Geometry-Aware MCTS for Extremal Problems in Combinatorial Geometry

Anonymous authors
Paper under double-blind review

Abstract

We study certain extremal problems in combinatorial geometry that ask about configurations of points in an $n \times n$ grid that satisfy strict, global geometric constraints. Classical exact solvers suffer from combinatorial explosion for these types of problems, and standard reinforcement learning and transformer-based models struggle with the sparse reward "validity cliff" and quadratic token-consumption limits. To overcome these bottlenecks, we propose a Geometry-Aware Monte Carlo Tree Search (MCTS) framework. Our approach strictly enforces geometric constraints through incremental updates to the feasible action space. For constraints about collections of collinear points, like those that occur in the classic No-Three-in-Line problem (Max-N3IL), this mechanism reduces the constraint checking complexity from $O(n^4)$ to $O(n^2)$. To improve search efficiency, we exploit geometric symmetries in two ways: canonical pruning during node expansion to reduce the branching factor, and symmetric batch transitions to accelerate the discovery of promising configurations. We perform extensive experiments and establish new best-known computational results on five out of six of the problems that we considered. Notably, for Max-N3IL we find configurations of size roughly $1.8n$ for grids of size $82 \leq n \leq 119$. For the Smallest Complete Set problem, we find configurations of size roughly $0.95n$, providing new upper bounds within the tested grids. This work establishes Geometry-Aware MCTS as a highly adaptable framework for discovering novel configurations in combinatorial geometry.

1 Introduction

We study certain extremal problems on $n \times n$ grids. These problems ask about the maximum or minimum number of points in the grid that satisfy certain geometric constraints. Representative examples include finding the largest point sets with no isosceles triangles, the largest point sets with no four points on a circle, or the smallest independent geometric dominating set. The No-Three-in-Line Problem (Max-N3IL) has attracted considerable attention, both as a theoretical problem and as an algorithmic one. This problem has a history going back at least to Dudeney (1917). It asks for the maximum number of points in an $n \times n$ grid such that no 3 points are collinear. We denote the answer to this problem by $M(n)$. Some configurations in a 3×3 grid are given in fig. 2.

Despite its elementary formulation, determining $M(n)$ remains an open challenge for large n. Since n parallel lines cover the $n \times n$ grid and we cannot have more than two points on any of these lines, we see that $M(n) \leq 2n$. Flammenkamp (1992; 1998) has shown that $M(n) = 2n$ for all $n \leq 46$ and for $n \in \{48, 50, 52\}$. Prellberg (2026) has very recently used advanced tools from constraint satisfaction programming to extend these results to show that $M(n) = 2n$ for all $n \leq 64$ and for $n \in \{66, 68\}$. These results are actively tracked in Flammenkamp's online database (2026). We do not know the exact value of $M(n)$ for $n \in \{65, 67\}$ or for any $n \geq 69$. The best-known asymptotic lower bound is due to Hall et al. (1975), who use a construction coming from finite fields to show that for any $\epsilon > 0$ and for all sufficiently large n, $M(n) > (1.5 - \epsilon)n$. This leaves a significant gap between the constructive $(1.5-\epsilon)n$ lower bound and the trivial $2n$ upper bound. Guy & Kelly (1968) used probabilistic arguments to suggest that as n goes to infinity, $M(n) \approx \frac{\pi}{\sqrt{3}}n \approx 1.814n$. In order to better understand the growth of $M(n)$, we would like to probe grid sizes that have historically been inaccessible to computation because of the combinatorial explosion in the search space.

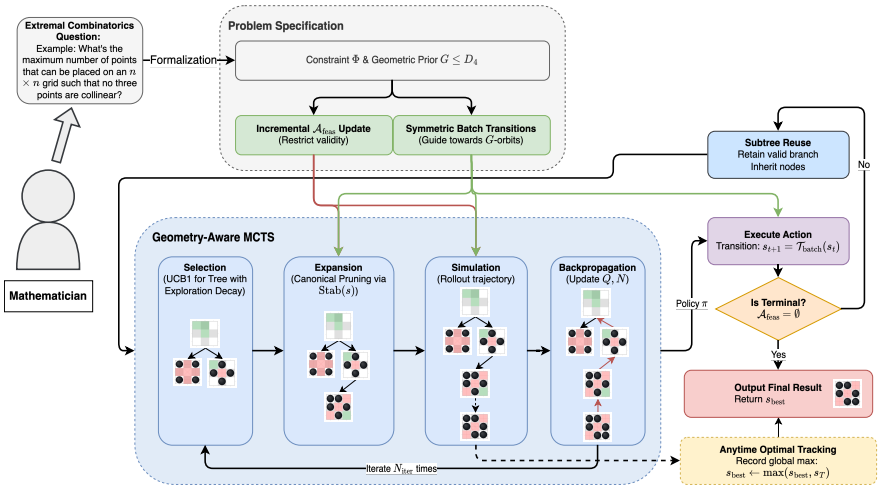
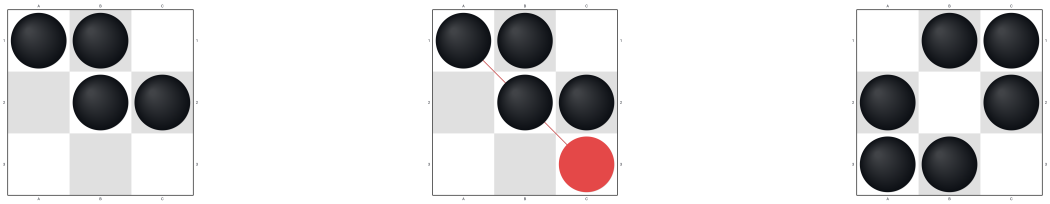


Figure 1: **Overview of the Geometry-Aware MCTS framework.** The search process is guided by problem specification (grey) including a Feasible Action Space defined by the specific problem constraint and Symmetric Batch Transitions (e.g., rotations and reflections) derived from a geometric prior. The core MCTS engine (blue) utilizes Canonical Pruning to reduce branching during expansion. The resulting policy dictates the main loop (right), which employs Subtree Reuse for efficiency and Anytime Optimal Tracking to asynchronously record the global best-found configuration across all simulations.



(a) An example configuration with no three collinear points.

(b) An example configuration with three collinear points (diagonal line).

(c) A configuration with no three collinear points of maximum size.

Figure 2: Three examples of point configurations on a 3x3 grid.

The complexity of these extremal problems on finite grids arises from the global nature of the geometric constraints. A single point placement affects the set of remaining points in complex ways, making local constraint satisfaction insufficient. While algebraic constructions provide rigorous guarantees, it is not at all clear whether we should expect extremal configurations to have algebraic structure. This motivates a need for computational methods to better understand grids that have previously been inaccessible.

In recent years, AI has achieved remarkable breakthroughs in combinatorial mathematics, from discovering graph theory counterexamples (Wagner, 2021) to predicting modular curve invariants (Zhuang et al., 2025). Applying these methods has not yet been successful on problems like Max-N3IL. A recent systematic evaluation by Ramanathan et al. (2025) highlighted severe scalability limits across three distinct computational paradigms: exact solvers, reinforcement learning, and transformer-based models. Traditional exact solvers, such as Integer Linear Programming (ILP), guarantee optimal solutions but hit a scalability wall at $n = 19$ due to the $O(n^3)$ constraint density and exponential search space. Conversely, standard model-free Reinforcement Learning (e.g., PPO) fails even earlier, finding optimal configurations only up to 10×10 grids before systematically failing at $n = 11$ (Ramanathan et al., 2025). Generic RL agents struggle to maintain global geometric constraint awareness solely from scalar reward signals; a single collinear placement renders the entire configuration invalid, creating a sparse-reward "validity cliff" that frustrates gradient-based optimization. While Prellberg (2026) employs constraint satisfaction programming to search for configurations

of size exactly $2n$, this incurs prohibitive computational costs, demanding 10 CPU days for $n \leq 60$ and extensive GPU parallelization to merely reach $n = 66$ (Flammenkamp, 2026).

More recently, advanced AI paradigms like Large Language Models (FunSearch, Romera-Paredes et al., 2024; AlphaEvolve, Novikov et al., 2025; and MCTS-AHD, Zheng et al., 2025) and Transformer-based search (PatternBoost, Charton et al., 2024) have discovered impressive new constructions for similar extremal problems, for example the famous Cap Set Problem about points in $(\mathbb{Z}/3\mathbb{Z})^n$. These models have not yet been successful when applied to the problems we consider here. When PatternBoost is applied to study Max-N3IL, it matches optimal performance on small grids but fails to scale beyond $n = 15$ (Ramanathan et al., 2025). This specific scalability wall arises because the transformer's pattern generalization is severely bottlenecked by the quality-speed trade-off inherent in the greedy algorithms used to generate its training data. Beyond this data generation bottleneck, transformer-based models are intrinsically highly *token-consuming* and computationally expensive. Representing an $n \times n$ grid requires $O(n^2)$ tokens, leading to attention mechanisms that scale poorly for larger grids. Furthermore, autoregressive generation often struggles to strictly adhere to *hard* global constraints without hallucinating invalid placements. LLM-driven methods like *FunSearch* excel at finding concise heuristics, but it may be the case that optimal configurations on large grids are highly irregular and difficult to express using simple rules. Furthermore, potentially constrained by their training distributions, these LLMs may exhibit limitations in the out-of-distribution (OOD) algorithmic generalization required to discover genuinely novel logic for unsolved problems in this area (Li & Ellis, 2024; Neuhaus et al., 2026). These limitations necessitate a different algorithmic paradigm that combines strict geometric constraint satisfaction with scalable search.

Why MCTS: A Geometry-Aware Approach To overcome the limitations of classical solvers and recent learning-based methods for problems like Max-N3IL, we propose a **Geometry-Aware MCTS** framework (Coulom, 2007; Kocsis & Szepesvári, 2006). Monte Carlo Tree Search (MCTS) avoids the sparse-reward "validity cliff" of standard RL by using simulated trajectories to provide dense, constructive reward signals. Furthermore, it effortlessly enforces hard geometric constraints by masking invalid placements, bypassing both the hallucination risks and $O(n^2)$ token overhead of sequence models. While the asymmetric tree growth of MCTS naturally balances exploration and exploitation in unstructured spaces, scaling it to large grids (e.g., $n > 50$) requires specific adaptations. Our framework makes this tractable by (1) incrementally maintaining a strict feasible action space ($\mathcal{A}_{\text{feas}}$) to minimize per-node overhead, (2) exploiting D_4 grid symmetry during expansion and simulation to reduce branching, and (3) augmenting the search strategy with information persistence, adaptive exploration, and global optimum tracking to maximize yield under a finite computational budget.

Main Contributions Our work advances the intersection of combinatorial geometry and reinforcement learning by demonstrating the efficacy of search-based paradigms. By formulating extremal construction as a geometric Markov Decision Process (MDP), we introduce specific algorithmic innovations to make MCTS tractable in vast, constraint-heavy spaces. The specific contributions are as follows:

- We propose an *Incremental Feasible Action Space* ($\mathcal{A}_{\text{feas}}$) maintenance mechanism. For problems based on constraints about collinear sets of points, such as Max-N3IL, this ray-casting approach reduces the per-node complexity for checking constraints from $O(n^4)$ to $O(n^2)$. This effectively bypasses the sparse-reward "validity cliff" that typically paralyzes standard RL agents.
- We introduce *Canonical Action Pruning* via dynamically computed state stabilizers $\text{Stab}(s) \leq D_4$, restricting tree expansion to orbit representatives while deliberately disabling pruning during rollouts to minimize computational overhead without sacrificing branching reduction.
- We propose *Symmetric Batch Transitions*, a novel MDP transition mechanism that executes the simultaneous placement of an orbit under a chosen collection of symmetries. This significantly accelerates the discovery of promising symmetric configurations and allows canonical pruning to remain effective deeper into the search tree.
- We improve the best-known computational bounds for five out of the six problems that we considered. Notably, we significantly extend the range in which we have lower bounds for Max-N3IL that are

better than the asymptotic lower bound of size approximately $1.5n$. We also find configurations giving upper bounds for the Smallest Complete Set problem that are smaller than the best-known construction.

