Automating Potential Discovery for the k-server Conjecture

Kirill Brilliantov [1] **Etienne Bamas** [1] **Emmanuel Abbé** [1]

Abstract

We aim to accelerate progress toward the k-server conjecture by automating the search for potential functions, the central object in the dominant proof paradigm for this problem. We formulate potential discovery as an executable optimization problem adapted for AI agents: an agent submits a Python program defining a candidate potential function, and an automatic verifier checks it against a large system of inequalities. The verifier provides dense feedback through the number of violated inequalities and a stringent success criterion of zero violations. The evaluation is sound but incomplete: any violation refutes the submitted potential on the tested instance, while zero violations do not by themselves constitute a full proof. Nevertheless, in the open $k = 4$ circle case, a zero-violation candidate would be mathematically interesting in its own right: no known potential function satisfies all constraints under our formulation, and such a candidate would identify a concrete new structure for a possible proof. Our experiments show that current agentic methods can solve the resolved $k = 3$ case and make partial progress in the open $k = 4$ regime. In particular, agents discover a new potential with fewer violations than previously known candidates under our formulation, though the challenge remains unresolved. At the same time, the contrast between successful recovery of zero-violation potentials in the resolved $k = 3$ case and persistent failure modes in the open $k = 4$ case suggests that this setting can also serve as a useful benchmark for code-based mathematical discovery agents.

1. Introduction

Recent advances in large language models (LLMs) and agentic scaffolds have made *automated science* a practical research direction rather than a slogan: contemporary systems can routinely write code, propose hypotheses, run experiments, and iterate on failures (Lu et al., 2024; Zheng et al., 2025). In mathematics, early task-specific efforts (Davies et al., 2021; Peifer et al., 2020; Wagner, 2021; Lample & Charton, 2019; Charton et al., 2024; Brilliantov et al., 2023) have recently been followed by more general-purpose agents that now can contribute beyond textbook problem solving, including program-search–driven discoveries and code-evolution workflows that improve sophisticated constructions (Romera-Paredes et al., 2024; Novikov et al., 2025; Georgiev et al., 2025; Nagda et al., 2025; 2026).

Inspired by these developments, we ask whether the same agentic capabilities can be brought to bear on *open-ended discovery* in foundational theoretical computer science. We focus on the *k-server conjecture*, widely regarded as the "holy grail of competitive analysis" and a flagship open problem in online algorithms. Since its formulation by Manasse, McGeoch, and Sleator in 1990 (Manasse et al., 1990), the conjecture has shaped the development of competitive analysis as a field (Chrobak et al., 1991; Chrobak & Larmore, 1991; Fiat et al., 1994; Koutsoupias, 2009; Bartal & Koutsoupias, 2004; Bein et al., 2002). It asks whether, for every parameter k and every metric space, there exists a deterministic online algorithm whose total cost is at most $c = k$ times the offline optimum up to an additive constant. We refer to c as the *competitive ratio*, and call an algorithm satisfying this guarantee *c-competitive*.

The best known general guarantee is due to Koutsoupias & Papadimitriou (1995), who proved that the *Work Function Algorithm* (WFA) — widely viewed as the leading candidate for resolving the conjecture — is $(2k - 1)$-competitive for any k and any metric space. In many resolved metric families where WFA *is* k-competitive, the proof proceeds by exhibiting a *potential function* that satisfies a prescribed collection of inequalities (Koutsoupias, 2009; Koutsoupias & Papadimitriou, 1995). Recent attention has concentrated on the *circle* metric, a highly structured setting that nonetheless remains a major obstacle for existing potentials. Coester & Koutsoupias (2021) introduced a unifying potential that resolves many known tight cases, yet it provably does not extend to the circle. By contrast, Huang & Zhang (2022) settled the $k = 3$ circle case by introducing a circle-specific modification, underscoring that the remaining difficulty ap-

[1]EPFL, Lausanne, Switzerland. Correspondence to: Kirill Brilliantov <kirill.brilliantov@epfl.ch>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

pears to be discovering the right potential function.

Analysis of the development of this problem suggests that progress largely comes from identifying the right potential (a construction) and verifying it—a workflow naturally amenable to automation. We operationalize this paradigm by representing candidate potentials as Python programs and posing an explicit, automatically verifiable search problem: minimize the number of violated inequalities, with the natural target of *zero* violations. A central roadblock is that the most interesting metric spaces are large or even infinite (such as the circle), making the full inequality system intractable to enumerate. To mitigate this, we evaluate potentials on a sequence of increasingly fine discretizations and on so-called k-taxi augmentations (Coester & Koutsoupias, 2021) that capture additional structure of the infinite case. For a fixed discretization, satisfying all generated inequalities gives, by explicit finite verification, a certificate of k-competitiveness of WFA on that finite metric instance. It does not, however, prove the infinite-circle case. The evaluation is therefore sound but *incomplete*: violations refute a candidate, whereas zero violations identify an interesting construction for subsequent mathematical analysis. Even this reduced task is challenging; to the best of our knowledge, no known potential satisfies all inequalities generated for our $k = 4$ circle instances, so finding one would already be a substantial contribution under our formulation.

We intentionally aim for a task that is challenging yet plausibly within reach of iterative LLM-based methods, rather than expecting models to produce fully formal proofs for arbitrary major conjectures. Three features support this: (i) the evaluator yields dense, automatic feedback enabling rapid iteration; (ii) the $k = 3$ circle case is solved, providing a concrete calibration target; and (iii) empirically, agents can reach zero detected violations on $k = 3$ instances and substantially reduce violations on the hardest $k = 4$ instances beyond prior human-designed potentials in our evaluation suite.

Moreover, our task can be viewed as a valuable *benchmark* for agentic open-ended discovery. By providing a clear target—achieving *zero* violations—it supports diagnostic evaluation beyond incremental score improvements. Even in the resolved $k = 3$ regime, this criterion cleanly ranks methods from naive random sampling to more structured, agentic methods. This directly addresses a limitation of recent code-based open-ended discovery benchmarks (Romera-Paredes et al., 2024; Novikov et al., 2025; Georgiev et al., 2025): their objectives often saturate and the progress is visible only through tiny decimal gains (Novikov et al., 2025; Lange et al., 2025; Wan et al., 2025; Wang et al., 2025), while random baselines can be surprisingly strong and obscure algorithmic contributions (Gideoni et al., 2025). In contrast, our setup can track progress via the *rate of perfect solutions* (e.g., the probability of reaching zero violations), not just marginal improvements in average score.

