to \mathbb{R}^N_{\geq 0}$, the $\mathbf{g}$-value and $\mathbf{f}$-value of $\mathcal{AP} = \langle \boldsymbol{A}, \pi \rangle$ are defined as $\mathbf{g}(\mathcal{AP}) := \boldsymbol{A}$ and $\mathbf{f}(\mathcal{AP}) := \boldsymbol{A} + \mathbf{h}(v(\mathcal{AP}))$. Finally, we say that $\mathcal{AP}$ is $\boldsymbol{\varepsilon}$-bounded iff $\mathbf{f}(\pi) \preceq_{\boldsymbol{\varepsilon}} \mathbf{f}(\mathcal{AP})$, where $\mathbf{f}(\pi) := \mathbf{c}(\pi) + \mathbf{h}(v(\pi))$. As we will see, all nodes in A*pex will be $\boldsymbol{\varepsilon}$-bounded.

Before we explain how apex-path pairs are used, let us define the operations on apex-path pairs: First, *extending* an apex-path pair $\mathcal{AP} = \langle \boldsymbol{A}, \pi \rangle$ by an edge e results in an apex-path pair $\mathcal{AP}' = \langle \boldsymbol{A} + \mathbf{c}(e), \pi' \rangle$, where π' is a path π extended by edge e. Conceptually, apex-path pair $\mathcal{AP}'$ corresponds to the set of paths $\Pi_{\mathcal{AP}'}$ that extends every path in $\Pi_{\mathcal{AP}}$ by e. It is easy to verify that the apex of $\mathcal{AP}'$ is the component-wise minimum of the costs of the paths in $\Pi_{\mathcal{AP}'}$.

The second operation is *merging* two apex-path pairs that contain the same vertex. Conceptually, merging two apex-path pairs corresponds to merging the two sets of paths that these two apex-path pairs correspond to. Hence, the apex of the merged apex-path pair is the component-wise minimum of the apexes of the two apex-path pairs. The representative path of the merged apex-path pair is either one of the two representative paths of the two apex-path pairs.

A*pex starts with a single apex-path pair $\langle \mathbf{0}, [s] \rangle$ in a priority-queue OPEN. In each iteration, A*pex extracts an apex-path pair $\mathcal{AP}$ from OPEN with the lexicographically smallest $\mathbf{f}$-value. The algorithm then performs dominance checks and discards $\mathcal{AP}$ if either (i) there exists a solution already found whose cost $\boldsymbol{\varepsilon}$-dominates the $\mathbf{f}$-value of $\mathcal{AP}$ or if (ii) there exists an expanded apex-path pair that contains $v(\mathcal{AP})$ and whose $\mathbf{g}$-value weakly dominates $\mathbf{g}(\mathcal{AP})$

When A*pex expands an apex-path pair that contains the goal vertex t, it adds the representative path of this apex-path pair to the set of found solutions. When A*pex expands an apex-path pair that does not contain t, it generates a child apex-path pair $\mathcal{AP}$ for each vertex v' such that $(v, v') \in E$ by extending the expanded apex-path pair with edge (v, v'). Consider that the child apex-path pair $\mathcal{AP}$ is not discarded after the dominance checks, and let OPEN$[v]$ be the set of apex-path pairs in OPEN that contains vertex v. A*pex then checks if there exists an apex-path pair $\mathcal{AP}'$ in OPEN$[v(\mathcal{AP})]$ that results in an $\boldsymbol{\varepsilon}$-bounded apex-path pair when merged with $\mathcal{AP}$. If so, A*pex removes $\mathcal{AP}'$ from OPEN and then adds the merged apex-path pair to OPEN. Otherwise, it adds $\mathcal{AP}$ to OPEN. When OPEN becomes empty, A*pex terminates and returns the set of solutions as an $\boldsymbol{\varepsilon}$-approximate Pareto frontier.

2.2 Rulebooks

To avoid ambiguity in terminology, we fix the following conventions for the remainder of the paper. Specifically, we follow standard robotics terminology and use the term *state* to refer to what is commonly called a *vertex* in graph-search literature, and *realization* to refer to a *path*. Readers familiar with graph terminology may interpret these terms interchangeably.

A rulebook is evaluated over a set of possible outcomes, referred to as the set of *realizations* and denoted by Σ. In the context of multi-objective search, a realization corresponds to a path in the underlying graph. A rulebook con-

sists of two main components: a set of rules and a preorder that specifies their relative importance. Each rule corresponds to a distinct objective and assigns a non-negative cost that reflects the degree to which the realization violates the objective. The preorder encodes priority relationships among rules, allowing the representation of both hierarchical objectives and objectives that are not comparable.

Definition 1. *A* rule *is a function $r : \Sigma \to \mathbb{R}_{\geq 0}$ that measures the degree of violation of its argument.*

For any realizations $x, y \in \Sigma$, $r(x) < r(y)$ indicates that y violates the rule r to a greater extent than x. In particular, $r(x) = 0$ indicates that x fully satisfies the rule. Although we refer to these functions as "rules", they are not limited to regulatory constraint. A rule may represent performance-related objectives like efficiency or user preferences that are desirable but not strictly required.

Definition 2. *A* preorder *on a set S is a binary relation $\lesssim$ that is reflexive ($s \lesssim s$ for all $s \in S$), and transitive ($s_1 \lesssim s_2$ and $s_2 \lesssim s_3$ imply $s_1 \lesssim s_3$ for all $s_1, s_2, s_3 \in S$).*

A preorder generalizes a partial order by relaxing the antisymmetry requirement. In particular, it allows both $s_1 \lesssim s_2$ and $s_2 \lesssim s_1$ to hold for distinct elements $s_1 \neq s_2$. This property enables the representation of objectives that are of equal priority [19]. One can define an equivalence relation $\sim$ on S so that $s_1 \sim s_2$ if and only if $s_1 \lesssim s_2$ and $s_2 \lesssim s_1$. The preorder on S can then be viewed as a partial order on $S \setminus \sim$ where each element is an equivalence class of $\sim$.

Definition 3. *A rulebook is defined as a tuple $\mathcal{R} = \langle R, \lesssim \rangle$, where R is the set of rules and $\lesssim$ is a preorder on R that specifies their relative importance.*

As proved by Slutsky et al. [6], a rulebook $\mathcal{R}$ induces a preorder $\lesssim_{\mathcal{R}}$ on Σ, ensuring consistency and preventing cyclic preferences among realizations. Intuitively, a realization x is considered "at least as desirable as" a realization y if any increased violation in a lower-priority rule is compensated by an improvement in a higher-priority rule. Formally, we say that x *weakly rule-dominates* y, denoted $x \lesssim_{\mathcal{R}} y$, if for any rule $r' \in R$ such that $r'(x) > r'(y)$, there exists a higher priority rule $r > r'$ such that $r(x) < r(y)$. We say that x *rule-dominates* y, denoted by $x <_{\mathcal{R}} y$, if $x \lesssim_{\mathcal{R}} y$ but $y \not\lesssim_{\mathcal{R}} x$.

3 Problem Formulation

We consider multi-objective search on a graph $G = (S, E)$, where S is a finite set of states and $E \subseteq S \times S$ is a finite set of edges. A realization is any path in G, regardless of its start or end states. We let Σ denote the set of all realizations. The quality of a realization is evaluated using a rulebook $\mathcal{R} = \langle R, \lesssim \rangle$ where $R = \{r_1, \ldots, r_N\}$, which induces a preorder $\lesssim_{\mathcal{R}}$ on Σ as defined in Section 2.2.