2 Problem Formulation

We study extremal subset construction on the discrete grid $\mathcal{G}_n = [n] \times [n]$ under hard geometric constraints. Unlike in probabilistic generation tasks, a construction is either valid or invalid according to a binary predicate Φ (e.g., "no three points are collinear"). We therefore cast the process as a sequential decision problem, enabling search and planning methods Vodopivec et al. (2017).

2.1 A General Geometric MDP Framework

We formulate geometric construction sequentially as a deterministic MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. Although the target configuration is permutation-invariant, this sequential formulation is computationally advantageous because it enforces hard constraints at every step, immediately pruning invalid branches. The state space $\mathcal{S} = \{s \mid s \subseteq \mathcal{G}_n\}$ is the power set of the grid, where a state $s \in \mathcal{S}$ represents a configuration of points, initialized at the empty set $s = \emptyset$. To maintain structural validity, we define a Boolean geometric invariant $\Phi : \mathcal{S} \to \{\text{True}, \text{False}\}$ (e.g., for Max-N3IL, $\Phi(s)$ holds iff no three points are collinear). Consequently, the action space is strictly confined to feasible placements, defined as $\mathcal{A}_{\text{feas}}(s) = \{p \in \mathcal{G}_n \setminus s \mid \Phi(s \cup \{p\}) = \text{True}\}$. This ensures that any action taken preserves Φ (efficient incremental maintenance of $\mathcal{A}_{\text{feas}}$ is detailed in Section 3.2). Transitions are deterministic and additive, given by $s' = s \cup \{a\}$ for $a \in \mathcal{A}_{\text{feas}}(s)$. A state s is terminal (denoted as s_T) if and only if $\mathcal{A}_{\text{feas}}(s) = \emptyset$, meaning the configuration is maximal under inclusion. Note that just because a configuration is maximal does not mean that it has optimal cardinality. Finally, the reward function $R(s_T, n)$ is evaluated at terminal states. For maximization variants (e.g., Max-N3IL), R increases with $|s_T|$; for minimization variants (e.g., Min-Complete), it decreases with $|s_T|$. Detailed formulations are given in Appendix C.1.

2.2 Problem Variants

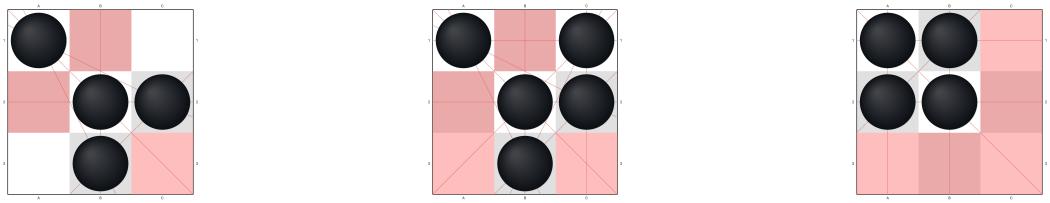
We apply the proposed MDP framework to six extremal combinatorial geometry problems. For each problem, the geometric constraints are strictly governed by the state validity predicate $\Phi(s)$, and the objective is to either maximize or minimize the terminal set cardinality $|s_T|$. Let $P \subseteq s$ denote an arbitrary subset of the current state.

Collinear Constraints: Max-N3IL and Min-Complete. Both problems share the same predicate enforcing general position: $\Phi(s) = \{\forall P \subseteq s \text{ with } |P| = 3, \neg\text{Coll}(P)\}$, where $\text{Coll}(P)$ is true if the points in P are collinear. Because our MDP naturally terminates only at inclusion-wise maximal states ($\mathcal{A}_{\text{feas}} = \emptyset$), any terminal configuration is inherently a "complete" set. As illustrated in fig. 3, the search stops exactly when all remaining empty cells lie on a line connecting two points already included in the set (figs. 3b and 3c). Max-N3IL seeks to maximize $|s_T|$ (Dudeney, 1917), whereas Min-Complete (a.k.a. Smallest Independent Geometric Dominating Set) seeks to minimize $|s_T|$ (Aichholzer et al., 2023).

Smallest Geometric Dominating Set (Min-Dom). This problem is like Min-Complete, except that we drop the requirement that no three points of the configuration are collinear. Let $\text{Cover}(X)$ be true if the lines spanned by all pairs of distinct points in X collectively cover every point in the grid $\mathcal{G}_n$ (i.e., every grid point is either in X or collinear with at least two distinct points in X). The predicate is defined as $\Phi(s) = \{\forall s' \subsetneq s, \neg\text{Cover}(s')\}$. If s is not a dominating set, then no subset of s is a dominating set. If s is a dominating set, then any superset of s is also a dominating set. In this way, we see that the search terminates exactly when a minimal dominating set is found. The objective is to minimize $|s_T|$ (Aichholzer et al., 2023).

Additional Problems. To demonstrate the flexibility of our framework, we explore three additional problems. In each one, we aim to maximize $|s_T|$.

1. **Max-N4IL**: $\Phi(s) = \{\forall P \subseteq s \text{ with } |P| = 4, \neg\text{Coll}(P)\}$, a natural relaxation of N3IL.
2. **Max-No-4-on-Circle**: $\Phi(s) = \{\forall P \subseteq s \text{ with } |P| = 4, \neg\text{Circ}(P)\}$, where $\text{Circ}(P)$ indicates the points are concyclic on a circle of finite radius (Dong & Xu, 2025).
3. **Max-No-Isosceles**: $\Phi(s) = \{\forall P \subseteq s \text{ with } |P| = 3, \neg\text{Isos}(P)\}$, where $\text{Isos}(P)$ is true if at least two Euclidean distances between points in P are equal, including degenerate cases (i.e., collinear triples where one point is the midpoint of the segment).



(a) This set is not complete. Either of the two white squares could be added without creating a collinear triple.

(b) A complete set on a 3×3 grid.

(c) A complete set on a 3×3 grid of minimum size.

Figure 3: Examples of incomplete and complete point configurations.

3 Method: Geometry-Aware MCTS

Monte Carlo Tree Search (MCTS) has demonstrated remarkable versatility and success across diverse domains, ranging from significantly improving real-time decision-making in heuristic-free general video game playing (Soemers et al., 2016), to enabling effective test-time scaling for Large Language Models (Wang et al., 2025). However, it is challenging to apply off-the-shelf MCTS to extremal problems in combinatorial geometry because of the strictness of the constraints and the $O(n^2)$ increase in the size of the action space. In this section, we first define the standard baseline and analyze its limitations, then introduce our Geometry-Aware framework.

3.1 Baseline MCTS and the Naive Action Space

We adopt the UCT (Upper Confidence bounds applied to Trees) algorithm (Kocsis & Szepesvári, 2006) as our baseline. During the selection phase, the agent traverses the tree by selecting actions that maximize the UCB1 (Upper Confidence Bound 1) criterion:

$$\text{UCB1}(s, a) = \frac{Q(s, a)}{N(s, a)} + C\sqrt{\frac{\ln N(s)}{N(s, a)}}, \tag{1}$$

where $Q(s, a)$ is the accumulated reward, $N(s, a)$ is the visit count, and C governs exploration. Since our MDP transitions are deterministic, $Q(s, a)$ simplifies to the value of the unique successor state, $V(s'|s, a)$, and $N(s, a)$ precisely equals the visit count of that child node. After executing the standard MCTS phases (selection, expansion, simulation, and backpropagation) for a computational budget H, the search policy selects the most visited action: $a = \text{argmax}_{a \in \mathcal{A}(s)} N(s, a)$.

Naive Action Space and Dual Bottlenecks. Lacking domain-specific geometric constraints, the baseline agent considers any unoccupied grid cell as a potential move. This defines the **Naive Action Space**:

$$\mathcal{A}_{\text{naive}}(s) = \mathcal{G}_n \setminus s. \tag{2}$$

Operating on $\mathcal{A}_{\text{naive}}$ introduces two fundamental scalability bottlenecks. First, the **combinatorial branching factor** ($|\mathcal{A}_{\text{naive}}| \approx n^2$) severely dilutes visit counts at the root (e.g., nearly 5,000 actions for $n = 70$),

rendering UCB1 estimates high-variance. Second, the **validity bottleneck** arises because the density of valid moves decays rapidly. A rollout policy selecting uniformly from $\mathcal{A}_{\text{naive}}$ almost certainly violates the invariant Φ, triggering immediate termination with sub-optimal reward. Consequently, discovering deep, valid configurations by chance becomes mathematically intractable for large n.

3.2 Feasible Action Space via Incremental Maintenance

To resolve the validity bottleneck described in Section 3.1, our framework restricts the search strictly to the **Feasible Action Space** ($\mathcal{A}_{\text{feas}}$) that is described in Section 2.1. This ensures that any action does not violate Φ. Although $\mathcal{A}_{\text{feas}}(s)$ was defined conceptually in Section 2, computing it from scratch at every search node is computationally expensive. For the Max-N3IL problem, verifying a single candidate point requires checking collinearity against all $\binom{|s|}{2}$ pairs of points in s. Since $|s| \leq 2n$, this check incurs $O(n^2)$ operations per candidate. When aggregated over all $O(n^2)$ potential locations on the grid, the computational cost for a full validity scan scales to $O(n^4)$, which becomes prohibitive for large n.

Monotonicity and Incremental Updates. We leverage the property of *constraint monotonicity*. In our target problems, placing a point never "frees up" a previously blocked location—it only imposes new constraints. Thus, $\mathcal{A}_{\text{feas}}(s') \subset \mathcal{A}_{\text{feas}}(s)$ for any successor state s'. Instead of recomputing constraints from scratch, upon transitioning $s \to s \cup \{p_{new}\}$, we update $\mathcal{A}_{\text{feas}}$ by solely identifying cells invalidated by the new point:

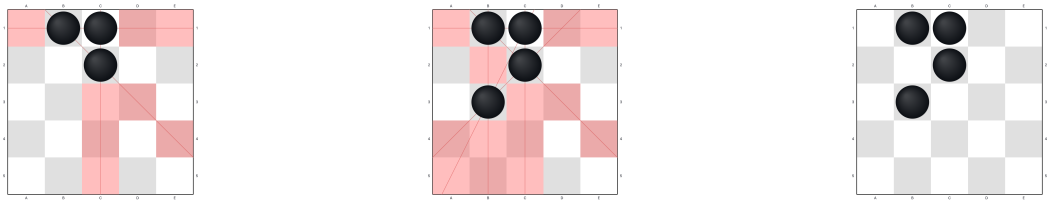
$$\mathcal{A}_{\text{feas}}(s \cup \{p_{new}\}) = \mathcal{A}_{\text{feas}}(s) \setminus (\{p_{new}\} \cup \text{NewConstraints}(p_{new}, s)) \,. \tag{3}$$

Example: Ray Casting for Max-N3IL. For the Max-N3IL problem, this update takes the form of *Ray Casting*. As illustrated in Figure 4, placing a new point p_{new} creates lines with each existing point $p_i \in s$. We write $\text{Ray}(u, v)$ for the set of points in the grid on the line connecting u and v. The update rule becomes:

$$\text{NewConstraints}(p_{new}, s) = \bigcup_{p_i \in s} \text{Ray}(p_{new}, p_i). \tag{4}$$

Computing a discrete line (e.g., via GCD of coordinates) and marking its cells takes $O(n)$ operations, and there are $|s| \leq 2n$ existing points. Therefore, the total update complexity is $O(n^2)$. This represents a significant improvement over the $O(n^4)$ baseline, enabling deeper search within the same time budget.

Furthermore, the strict enforcement of geometric constraints naturally prunes the search tree. As illustrated by comparing Figure 4b and Figure 4c, the size of the feasible action space $|\mathcal{A}_{\text{feas}}|$ shrinks drastically as more points are placed, whereas $|\mathcal{A}_{\text{naive}}|$ only decreases by one per step. This effectively mitigates the **combinatorial branching factor** bottleneck identified in Section 3.1, allowing the UCB estimates to converge significantly faster on the remaining highly-promising branches.



(a) State s with 3 points. Red cells indicate unoccupied points that are not in $\mathcal{A}_{\text{feas}}$.

(b) State s' (4 points). Ray casting incrementally updates $\mathcal{A}_{\text{feas}}$.