Contributions.

1. **Proof search as executable optimization.** We formulate potential-function discovery for the k-server conjecture as an executable optimization problem: candidate potentials are programs, and a verifier scores them by violations of the inequalities required for a Work Function Algorithm certificate, with zero violations serving as the target for a finite proof candidate.
2. **Tooling for potential discovery.**[1] We implement the full pipeline needed to search for and evaluate candidate potentials: construction of work-function inequality systems, generation of circle instances with varying discretization levels and optional k-taxi augmentations, fast inequality verification, diagnostic violation reports, and a representation format that supports rapid iteration and reproducible comparison across methods.
3. **Empirical progress on resolved and open regimes.** We show that agentic methods can recover zero-violation potentials in the resolved $k = 3$ circle case and make partial progress in the open $k = 4$ regime, where our best candidate reduces the violation count on the augmented $k = 4, m = 6$ instance from 17 to 3 under our finite formulation.

Section 2 formalizes the potential-based certification task for WFA as a collection of automatically checkable inequalities over an explicit *work-function graph*, and describes how we instantiate this framework on discretized circle metrics and their k-taxi augmentations. Section 3 then presents empirical calibration of the framework, followed by section 4 presenting results and failure modes in the more challenging $k = 4$ regime. We conclude with related works section 5 (with extended version in Appendix H), a discussion of limitations (section 6), and conclusions (section 7).

2. Potential Discovery as Executable Optimization

We begin with a practically oriented exposition of our optimization reformulation, deferring the mathematical background and additional details to Appendix A.

Problem Formulation. Given a *finite* metric space M, represented by a distance matrix in $\mathbb{R}^{|M|\times|M|}$, and an integer k, we construct a finite directed graph (V, E), commonly referred to as the *work-function graph*. Each node $u \in V$ is associated with a vector $w_u \in \mathbb{R}^{|\mathcal{C}|}$, where $\mathcal{C}$ is the set

[1] Code and artifacts; see Appendix I for raw URLs.

of all possible *configurations* of servers in the metric space, i.e. the set of all possible k-uplets of locations in the metric space. For simplicity, we define w as a vector but in the infinite case it is a function that maps elements of $\mathcal{C}$ to a real value. Each directed edge is parameterized by a triple (u, r, v): u is the start vertex, r is an edge parameter, and the end vertex v is uniquely determined by the pair (u, r). Edges carry weights, denoted by $\gamma(u, r, v)$.

The objective is to find a *potential function* $\Phi : \mathbb{R}^{|\mathcal{C}|} \to \mathbb{R}$ such that the following inequality holds for every edge:

$$\forall (u, r, v) \in E : \qquad \Phi(w_v) - \Phi(w_u) \;\geq\; \gamma(u, r, v).$$

Proposition 2.1. *For a finite metric instance M, if a function potential Φ satisfies all verifier inequalities at competitiveness c, then it gives a certificate for c-competitiveness of WFA on M.*

Note: to the best of our knowledge, the explicit notion of a *work-function graph* has not been formalized in the k-server literature. Rather, closely related graph-based viewpoints appear as folklore and have likely been used implicitly in earlier analyses (Chrobak & Larmore, 1991; Coester & Koutsoupias, 2021).

Family of Metric Spaces. As suggested by the formulation above, the task is closely related to detecting negative cycles in a directed graph. Indeed, running the Bellman–Ford algorithm on (V, E) yields a certificate in the form of node potentials Φ, but this certificate is purely *numerical* and tied to a single finite instance.

Our goal, in contrast, is to obtain a *functional* potential: a parameterized form that generalizes across multiple metric instances drawn from the *same family* and can plausibly be integrated into a subsequent proof. We illustrate this with the circle metric. Ultimately, we seek a certificate for WFA on the *infinite* circle, but the associated work-function graph is infinite, making direct verification infeasible. Instead, we evaluate candidates on a sequence of finite discretizations, i.e., finite metric subspaces of the circle obtained by restricting the circle metric to sampled point sets: for each m, we place m equidistant points on the circle to obtain a finite metric, with values of m accessible to modern computation (e.g., $m = 4, 5, 6, \dots$). The desired potential should apply *without modification* across these discretized instances. Both empirically and mathematically, success on an expanding range of m provides strong evidence that the same functional form extends to the infinite-circle limit.

This evaluation is sound but incomplete: any detected violation definitively refutes a candidate, whereas a candidate with no detected violations is best viewed as a strong hypothesis rather than a complete infinite-circle proof. Even so, such a candidate would be of independent interest—especially because, to our knowledge, no potential is currently known that achieves zero violations for the $k = 4$ circle case in our discrete formulation. Finally, across all instances we can currently probe, we find no evidence that the resulting inequality systems are infeasible. Concretely, we use Bellman–Ford–style negative-cycle checks, which require at least one full pass over the edge set and thus become prohibitively expensive for the largest instance we consider ($k = 4, m = 8$ with k-taxi augmentation), whose work-function graph has on the order of billions of edges.

Table 1. Summary of the pre-generated circle discretizations. Extended discussion: Appendix B.3.

k	m	k-taxi	edges	violations
3	6	no	$2'100$	0
3	8	no	$41'920$	0
3	6	yes	$193'662$	0
3	8	yes	$20'068'416$	1
4	6	no	$6'006$	0
4	8	no	$261'200$	0
4	6	yes	$7'000'602$	17
4	8	yes	$7 \cdot 10^{9*}$	$30'000^*$

Generated Instances. From a practical standpoint, for the family of circle metrics we provide several pre-generated instances (see Table 1). In addition, we employ a convenient augmentation mechanism introduced by Coester & Koutsoupias (2021), known as *k-taxi requests*. Intuitively, k-taxi requests allow us to substantially enrich the work-function graph over the same underlying set of sampled circle points: we add additional nodes and edges that approximate carefully constructed (infinite) request sequences. This augmentation both enlarges the constraint set and makes the potential inequalities more stringent, thereby increasing the likelihood that a potential satisfying all constraints will extend to the infinite-circle setting.