Let $s_{\text{start}} \in S$ and $s_{\text{goal}} \in S$ denote the start and goal states, respectively. We define the set of solutions as the subset of realizations that start at s_{start} and end at s_{goal} and denote this set by $\Sigma_{\text{sol}} \subseteq \Sigma$.

We first define the notion of optimality with respect to a rulebook. A solution is said to be *rulebook-optimal* if it is not rule-dominated by any other solution. The set of all such solutions forms the *rulebook-optimal solution set.*

Definition 4. *The* rulebook-optimal solution set *is defined as* $\mathcal{P}_{\mathcal{R}} = \{x \in \Sigma_{sol} \mid \nexists y \in \Sigma_{sol} \text{ such that } y <_{\mathcal{R}} x\}$

The set $\mathcal{P}_{\mathcal{R}}$ generalizes the classical Pareto-optimal set and captures both hierarchical and non-comparable objectives. As in standard multi-objective search, the size of $\mathcal{P}_{\mathcal{R}}$ can grow exponentially in the size of the graph, making exact computation impractical for large problems. To enable approximation of $\mathcal{P}_{\mathcal{R}}$, we introduce a relaxed notion of rule-dominance.

Definition 5. *Let* $\mathcal{R} = \langle R, \lesssim \rangle$ *be a rulebook, where* $R = \{r_1, \ldots, r_N\}$*, and* $\boldsymbol{\varepsilon} \in \mathbb{R}^N_{\geq 0}$*. For realizations* $x, y \in \Sigma$*, we say that* x $\boldsymbol{\varepsilon}$-rule-dominates y*, denoted* $x \lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}} y$ *if for any rule* $r_j \in R$ *such that* $r_j(x) > (1+\varepsilon_j) r_j(y)$*, there exists a higher-priority rule* $r_i > r_j$ *such that* $r_i(x) < (1+\varepsilon_i) r_i(y)$*.*

Example 1. Consider a rulebook $\mathcal{R} = \langle R, \lesssim \rangle$ with $R = \{r_1, r_2, r_3\}$, $r_1 > r_2$ and $r_1 > r_3$, i.e., r_1 is the highest priority rule while r_2 and r_3 are not comparable. Consider two realizations $x, y \in \Sigma$ with $r_1(x) = 1.9$, $r_2(x) = r_3(x) = 2$ and $r_1(y) = r_2(y) = r_3(y) = 1$. Then, $y \lesssim_{\mathcal{R}} x$ but $x \not\lesssim_{\mathcal{R}} y$ because $r_1(x) > r_1(y)$ and there is no rule r_i satisfying $r_i > r_1$. However, with $\boldsymbol{\varepsilon} = \mathbf{1}$, we get $y \lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}} x$ and $x \lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}} y$ because $r_1(x) < (1+\varepsilon_1) r_1(y)$.

This definition relaxes rule-dominance by allowing bounded degradation, controlled by $\boldsymbol{\varepsilon}$. Using $\boldsymbol{\varepsilon}$-rule-dominance, we define a notion of an approximate optimal solution set.

Definition 6. *A set* $\mathcal{P}^{\boldsymbol{\varepsilon}}_{\mathcal{R}} \subseteq \Sigma_{sol}$ *is an* $\boldsymbol{\varepsilon}$-approximate rulebook-optimal solution set *if for every rulebook-optimal solution* $x' \in \mathcal{P}_{\mathcal{R}}$*, there exists a solution* $x \in \mathcal{P}^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$ *such that* $x \lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}} x'$*.*

We note that this definition does not require the elements of $\mathcal{P}^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$ to be rulebook-optimal or even near-optimal. In the extreme case, the entire solution set Σ_{sol} trivially satisfies this definition. Our goal is not to minimize the suboptimality of the returned solutions, but to efficiently compute $\mathcal{P}^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$, ideally one of small size.

Problem 1 (Approximate Multi-Objective Search under Rulebooks:). Given a graph $G = (S, E)$, start and goal states $s_{\text{start}}, s_{\text{goal}} \in S$, a rulebook $\mathcal{R} = \langle R = \{r_1, \ldots, r_N\}, \lesssim \rangle$, and a tolerance vector $\boldsymbol{\varepsilon} \in \mathbb{R}^N_{\geq 0}$, compute an $\boldsymbol{\varepsilon}$-approximate rulebook-optimal solution set $\mathcal{P}^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$.

4 Rulebook-A*pex (RA*pex)

This section presents rulebook-A*pex (RA*pex), our algorithm for Problem 1. At a high level, RA*pex follows the same best-first "expand-and-prune" template as A*pex, but replaces component-wise $\boldsymbol{\varepsilon}$-dominance $\preceq_{\boldsymbol{\varepsilon}}$ with the rulebook-based relation $\lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$. This change is conceptually simple, yet it has two important consequences that drive the technical development in the rest of the section.

A*pex can rely on purely component-wise reasoning: when OPEN is ordered lexicographically by $\mathbf{f}$, the first component becomes monotone along the search and can be ignored in dominance checks (dimensionality reduction). Under rulebooks, dominance depends on the *priority structure* among rules and, crucially, on whether a difference is *strict* (an improvement in a higher-priority rule can

compensate for a degradation in a lower-priority rule). As a result, the simple component-wise dominance machinery of A*pex no longer applies directly.

We first give a baseline RA*pex that operates on full rule-value vectors and uses $\lesssim_{\mathcal{R}}$ and $\lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$ directly for pruning. This baseline introduces the algorithmic logic but does not benefit from dimensionality reduction. We then develop a dimensionality-reduction variant: we explain why the A*pex argument fails under rulebooks, and show how to restore an efficient dominance test by partitioning expanded nodes while preserving correctness.

4.1 Baseline RA*pex

To simplify notation, we define a mapping that associates each realization with its vector representation of rule-violation values.

Definition 7. *Let $\mathcal{R} = \langle R, \lesssim \rangle$ be a rulebook with $R = \{r_1, \ldots, r_N\}$. We define the mapping $\mathbf{c}_{\mathcal{R}} : \Sigma \to \mathbb{R}^N_{\geq 0}$ by $\mathbf{c}_{\mathcal{R}}(x) = [r_1(x), \ldots, r_N(x)]$ so that the i-th component of $\mathbf{c}_{\mathcal{R}}(x)$ represents the degree to which realization x violates rule r_i.*

This vector representation allows us to reason directly about N-dimensional vectors (similar to the way that A*pex reasons about path costs). In particular, both rule-dominance and $\boldsymbol{\varepsilon}$-rule-dominance can be defined and evaluated purely at the vector level, without reference to the underlying realization.

Definition 8. *Let $\mathcal{R} = \langle R, \lesssim \rangle$ be a rulebook with $R = \{r_1, \ldots, r_N\}$ and let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N_{\geq 0}$ be N-dimensional vectors. We say that $\mathbf{u} \lesssim_{\mathcal{R}} \mathbf{v}$, if for any rule $r_j \in R$ such that $u_j > v_j$, there exists a higher-priority rule $r_i > r_j$ such that $u_i < v_i$.*

Similarly, $\boldsymbol{\varepsilon}$-rule-dominance $\lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$ can be defined for vectors in the same way as for realizations in Definition 5. Motivated by A*pex, we represent each node by a *rule-apex-realization pair*.