(c) The naive action space $\mathcal{A}_{\text{naive}}$ for the same 4 points (only occupied cells are illegal).

Figure 4: Visualizing the Incremental Ray Casting update and comparing $\mathcal{A}_{\text{feas}}$ with $\mathcal{A}_{\text{naive}}$ on a 5×5 grid.

3.3 Exploiting Grid Symmetry

The $n \times n$ grid inherently possesses geometric symmetries described by the dihedral group D_4, which consists of eight isometric transformations (four rotations and four reflections). Because the validity predicate Φ is

invariant under these transformations, the MDP induces highly redundant symmetric branches during tree search. We exploit this symmetry through two complementary mechanisms: reducing the search width via state-dependent canonical pruning, and accelerating the search depth via global symmetric batch transitions.

3.3.1 Canonical Pruning via State Stabilizers

To mitigate the combinatorial branching factor during the *Expansion* phase of MCTS, we dynamically prune geometrically equivalent actions based on the symmetry of the current state. Let $\mathrm{Stab}(s) \leq D_4$ denote the stabilizer subgroup of a valid state s, defined as the set of transformations that leave the state configuration unchanged:

$$\mathrm{Stab}(s) = \{g \in D_4 \mid g \cdot s = s\}. \tag{5}$$

The group action of $\mathrm{Stab}(s)$ partitions the feasible action space $\mathcal{A}_{\mathrm{feas}}(s)$ into equivalence classes, or *orbits*. For any candidate action $a \in \mathcal{A}_{\mathrm{feas}}(s)$, its orbit under the stabilizer is denoted by $\mathcal{O}_{\mathrm{Stab}(s)}(a) = \{g \cdot a \mid g \in \mathrm{Stab}(s)\}$. Because transitioning to any action within the same orbit leads to isomorphic subtrees, we define a canonical pruning mechanism that restricts the expansion to exactly one representative action per orbit:

$$\mathcal{A}_{\mathrm{pruned}}(s) = \{\mathrm{rep}(\mathcal{O}_{\mathrm{Stab}(s)}(a)) \mid a \in \mathcal{A}_{\mathrm{feas}}(s)\}, \tag{6}$$

where $\mathrm{rep}(\cdot)$ is a deterministic tie-breaking function, such as selecting the action with the lexicographically smallest coordinate index.

Crucially, this canonical pruning is applied *exclusively* during node expansion. We intentionally disable it during the *Simulation* (rollout) phase. Since a simulation merely generates a single random trajectory to estimate the value function, computing stabilizers and orbits at each simulation step incurs severe computational overhead without the benefit of permanent branching factor reduction. The impact of this pruning is most impactful at the beginning of the search; for instance, at the empty root node where $|\mathrm{Stab}(\emptyset)| = 8$, the initial branching factor is immediately reduced by nearly 7/8 (as visualized in fig. 5a).

3.3.2 Symmetric Batch Transitions

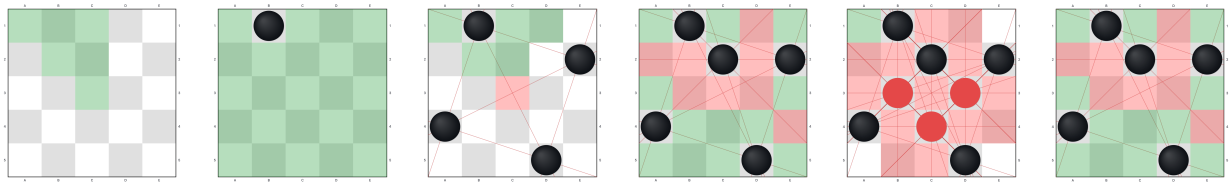
While canonical pruning optimizes the tree width, empirical observations on the problems we tested suggest that optimal configurations frequently possess symmetries (Flammenkamp, 1992; 1998; 2026; Prellberg, 2026; Aichholzer et al., 2023). To incorporate this structural prior, we introduce *Symmetric Batch Transitions*, which fundamentally redefines the MDP transition function to accelerate search depth and guide the agent toward highly symmetric subspaces.

Before the search begins, we pre-select a group of symmetries $G \leq D_4$ (e.g., the rotational group C_4). When an action $a \in \mathcal{A}_{\mathrm{feas}}$ is selected by the agent (either in the tree or during rollout), we compute its orbit under the action of G, denoted as $\mathcal{O}_G(a) = \{g \cdot a \mid g \in G\}$. We then evaluate whether these points can be placed simultaneously without violating the geometric invariant. The modified transition function $\mathcal{T}_{\mathrm{batch}}$ follows an "all-or-nothing" rule:

$$\mathcal{T}_{\mathrm{batch}}(s) = \begin{cases} s \cup \mathcal{O}_G(a) & \text{if } \Phi(s \cup \mathcal{O}_G(a)) = \text{True}, \\ s \cup \{a\} & \text{otherwise.} \end{cases} \tag{7}$$

The batch transition is executed internally as a sequence of incremental additions, ensuring that each point is validated against the updated feasible action space $\mathcal{A}_{\mathrm{feas}}$. As a result, the agent commits the entire orbit $\mathcal{O}_G(a)$ in a single MDP transition if mutually compatible (figs. 5b and 5c); otherwise, it gracefully falls back to placing only the single action a (figs. 5d to 5f).

This batching mechanism operates throughout the entire MCTS pipeline, drastically accelerating terminal state evaluations during simulations. Furthermore, it creates a powerful synergy with the canonical pruning described in Section 3.3.1. Batch transitions are designed to generate states that preserve the global symmetry G deeper within the search tree, provided the fallback is not triggered. Consequently, canonical pruning remains highly effective at reducing $|\mathcal{A}_{\mathrm{feas}}|$ at deeper search depths.



(a) Pruned $\mathcal{A}_{\text{feas}}$ on an empty grid. (b) Place a point and attempt batch actions under C_4. (c) Batch transition successful. (d) Place another point and try batch action under C_4. (e) Constraint violated (red points created collinear triples). (f) Fallback to single action.

Figure 5: Visualization of Canonical Pruning via State Stabilizers (green-shaded spaces are pruned $\mathcal{A}_{\text{feas}}$) and Symmetric Batch Transitions (succeeded from action b to state c but failed from action d to state e). Note that canonical pruning is effective on states a and c.

3.4 Enhanced Search Strategy

To further maximize the performance of Geometry-Aware MCTS under a finite computational budget, we adopt and adapt three targeted optimization strategies. These enhancements focus on *information persistence*, *dynamic exploration adjustment*, and *global result harvesting*, transforming the standard MCTS into a more efficient solver for deterministic combinatorial tasks.

3.4.1 Subtree Reuse (Information Persistence)

In standard MCTS implementations, the search tree is typically discarded after each decision step, resetting the agent's knowledge to zero. Given that our MDP transitions are deterministic, discarding the tree wastes significant computational effort. We therefore adopt a **Subtree Reuse** strategy (Silver et al., 2016). Upon executing an action a and transitioning from the current root s to a child state s', we do not re-initialize the tree. Instead, we promote the child node corresponding to s' to become the new root for the subsequent search phase. To maintain memory efficiency, the previous root and its unselected branches are detached and garbage-collected. This strategy ensures that the new search phase begins with a "warm start," where $N_{\text{existing}}(s') > 0$, allowing the agent to immediately focus on deeper lookaheads rather than re-exploring the immediate action space.

3.4.2 Time-Dependent Exploration Decay

A critical challenge in the extremal problems we study is the *small suboptimality gap*: the value difference between optimal and near-optimal actions is often extremely small. Even with exponential reward shaping, the exploration term $C\sqrt{\ln N(s)/N(s,a)}$ in the UCB1 formula (1) can dominate the exploitation term $Q(s,a)/N(s,a)$, hindering convergence. To address this, we implement a time-dependent exploration decay mechanism (Ruijl et al., 2013), modulating the exploration constant via a decay function $\lambda(r)$: $C(t) = C_{\text{base}} \cdot \lambda(r)$, where $\lambda(r) = 1 - 0.85\sqrt{r}$. Here, $r = i/N_{\text{total}} \in [0,1]$ represents the progress ratio of the search, where i is the current iteration index and N_{total} is the allocated search budget. A critical design choice is formulating r under the Subtree Reuse paradigm. If r is computed relative only to the newly allocated budget for the current step, $C(t)$ resets to its maximum at every transition, destroying accumulated confidence. Instead, we incorporate the inherited visit count $N_{\text{existing}}(s')$ from the previous tree as valid prior exploration:

$$r = \frac{N_{\text{existing}}(s') + i}{N_{\text{total}}}. \tag{8}$$

This formulation ensures the exploration factor $C(t)$ remains continuous and monotonically decreasing across sequential decisions. As the effective visit count approaches N_{total}, the decay aggressively reduces exploration noise, committing the policy to the empirically superior branch.

3.4.3 Anytime Optimal State Tracking

The standard output of MCTS is a policy derived from the visit counts of the root node, intended to maximize the expected future reward. However, in extremal problems, our objective is to identify the *global maximum* configuration (e.g., the largest possible set), regardless of whether it lies on the most probable path. To address this, we augment MCTS with an **Anytime Optimal State Tracking** mechanism. During the simulation phase, whenever the rollout policy reaches a terminal state s_T, we evaluate its reward $R(s_T, n)$. If $R(s_T, n)$ exceeds the best reward found so far globally, we record the state s_T, independent of the tree search statistics. This decoupling ensures that valid "outlier" solutions found via serendipity in deep rollouts are never lost, effectively treating the simulation phase as a global random sampler running in parallel with the policy optimization.

4 Experiments and Results

4.1 Experimental Setup

We conducted all experiments on a High Performance Computing Cluster using a single CPU core and a strict 6GB memory limit per trial. We chose this limit to show the memory efficiency of our framework. Time limits are set to 14 days for Max-N3IL and 7 days for the other problems we tested. Comprehensive hardware details, hyperparameter configurations, and ablation constraints are provided in Appendix C and D.

Our baseline strategy was a single search trial per configuration to maximize the exploration of large grid sizes under our strict compute budget. However, due to opportunistic job scheduling on the shared cluster, a subset of configurations received multiple evaluations. Because our objective is to establish theoretical existence bounds rather than expected average returns, we report the absolute best (global maximum or minimum) valid configuration discovered across all available trials for any given n.

4.2 Main Results

Table 1: Comparison of our MCTS results against best-known theoretical and computational bounds. See Appendix A and E for detailed previous best known results. **Note:** "Ours" reports the empirical performance trend observed or specific improvements found within the tested range. All baseline comparisons are based on the records available as of April 8, 2026.