Table 1 also reports the number of violations incurred by the unifying potential of Coester & Koutsoupias (2021). That work pushed the frontier of *solved* metric spaces toward the circle by unifying several potentials originally developed for special cases, including the line metric, weighted star metrics, and the case $k = 2$. To the best of our knowledge this unifying potential remains the state of the art among human-proposed candidates for the case $k = 4$.

Note that, the pipeline is not specific to the circle. It can be adapted to other metric families with minimal changes, essentially by replacing the metric-defining parameter matrix while keeping the rest of the generation and evaluation framework unchanged.

Submission and Scoring. Submissions are provided as Python code defining a function `potential` that takes as input k, a metric M (represented as a real-valued matrix), and a *work function* w (a vector associated with each node), and returns a single real value. See Appendix B.4 for the complete specification.

Given a submission, we evaluate Φ by computing potential values for all nodes of the given instances and then check all inequalities. The primary benchmark target is the *number of violated inequalities* incurred by the submitted potential (lower is better, with the ideal target being zero violations). Or alternatively, we use violations *score*: $1 - \frac{\text{violations}}{\text{edges}}$ (higher is better).

In addition to the headline score, we report several diagnostic metrics to facilitate analysis and debugging, including the correlation between the submitted potential and the instance-specific Bellman–Ford potential, violation counts stratified by the type of edge weight $\nabla(u, r, c)$ and several more. Full definitions and reporting details are provided in the Appendix B.5.

Functional potentials and robustness. The objects we seek are *functional* potentials: compact expressions that depend on the mathematical data of the instance, rather than on the identifiers of a particular finite work-function graph. A simple example is the linear potential

$$\Phi_{\text{sum}}(w) = \sum_{X \in \mathcal{C}} w(X).$$

If such a potential satisfied the required inequalities, it could in principle be incorporated into an infinite-case argument, because it is given as an explicit function of the work function. More sophisticated examples include the unifying potential of Coester and Koutsoupias (Coester & Koutsoupias, 2021) and the circle-specific canonical potentials of Huang and Zhang (Huang & Zhang, 2022): although their formulas are more structured, they are still explicit functions of (k, M, w) and can be evaluated across different discretizations without referring to a particular enumeration of graph nodes (see Appendix A.5 for explicit definitions).

By contrast, a Bellman–Ford certificate for a finite work-function graph is only an instance-specific numerical assignment to vertices. Such a certificate may prove feasibility of one finite inequality system, but it is essentially a lookup table for that discretized instance and does not provide a functional expression that can be transferred to larger discretizations or to the infinite circle. Since our goal is potential discovery rather than finite-graph memorization, we explicitly distinguish these two notions.

To discourage submissions that exploit artifacts of a finite enumeration, such as Bellman–Ford lookup tables or case-dependent logic tied to the public work-function graph, we include a robustness protocol. Candidate potentials are evaluated under mathematically valid transformations of the instance representation, including global rescalings of the metric, permutations of the configuration indexing, and additive shifts of the work function. These checks rule out simple representation-level memorization. They are not, however, a complete anti-overfitting guarantee: they do not by themselves prevent specialization to a public finite graph, a cache of hard edges, or a particular discretization level. We therefore also hold out the $m = 8$ instances with k-taxi augmentation for private evaluation for both $k = 3$ and $k = 4$; for the very large $k = 4, m = 8$ instance, we use a ladder of private subsamples before the largest held-out evaluation. These safeguards are pragmatic rather than absolute (see Appendix B.7 for more details).

3. Calibration on the $k = 3$ case

We first calibrate the framework in the resolved $k = 3$ circle setting. As discussed above, this case is known to admit valid potential-function certificates, and the corresponding generated instances are substantially smaller than those in the open $k = 4$ regime. This makes $k = 3$ a natural setting for controlled experiments: it allows us to test whether agents can recover potentials with zero detected violations, while still preserving the central challenge of the open case—discovering potentials that generalize across discretizations rather than overfitting to a single finite instance.

Methods. We focus on three representative search strategies that provide diversity to justify the design choices of the framework while keeping the study resource-bounded. All methods use the same `openai/gpt-oss-120b` model. BEST-OF-N samples independent candidate implementations of `potential` from a fixed prompt and keeps the candidate with the fewest violations, isolating gains from inference-time sampling alone. SHINKAEVOLVE (Lange et al., 2025) and LOONGFLOW (Wan et al., 2025) follow an LLM-driven program-evolution paradigm: they maintain previously evaluated candidates, use the LLM to propose edits or mutations, evaluate candidates by violation-based feedback, and iterate via selection. See Appendix C.2 for hyperparameters and method-specific details.

Design Choices. We adopt two protocol choices aimed at the open $k = 4$ regime — evaluating *search procedures* rather single constructions and restricting to a *canonical potential* hypothesis class — and validate them in the $k = 3$ testbed. See Appendix D for full information and supporting evidence on the following ablations.

Search vs. construction. Following the search-over-programs recipe used in recent code-based discovery systems (Georgiev et al., 2025), we found across the $k = 3$

Table 2. Perfect runs and final scores summary (higher is better) over 10 independent runs of BEST-OF-N, SHINKA, and LOONGFLOW all using `gpt-oss-120b` capped at 100 evaluations, evaluated on two instances ($k = 3$) with $m \in 6, 8$ and no k-taxi augmentations. Methods are given with the same initial solutions and task descriptions. "Perfect Runs" reports the fraction of runs achieving perfect score for each corresponding metric. "Final Score" reports per-run maxima for the combined metric (multiplicative across instances) and each instance, aggregated as mean ± std across runs.

Method	Perfect Runs			Max Score		
	combined	$m = 6$	$m = 8$	combined	$m = 6$	$m = 8$
BEST-OF-N	0/10	0/10	0/10	0.907±0.064	0.958±0.027	0.948±0.039
SHINKA	0/10	4/10	0/10	0.913±0.146	0.956±0.079	0.949±0.090
LOONGFLOW	2/10	5/10	2/10	0.938±0.084	0.972±0.038	0.964±0.052

testbed that improving *search procedures* over potentials is more effective than directly improving a single candidate potential. Beyond yielding better scores, this formulation provides an additional scaling axis: the learned search procedure can be run longer or with a larger evaluation budget. It also better matches the strengths of current LLMs. Rather than asking the model to synthesize a compact symmetric construction in one step, we ask it to design heuristics for exploring a structured space of such constructions. In our setting, this makes "tuning a search algorithm" a more natural use of the budget than fine-grained editing of one proposed potential.