Definition 9. *A* rule-apex-realization pair $\mathcal{RP} = \langle \mathbf{r}, x \rangle$ *consists of a* rule apex $\mathbf{r} \in \mathbb{R}^N_{\geq 0}$ *and a* representative realization $\mathbf{r} \lesssim_{\mathcal{R}} \mathbf{c}_{\mathcal{R}}(x)$.

A state of $\mathcal{RP} = \langle \mathbf{r}, x \rangle$, denoted $s(\mathcal{RP})$, is defined as the last state of x. Similar to A*pex, for a given heuristic function $\mathbf{h} : S \to \mathbb{R}^N_{\geq 0}$, we define the $\mathbf{g}$-value and $\mathbf{f}$-value of $\mathcal{RP}$ as $\mathbf{g}(\mathcal{RP}) := \mathbf{r}$ and $\mathbf{f}(\mathcal{RP}) := \mathbf{r} + \mathbf{h}(s(\mathcal{RP}))$. We say that $\mathcal{RP}$ is $\boldsymbol{\varepsilon}$-bounded iff $\mathbf{f}(x) \lesssim^{\varepsilon}_{\mathcal{R}} \mathbf{f}(\mathcal{RP})$, where $\mathbf{f}(x) := \mathbf{c}_{\mathcal{R}}(x) + \mathbf{h}(s(x))$. RA*pex ensures that all the rule-apex-realization pairs are $\boldsymbol{\varepsilon}$-bounded.

The baseline version of RA*pex, outlined in Alg. 1 closely follows the structure of A*pex, but replaces vector-based dominance ($\preceq$ and $\preceq_{\boldsymbol{\varepsilon}}$) with rule-dominance ($\lesssim_{\mathcal{R}}$ and $\lesssim^{\boldsymbol{\varepsilon}}_{\mathcal{R}}$) to account for hierarchical rule priorities. This difference is reflected in the is_dominated function, outlined in Alg. 2.

RA*pex maintains three main data structures throughout the search: (i) the priority queue OPEN, storing candidate rule-apex-realization pairs and ordered according to rule-dominance $\lesssim_{\mathcal{R}}$ (instead of lexicographic ordering as in A*pex), since dominance checks are performed over the full rule-value vectors; (ii) the set *solutions*, storing the current representative solutions; and (iii) a per-state closed structure $G_{\text{cl}}(\cdot)$ (initialized in Line 4), storing the $\mathbf{f}$-values of rule-apex-realization pairs that have already been expanded at state s.

At each iteration, RA*pex extracts a rule-apex-realization pair $\mathcal{RP} = \langle \mathbf{r}, x \rangle$ from OPEN (Line 6) and immediately applies the pruning test is_dominated

Algorithm 1: RA*pex$(S, E, \mathcal{R}, s_{\text{start}}, s_{\text{goal}}, \mathbf{h}, \boldsymbol{\varepsilon})$

```
1  OPEN ← {⟨ 0, [s_start]⟩} ;        // ordered according to rule-dominance ≲_R
2  solutions ← ∅;
3  for s ∈ S do
4  |  G_cl(s) ← ∅ ;
5  while OPEN ≠ ∅ do
6  |  RP = ⟨r, x⟩ ← OPEN.extract_min()
7  |  if is_dominated(RP, G_cl, solutions) then                    // Alg. 2
8  |  |  continue ;
9  |  G_cl(s(RP)).add(f(RP)) ;
10 |  if s(RP) = s_goal
11 |  |  insert(RP, solutions) ;                                   // Alg. 4
12 |  |  continue ;
13 |  for s' ∈ succ(s(RP)) do
14 |  |  RP' ← ⟨r + c_R(⟨s(RP), s'⟩), extend(x, ⟨s(RP), s'⟩)⟩ ;
15 |  |  if is_dominated(RP', G_cl, solutions) then                // Alg. 2
16 |  |  |  continue ;
17 |  |  insert(RP', OPEN) ;                                      // Alg. 4
18 |  return{x : ⟨r, x⟩ ∈ solutions}
```

Algorithm 2: is_dominated$(\mathcal{RP} = \langle \mathbf{r}, x\rangle, G_{\text{cl}}, solutions)$

```
1  if ∃RP' = ⟨r', x'⟩ ∈ solutions : f(x') ≲^ε_R f(RP) then          // Alg. 3
2  |  remove ⟨r', x'⟩ from solutions ;
3  |  add ⟨comp_wise_min(r, r'), x'⟩ to solutions ;
4  |  return TRUE ;
5  if ∃w ∈ G_cl(s(RP)) : w ≲_R r then
6  |  return TRUE ;
7  return FALSE ;
```

(Line 7, implemented in Alg. 2). This test consults both *solutions* and the state-local closed set $G_{\text{cl}}(s(\mathcal{RP}))$ to decide whether $\mathcal{RP}$ can be discarded; the approximate dominance checks within is_dominated rely on Alg. 3. If $\mathcal{RP}$ is not pruned, we insert its $\mathbf{f}$-value into $G_{\text{cl}}(s(\mathcal{RP}))$ (Line 9).

If $\mathcal{RP}$ reaches the goal state (Line 10), we insert it into *solutions* (Line 11) using Alg. 4. Similar to A*pex, RA*pex merges rule-apex-realization pairs by taking the component-wise minimum of their rule apexes (i.e., the merged apex is $\min\{\mathbf{r}, \mathbf{r}'\}$) and keeping one of the two representative realizations; this is performed by merge in Alg. 4, Line 2. Otherwise, we generate successors (Line 14); each child $\mathcal{RP}'$ is again filtered by the same pruning test (Line 15) and, if it survives, inserted into OPEN via Alg. 4 (Line 17). Upon termination, the algorithm returns the representative realizations stored in *solutions* (Line 18).

The core operation underlying all dominance checks is the $\boldsymbol{\varepsilon}$-rule-dominance test $\lesssim^{\varepsilon}_{\mathcal{R}}$, shown in Alg. 3. Since the rulebook $\mathcal{R}$ forms a partial order on $R\backslash\sim$, it can be represented by a directed acyclic graph $G_{\mathcal{R}} = (V_{\mathcal{R}}, E_{\mathcal{R}})$. Every vertex $v_i^{\mathcal{R}} \in V_{\mathcal{R}}$ represents a set of equivalent rules $[v_i^{\mathcal{R}}] \subseteq R$, such that $r \sim r'$ for all $r, r' \in [v_i^{\mathcal{R}}]$. An edge $v_i^{\mathcal{R}} \to v_j^{\mathcal{R}}$ means that there are rules $r_i \in [v_i^{\mathcal{R}}]$ and $r_j \in [v_j^{\mathcal{R}}]$ such that $r_i > r_j$, meaning that r_i has a higher rank that r_j, also implying that the rules in $[v_i^{\mathcal{R}}]$ have a higher rank than the rules in $[r_j^{\mathcal{R}}]$. The procedure in Alg.