Extremal Problem	Metric	Previous Best Known (Baseline)	Ours	Improved Range (n)
Max No-Three-in-Line	Lower Bound	$(1.5-\varepsilon)n$ (Hall et al., 1975) See table 4 for known optimal and near-optimal results	$\approx$ **1.8n** (Empirical Fit)	$82 \leq n \leq 119$
Min Complete	Upper Bound	See Table 5 and 6	$\approx$ **0.95n** (Empirical Fit)	$37 \leq n \leq 96$
Min Dominating Set	Upper Bound	$2\lceil n/2 \rceil$ (Aichholzer et al., 2023) For small grids, see Table 5 and 6	$\approx$ **0.95n** (Empirical Fit)	$37 \leq n \leq 96$, $31 \leq n \leq 35$, and $n \in \{7, 16, 26\}$
Max No-Four-in-Line	Lower Bound	$1.973n$ for sufficiently large n (Kovács et al., 2025)	= **3n**	$n \leq 100$
Max No Isosceles Triangle	Lower Bound	$\Omega(n/\sqrt{\log n})$ (Charton et al., 2024) 112 for $n = 64$ and 164 for $n = 100$ (Georgiev et al., 2025)	$\approx 1.4n$ (Empirical Fit)	-
Max No-Four-on-a-Circle	Lower Bound	$n - o(n)$ (Dong & Xu, 2025)	$\approx$ **1.8n** (Empirical Fit)	$n \leq 39$

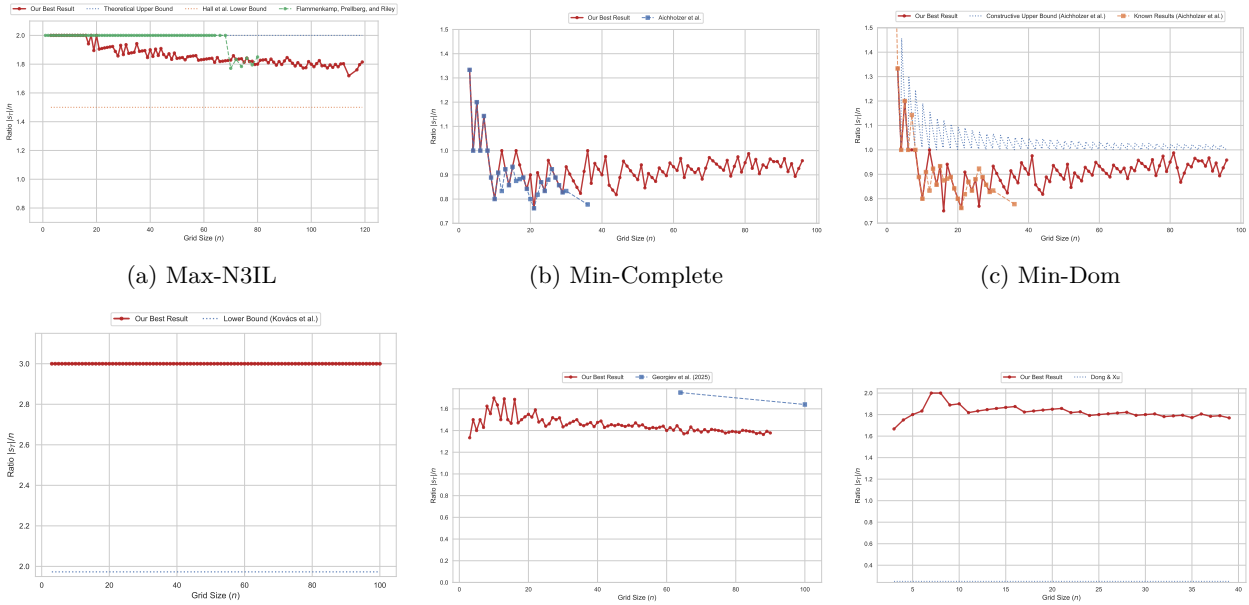
The proposed framework establishes new best-known computational bounds for five out of the six problems we evaluated (table 1). Figure 6 illustrates the detailed scaling behaviors (optimality ratio $|s_T|/n$). While minor local variance exists due to MCTS stochasticity and opportunistic multi-trial sampling, the macroscopic trends robustly support our empirical bounds. The framework demonstrates exceptional search efficiency and generalizability across diverse constraints:

Collinearity Constraints (figs. 6a and 6d): Our MCTS framework finds sets of points of size roughly $1.8n$ for each n satisfying $82 \leq n \leq 119$. This sits squarely between the algebraic lower bound of size approximately $1.5n$ and the upper bound of size $2n$. This demonstrates the success of this method on grids that were previously inaccessible.

For the Max-N4IL problem, for each n satisfying $3 \leq n \leq 100$ we were able to find optimal configurations with $3n$ points with no 4 collinear points, outperforming the previous $1.973n$ lower bound. It would be interesting to see if we can produce these kinds of configurations for even larger grids.

Coverage Objectives (figs. 6b and 6c): For minimization tasks, our MCTS framework was most successful in finding configurations with full D_4-symmetry. We find complete sets of size approximately $0.95n$ for each n in the range $37 \leq n \leq 96$. Since $\mathscr{D}_n \leq \mathscr{I}_n$ for each n, this provides a new upper bound for both Min-Complete and Min-Dom in this range. For some n (e.g., $n = 83$), we find a geometric dominating set of size smaller than the smallest complete set that we found.

Non-linear & Distance Constraints (figs. 6e and 6f): We find configurations of approximately $1.8n$ points with no 4 points on a circle of finite radius for each $n \leq 39$. For Max-No-Isosceles, the framework finds configurations of size approximately $1.4n$. Highly specialized constructions using recent AI tools find larger sets for specific grid sizes (e.g., $n \in \{64, 100\}$) (Georgiev et al., 2025), and could likely be applied to a whole range of values of n. We still include the performance of our MCTS framework to demonstrate that it remains competitive with these more specialized approaches without requiring domain-specific heuristic redesign. This establishes a boundary of our current search capabilities.



(a) Max-N3IL (b) Min-Complete (c) Min-Dom

(d) Max-N4IL (e) Max-No-Isosceles (f) Max-No-4-on-Circle

Figure 6: Scaling behaviors (optimality ratio $|s_T|/n$) across six constraints. Our results (red) are compared against previous best-known bounds. The framework successfully pushes empirical lower bounds upwards (a, d, f) and tightens upper bounds downwards (b, c), demonstrating strong generalization across these kinds of geometric problems.

4.3 Effectiveness of Strategies

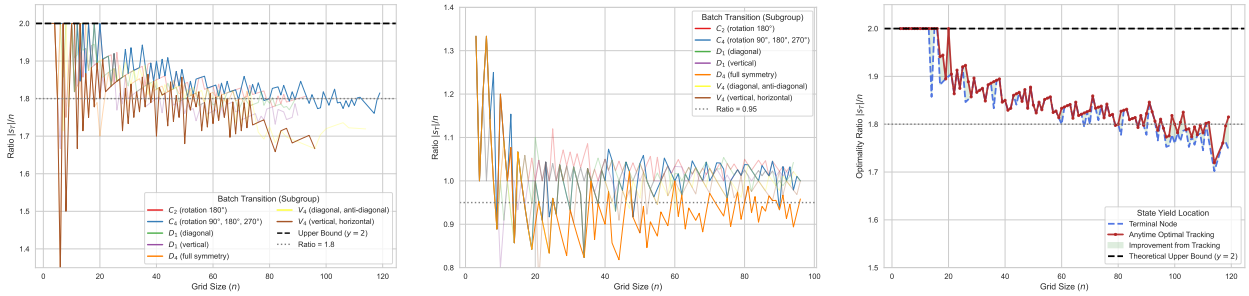
The effectiveness of symmetric batch transitions heavily depends on aligning the chosen subgroup G with the geometric constraints of the problem. These geometric priors can be acquired through empirical exploration or existing mathematical literature.

Symmetric Batch Transitions. For the Max-N3IL problem, empirical exploration (fig. 7a) shows that the rotational subgroup C_4 yields the highest optimality ratios overall. Conversely, axis reflections (V_4) perform poorly, likely because they tend to rapidly remove whole rows and columns from $\mathcal{A}_{\text{feas}}$. However, this exact rapid-elimination mechanism becomes a powerful advantage for minimization tasks. For the Min-Complete problem (fig. 7b), the full symmetry group D_4 leverages these reflections to rapidly cover the grid,

successfully driving the search toward sparser dominating sets. Furthermore, the literature confirms that known optimal N3IL configurations often exhibit C_4 or diagonal symmetries (Flammenkamp, 1992; 1998; 2026), motivating our specific subgroup selections for large grids ($n \geq 96$). If symmetric placements prove detrimental to a specific problem, the framework adapts to a single-action search.

Anytime Optimal Tracking. Figure 7c demonstrates the value of asynchronously recording the global best state during random simulations. This tracking permanently harvests valid "outlier" solutions that might otherwise be discarded when trees are reset or pruned, ensuring the final output is strictly no worse than terminal node evaluations.

We analyze the incremental contribution of each algorithmic component in our ablation study (Section 4.4).



(a) Max-N3IL Batch Transitions (b) Min-Complete Batch Transitions (c) Anytime Optimal Tracking

Figure 7: Effectiveness of algorithmic strategies. The optimal symmetric subgroup is heavily problem-dependent: C_4 excels for Max-N3IL (a) while D_4 excels for Min-Complete by rapidly covering the grid (b). Asynchronous tracking (c) effectively captures outlier configurations discovered deep in rollouts.

4.4 Ablation Study

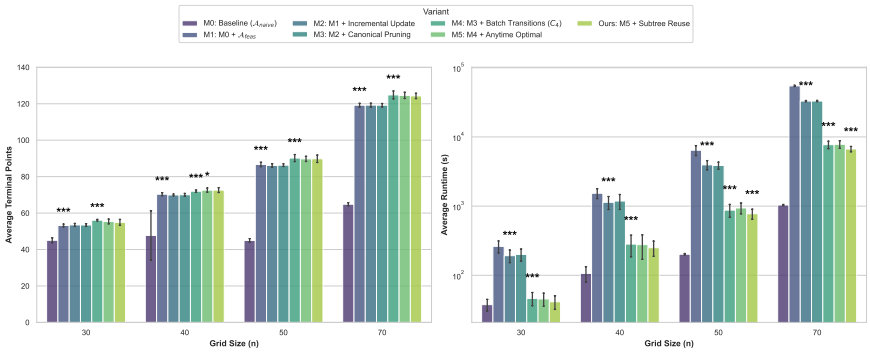


Figure 8: Ablation Study: Comparison of average terminal points (left) and average runtime (right) across different algorithm variants and grid sizes. Bar heights represent the mean value over $N = 20$ independent random seeds, and error bars indicate the standard deviation (± 1 std). Statistical significance is evaluated incrementally against the immediately preceding variant using a one-sided Welch's t-test (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). Specifically, the alternative hypothesis for terminal points is that the current variant yields a strictly greater mean ($\mu_{\text{curr}} > \mu_{\text{prev}}$), whereas for runtime, the alternative hypothesis is a strictly lower mean ($\mu_{\text{curr}} < \mu_{\text{prev}}$). Note that runtime is plotted on a logarithmic scale. See table 7 for detailed numerical results.

The different components of our framework exhibit hierarchical dependencies; for instance, *Canonical Pruning* relies on the structure of the *Feasible Action Space* ($\mathcal{A}_{\text{feas}}$), and *Symmetric Batch Transitions* build directly upon this pruning mechanism. Therefore, we start with the baseline MCTS and cumulatively add components to evaluate the marginal performance gain of each layer.

As shown in fig. 8 and table 7, our final variant achieves superior performance in producing successful configurations and also demonstrates highly optimized computational efficiency. While minor runtime fluctuations exist—likely attributable to the stochastic nature of the shared cluster environment—the overall improvements remain robust across all tested grid sizes.

The first major performance leap occurs with the strict enforcement of the feasible action space (**M1**). Compared to the baseline (**M0**), which operates on the naive action space $\mathcal{A}_{\text{naive}}$ and frequently terminates early due to invalid random selections, M1 exhibits a significant improvement in average terminal number of points ($p < 0.001$ across all grid sizes) by ensuring the deep exploration of valid configurations. However, computing $\mathcal{A}_{\text{feas}}$ from scratch at each node severely degrades runtime. This bottleneck is resolved by the incremental update mechanism (**M2**), which dramatically improves runtime compared to M1 ($p < 0.001$) while incurring no statistically significant performance loss ($p \geq 0.05$). This efficiency gain scales exceptionally well for larger grids, such as $n \in \{50, 70\}$, empirically validating the theoretical $O(n^2)$ update complexity.

The subsequent integration of canonical pruning (**M3**) yields no statistically significant improvement in either performance or efficiency on its own. Without forced symmetric transitions, symmetric states become increasingly sparse at deeper search levels. Consequently, the computational overhead of computing the stabilizer at each step outweighs the minimal branching reduction achieved. This limitation is directly overcome by the introduction of C_4 symmetric batch transitions (**M4**), which creates a breakthrough in both metrics. By actively maintaining structural symmetry deeper within the search tree, the batch mechanism unlocks the potential of canonical pruning. As a result, M4 establishes new empirical peaks for terminal points ($p < 0.001$) and drastically reduces runtime compared to the M1–M3 variants.

Finally, we evaluate the utility-focused enhancements. Although the anytime optimal tracking (**M5**) shows no statistically significant shift in average performance within this specific randomized ablation study, it serves as a critical, zero-cost fail-safe. It ensures that the globally best configuration discovered during any rollout is permanently recorded, independent of the final MCTS policy convergence. The framework is finalized with the subtree reuse strategy (**Ours**), which significantly improves overall efficiency on larger grids ($p < 0.001$ for $n \in \{50, 70\}$) with no statistically significant loss in terminal points. This confirms that persisting tree statistics across deterministic transitions effectively reduces redundant explorations, demonstrating the scalability of the framework for a wide array of extremal problems in combinatorial geometry.