Canonical potentials. We find that current methods make little progress without a strong hint in the form of a substantially reduced hypothesis class: even for the smallest setting ($k = 3, m = 6$), unbiased methods never discover a *perfect* potential. Accordingly, we adopt the *canonical potential* representation from Huang & Zhang (2022), in which potentials are parameterized by small integers h, n, an index matrix of size $h \times k$, and a coefficient vector of size $\binom{n}{2}$. This form is computationally efficient to evaluate and captures the structure of most of the known potentials (Appendix A.5). Imposing this restriction improves performance dramatically and enables methods to reliably discover perfect potentials.

Stress test. After the ablations above, we run a $k = 3$ generalization stress test designed to mimic the pressure of the $k = 4$ regime: a single potential must satisfy the constraint suites for both $m = 6$ and $m = 8$. The best methods can still reach zero detected violations on the combined task, indicating that the proposed search-over-canonical-templates protocol admits feasible solutions and current agentic methods can find it. At the same time, Table 2 shows that aggregate scores (mean ± std.) overlap across methods, whereas the rate of zero-violation solutions ("Perfect Runs") ranks them cleanly, supporting the usefulness of the proposed task as a *benchmark*.

4. Open case: $k = 4$

The open $k = 4$ circle case is the main regime considered in this work. While the resolved $k = 3$ setting can also be viewed as a *benchmark*, the $k = 4$ setting is better described as an open **challenge**: no known *functional* potential satisfies all generated inequalities.

Unlike the $k = 3$ circle case, which was directly resolved by a circle-specific analysis, the $k = 4$ case has not, to our knowledge, been the focus of a comparable dedicated analysis. The circle-specific construction of Huang & Zhang (2022) is tailored to $k = 3$ and does not directly extend to $k = 4$ without structural modifications; see Appendix E.1. On the other hand, the unifying potential of Coester & Koutsoupias (2021) was designed as a general construction for a broad class of metrics and values of k, rather than optimized for the circle. Nevertheless, it remains the strongest known functional candidate applicable to our $k = 4$ circle instances: on the augmented $k = 4, m = 6$ instance, it incurs only 17 violations among roughly 7 million inequalities (and zero violations on $k = 4, m = 6, 8$ without augmentations).

Moreover, moving from $k = 3$ to $k = 4$ changes the computational regime. Full verification becomes substantially more expensive: while evaluating a canonical potential on the $k = 3$ calibration instances takes only a few seconds, evaluating one on the augmented $k = 4, m = 6$ instance takes over a minute even with parallelization. This sharply limits candidate throughput, and simple approximations such as checking random subsets of inequalities are ineffective at this resolution, since they do not reliably distinguish near-feasible candidates. At the same time, the unifying potential of Coester & Koutsoupias (2021), is already close to feasibility. Thus, meaningful progress requires operating in the "tens of violations out of millions" regime.

In practice, we found that a direct transfer of the $k = 3$ search protocol was insufficient. Without additional guidance, agents did not reliably develop efficient search procedures: they evaluated too few candidates and spent much of the budget on candidates that could have been rejected cheaply. We therefore introduced an explicit hint encourag-

Table 3. Best-found k-competitive potential. Candidates are evaluated on the k-taxi augmented $k = 4, m = 6$ circle metric.

Found by	Violations
Codex (gpt-5.2)	3
SHINKA (gpt-oss-120b)	14
Coester et al. (2021)	17

ing agents to use *proxy evaluation*. Rather than checking every candidate against the full inequality system during search, agents were prompted to design cheaper intermediate evaluators—more structured than naive random subsets—and reserve full verification for the most promising candidates. Useful proxy strategies included evaluating on caches of historically hard edges, staging evaluation from smaller to larger edge sets, and applying early stopping once a candidate accumulated too many violations.

This proxy-based formulation substantially increases candidate throughput, but it also changes the engineering profile of the task. Search programs become longer and more brittle, prompts must describe both the mathematical objective and the proxy-evaluation protocol, and each iteration requires the agent to maintain a larger codebase and more state across attempts. These costs limit large-scale experimentation in the open $k = 4$ regime. Even modest sweeps with frontier models become expensive, while self-hosted models are cheaper but less effective in this niche mathematical setting.

Across our self-hosted model runs with canonical or structured hints, we did not find qualitatively new candidates beyond variants of the unifying potential. Most non-unifying constructions plateaued around thousands of violations, far from the near-feasible regime reached by the unifying construction. This suggests that the open $k = 4$ case requires both stronger mathematical priors and more scalable search machinery: near-feasible candidates differ by only a handful of inequalities among millions, and identifying useful corrections requires more than broad random exploration.

See Appendix E.2 for an extended discussion of token costs, proxy-evaluation design, and additional experiments.

4.1. Pushing beyond the unifying potential

We organized the $k = 4$ search around two structured sources of guidance. The first was the strongest available general candidate: the unifying potential of Coester & Koutsoupias (2021). We gave SHINKAEVOLVE, using `gpt-oss-120b`, an explicit implementation of this potential and asked it to search for improvements. This produced a small modification reducing the violation count from 17 to 14 on the augmented $k = 4, m = 6$ instance (preserving zero violations on $k = 4, m = 6, 8$ without augmentations). However, the improvement did not lead to a broader search trajectory: subsequent candidates remained close to the same local structure, suggesting that small perturbations of the unifying potential alone may be insufficient to eliminate the remaining violations.

The second source of guidance came from the solved $k = 3$ circle case. In an iterative GPT-5 Pro assisted loop, we analyzed which features of the Huang–Zhang construction appeared responsible for its success, including symmetries, index relations in the canonical matrices, and coefficient patterns. We then turned these observations into structured search hypotheses for the $k = 4$ case. However, SHINKAEVOLVE and LOONGFLOW powered by `gpt-oss-120b`, could not find notable improvements.