Algorithm 3: Approximate dominance check $x \lesssim^{\varepsilon}_{\mathcal{R}} y$

Let $\langle v_1^{\mathcal{R}}, v_2^{\mathcal{R}} \dots v_M^{\mathcal{R}} \rangle$ be a topological order of $R \backslash \sim$;
$Q \leftarrow \langle v_1^{\mathcal{R}}, v_2^{\mathcal{R}}, \dots, v_M^{\mathcal{R}} \rangle$;
while $Q \neq \emptyset$ **do**
  $v = Q.pop()$;
  if $\exists r \in [v]$ *such that* $r(x) > (1+\varepsilon) r(y)$ **then**
    return `FALSE`;
  else if $\exists r \in [v]$ *such that* $r(x) < (1+\varepsilon) r(y)$
    remove all successors of v from Q ;
return `TRUE` ;

Algorithm 4: insert$(\mathcal{RP}, list)$

for $\mathcal{RP}' \in list$ **do**
  $\mathcal{RP}_{new} \leftarrow \mathsf{merge}(\mathcal{RP}, \mathcal{RP}')$;
  if $\mathcal{RP}_{new}$ *is* $\boldsymbol{\varepsilon}$*-bounded*
    remove $\mathcal{RP}'$ from *list* ;
    add $\mathcal{RP}_{new}$ to *list* ;
    return ;
add $\mathcal{RP}$ to *list* ;
return ;

3 follows the rulebook structure by traversing the rules in the topological order $\langle v_1^{\mathcal{R}}, v_2^{\mathcal{R}}, \dots, v_M^{\mathcal{R}} \rangle$ of their equivalence classes in $V_{\mathcal{R}}$, where $M \leq N$ is the number of equivalence classes. It checks for violations that exceed the allowed $\boldsymbol{\varepsilon}$ bound and prunes the lower-priority rules once sufficient improvement is detected at a higher level. This hierarchical filtering is what fundamentally distinguishes RA*pex from A*pex and explains why dominance checks are more complex in the rulebook setting.

4.2 Dimensionality Reduction

Recall that dimensionality reduction is more challenging with rulebooks because of the structure of rule-dominance. In A*pex, OPEN is ordered lexicographically by the $\mathbf{f}$-values. Thus, when an apex-path pair $\mathcal{AP}$ is extracted from OPEN, lexicographic best-first ordering guarantees that the first component of $\mathbf{f}(\mathcal{AP})$ cannot improve upon that of any previously expanded pair that ends at the same state or any pair already in the solution set, i.e., $f_1(\mathcal{AP}) \geq f_1(\mathcal{AP}')$ for all $\mathcal{AP}' \in G_{\mathrm{cl}}(s(\mathcal{AP})) \cup solutions$. This property enables a simple form of dimensionality reduction, i.e., when checking whether $\mathbf{f}(\mathcal{AP})$ is dominated by any vector in $G_{\mathrm{cl}}(s(\mathcal{AP})) \cup solutions$, the first component can be ignored and dominance checks reduce to comparisons over the remaining components only.

Example 2. Given three vectors $\mathbf{u} = [1,3,4]$, $\mathbf{v} = [2,3,4]$, and $\mathbf{w} = [2,4,1]$, consider testing whether $\mathbf{u} \preceq \mathbf{w}$ or $\mathbf{v} \preceq \mathbf{w}$. Since $u_1 \leq w_1$, it suffices to compare the suffixes $[3,4]$ and $[4,1]$. Since $[3,4] \not\preceq [4,1]$, we can conclude that $u \not\preceq w$. The same reasoning applies to $\mathbf{v}$, yielding $\mathbf{v} \not\preceq \mathbf{w}$.

When there are hierarchies among the rules, this simplification no longer applies. Although RA*pex also guarantees that the first component of $\mathbf{f}(\mathcal{RP})$ is not smaller than that of any previously expanded pair that ends at $s(\mathcal{RP})$

Algorithm 5: is_dominated$_{\mathrm{dr}}(\mathcal{RP} = \langle \mathbf{r}, x\rangle, G^{=}_{\mathrm{cl}}, G^{<}_{\mathrm{cl}}, \mathit{solutions})$

if $\exists \mathcal{RP}^{=} \in G^{=}_{\mathrm{cl}}(s(\mathcal{RP})) : f_1(\mathcal{RP}^{=}) < f_1(\mathcal{RP})$ **then** `// transfer` $G^{=}_{\mathrm{cl}}$ `to` $G^{<}_{\mathrm{cl}}$
  for $\mathcal{RP}' \in G^{<}_{\mathrm{cl}}(s(\mathcal{RP}))$ **do**
    if $\exists \mathcal{RP}'' \in G^{=}_{\mathrm{cl}}(s(\mathcal{RP})) : Tr(\mathbf{f}(\mathcal{RP}'')) \lesssim_{\mathcal{R}} Tr(\mathbf{f}(\mathcal{RP}'))$
      remove $\mathcal{RP}'$ from $G^{<}_{\mathrm{cl}}(s(\mathcal{RP}))$;
  $G^{<}_{\mathrm{cl}}(s(\mathcal{RP})) \leftarrow G^{<}_{\mathrm{cl}}(s(\mathcal{RP})) \cup G^{=}_{\mathrm{cl}}(s(\mathcal{RP}))$;
  $G^{=}_{\mathrm{cl}}(s(\mathcal{RP})) \leftarrow \emptyset$;
if $\exists \mathcal{RP}^{=} \in G^{=}_{\mathrm{cl}}(s(\mathcal{RP})) : Tr(\mathbf{f}(\mathcal{RP}^{=})) \lesssim_{\mathcal{R}} Tr(\mathbf{f}(\mathcal{RP}))$ **then**
  return `TRUE` ;
if $\exists \mathcal{RP}^{<} \in G^{<}_{\mathrm{cl}}(s(\mathcal{RP})) : \alpha(\mathbf{f}(\mathcal{RP}^{<})) \lesssim_{\mathcal{R}} \alpha(\mathbf{f}(\mathcal{RP}))$ **then**
  return `TRUE` ;
if $\exists \mathcal{RP}' = \langle \mathbf{r}', x'\rangle \in \mathit{solutions} : \mathbf{f}(x') \lesssim^{\epsilon}_{R} \mathbf{f}(\mathcal{RP})$ **then**
  remove $\langle \mathbf{r}', x'\rangle$ from *solutions* ;
  add $\langle$comp_wise_min$(\mathbf{r}, \mathbf{r}'), x'\rangle$ to *solutions* ;
  return `TRUE` ;
return `FALSE` ;

or any pair in *solutions*, this information alone is insufficient to determine rule-dominance. Unlike component-wise dominance, rule-dominance depends not only on non-inferiority at higher-priority rules, but also on the presence of strict improvements that may compensate for degradations at lower-priority rules. Consequently, lexicographic ordering cannot isolate a single component, and dominance checks must explicitly account for the rule hierarchy and distinguish strict from non-strict improvements, as illustrated in the following example.

Example 3. Consider the same rulebook as in Example 1, namely $\mathcal{R} = \langle R, \lesssim\rangle$ with $R = \{r_1, r_2, r_3\}$, $r_1 > r_2$ and $r_1 > r_3$. Let x, y, z be realizations with $\mathbf{c}_{\mathcal{R}}(x) = [1, 3, 4]$, $\mathbf{c}_{\mathcal{R}}(y) = [2, 3, 4]$ and $\mathbf{c}_{\mathcal{R}}(z) = [2, 4, 1]$. For consistency with Example 2, let $\mathbf{u} = \mathbf{c}_{\mathcal{R}}(x)$, $\mathbf{v} = \mathbf{c}_{\mathcal{R}}(y)$ and $\mathbf{w} = \mathbf{c}_{\mathcal{R}}(z)$.