5 Conclusions and Limitations

In this work, we presented a Geometry-Aware Monte Carlo Tree Search framework tailored for extremal problems with hard geometric constraints. By integrating *Incremental Feasible Action Space* maintenance and leveraging symmetries through *Canonical Pruning* and *Symmetric Batch Transitions*, we help to address the "validity cliff" and combinatorial explosion that hinder generic solvers.

Our empirical results demonstrate the effectiveness of this approach. The MCTS framework we propose achieves a significant improvement in performance over baseline implementations. Our ablation study shows how the various aspects of this approach lead to improved efficiency. We establish new best-known bounds for five of the six problems within the grid sizes we evaluated. This demonstrates the capability of our method to generalize and scale, suggesting that it can serve as a versatile tool for constructions in discrete geometry.

Despite the combinatorial explosion inherent to these problems, our framework demonstrates remarkable scalability. All reported main results, including the exploration of grids up to $n = 119$ for Max-N3IL, were achieved using only a **single CPU core with a strict 6GB memory limit** within a two-week cutoff. This confirms that our geometry-aware pruning effectively mitigates the curse of dimensionality for a significant range of n.

Our method provides empirical constructions rather than closed-form bounds. The configurations that we produce are not necessarily optimal, but still provide valuable information for grids of larger sizes that could be used to guide future theoretical work. Future work will leverage parallelization and GPU acceleration (Klęsk, 2025) to explore even larger spaces. Integrating neural networks for value/policy guidance for this geometry-aware search is a promising direction for future research.

References

Oswin Aichholzer, David Eppstein, and Eva-Maria Hainzl. Geometric dominating sets - a minimum version of the No-Three-In-Line Problem. *Computational Geometry*, 108:101913, 2023. ISSN 0925-7721. doi: 10.1016/j.comgeo.2022.101913.

Thomas F. Bloom and Olof Sisask. An improvement to the Kelley-Meka bounds on three-term arithmetic progressions, 2023. URL `https://arxiv.org/abs/2309.02353`.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1): 1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

Tristan Cazenave. Monte Carlo permutation search, 2025. URL `https://arxiv.org/abs/2510.06381`.

François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. PatternBoost: Constructions in mathematics with a little help from AI, 2024. URL `https://arxiv.org/abs/2411.00566`.

Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers (eds.), *Computers and Games*, pp. 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-75538-8.

Zichao Dong and Zijian Xu. Large grid subsets without many cospherical points, 2025. URL `https://arxiv.org/abs/2506.18113`.

Henry E. Dudeney. *Amusements in Mathematics*. Nelson, Edinburgh, 1917.

Achim Flammenkamp. Progress in the no-three-in-line-problem. *Journal of Combinatorial Theory, Series A*, 60(2):305–311, 1992. ISSN 0097-3165. doi: 10.1016/0097-3165(92)90012-J.

Achim Flammenkamp. Progress in the No-Three-in-Line problem, II. *Journal of Combinatorial Theory, Series A*, 81(1):108–113, 1998. ISSN 0097-3165. doi: 10.1006/jcta.1997.2829.

Achim Flammenkamp. The No-Three-in-Line problem, 2026. URL `https://wwwhomes.uni-bielefeld.de/achim/no3in/readme.html`. Accessed: 2026-04-08.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale, 2025. URL `https://arxiv.org/abs/2511.02864`.

Richard K. Guy and Patrick A. Kelly. The No-Three-In-Line problem. *Canadian Mathematical Bulletin*, 11 (4):527–531, 1968. doi: 10.4153/CMB-1968-062-3.

R. R. Hall, T. H. Jackson, A. Sudbery, and K. Wild. Some advances in the no-three-in-line problem. *Journal of Combinatorial Theory, Series A*, 18(3):336–341, 1975. ISSN 0097-3165. doi: 10.1016/0097-3165(75)90043-6.

Jorik Jooken, Pieter Leyman, Tony Wauters, and Patrick De Causmaecker. Exploring search space trees using an adapted version of Monte Carlo tree search for combinatorial optimization problems. *Computers & Operations Research*, 150:106070, 2023. ISSN 0305-0548. doi: 10.1016/j.cor.2022.106070.

Przemysław Klęsk. MCTS-NC: A thorough GPU parallelization of Monte Carlo tree search implemented in Python via numba.cuda. *SoftwareX*, 30:102139, 2025. ISSN 2352-7110. doi: 10.1016/j.softx.2025.102139.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Benedek Kovács, Zoltán Lóránt Nagy, and Dávid R. Szabó. Randomised algebraic constructions for the no-$(k+1)$-in-line problem, 2025. URL `https://arxiv.org/abs/2508.07632`.

Wen-Ding Li and Kevin Ellis. Is programming by example solved by LLMs? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 44761–44790. Curran Associates, Inc., 2024. doi: 10.52202/079017-1422.

Yannic Neuhaus, Nicolas Flammarion, Matthias Hein, and Francesco Croce. On the out-of-distribution generalization of reasoning in multimodal LLMs for simple visual planning tasks, 2026. URL https://arxiv.org/abs/2602.15460.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Thomas Prellberg. Constraint satisfaction programming for the No-three-in-line Problem, 2026. URL https://arxiv.org/abs/2602.07751.

Rushil Raghavan. Improved bounds for 3-progressions, 2026. URL https://arxiv.org/abs/2603.27045.

Pranav Ramanathan, Thomas Prellberg, Matthew Lewis, Prathamesh Dinesh Joshi, Raj Abhijit Dandekar, Rajat Dandekar, and Sreedath Panat. Three methods, one problem: Classical and AI approaches to no-three-in-line, 2025. URL https://arxiv.org/abs/2512.11469.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Ben Ruijl, Jos Vermaseren, Aske Plaat, and Jaap van den Herik. Combining simulated annealing and Monte Carlo tree search for expression simplification, 2013. URL https://arxiv.org/abs/1312.0841.

Stephan Schiffel. Symmetry detection in general game playing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):980–985, July 2010. doi: 10.1609/aaai.v24i1.7649.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Dennis J. N. J. Soemers, Chiara F. Sironi, Torsten Schuster, and Mark H. M. Winands. Enhancements for real-time Monte-Carlo tree search in general video game playing. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, 2016. doi: 10.1109/CIG.2016.7860448.

Tom Vodopivec, Spyridon Samothrakis, and Branko Šter. On Monte Carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017. doi: 10.1613/jair.5507.

Adam Zsolt Wagner. Constructions in combinatorics via neural networks, 2021. URL https://arxiv.org/abs/2104.14516.

Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. MCTS-Judge: Test-time scaling in LLM-as-a-judge for code correctness evaluation, 2025. URL https://arxiv.org/abs/2502.12468.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design, 2025. URL https://arxiv.org/abs/2501.08603.

Xu Zhuang, Yuxiang Yao, Po-Chu Hsu, Xiaokang Wang, and Peikai Qi. Machine learning and LLM-boost symbolic regression for predicting $\mathbb{Q}$-gonality of modular curves. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=LDcFa3E5vJ.

A Related Works

We categorize existing approaches to the extremal problems into two distinct streams: theoretical mathematical constructions and constrained tree search.

A.1 Mathematical Bounds and Constructions

A.1.1 No-3-in-Line

Early work focused on number-theoretic constructions to establish foundational lower bounds. Hall et al. (1975) utilized points on modular hyperbolas ($xy \equiv k \pmod p$) to achieve the current best asymptotic lower bound of roughly $1.5n$. However, a significant gap remains between this constructive lower bound and the trivial upper bound of $2n$. This substantial margin suggests that for finite grid sizes (e.g., $n \in [50, 100]$), $M(n)$—the maximum number of points in an $n \times n$ grid such that no 3 points are collinear—may be significantly higher than what algebraic constructions can guarantee. Motivated by this gap, our work employs computational search to explore the solution space within this interval, aiming to empirically push the lower bound upward and reveal larger configurations than the ones that we can produce with algebraic constructions.

A.1.2 Geometric Dominating Sets and Independent Geometric Dominating Sets

Following Aichholzer et al. (2023), a set $S \subseteq \mathcal{G}_n$ is a *geometric dominating set* if every grid point p is either in S or lies on a line determined by two points of S; equivalently, every p is collinear with two (distinct) points of S. The minimum size of such a set is the *geometric domination number* $\mathscr{D}_n$. If additionally no three points of S are collinear, then S is an *independent geometric dominating set*. Let $\mathscr{I}_n$ denote the minimum size of a independent geometric dominating set in an $n \times n$ grid. The authors also establish the first general asymptotic bounds for $\mathscr{D}_n$: they prove a lower bound $\mathscr{D}_n = \Omega(n^{2/3})$ and give an explicit construction of a geometric dominating set of size $2\lceil n/2 \rceil$. These results leave a substantial gap between the best known lower and upper bounds in the grid, motivating both improved constructions and stronger counting/packing arguments.

We understand much less about the minimum size of an independent geometric dominating set. It is clear that $\mathscr{I}_n \leq 2n$, but we do not even know that there exist infinitely many n for which $\mathscr{I}_n < 2n$.

On the computational side, Aichholzer et al. (2023) compute the value of $\mathscr{I}_n$ for $n \leq 12$ and give upper bounds for n satisfying $13 \leq n \leq 30$ and $n = 36$, providing evidence and test cases for conjectures about the true growth of $\mathscr{I}_n$.

A.1.3 No-4-in-Line

Let $f_{\text{4-coll}}(n)$ denote the maximum cardinality of a subset of $\mathcal{G}_n$ containing no four collinear points. The best asymptotic bounds currently known, established by Kovács et al. (2025), are:

$$1.973\, n \leq f_{\text{4-coll}}(n) \leq 3n$$

for all sufficiently large n. Consequently, while the maximum size is known to scale linearly with n, we do not even have a conjecture for how $f_{\text{4-coll}}(n)$ grows with n.

A.1.4 No-4-on-a-Circle

Let $f_{\text{circ}}(n)$ denote the maximum cardinality of a subset of $[n]^2$ containing no four concyclic points. The tightest asymptotic bounds currently known, established by Dong & Xu (2025), are:

$$n - o(n) \leq f_{\text{circ}}(n) \leq \frac{5}{2}n - \frac{3}{2}.$$

Consequently, while the maximum size is guaranteed to grow linearly with n, we do not even have a conjecture for how $f_{\text{circ}}(n)$ grows with n.

A.1.5 No-Isosceles-Triangle

Let $f_{\text{iso}}(n)$ denote the maximum size of a subset of $[n]^2$ containing no three points that form an isosceles triangle, including degenerate collinear configurations. The best known lower bound is:

$$f_{\text{iso}}(n) \geq c\,\frac{n}{\sqrt{\log n}}$$

for some absolute constant $c > 0$ (Charton et al., 2024).

The best known upper bounds for this problem come from bounding the number of points on each horizontal line of the grid: three points at positions (a, y), $(a + d, y)$, $(a + 2d, y)$ on the same horizontal line form a (degenerate) isosceles triangle, so the restriction of any isosceles-free set to a single horizontal line must avoid three-term arithmetic progressions. Applying the best known bounds for subsets of $\{1, 2, \ldots, n\}$ containing no three-term arithmetic progression to each of the n horizontal lines, Bloom & Sisask (2023) established

$$f_{\text{iso}}(n) \leq \exp(-O((\log n)^{1/9}))\, n^2.$$

This was further improved by Raghavan (2026), giving

$$f_{\text{iso}}(n) \leq \exp(-O((\log n)^{1/6}(\log\log n)^{-1}))\, n^2.$$

Like in the problems about sizes of dominating sets, the upper and lower bounds here have different orders of magnitude.

A.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) was introduced by Coulom (2007) as a framework combining selective tree search with random rollout evaluations. The UCT algorithm of Kocsis & Szepesvári (2006) provided a principled exploration-exploitation strategy by treating node selection as a multi-armed bandit problem, and remains the backbone of most modern MCTS implementations (Browne et al., 2012). MCTS gained widespread attention through AlphaGo and AlphaZero (Silver et al., 2016; 2018), which demonstrated superhuman performance on Go, Chess, and Shogi by coupling MCTS with deep neural networks. These works also established engineering practices—most notably subtree reuse across decision steps—that we adopt in our framework. However, neural networks are not included in our current implementation, and we leave the integration of learned value and policy functions to future work.