Our best result came from applying this second guidance source in a coding-agent plus human-guided workflow using Codex. This interface was useful because it combined autonomous search with a faster feedback loop: the human user could inspect failures, steer the search at the level of mathematical structure, and let the coding agent immediately translate those directions into executable experiments. This mattered especially in the $k = 4$ regime, where progress depended on engineering as well as on mathematical hypothesis formation. The agent implemented proxy evaluation routines, maintained caches of frequently violated edges, staged evaluation across increasingly difficult constraint sets, and used early stopping to discard weak candidates before full verification. This allowed the search to screen many more candidates than would be feasible with exhaustive evaluation against all roughly 7 million inequalities at every iteration.

This workflow produced our best k-competitive candidate, with only 3 violations on the augmented $k = 4, m = 6$ instance (and zero on all non-augmented instances). Under our finite formulation, this improves over both the original unifying potential and the 14-violation modification found by SHINKAEVOLVE. The same workflow also found a zero-violation candidate for a relaxed augmented $k = 4, m = 6$ setting with competitiveness $c = k + 1$, although this candidate does not generalize to the augmented $m = 8$ instance.

Details of the discovered potentials, additional exhaustive brute-force searches around the found candidates, and violation analyses are given in Appendix E.5.

These results do not resolve the open $k = 4$ challenge: a candidate with 3 remaining violations is still refuted on the checked finite instance, and zero violations in a relaxed setting do not imply a proof for the infinite circle. Nevertheless, the improvement is mathematically useful. It reduces the obstruction set relative to the previous best functional candidate and provides concrete objects for further analysis: characterizing the geometry of the remaining vio-

lated inequalities, simplifying the discovered potentials, and searching for correction terms that preserve the near-feasible structure while eliminating the final violations.

5. Related Work

Code-based mathematical discovery. Recent work has shown that LLM-based coding agents can be used not only for problem solving, but also for scientific and mathematical exploration through executable search loops. AlphaEvolve and related systems use language models to propose, mutate, and refine programs whose outputs can be automatically evaluated, leading to improvements in algorithmic and combinatorial constructions (Novikov et al., 2025; Georgiev et al., 2025; Nagda et al., 2025; 2026). Our work follows this general "search over executable constructions" paradigm, but applies it, to our knowledge for the first time, to potential-function proof search for the k-server conjecture, a concrete open problem in online algorithms. The goal is not merely to improve a numerical constant, but to discover functional potentials satisfying proof-relevant inequalities, where zero violations identify finite proof candidates for subsequent mathematical analysis.

Benchmarks for mathematical discovery. The success of code-based discovery systems has spurred several benchmark efforts for measuring progress in AI-assisted mathematical and scientific reasoning, including optimization-constant collections and broader mathematical challenge sets (Davis et al., 2026; Wang et al., 2026; Glazer et al., 2025; Epoch AI, 2026). Our task can also be viewed as a benchmark for agentic mathematical search, but its primary role in this work is as a discovery environment. It comes from competitive analysis, a relatively underexplored domain in current AI-for-mathematics benchmarks, and targets the k-server conjecture, arguably the most prominent open problem in that area. Rather than merely posing the problem as an evaluation task, we use the environment to run ablations over problem formulations, study which interfaces make the search tractable, and attempt to push beyond the best known human-designed potentials.

AI-assisted scientific and theoretical discovery. A growing body of work studies AI systems as collaborators in research-level scientific and mathematical discovery, including applications in theoretical computer science, physics, and symbolic computation (Bubeck et al., 2025; Dobriban, 2025; Guevara et al., 2026; Brenner et al., 2026; Feng et al., 2026a). These works emphasize that progress often requires more than producing a final answer: systems must propose hypotheses, run computational experiments, interpret failures, and refine search strategies. Our $k = 4$ experiments exhibit the same pattern. The most effective workflow combined mathematical guidance with coding-agent engineering, including proxy evaluation, hard-edge caches, and staged verification.

6. Limitations

Our method inherits several fundamental limitations from the underlying proof-search problem. First, instance size grows rapidly with k, discretization level m, and augmentations, leading to very rapid growth in the number of states and constraints that must be checked. This makes exhaustive evaluation increasingly expensive and forces the use of proxy objectives or partial checking at larger scales. Second, while the circle (and its k-taxi augmentation) provides a particularly crisp and challenging stress test—including a strong human baseline that is close to feasibility—there are relatively few metric families with similarly well-motivated, discretizable open cases and comparable "near-miss" potentials. Third, our representation assumes candidate potentials are efficiently computable programs; however, a correct potential might require structure that is difficult to express or evaluate efficiently in Python under our current interface. Finally, although canonical parameterizations are computationally convenient, existing results suggest that broad canonical classes have intrinsic limitations (Huang & Zhang, 2022), and over-reliance on them may bias search away from genuinely new potential forms.

7. Conclusion and Future Work

We presented a verifier-backed approach to potential discovery for the k-server conjecture on the circle. Instead of asking an AI system to produce a complete proof directly, we reformulate a central proof-search bottleneck—finding a suitable potential function—as an executable optimization problem over automatically checkable inequality systems. In this formulation, candidate potentials are programs, violations provide dense feedback, and zero violations identify finite proof candidates for further mathematical analysis.

Our experiments suggest that this form of executable verification can support meaningful AI-assisted mathematical search. In the resolved $k = 3$ circle case, agentic methods can recover zero-violation potentials, validating the framework as a calibration setting. In the open $k = 4$ regime, the problem remains unsolved, but verifier-guided search improves over the strongest known functional candidate under our finite formulation, reducing the violation count on the augmented $k = 4, m = 6$ instance from 17 to 3.

More broadly, our results highlight that progress in AI-assisted mathematics often requires both mathematical structure and engineering infrastructure. The most effective workflow combined human conceptual guidance, coding-agent iteration, proxy evaluation, hard-edge caching, and full verifier checks.

A central direction for future work is to move beyond the canonical potential classes explored here. As suggested by the limitations identified by Huang & Zhang (2022), canonical constructions may not be expressive enough for the open $k = 4$ circle case. The next step is therefore to let agents explore richer non-canonical potential structures, while developing diagnostics that distinguish genuinely new structure from reformulations of known canonical forms. See Appendix G for more details.