First, consider whether $\mathbf{u} \lesssim_{\mathcal{R}} \mathbf{w}$. As in Example 2, we have $u_1 \leq w_1$ and the suffixes satisfy $[3, 4] \not\lesssim_{\mathcal{R}} [4, 1]$ because $u_3 > w_3$. Under component-wise dominance, this would be sufficient to conclude non-dominance. However, under rule-dominance, the degradation at r_3 is offset by a strict improvement at a higher-priority rule $r_1 > r_3$ where $u_1 < w_1$. This strict improvement at a more important rule compensates for the degradation at r_3, and therefore $\mathbf{u} \lesssim_{\mathcal{R}} \mathbf{w}$.

Now consider whether $\mathbf{v} \lesssim_{\mathcal{R}} \mathbf{w}$. Again, the suffixes satisfy $[3, 4] \not\lesssim_{\mathbf{R}} [4, 1]$. However, unlike the previous case, there is no strict improvement at any higher-priority rule since $v_1 = w_1$. Since the degradation at r_3 is not compensated by a strict improvement at a more important rule, we conclude that $\mathbf{v} \not\lesssim_{\mathcal{R}} \mathbf{w}$.

Note that in both this example and Example 2, we have $v_1, u_1 \leq w_1$. Under rule-dominance, however, the distinction between a strict improvement ($u_1 < w_1$) and equality ($v_1 = w_1$) is critical. This illustrates why lexicographic ordering alone is insufficient for dimensionality reduction under rule-dominance.

Supporting dimensionality reduction under rule-dominance thus requires maintaining two closed sets for each state s instead of a single set $G_{\mathrm{cl}}(s)$ used in the baseline RA*pex (see Alg. 1) and in A*pex. Specifically, RA*pex maintains $G^{<}_{\mathrm{cl}}$ and $G^{=}_{\mathrm{cl}}$ for local dominance checks at each state. Intuitively, for any rule-apex-realization pair $\mathcal{RP}$, the set $G^{=}_{\mathrm{cl}}(s(\mathcal{RP}))$ stores previously expanded pairs $\mathcal{RP}'$

that end at the same state and have the same f_1-value, i.e., $s(\mathcal{RP}') = s(\mathcal{RP})$ and $f_1(\mathcal{RP}') = f_1(\mathcal{RP})$, while $G^{<}_{\mathrm{cl}}(s(\mathcal{RP}))$ stores those that end at the same state but have strictly smaller f_1-value. Formally, whenever $\mathcal{RP}$ is checked for dominance, it holds that $\forall \mathcal{RP}' \in G^{=}_{\mathrm{cl}}(s(\mathcal{RP})), f_1(\mathcal{RP}') = f_1(\mathcal{RP})$ and $\forall \mathcal{RP}' \in G^{<}_{\mathrm{cl}}(s(\mathcal{RP})) : f_1(\mathcal{RP}') < f_1(\mathcal{RP})$.

With this partitioning, we update the dominance test by introducing the $\mathsf{is\_dominated}_{\mathrm{dr}}$ function presented in Alg. 5 and using it in Alg. 1 Lines 7 and 15. To formalize the dimensionality reduction, we first compute a topological ordering [7] $\langle v_1^{\mathcal{R}}, v_2^{\mathcal{R}}, \ldots, v_M^{\mathcal{R}} \rangle$ of the directed acyclic graph $G_{\mathcal{R}}$. We then obtain an ordering $\langle r_{\sigma(1)}, r_{\sigma(2)}, \ldots, r_{\sigma(N)} \rangle$ of rules such that $r_{\sigma(i)} \not< r_{\sigma(j)}$ for all $i > j$ by simply expanding each equivalence class $[v_i^{\mathcal{R}}]$ in their topological order. Intuitively, this ordering arranges rules in non-increasing priority. Given a rule-value vector $\mathbf{u} = [u_1, \ldots, u_N]$, we reorder its components according to this permutation to obtain $\mathbf{u}_\sigma = [u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(N)}]$. Open is subsequently ordered lexicographically with respect to these reordered rule-value vectors, as in A*pex. For notational simplicity, we henceforth assume that the rules are indexed in topological order, so that $r_i \not< r_j$ for all $i > j$.

Under this convention, r_1 is a highest-priority rule, meaning that there is no rule $r_i \in R$ such that $r_i > r_1$ (r_i may have equal priority as r_1). We then define the *truncated rule set* $Tr(R) := \{r_2, \ldots, r_N\}$, which excludes r_1 as in A*pex. Additionally, we define the *residual rule set* $\alpha(R) := \{r_i \in Tr(R) \mid \neg(r_i < r_1)\}$, which excludes r_1 and any rule that is strictly lower in priority than r_1. By a slight abuse of notation, we use the same operator $Tr(\cdot)$ and $\alpha(\cdot)$ to denote the corresponding operations on rule-value vectors. Specifically, for any $\mathbf{u} \in \mathbb{R}^N$, we define $Tr(\mathbf{u})$ and $\alpha(\mathbf{u})$ as the subvector of $\mathbf{u}$ obtained by keeping only the components corresponding to the rules in $Tr(R)$ and $\alpha(R)$, respectively, with the original order preserved. Alg. 5 uses $Tr(\cdot)$ and $\alpha(\cdot)$ to perform dimensionality reduction-based rule-dominance. Conceptually, the algorithm maintains two state-local closed sets: $G^{=}_{\mathrm{cl}}(s)$ for expanded pairs whose extracted f_1-value equals the current best-known value at s, and $G^{<}_{\mathrm{cl}}(s)$ for expanded pairs with strictly smaller f_1. Whenever the extracted f_1 at s increases (Line 1), all pairs in $G^{=}_{\mathrm{cl}}(s)$ become "strictly better in r_1" than the new pair and are therefore moved into $G^{<}_{\mathrm{cl}}(s)$ (Lines 5–6), after filtering out entries that are already dominated in the residual space $\alpha(\cdot)$.

After this bookkeeping, the dominance test proceeds in three stages. First, it checks for a witness in $G^{=}_{\mathrm{cl}}(s(\mathcal{RP}))$ using only the truncated vector $Tr(\mathbf{f}(\cdot))$ (Line 7): since all candidates have the same f_1-value, rule-dominance cannot rely on a strict improvement in r_1, and we can safely ignore r_1 and compare the remaining rules. Second, it checks $G^{<}_{\mathrm{cl}}(s(\mathcal{RP}))$ in the residual space $\alpha(\cdot)$ (Line 9): here every candidate already has a strict improvement in r_1, so any rule that is strictly lower priority than r_1 is automatically "covered" and can be removed from the comparison. Finally, it performs the global approximate-dominance check against *solutions* exactly as in the baseline algorithm (Line 11).

5 Theoretical Analysis

This section provides a formal guarantee that RA*pex solves Problem 1. Due to space limitations, complete proofs are provided in the supplementary material.

In summary, the correctness of RA*pex hinges on a key structural property of $\boldsymbol{\varepsilon}$-rule-dominance $\lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}}$, namely its one-sided transitivity with respect to exact rule-dominance $\lesssim_{\mathcal{R}}$.

Definition 10. *Let $\precsim$ and $\precsim'$ be two binary relations on a set X. We say that $\precsim'$ is* one-sided transitive with respect to $\precsim$ *if, for any $x, y, z \in X$, $x \precsim' y$ and $y \precsim z$ implies $x \precsim' z$.*

With the definition of $\boldsymbol{\varepsilon}$-rule-dominance given in Definition 5, it can be shown that $\lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}}$ is one-sided transitive with respect to $\lesssim_{\mathcal{R}}$ (see Lemma **??** in the supplementary material). This property is not guaranteed for alternative notions of $\boldsymbol{\varepsilon}$-rule-dominance as demonstrated in the following example.