Beyond game playing, MCTS has been applied to combinatorial optimization problems such as scheduling and vehicle routing (Jooken et al., 2023). However, these settings typically feature smooth objective functions and separable constraints, where random rollouts naturally produce valid solutions. The geometric constraints in this work (e.g., collinearity, concyclicity) are *global*: a single point placement invalidates an $O(n)$ set of grid cells, and uniform random rollouts over the unrestricted action space collapse with high probability. This incompatibility motivates our Incremental $\mathcal{A}_{\text{feas}}$ mechanism, which maintains geometric validity throughout the search at $O(n^2)$ cost per step.

Two prior lines of work address symmetry within MCTS. Schiffel (2010) exploited symmetry in general game playing (GGP) by detecting automorphisms at the *game rule level* and merging statistics of globally

equivalent states via a transposition table. Cazenave (2025) proposed Monte Carlo Permutation Search, which fuses visit statistics of permutation-equivalent actions to reduce effective branching. Our approach differs from both in two respects. First, rather than detecting global game symmetry or fusing statistics, we dynamically compute the *state-level stabilizer subgroup* $\mathrm{Stab}(s) \leq D_4$ at each node and restrict expansion to canonical orbit representatives—a finer-grained, state-dependent pruning. Second, our Symmetric Batch Transitions redefine the MDP transition itself by attempting simultaneous orbit-wide point placements, a mechanism with no direct counterpart in prior MCTS literature.

The challenge of tuning the exploration–exploitation balance over a long horizon has been addressed by SA-UCT (Ruijl et al., 2013), which anneals the UCB exploration constant over iterations. We adopt a similar decay schedule but introduce a refinement for the subtree reuse setting: our progress ratio r incorporates inherited visit counts N_{existing}, ensuring that the exploration schedule remains continuous and monotonically decreasing across decision steps rather than resetting at the start of each move.

B Computational Complexity Analysis

We analyze the time complexity of our Geometry-Aware MCTS framework, using the Max-N3IL problem on the $n \times n$ grid as a representative case. We first derive the per-iteration cost, then aggregate over the full search to establish the total complexity, and finally contrast this polynomial bound with the intractability of exact methods.

Per-Iteration Cost Breakdown. Each MCTS iteration consists of four phases. We analyze their worst-case costs in terms of n:

Selection. The agent traverses the tree from root to leaf. Since each level corresponds to a point placement and the trajectory length is bounded by $2n$, the tree depth is $O(n)$. At each internal node, the UCB1 score is evaluated over all expanded children, costing $O(|\mathcal{A}_{\text{feas}}|)$ where $|\mathcal{A}_{\text{feas}}|$ is equal to the local branching factor. In the worst case $|\mathcal{A}_{\text{feas}}| = O(n^2)$, but in practice the feasible action space shrinks rapidly with depth (cf. Section 3.2). The total selection cost is $O(n) \cdot O(|\mathcal{A}_{\text{feas}}|)$.

Expansion. A single child node is created. The dominant cost is the incremental update of the feasible action space via ray casting (Section 3.2): the new point forms lines with each of the $|s| \leq 2n$ existing points, and each line is traced across $O(n)$ grid cells. This yields an expansion cost of $O(n^2)$ per node.

Simulation (Rollout). Starting from the expanded node, the agent performs a random rollout until the terminal condition $\mathcal{A}_{\text{feas}} = \emptyset$ is reached. Each rollout step selects a random feasible action and incrementally updates $\mathcal{A}_{\text{feas}}$ at a cost of $O(n^2)$. Since at most $2n$ points can be placed in total, the rollout traverses at most $O(n)$ steps. The simulation cost is therefore $O(n) \cdot O(n^2) = O(n^3)$.

Backpropagation. The reward is propagated from the leaf back to the root. Each update is $O(1)$, and the path length is $O(n)$, giving a total cost of $O(n)$.

The simulation phase dominates, yielding a **per-iteration cost of** $O(n^3)$.

Total Complexity. The full algorithm proceeds through two nested loops. The *outer loop* executes the sequential MDP: at each decision step, MCTS selects and commits to an action, advancing the trajectory by one point. Since the trajectory length is at most $2n$, the outer loop runs for $O(n)$ steps. The *inner loop* performs MCTS search at each decision step, executing $N_{\text{iter}} = O(n^2)$ iterations (we set $N_{\text{iter}} = O(n^2)$ in all experiments). Combining with the per-iteration cost:

$$T_{\text{total}} = \underbrace{O(n)}_{\text{trajectory}} \times \underbrace{O(n^2)}_{\text{iterations per step}} \times \underbrace{O(n^3)}_{\text{cost per iteration}} = O(n^6). \tag{9}$$

The total time complexity of our framework is therefore **polynomial** in the grid size n.

Contrast with Exact Methods. The polynomial complexity of our approach stands in sharp contrast to the cost of exact enumeration. An exhaustive search must, in principle, examine all candidate subsets of the n^2 grid points up to the trivial upper bound of $2n$. The size of this search space is

$$\sum_{k=1}^{2n} \binom{n^2}{k}, \tag{10}$$

which grows super-exponentially in n. Therefore, exact methods are severely limited for large grid sizes.

The Tractability–Optimality Trade-off. Our framework does not guarantee global optimality; it is a heuristic search that trades optimality guarantees for polynomial-time tractability. However, this trade-off is precisely what enables the exploration of grid sizes ($n > 68$) that up to this point have been too large for exact computation. As demonstrated in Section 3, the components of our geometry-aware framework—incremental $\mathcal{A}_{\text{feas}}$ maintenance, canonical pruning, and symmetric batch transitions—ensure that this polynomial budget is spent efficiently, concentrating search effort on the most promising regions of the combinatorial landscape. The empirical results (Section 4) confirm that this approach yields configurations that significantly exceed the best-known constructive lower bound of $\approx 1.5n$ and closely approach the conjectured density of $\approx 1.814n$ (Guy & Kelly, 1968), suggesting that $O(n^6)$ may be capable of identifying structures that approach the current conjectured bounds.

Generality Across Problem Variants. The $O(n^3)$ per-iteration bottleneck arises from the incremental maintenance of $\mathcal{A}_{\text{feas}}$ during simulation, which is specific to the geometric invariant Φ. For other problem variants (e.g., No-Four-in-Line, No-Four-on-a-Circle), the per-step update cost may differ depending on the constraint structure, but the overall algorithmic skeleton—and its polynomial-time guarantee—remains unchanged.

C Detailed Experimental Configurations

We list the detailed hyperparameters used in our experiments to ensure reproducibility. Table 2 summarizes the configurations for the MCTS algorithm. The UCT algorithm specified in Section 3.4.2 was utilized for all experiments.

C.1 Reward Functions

For every extremal problem with the goal of maximizing $|s_T|$ and minimizing $|s_T|$, we use these two reward functions respectively:

$$R(s_T, n) = \exp\left(2\frac{|s_T| - n}{n}\right) \quad \text{and} \quad R(s_T, n) = \exp\left(-2\frac{|s_T| - n}{n}\right). \tag{11}$$

The design of the reward function is an empirical choice aimed at amplifying signal when the optimality gap is small. Optimizing the reward shaping is orthogonal to the main contribution of this work.

C.2 Hyperparameter Settings for Main Experiments

We set these hyperparameters and configurations based on complexity, best-known geometric prior, and heuristics for each problem (see Table 2).

C.3 Ablation Study Configurations

The ablation study was conducted on the Max-N3IL problem. Table 3 details the algorithmic configurations for each variant evaluated in the incremental ablation study.

Table 2: Detailed hyperparameter settings and algorithmic configurations across problems.

Hyperparameters Symbol	**Max Iterations** N_{iter}	**Batch Transition** G	**Max Depth for Canonical Pruning**	**Exploration Constant** C
Max-N3IL ($3 \leq n \leq 95$)	$100n^2$	1 trial of each symmetry	$2n$	1.41
Max-N3IL ($96 \leq n \leq 119$)	$25n^2$	1 trial each in C_4 and V_4 [(a)]		
Min-Complete	$100n^2$	1 trial of each symmetry		
Min-Dom	$100n^2$	1 trial of each symmetry		
Max-N4IL	$10n^2$ & $25n^2$ [(b)]	C_4	$3n$	
Max-No-Isosceles	$100n^2$	None	2	
Max-No-4-on-Circle	$25n^2$	None		

[(a)] V_4 here specifically denotes diagonal and anti-diagonal reflections.
[(b)] We ran 1 trial for each value of max iteration. Figure 6d shows the max number of points found for each n.

Table 3: Algorithmic configurations for each variant in the incremental ablation study. "Max Depth" indicates the maximum search depth up to which canonical pruning is applied. $2n$ represents full-depth pruning, as the maximum possible number of points is bounded by $2n$ for Max-N3IL. "Opt. Tracking" represents anytime optimal tracking.

Variant	**Description**	**Action Space**	**Max Depth**	**Batch Transitions**	**Opt. Tracking**	**Tree Reuse**
M0	Baseline	$\mathcal{A}_{\text{naive}}$	None	None	Disabled	Disabled
M1	M0 + $\mathcal{A}_{\text{feas}}$	$\mathcal{A}_{\text{feas}}$ (Scratch)	None	None	Disabled	Disabled
M2	M1 + Incremental	$\mathcal{A}_{\text{feas}}$ (Incremental)	None	None	Disabled	Disabled
M3	M2 + Canonical Pruning	$\mathcal{A}_{\text{feas}}$ (Incremental)	$2n$	None	Disabled	Disabled
M4	M3 + Batch Transitions	$\mathcal{A}_{\text{feas}}$ (Incremental)	$2n$	Rotational (C_4)	Disabled	Disabled
M5	M4 + Anytime Optimal	$\mathcal{A}_{\text{feas}}$ (Incremental)	$2n$	Rotational (C_4)	Enabled	Disabled
Ours	M5 + Subtree Reuse	$\mathcal{A}_{\text{feas}}$ (Incremental)	$2n$	Rotational (C_4)	Enabled	Enabled

Other Settings for Ablation Study: The computational budget and evaluation metrics were strictly controlled across all variants. The experiments were conducted on grid sizes $n \in \{30, 40, 50, 70\}$. For each algorithmic variant and grid size, we evaluated the performance over 20 independent trials, initialized with random seeds 0 through 19. The search budget (N_{iter}) was uniformly set to $10n^2$ iterations per decision step.

D Technical Details

Memory Efficiency and State Space Complexity: As stated in the main text, a strict 6GB memory limit was intentionally imposed across all trials to rigorously test the memory efficiency of our framework. The underlying state space complexity $|\mathcal{S}| = 2^{n^2}$ grows explosively; for instance, a mere 20×20 grid already contains $|\mathcal{S}| = 2^{400} \approx 2.58 \times 10^{120}$ states. By achieving superior results on significantly larger grids (e.g., up to $n = 119$) within this modest memory budget, we empirically highlight the effectiveness of our geometry-aware pruning and incremental update mechanisms in mitigating the curse of dimensionality.

Hardware Heterogeneity and Control: The HPC cluster is a heterogeneous environment comprising various CPU architectures (e.g., Intel Skylake, Cascade Lake, and Ice Lake). To ensure the scientific rigor of our **ablation studies**, we enforced strict hardware consistency by using Slurm constraints to the best of our ability. Specifically, all ablation trials were restricted to nodes supporting the following features: `intel, avx512, mlx5_ib, nvme, fastscratch`. This ensures that performance variations are attributable to algorithmic changes rather than underlying hardware differences.

Resource Optimization for Main Experiments: For non-ablation experiments (e.g., searching for maximum n sets), where the primary goal is sheer computational throughput rather than controlled timing comparisons, we did not impose these hardware constraints. This allowed the scheduler to utilize any available CPU-only nodes, thereby maximizing resource utilization and reducing queue wait times under the 7-day or 14-day time limits.