We hope that the released verifier, instances, and discovered candidates will help turn the open $k = 4$ circle case into a shared executable challenge for both human mathematicians and AI systems and eventually lead to the resolution of the k-server conjecture.

References

Abouzaid, M., Blumberg, A. J., Hairer, M., Kileel, J., Kolda, T. G., Nelson, P. D., Spielman, D., Srivastava, N., Ward, R., Weinberger, S., and Williams, L. First proof, 2026. URL https://arxiv.org/abs/2602.05192.

Angdinata, D. K., Chen, E., Cummins, C., Eltschig, B., Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Mahns, S., Markosyan, A. H., Muddana, R., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Abc implies that ramanujan's tau function misses almost all primes, 2026a. URL https://arxiv.org/abs/2603.29970.

Angdinata, D. K., Chen, E., Ono, K., Zhang, J., and Zhang, J. On the paucity of lattice triangles, 2026b. URL https://arxiv.org/abs/2603.23928.

Bartal, Y. and Koutsoupias, E. On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical computer science*, 324(2-3):337–345, 2004.

Bein, W. W., Chrobak, M., and Larmore, L. L. The 3-server problem in the plane. *Theoretical Computer Science*, 289 (1):335–354, 2002.

Biswal, R., Ono, K., and Zhang, J. Chebyshev quotients, demazure multiplicities, and dyck-path models, 2026. URL https://arxiv.org/abs/2604.25246.

Brenner, M. P., Cohen-Addad, V., and Woodruff, D. Solving an open problem in theoretical physics using ai-assisted discovery, 2026. URL https://arxiv.org/abs/2603.04735.

Brilliantov, K., Pavutnitskiy, F., Pasechnyuk, D., and Magai, G. Applying language models to algebraic topology: generating simplicial cycles using multi-labeling in wu's formula, 2023. URL https://arxiv.org/abs/2306.16951.

Bubeck, S., Coester, C., Eldan, R., Gowers, T., Lee, Y. T., Lupsasca, A., Sawhney, M., Scherrer, R., Sellke, M., Spears, B. K., Unutmaz, D., Weil, K., Yin, S., and Zhivotovskiy, N. Early science acceleration experiments with gpt-5, 2025. URL https://arxiv.org/abs/2511.16072.

Charton, F., Ellenberg, J. S., Wagner, A. Z., and Williamson, G. Patternboost: Constructions in mathematics with a little help from ai, 2024. URL https://arxiv.org/abs/2411.00566.

Chen, E., Cummins, C., Eltschig, B., Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Markosyan, A., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Dead ends in square-free digit walks, 2026a. URL https://arxiv.org/abs/2602.05095.

Chen, E., Cummins, C., Eltschig, B., Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Markosyan, A., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Almost all primes are partially regular, 2026b. URL https://arxiv.org/abs/2602.05090.

Chen, E., Cummins, C., GSM, Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Markosyan, A., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Fel's conjecture on syzygies of numerical semigroups, 2026c. URL https://arxiv.org/abs/2602.03716.

Chrobak, M. and Larmore, L. L. An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.

Chrobak, M., Karloof, H., Payne, T., and Vishwnathan, S. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

Coester, C. and Koutsoupias, E. Towards the k-server conjecture: A unifying potential, pushing the frontier to the circle, 2021. URL https://arxiv.org/abs/2102.10474.

Davies, A., Veličković, P., Buesing, L., and et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600:70–74, 2021.

Davis, D., Ivanisvili, P., Tao, T., and contributors. Optimization constants in mathematics. GitHub repository, 2026. URL https://github.com/teorth/optimizationproblems.

Dobriban, E. Solving a research problem in mathematical statistics with ai assistance, 2025. URL https://arxiv.org/abs/2511.18828.

Epoch AI. Frontiermath open problems, 2026. URL https://epoch.ai/frontiermath/open-problems.

Feng, T., Trinh, T., Bingham, G., Kang, J., Zhang, S., hyun Kim, S., Barreto, K., Schildkraut, C., Jung, J., Seo, J., Pagano, C., Chervonyi, Y., Hwang, D., Hou, K., Gukov, S., Tsai, C.-C., Choi, H., Jin, Y., Li, W.-Y., Wu, H.-A., Shiu, R.-A., Shih, Y.-S., Le, Q. V., and Luong, T. Semi-autonomous mathematics discovery with gemini: A case study on the erdős problems, 2026a. URL https://arxiv.org/abs/2601.22401.

Feng, T., Trinh, T. H., Bingham, G., Hwang, D., Chervonyi, Y., Jung, J., Lee, J., Pagano, C., hyun Kim, S., Pasqualotto, F., Gukov, S., Lee, J. N., Kim, J., Hou, K., Ghiasi, G., Tay, Y., Li, Y., Kuang, C., Liu, Y., Lin, H., Liu, E. Z., Nayakanti, N., Yang, X., Cheng, H.-T., Hassabis, D., Kavukcuoglu, K., Le, Q. V., and Luong, T. Towards autonomous mathematics research, 2026b. URL https://arxiv.org/abs/2602.10177.

Fiat, A., Rabani, Y., and Ravid, Y. Competitive k-server algorithms. *Journal of Computer and System Sciences*, 48(3):410–428, 1994.

Georgiev, B., Gómez-Serrano, J., Tao, T., and Wagner, A. Z. Mathematical exploration and discovery at scale, 2025. URL https://arxiv.org/abs/2511.02864.

Gideoni, Y., Tang, Y., Risi, S., and Gal, Y. Random baselines for simple code problems are competitive with code evolution. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025. URL https://openreview.net/forum?id=rbVIpbmbTc.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., de Oliveira Santos, E., Järviniemi, O., Barnett, M., Sandler, R., Vrzala, M., Sevilla, J., Ren, Q., Pratt, E., Levine, L., Barkley, G., Stewart, N., Grechuk, B., Grechuk, T., Enugandla, S. V., and Wildon, M. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai, 2025. URL https://arxiv.org/abs/2411.04872.

Guevara, A., Lupsasca, A., Skinner, D., Strominger, A., and Weil, K. Single-minus gluon tree amplitudes are nonzero, 2026. URL https://arxiv.org/abs/2602.12176.