Example 4. Consider an alternative definition of $\lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}}$: Given realizations $x, y \in \Sigma$ and a vector $\boldsymbol{\varepsilon} \geq \mathbf{0}$, define $x \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} y$ if for any rule $r_j \in R$ such that $r_j(x) > (1+\varepsilon_j) r_j(y)$, there exists a higher-priority rule $r_i > r_j$ such that $r_i(x) < \frac{r_i(y)}{1+\varepsilon_i}$. Intuitively, any degradation at a rule beyond the allowed tolerance must be compensated by a sufficiently strong improvement at a higher-priority rule. The motivation behind this definition is to treat small relative differences at higher-priority rules as negligible under approximation. In particular, if two realizations differ only slightly at a high-priority rule, then differences at lower-priority rules may still be considered meaningful. For example, consider rules $R = \{r_1, r_2\}$ with $r_1 > r_2$ and rule-value vectors $\mathbf{u} = [1, 10^6]$ and $\mathbf{v} = [1.001, 1]$. With $\boldsymbol{\varepsilon} = [0.1, 0.1]$, the relative difference at the higher-priority rule r_1 is within tolerance, while $\mathbf{v}$ is significantly better at the lower-priority rule r_2. Under this alternative definition, we obtain $\mathbf{v} \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{u}$ but $\mathbf{u} \not\lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{v}$, i.e., $\mathbf{v}$ is strictly preferred to $\mathbf{u}$. In contrast, under Definition 5, we have both $\mathbf{v} \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{u}$ and $\mathbf{u} \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{v}$, so $\mathbf{u}$ and $\mathbf{v}$ are treated as equivalent.

Despite this intuition, this alternative definition does not satisfy one-sided transitivity. Consider rules $R = \{r_1, r_2, r_3\}$ with $r_1 > r_2$ and $r_1 > r_3$, and rule-value vectors $\mathbf{u} = [2, 2, 2], \mathbf{v} = [4, 1, 1], \mathbf{w} = [3, 4, 2]$ with $\boldsymbol{\varepsilon} = [1, 1, 1]$. It is easy to check that $\mathbf{w} \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{u}$ (since there is no rule r_j such that $w_j > 2u_j$) and $\mathbf{u} \lesssim_{\mathcal{R}} \mathbf{v}$ (since $u_1 < v_1$). However, based on the alternative definition, $\mathbf{w} \not\lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} \mathbf{v}$ because $w_1 = 3 \not< 0.5 v_1 = 2$.

One-sided transitivity allows us to reason about sets of realizations with ε-bounded rule-apex-realization pairs and ensures that dominance relations are safely propagated across search expansions. The following lemma and the theorem, with $\boldsymbol{\varepsilon}$-rule-dominance defined in Definition 5, show that RA*pex correctly solves Problem 1 (Lemma is numbered 6 to match the supplementary materials).

Lemma 6. *For any prefix $x_l = [s_1, s_2 \dots s_l]$ of any solution*

$$x = [s_1 (= s_{start}), s_2 \dots s_L (= s_{goal})]$$

*with $1 \leq l \leq L$, when the algorithm terminates, there exists either (*Case 1:*) an expanded rule-apex-realization pair $\mathcal{RP}$ (that is, one that reaches Line 9 of the Alg 1) that ends at the state s_l and whose rule apex weakly rule-dominates the $\mathbf{g}$-value of the realization x_l or (*Case 2:*) a rule-apex-realization pair $\mathcal{RP}$ in the solution set such that the $\mathbf{f}$-value of its representative realization $\boldsymbol{\varepsilon}$-rule-dominates the $\mathbf{f}$-value of the realization x_l.*

Proof (sketch). The proof is via induction. The lemma holds for $l = 1$ and any solution since $\mathcal{RP} = \langle \mathbf{0}, [s_{start}] \rangle$ is expanded and satisfies Case 1. Assuming that the lemma holds for l and any solution, we show that it holds for $l + 1$ and this solution x.

Assume that Case 1 holds for l and consider the pair $\mathcal{RP}$ mentioned there and its child pair $\mathcal{RP}'$ created for $s' = s_{l+1}$, where $\mathbf{g}(\mathcal{RP}') \lesssim_{\mathcal{R}} \mathbf{g}(x_{l+1})$ implying that $\mathbf{f}(\mathcal{RP}') \lesssim_{\mathcal{R}} \mathbf{f}(x_{l+1})$. The pair $\mathcal{RP}'$ is either expanded, in which case it satisfies Case 1 for $l + 1$, or is pruned by local or global dominance checks. If it's pruned locally, then there is a realization pair that satisfies Case 1 for $l + 1$. If it's pruned globally, then there is a realization pair in the solution set that satisfies Case 2 for $l + 1$.

Assume that Case 2 holds for l and consider the pair $\mathcal{RP}$ mentioned there. Since the $\mathbf{f}$-value of the representative realization of $\mathcal{RP}$ $\boldsymbol{\varepsilon}$-rule-dominates the $\mathbf{f}$-value of the realization x_l, it also $\boldsymbol{\varepsilon}$-rule-dominates the $\mathbf{f}$-value of x_{l+1}, satisfying the Case 2 for $l + 1$.

Theorem 1. *Upon termination of RA*pex, for every solution $y \in \Sigma_{sol}$, there exists a rule-apex-realization $\mathcal{RP} = \langle \mathbf{r}, x \rangle \in$ solution whose representative path $\boldsymbol{\varepsilon}$-rule-dominates y, i.e, $x \lesssim_{\mathcal{R}}^{\boldsymbol{\varepsilon}} y$.*

Proof (sketch). Lemma 6 holds for prefix $x_L = x$ of any solution x. When Case 2 holds, the theorem holds by definition since the $\mathbf{f}$-values of the solutions are equal to their costs. If Case 1 holds, then the realization pair mentioned there is inserted into the solution set, and the $\mathbf{f}$-value of its representative realization $\boldsymbol{\varepsilon}$-rule-dominates the $\mathbf{g}$-value of x. Thus, again the theorem holds by definition for x since the $\mathbf{g}$- and $\mathbf{f}$-values of the solutions are equal to their costs.