E Detailed Previous Best-Known Results of Max-N3IL and Min-Complete Sets

Previous Best-Known Results of Largest No-Three-in-Line Sets: Table 4 shows the best detailed results known results according to Flammenkamp (2026); Prellberg (2026).

Table 4: Previous best known values for the sizes of the N3IL sets ($n \leq 80$). Bold values indicate that the value is optimal (i.e., reaches the theoretical upper bound $2n$).

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$M(n)$	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**	**30**
n	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$M(n)$	**32**	**34**	**36**	**38**	**40**	**42**	**44**	**46**	**48**	**50**	**52**	**54**	**56**	**58**	**60**
n	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
$M(n)$	**62**	**64**	**66**	**68**	**70**	**72**	**74**	**76**	**78**	**80**	**82**	**84**	**86**	**88**	**90**
n	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
$M(n)$	**92**	**94**	**96**	**98**	**100**	**102**	**104**	**106**	**108**	**110**	**112**	**114**	**116**	**118**	**120**
n	61	62	63	64	66	68	70	72	74	76	78	80			
$M(n)$	**122**	**124**	**126**	**128**	**132**	**136**	124	132	132	140	140	148			

Previous Best-known Results of Smallest Complete Sets: Table 5 shows the sizes of the smallest complete sets for $2 \leq n \leq 12$. These results are obtained by exhaustive enumeration, so they are optimal (Aichholzer et al., 2023). Table 6 shows the best-known sizes of smallest complete sets for $n > 12$. Note that the definition of "independent dominating set" in their work is equivalent to "complete set" in our context.

Table 5: Known optimal values for sizes of the smallest complete sets ($n \leq 12$).

n	2	3	4	5	6	7	8	9	10	11	12
$\mathscr{I}_n$	4	4	4	6	6	8	8	8	8	10	10

Table 6: Current best upper bounds for sizes of smallest complete sets ($n > 12$).

n	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	36
$\mathscr{I}_n \leq$	12	12	14	14	15	16	16	16	16	18	20	20	22	24	24	24	24	25	28

F Detailed Experiment Results

F.1 Ablation Study

Table 7: Incremental ablation study on $n \times n$ grids (Average terminal number of points and average runtime in seconds, with std in parentheses). Statistical significance vs. the preceding variant is denoted by $^{*}p < 0.05$,$^{**}p < 0.01$,$^{***}p < 0.001$. Search budget is $10n^2$ iterations.

Algorithm Variant	$n = 30$		$n = 40$	
	Points	**Time**	**Points**	**Time**
M0: Baseline ($\mathcal{A}_{\text{naive}}$)	45.0 (1.4)	37.64 (7.27)	47.8 (13.6)	106.53 (26.91)
M1: M0 + $\mathcal{A}_{\text{feas}}$	53.2 (0.8)***	261.93 (51.89)	70.3 (0.8)***	1533.96 (250.77)
M2: M1 + Incremental Update	53.5 (0.8)	192.11 (39.79)***	70.0 (0.5)	1134.80 (241.91)***
M3: M2 + Canonical Pruning	53.5 (0.8)	200.34 (39.43)	70.0 (0.8)	1184.56 (288.49)
M4: M3 + Batch Transitions (C_4)	**56.1** (0.4)***	46.34 (10.06)***	72.0 (0.6)***	282.59 (97.78)***
M5: M4 + Anytime Optimal	55.4 (1.3)	45.52 (9.69)	**72.6** (1.1)*	277.33 (106.73)
Ours: M5 + Subtree Reuse	54.9 (1.7)	**41.41** (9.28)	**72.6** (1.3)	**250.42** (61.93)

Algorithm Variant	$n = 50$		$n = 70$	
	Points	**Time**	**Points**	**Time**
M0: Baseline ($\mathcal{A}_{\text{naive}}$)	45.0 (1.0)	201.67 (4.96)	64.8 (0.8)	1036.85 (19.50)
M1: M0 + $\mathcal{A}_{\text{feas}}$	86.7 (1.2)***	6421.70 (1065.50)	119.2 (1.1)***	54667.12 (1237.59)
M2: M1 + Incremental Update	86.2 (0.9)	3945.52 (603.08)***	119.2 (1.1)	32974.75 (676.13)***
M3: M2 + Canonical Pruning	86.3 (0.7)	3868.81 (456.44)	119.2 (1.0)	33042.73 (719.21)
M4: M3 + Batch Transitions (C_4)	**90.2** (1.9)***	874.99 (183.97)***	**124.8** (2.2)***	7722.33 (963.06)***
M5: M4 + Anytime Optimal	89.8 (1.4)	940.59 (167.67)	124.6 (1.7)	7822.67 (967.08)
Ours: M5 + Subtree Reuse	89.8 (2.0)	**775.19** (128.46)***	124.3 (1.5)	**6693.15** (599.49)***

G Examples

To best demonstrate the scalability of our framework, we show examples of configurations discovered on larger grids for every problem evaluated in this study.

G.1 Max No-Three-in-Line

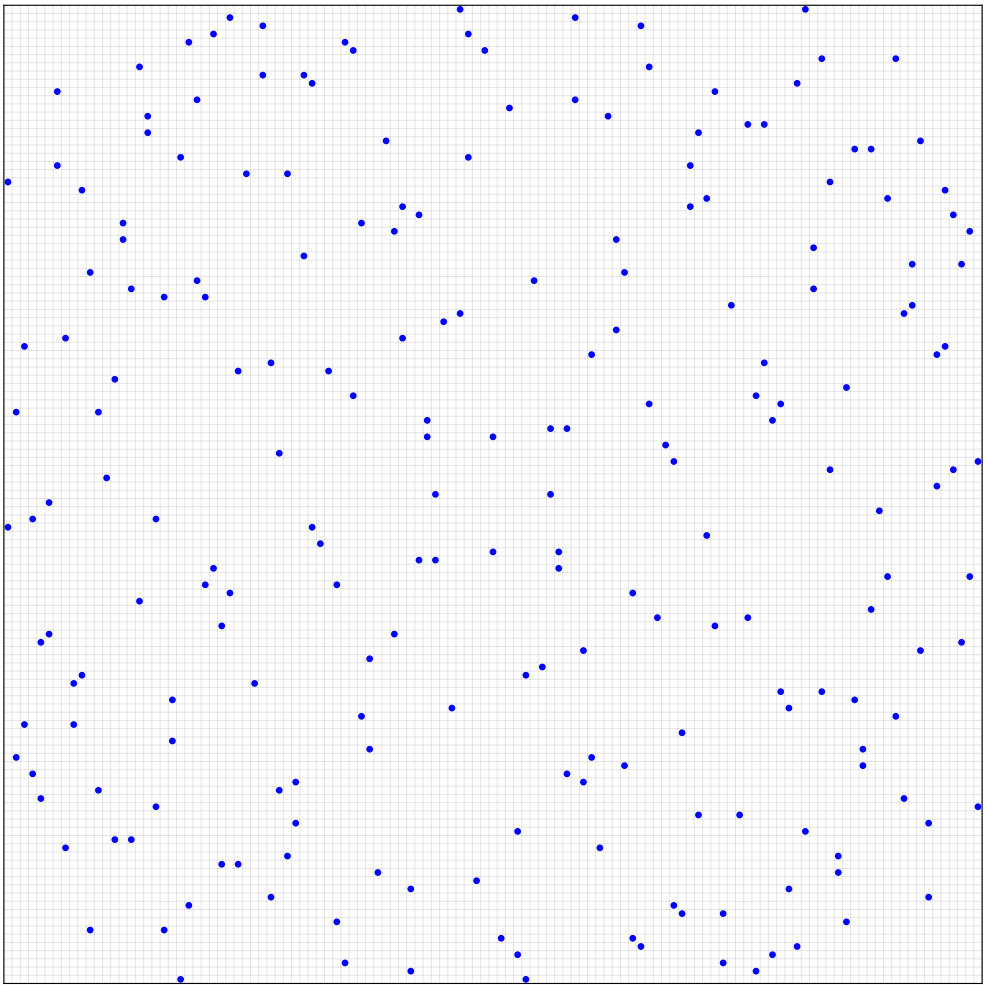
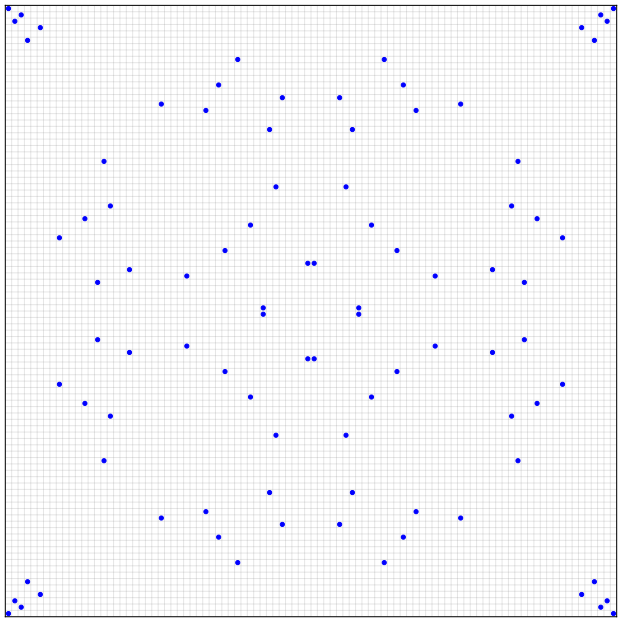
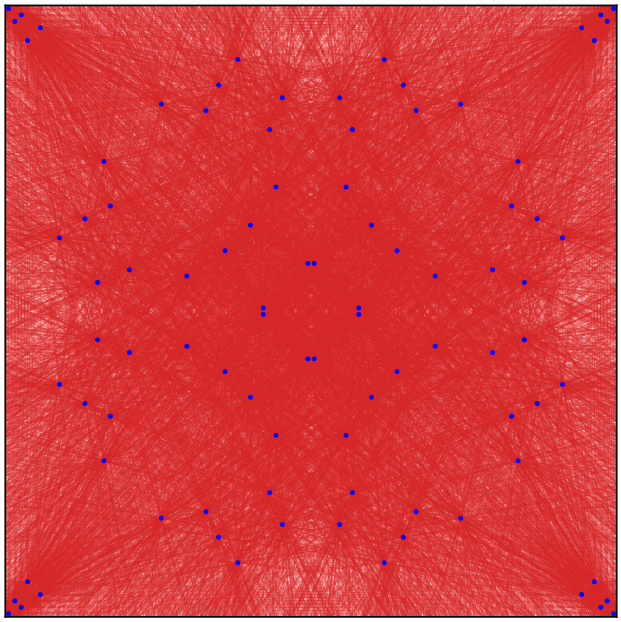


Figure 9: A discovered configuration with 216 points on a 119×119 grid, satisfying the No-Three-in-Line constraint.

G.2 Smallest Complete Set



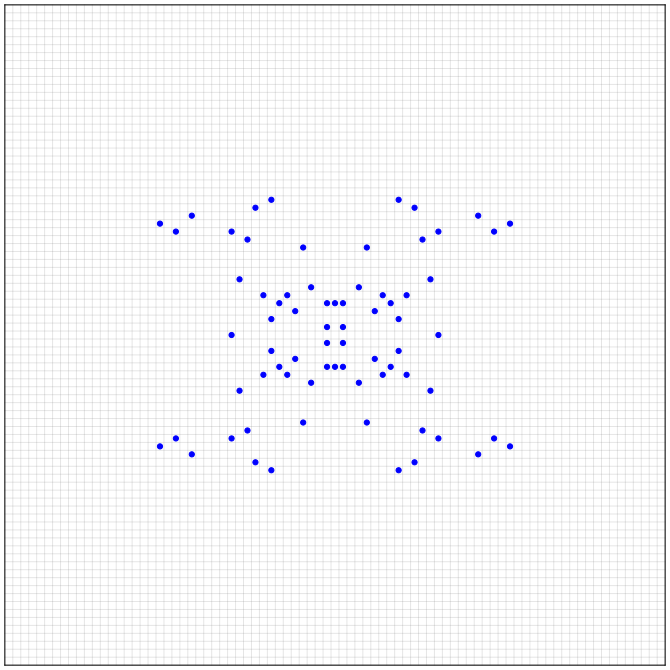
(a) A complete set with 92 points.