Huang, Z. and Zhang, H. Deterministic 3-server on a circle and the limitation of canonical potentials, 2022. URL https://arxiv.org/abs/2205.08103.

Jiang, J., Ding, T., and Zhu, Z. Deltaevolve: Accelerating scientific discovery through momentum-driven evolution, 2026. URL https://arxiv.org/abs/2602.02919.

Koutsoupias, E. The k-server problem. *Comput. Sci. Rev.*, 3(2):105–118, May 2009.

Koutsoupias, E. and Papadimitriou, C. H. On the k-server conjecture. *J. ACM*, 42(5):971–983, September 1995.

Lample, G. and Charton, F. Deep learning for symbolic mathematics, 2019.

Lange, R. T., Imajuku, Y., and Cetin, E. Shinkaevolve: Towards open-ended and sample-efficient program evolution, 2025. URL https://arxiv.org/abs/2509.19349.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Manasse, M. S., McGeoch, L. A., and Sleator, D. D. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990. ISSN 0196-6774. doi: https://doi.org/10.1016/0196-6774(90)90003-W. URL https://www.sciencedirect.com/science/article/pii/019667749090003W.

Nagda, A., Raghavan, P., and Thakurta, A. Reinforced generation of combinatorial structures: Hardness of approximation, 2025. URL https://arxiv.org/abs/2509.18057.

Nagda, A., Raghavan, P., and Thakurta, A. Reinforced generation of combinatorial structures: Ramsey numbers, 2026. URL https://arxiv.org/abs/2603.09172.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Peifer, D., Stillman, M., and Halpern-Leistner, D. Learning selection strategies in buchberger's algorithm, 2020.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, January 2024.

Sharma, A. Openevolve: an open-source evolutionary coding agent, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve.

Sothanaphan, N. Resolution of erdős problem #728: a writeup of aristotle's lean proof, 2026. URL https://arxiv.org/abs/2601.07421.

Surina, A., Mansouri, A., Quaedvlieg, L., Seddas, A., Viazovska, M., Abbe, E., and Gulcehre, C. Algorithm discovery with llms: Evolutionary search meets reinforcement learning, 2025. URL https://arxiv.org/abs/2504.05108.

Wagner, A. Z. Constructions in combinatorics via neural networks, 2021.

Wan, C., Dai, X., Wang, Z., Li, M., Wang, Y., Mao, Y., Lan, Y., and Xiao, Z. Loongflow: Directed evolutionary search via a cognitive plan-execute-summarize paradigm, 2025. URL https://arxiv.org/abs/2512.24077.

Wang, E. Y., Motwani, S., Roggeveen, J. V., Hodges, E., Jayalath, D., London, C., Ramakrishnan, K., Cipcigan, F., Torr, P., and Abate, A. Horizonmath: Measuring ai progress toward mathematical discovery with automatic verification, 2026. URL https://arxiv.org/abs/2603.15617.

Wang, Y., Su, S.-R., Zeng, Z., Xu, E., Ren, L., Yang, X., Huang, Z., He, X., Ma, L., Peng, B., Cheng, H., He, P., Chen, W., Wang, S., Du, S. S., and Shen, Y. Thetaevolve: Test-time learning on open problems, 2025. URL https://arxiv.org/abs/2511.23473.

Zhang, Q., Shen, A., Zhang, B., Jiang, H., and Du, B. Online matching with stochastic rewards: Provable better bound via adversarial reinforcement learning. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59806–59826. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24bf.html.

Zheng, T., Deng, Z., Tsang, H. T., Wang, W., Bai, J., Wang, Z., and Song, Y. From automation to autonomy: A survey on large language models in scientific discovery, 2025. URL https://arxiv.org/abs/2505.13259.

A. Mathematical Background and Benchmark Derivation

A.1. The k-server problem and WFA

We focus on the *k-server conjecture*, widely regarded as the "holy grail of competitive analysis" and a flagship open problem in online algorithms. Since its formulation by Manasse, McGeoch, and Sleator in 1990 (Manasse et al., 1990), the conjecture has shaped the development of competitive analysis as a field (Chrobak et al., 1991; Chrobak & Larmore, 1991; Fiat et al., 1994; Koutsoupias, 2009; Bartal & Koutsoupias, 2004; Bein et al., 2002). It asks whether, for every parameter k and every metric space, there exists a deterministic online algorithm whose total cost is at most $c = k$ times the offline optimum up to an additive constant. We refer to c as the *competitive ratio*, and call an algorithm satisfying this guarantee *c-competitive*.

The best known general guarantee is due to Koutsoupias & Papadimitriou (1995), who proved that the *Work Function Algorithm* (WFA) — widely viewed as the leading candidate for resolving the conjecture — is $(2k-1)$-competitive for any k and any metric space. In many resolved metric families where WFA *is* k-competitive, the proof proceeds by exhibiting a *potential function* that satisfies a prescribed collection of inequalities (Koutsoupias, 2009; Koutsoupias & Papadimitriou, 1995). Recent attention has concentrated on the *circle* metric, a highly structured setting that nonetheless remains a major obstacle for existing potentials. Coester & Koutsoupias (2021) introduced a unifying potential that resolves many known tight cases, yet it provably does not extend to the circle. By contrast, Huang & Zhang (2022) settled the $k = 3$ circle case by introducing a circle-specific modification, underscoring that the remaining difficulty appears to be discovering the right potential function.

In this terminology, a *k-competitive* potential is one whose inequalities would certify the conjectured optimal competitive ratio $c = k$. We sometimes also consider a relaxed *$(k+1)$-competitive* target, obtained by setting $c = k + 1$ in the same potential-inequality framework. This relaxation is weaker: a $(k+1)$-competitive certificate would not resolve the k-server conjecture, but it is still useful diagnostically because it tests whether a proposed potential form can satisfy the benchmark inequalities under a slightly less stringent competitive ratio. In our experiments, such relaxed candidates are therefore treated as evidence about the expressiveness of a potential family, not as solutions to the main k-competitive benchmark.

A.2. Work Function

We call a multiset of k points in the metric space M a configuration (of servers) and denote $\mathcal{C}$ with a set of all possible configurations.

The central theoretical object in the k-server problem is the *work function*.