6 Experimental Results

We evaluate the efficiency of RA*pex across three different problem settings that differ in both the presence of an objective hierarchy and the use of approximation. In setting **(S1)** objectives are non-hierarchical while in setting **(S2)** and **(S3)** objectives are hierarchical and we do not / do allow an approximation, respectively. Consequentially, we compare RA*pex against a state-of-the-art algorithm appropriate for each setting: in **(S1)** against A*pex, in **(S2)** against rulebook-based complete control synthesis [19], which we refer to as RB-Exact and in **(S3)** against an approximate extension of RB-Exact, which we refer to as RB-Approx. The experiments were conducted on the BAY roadmap from the 9th DIMACS Implementation Challenge: Shortest Path [1], as well as on randomly generated graphs. All experiments were run on the Macbook Air with an Apple M4 Chip, 24GB of memory, and a five-minute runtime limit per instance. All algorithms were implemented in C++, reusing common code from A*pex and RB-Exact wherever possible.[3]

We used the same 25 generated 3-objective roadmap instances that Zhang et al. [20] used with randomly selected start and goal states for each roadmap. A fourth objective was added by assigning to each edge a value randomly drawn from $\{0, 1\}$. The heuristic function $\mathbf{h}$ was set as a vector, where each component

[3] https://github.com/Infus3d/Rulebook_approximation

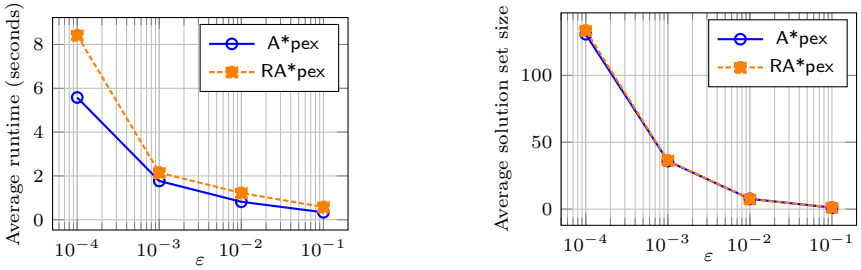

Fig. 1. Comparison of average runtimes (left) and average solution set sizes (right) on 3-objective roadmap instances with 321,270 states and 794,830 edges, for different approximation factors ε.

corresponds to a single objective and is given by the minimum achievable cost from a state s to the goal with respect to that objective, computed using Dijkstra's algorithm. Since computing $\mathbf{h}$ accounted for only a small fraction of the total runtime, all reported runtimes exclude this preprocessing step.

While RA*pex outperforms the existing methods in both settings **(S2)** and **(S3)**, we observed that it performs the best in the setting **(S3)** since the approximated planning takes advantage of the grouping of the similar cost paths in the search frontier, making it up to two orders of magnitude faster than the exact planning in **(S2)** on the 4-objective roadmap instances from the BAY dataset.

Setting (S1): Approximate Planning without Objective Hierarchy

Fig. 1 shows the average runtimes (in seconds) of A*pex and RA*pex over all roadmap instances in the BAY dataset, plotted as a function of the approximation factor. In these experiments, $\boldsymbol{\varepsilon}$ is a vector whose components are set to the same scalar value ε, which is now represented on the x-axis for clarity..

Across all instances, the runtime of RA*pex is within 1.5× the runtime of A*pex while producing approximate solution sets of comparable size. This shows that, in the absence of objective hierarchies, RA*pex incurs only modest overhead relative to A*pex while maintaining similar solution quality.

Setting (S2): Exact Planning with Hierarchical Objectives

Fig. 2 (Left) compares RA*pex and RB-Exact on 3-objective random graph instances of increasing size, reporting both runtime (in seconds) and the size of the returned solution sets. The rule hierarchy is $r_1 > r_2$ and $r_1 > r_3$. On the smallest instances, both algorithms exhibit comparable runtimes. As graph size increases, however, the runtime of RB-Exact grows rapidly, becoming more than two orders of magnitude slower than RA*pex on the largest instances.

Fig. 2 (Right) shows detailed results for 4-objective roadmap instances from the BAY dataset, with rule hierarchy $r_4 > r_1$, $r_4 > r_2$, and $r_4 > r_3$. In all of the instances, RB-Exact failed to finish within five minutes, while RA*pex completed all but the three largest instances. For instances with up to one million explored nodes, RA*pex found the solution set in about a minute, while for instances with up to 100,000 explored nodes, it completed in under 5 seconds.

Setting (S3): Approximate Planning with Hierarchical Objectives

Lastly, we report the performance of RA*pex in the presence of rule hierarchies with non-zero approximation vector $\boldsymbol{\varepsilon}$. Due to the absence of a prior work done

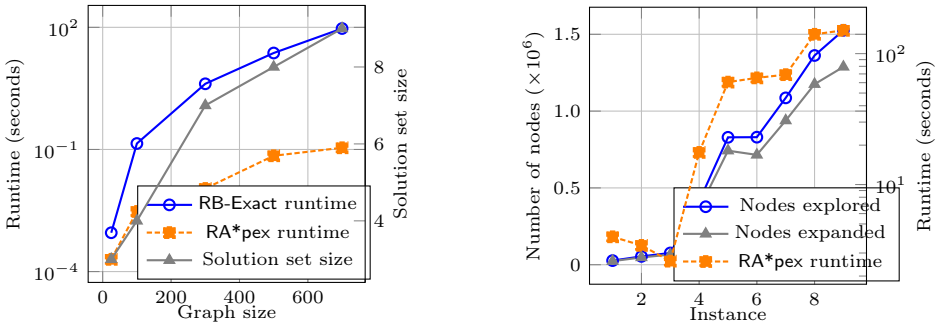

Fig. 2. (Left) Runtimes (in seconds) and sizes of solution sets for RB-Exact and RA*pex on random graphs with 3 rules ($r_1 > r_2, r_3$) under exact dominance (i.e., for $\boldsymbol{\varepsilon = 0}$). (Right) Runtimes (in seconds) for RA*pex on roadmap instances with 4 rules ($r_4 > r_1, r_2, r_3$), along with the number of nodes explored and expanded during the search.

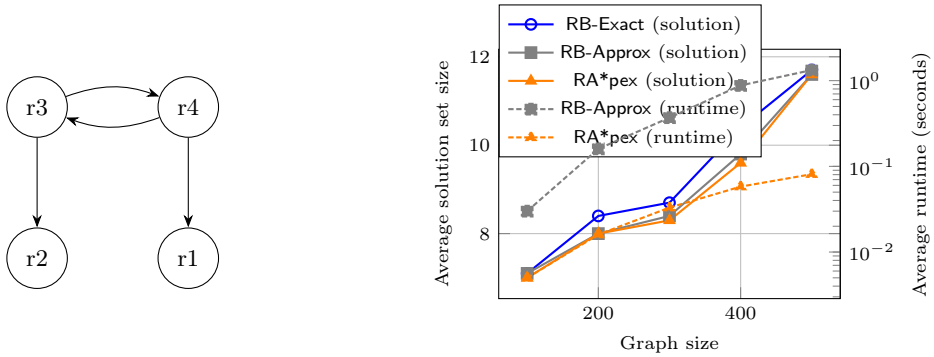

Fig. 3. (Left) A preorder of four rules in **(S3)**). (Right) The average solution set sizes and runtimes for RB-Exact, RB-Approx and RA*pex on random graphs with this rulebook under the approximation $\boldsymbol{\varepsilon} = [0.01, 0.01, 0.01, 0.01]$.

on this, we compared the results against the approximate extension of RB-Exact, where an additional global dominance check was incorporated. In this variant, which we call RB-Approx, whenever a new candidate realization x is constructed, this global dominance check examines whether or not the cost of x is approximately dominated by some realization x' already present in the solution set. If so, then the realization x is discarded since the cost of any realization with the prefix x will also be approximately dominated by x'. Consequently, RB-Approx not only finds an $\boldsymbol{\varepsilon}$-approximate rulebook-optimal solution set but also guarantees that every solution is rulebook-optimal.