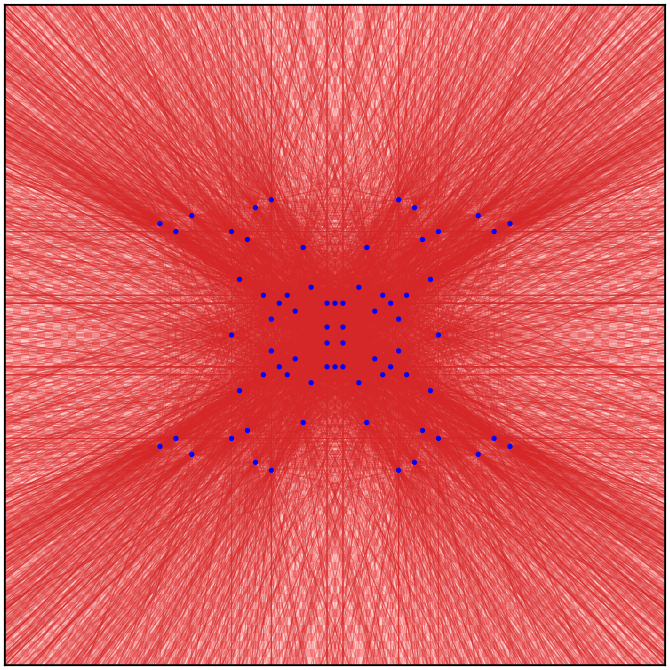
(b) Raycast visualization demonstrating full grid coverage without violating collinearity constraints.

Figure 10: A valid smallest complete set (independent geometric dominating set) discovered by our framework, consisting of 92 points on 96×96 grid. The bottom panel visually confirms the validity of the complete set, illustrating how the rays generated by the points cover the entire grid while satisfying the No-Three-in-Line constraint.

G.3 Smallest Geometric Dominating Set



(a) A geometric dominating set with 72 points.



(b) Raycast visualization demonstrating full grid coverage.

Figure 11: An example of a geometric dominating set on an 83×83 grid with 72 points. The bottom panel visually confirms the validity of the dominating set, illustrating how the lines generated by the 72 points completely cover the entire grid.

G.4 Max No-Four-in-Line

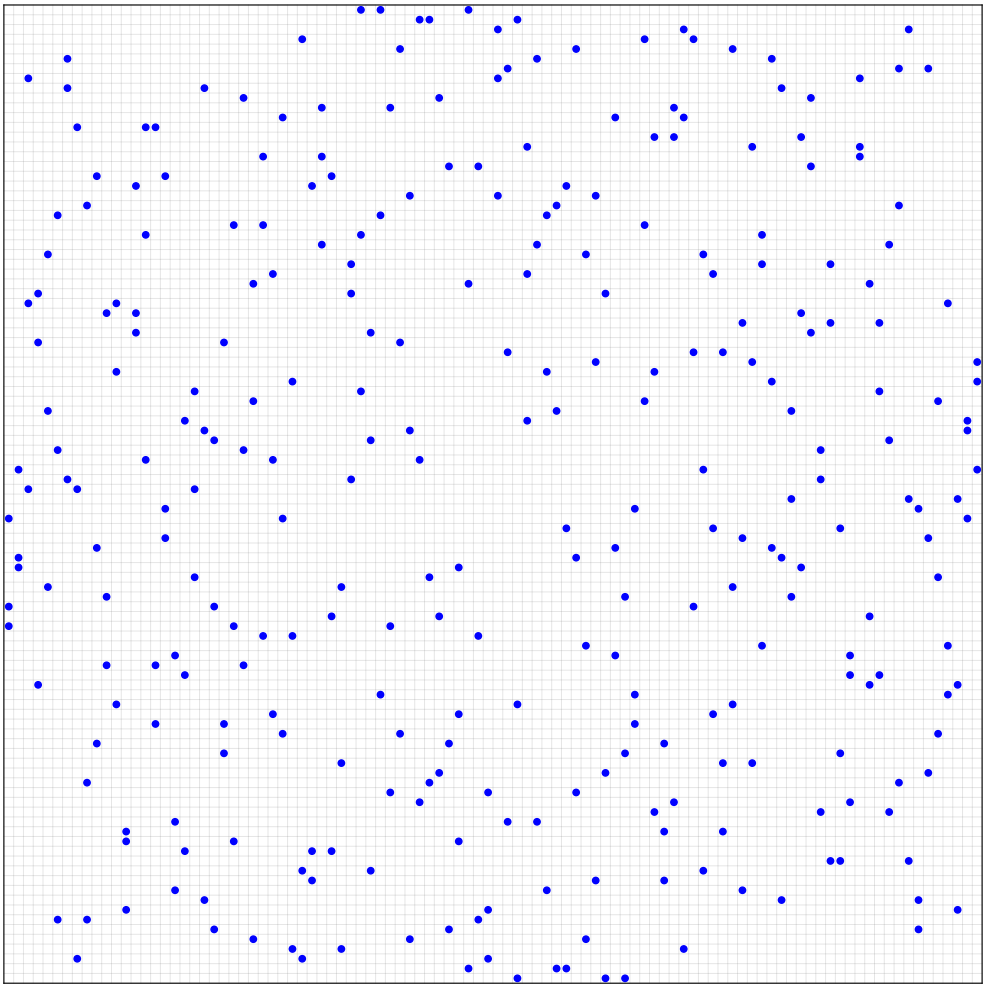


Figure 12: An optimal configuration with 300 points on a 100×100 grid, satisfying the No-Four-in-Line constraint.

G.5 Max No-Isosceles Triangle

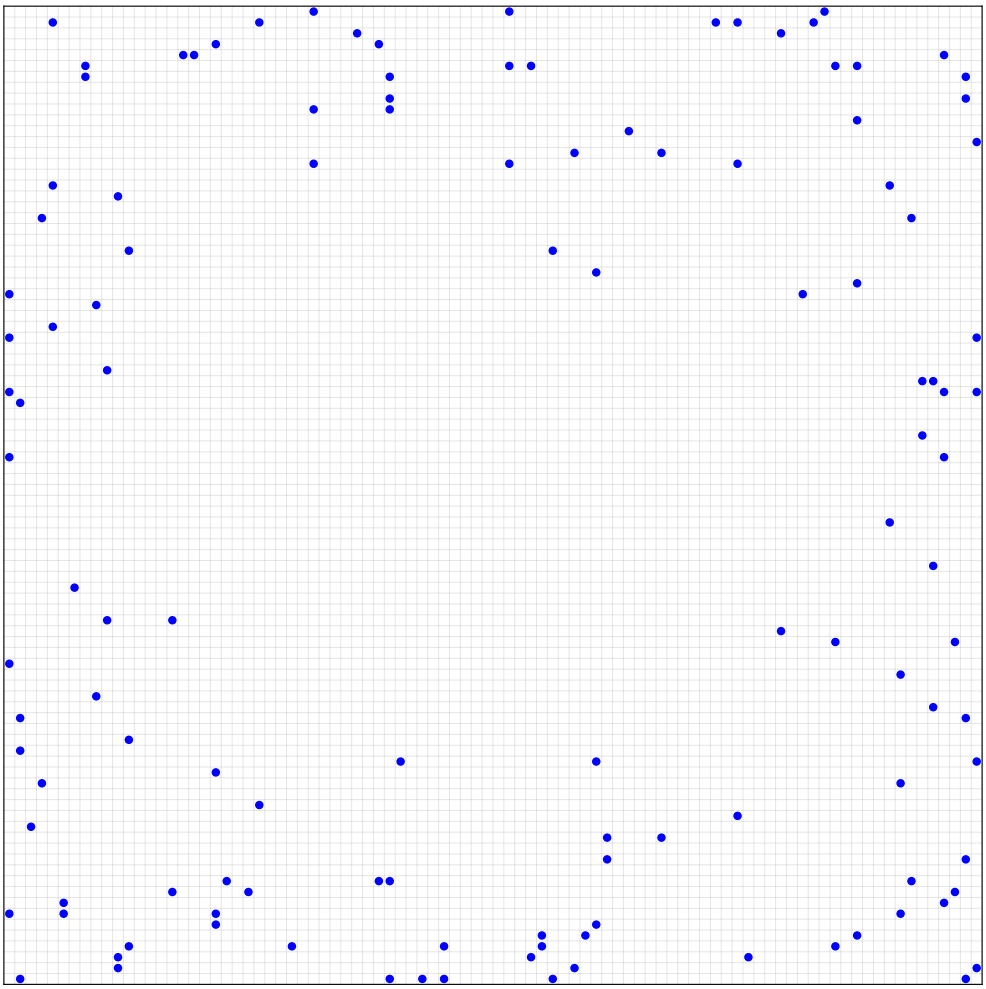


Figure 13: A configuration with 124 points on a 90×90 grid, where no three points form an isosceles triangle.

G.6 Max No-Four-on-a-Circle

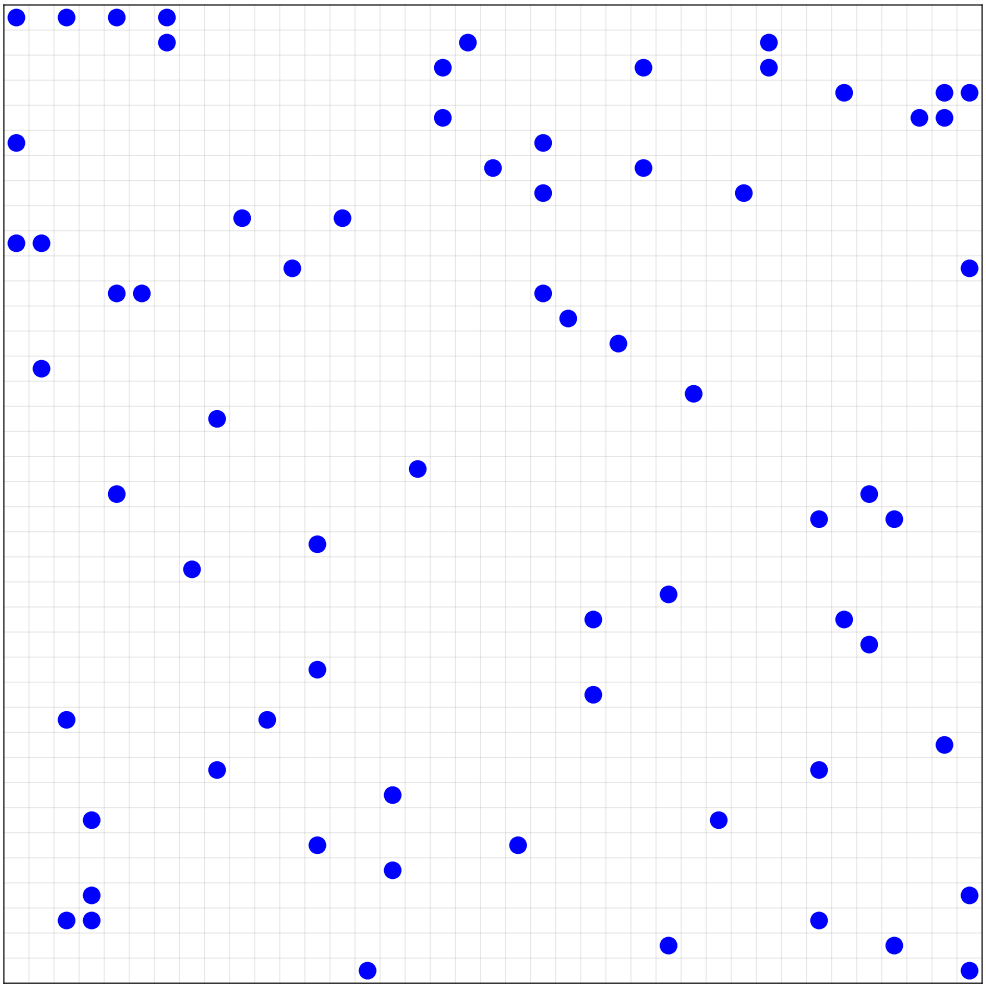


Figure 14: A configuration with 69 points on a 39×39 grid, satisfying the No-Four-on-a-Circle constraint.