For this experiment, we randomly generated graphs of varying sizes with 4 objectives. The rule hierarchy assigns equal importance to r_3 and r_4, while r_1 and r_2 have lower priority than both. Specifically, the hierarchy (illustrated in Fig. 3) is given by $r_3 \lesssim r_4$, $r_4 \lesssim r_3$, $r_2 < r_3$, $r_1 < r_4$. Fig. 3 shows the average runtimes and the solution sizes for RB-Approx and RA*pex. In all of them, RA*pex runs more than an order of magnitude faster than RB-Approx, with the difference growing more and going up to 16× the runtime of RB-Approx on the largest graph. The solution set sizes, however, remain relatively comparable, with RA*pex having a slight advantage on average while RB-Approx being closer

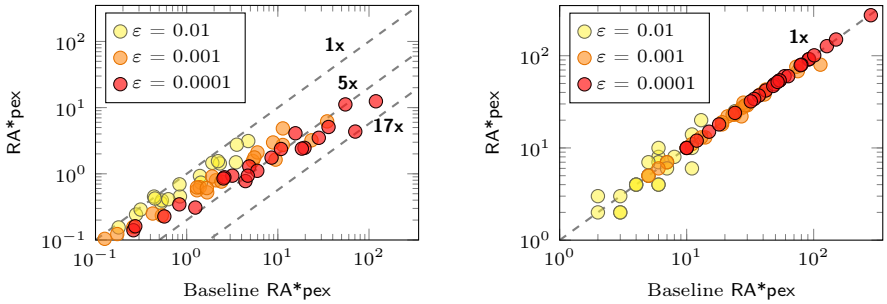

Fig. 4. Individual runtimes (left) and solution sizes (right) of the baseline RA*pex and RA*pex with dimensionality reduction on 4-objective road map instances with 321,270 states and 794,830 edges for varying $\boldsymbol{\varepsilon}$ approximations with the rule hierarchy being $r_1 \sim r_2$, $r_1 > r_3$, $r_2 > r_4$ (similar to Figure 3).

to the rulebook-optimal frontier sizes. This makes sense when we consider that the solution set returned by RB-Approx is also a subset of the rulebook-optimal frontier, and hence might require more solutions from the frontier to cover what a few solutions not on the frontier could approximately cover.

We also evaluate the effect of dimensionality reduction in RA*pex by comparing it against a baseline version that operates in the full objective space. As shown in Figure 4, dimensionality reduction leads to a substantial decrease in runtime, often by an order of magnitude (up to 17×) for better $\boldsymbol{\varepsilon}$ factors, while producing solution sets of comparable size. Although the extent of this improvement depends on the specific rule hierarchy, these results demonstrate that exploiting rulebook structure to reduce effective dimensionality can significantly improve computational efficiency without sacrificing solution quality.

7 Conclusions

We presented RA*pex, an approximate multi-objective search algorithm that leverages the rulebook formalism to precisely express complex relationships among objectives, while providing provable approximation guarantees. RA*pex generalizes A*pex to settings where objectives are partially ordered by priority. A key theoretical contribution of this work is the identification of one-sided transitivity as a crucial structural property for approximate rule-dominance. We showed that the proposed definition of $\boldsymbol{\varepsilon}$-rule-dominance satisfies one-sided transitivity with respect to exact rule-dominance, which allows dominance relations to be safely propagated across search expansions and establish the correctness of RA*pex.

Our experimental results evaluated RA*pex across three problem settings that vary in the presence of objective hierarchy and the use of approximation. Across all settings, RA*pex consistently matched or significantly outperformed specialized state-of-the-art algorithms. Overall, RA*pex bridges the gap between exact rulebook-based planning and approximate multi-objective search, offering a principled and efficient approach to approximate multi-objective search under rulebooks. We believe this approach opens the door to scalable planning and control synthesis in complex systems where objectives are partially ordered and exact optimality is neither required nor tractable.

This work was supported in part by NSF under Grant CNS-2141153.

References

1. DIMACS implementation challenge: Shortest paths. https://www.diag.uniroma1.it/challenge9/download.shtml
2. Ahmadi, S., Tack, G., Harabor, D., Kilby, P.: Bi-objective search with bi-directional A*. In: European Symposium on Algorithms (ESA). LIPIcs, vol. 204, pp. 3:1–3:15 (2021)
3. Breugem, T., Dollevoet, T., van den Heuvel, W.: Analysis of fptases for the multi-objective shortest path problem. Computers & Operations Research **78**, 44–58 (2017)
4. de las Casas, P.M., Sedeño-Noda, A., Borndörfer, R.: An improved multiobjective shortest path algorithm. Comput. Oper. Res. **135**, 105424 (2021)
5. de las Casas, P.M., Kraus, L., Sedeño-Noda, A., Borndörfer, R.: Targeted multi-objective dijkstra algorithm. Networks **82**(3), 277–298 (2023)
6. Censi, A., Slutsky, K., Wongpiromsarn, T., Yershov, D., Pendleton, S., Fu, J., Frazzoli, E.: Liability, ethics, and culture-aware behavior specification using rulebooks. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8536–8542 (2019)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, 4th edn. (2022)
8. Ehrgott, M.: Multicriteria Optimization (2nd ed.). Springer (2005)
9. Goldin, B., Salzman, O.: Approximate bi-criteria search by efficient representation of subsets of the Pareto-optimal frontier. In: International Conference on Automated Planning and Scheduling (ICAPS). pp. 149–158 (2021)
10. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
11. Hernández, C., Yeoh, W., Baier, J.A., Zhang, H., Suazo, L., Koenig, S., Salzman, O.: Simple and efficient bi-objective search algorithms via fast dominance checks. Artificial intelligence **314**, 103807 (2023)
12. Mandow, L., De La Cruz, J.L.P.: A new approach to multiobjective A* search. In: International Joint Conferences on Artificial Intelligence (IJCAI). pp. 218–223 (2005)
13. Mandow, L., De La Cruz, J.L.P.: Multiobjective A* search with consistent heuristics. J. ACM **57**(5), 1–25 (2010)
14. Peer, H., Weiss, E., Alterovitz, R., Salzman, O.: Generalizing multi-objective search via objective-aggregation functions. CoRR **abs/2509.22085** (2025)
15. Pulido, F.J., Mandow, L., Pérez-de-la-Cruz, J.L.: Dimensionality reduction in multiobjective shortest path search. Computers & Operations Research **64**, 60–70 (2015)
16. Ren, Z., Hernández, C., Likhachev, M., Felner, A., Koenig, S., Salzman, O., Rathinam, S., Choset, H.: EMOA*: A framework for search-based multi-objective path planning. Artificial intelligence **339**, 104260 (2025)
17. Salzman, O., Felner, A., Hernández, C., Zhang, H., Chan, S., Koenig, S.: Heuristic-search approaches for the multi-objective shortest-path problem: Progress and research opportunities. In: International Joint Conferences on Artificial Intelligence (IJCAI). pp. 6759–6768 (2023)
18. Slutsky, K., Yershov, D., Wongpiromsarn, T., Frazzoli, E.: Hierarchical multiobjective shortest path problems. In: LaValle, S.M., Lin, M., Ojala, T., Shell, D., Yu, J. (eds.) Algorithmic Foundations of Robotics XIV. pp. 261–276. Springer International Publishing, Cham (2021)

19. Wongpiromsarn, T., Slutsky, K., Frazzoli, E.: Formal specification and control synthesis of autonomous robots using rulebooks. IEEE Trans. Robotics (2026), to appear, included in supplementary material
20. Zhang, H., Salzman, O., Kumar, T.S., Felner, A., Ulloa, C.H., Koenig, S.: A*pex: Efficient approximate multi-objective search on graphs. In: International Conference on Automated Planning and Scheduling (ICAPS). vol. 32, pp. 394–403 (2022)