Definition A.1. The work function $w_t : \mathcal{C} \to \mathbb{R}$ maps a configuration C to the minimum cost of serving the request sequence $r_1, \ldots, r_t$ and ending in configuration C.

It satisfies the recurrence

$$w_{t+1}(X) = T_r(w_t)(X) = \min_{\substack{Y \in \mathcal{C} \\ r_{t+1} \in Y}} \{w_t(Y) + d(X, Y)\},$$

where the minimum is taken over all configurations Y that contain r_{t+1} and T_r is the convenient notation for operator returning the "next" work function. Here, with slight abuse of notation, $d(X, Y)$ denotes the minimum matching cost between configurations X and Y.

The work function enjoys several important structural properties, including monotonicity, quasi-convexity, and 1-Lipschitzness (Koutsoupias, 2009).

Definition A.2 (Extended Cost)**.** Let w be a work function. The *extended cost* is defined as

$$\nabla(w, r) := \max_{X \in \mathcal{C}} \big(T_r(w)(X) - w(X)\big).$$

Definition A.3 (OPT Increase)**.** Let w be a work function. The *OPT increase* is defined as

$$\Delta\,\mathrm{OPT}(w, r) := \min_{X \in \mathcal{C}} T_r(w)(X) - \min_{X \in \mathcal{C}} w(X).$$

A.3. Competitiveness of the WFA

The Work Function Algorithm (WFA) uses the work function explicitly to determine its moves. While WFA is defined algorithmically, the work function itself is a more general analytical object.

If the current configuration after t requests is X_t, WFA chooses the next configuration according to

$$X_{t+1} = \arg\min_{Y \in \mathcal{C}} \{w_{t+1}(Y) + d(X_t, Y)\}.$$

Lemma A.4. *For any request sequence $\rho = (r_1, \ldots, r_t)$,*

$$\mathrm{OPT}(\rho) + \mathrm{WFA}(\rho) \leq \sum_{i=0}^{t-1} \nabla(w_i, r_{i+1}).$$

A standard way to prove c-competitiveness of WFA is to construct a potential Φ on work functions.

Theorem A.5. *Suppose there exists Φ such that:*

1. *For all w and requests r,*
$$\Phi(w') - \Phi(w) \geq \nabla(w, r),$$
where $w' = T_r(w)$ is the work function after r,
2. *For all w,*
$$\Phi(w) \leq (c+1) \min_{X \in \mathcal{C}} w(X) + \mathrm{const}.$$

Then WFA is c-competitive.

Proof. Using Lemma A.4 and telescoping,

$$\mathrm{OPT}(\rho) + \mathrm{WFA}(\rho) \leq \sum_i \nabla(w_i, r_{i+1}) \leq \Phi(w_{t+1}) - \Phi(w_0),$$

and the second property yields the claim after absorbing constants. □

A.4. Work-Function Graph

We first note that work functions are shift invariant in the following sense:

Lemma A.6 (Shift-Invariance)**.** *For every $w : \mathcal{C} \mapsto \mathbb{R}$, request $r \in M$ and scalar $\alpha \in \mathbb{R}$ we have:*

$$T_r(w + \alpha \cdot \mathbf{1}) = T_r(w) + \alpha \cdot \mathbf{1},$$

where $+\alpha \cdot \mathbf{1}$ means adding α to every value of the work function.

Consequently,

$$\nabla(w + \alpha \cdot 1, r) = \nabla(w, r), \quad \Delta\mathrm{OPT}(w + \alpha \cdot \mathbf{1}, r) = \Delta\mathrm{OPT}(w, r)$$

Definition A.7 (Normalized Work Function)**.** For a work function w we call

$$\widehat{w} = w - \min_{X \in \mathcal{C}} w(X) \cdot \mathbf{1}$$

to be normalized work function.

Definition A.8 (Reachable state)**.** Normalized work function $\widehat{w}$ is reachable from w_0 if there exists a finite sequence of requests $r_1, ..., r_n$ such that $\widehat{w} = \widehat{w_n}$, where $w_i = T_{r_i}(w_{i-1})$

Lemma A.9 (Finite number of Reachable states)**.** *For a finite metric space M with integer (for simplicity) distances, the number of reachable normalized work functions from an integer-valued work function, w_0, is finite.*

Proof. First, we note that the application of T_r operator keeps the work function to be integer valued since all distances are as well integers.

Next we note that because of Lipschitzness of the work function all normalized work functions values in the given metric space are bounded:

$$\forall w, X : \widehat{w}(X) \leq \max_{X,Y \in \mathcal{C}} d(X, Y)$$

Given that all values of the normalized work functions are bounded and integer valued we have that there are only finite number of reachable normalized work functions. □

Definition A.10 (Work Function Graph)**.** The work function graph is the directed graph $G = (V, E)$ whose vertex set is the finite set V of reachable normalized work functions and where for each $\widehat{w} \in V$ and request $r \in M$ we add the edge

$$e = (\widehat{w_u}, r, \widehat{w_v}) \quad \text{whenever} \quad \widehat{w_v} = \widehat{T_r(\widehat{w_u})}$$

For such an edge, define its weight

$$\text{weight}_c(u, r, v) = \nabla(w_u, r) - (c + 1)\Delta\text{OPT}(w_u, r)$$

Lemma A.11 (Normalized Potential Criterion)**.** *Suppose* $\Psi : V \mapsto \mathbb{R}$ *satisfies*

$$\Psi(\widehat{w_v}) - \Psi(\widehat{w_u}) \geq \text{weight}_c(u, r, v)$$

for every edge $e = (u, v, r) \in E$*. Define*

$$\Phi(w) := \Psi(\widehat{w}) + (c + 1) \min_{X \in \mathcal{C}} w(X)$$

Then for every reachable work function w *and* $w' = T_r(w)$*,*

$$\Phi(w') - \Phi(w) \geq \nabla(w, r) \quad \textit{and} \quad \Phi(w) \leq (c + 1) \min_X w(X) + B,$$

for some constant B *that does not depend on the request sequence to reach* w*.*

Proof. By definition,

$$\Phi(w') - \Phi(w) = \Psi(\widehat{w'}) - \Psi(\widehat{w}) + (c + 1)\Delta\text{OPT}(w, r)$$

The assumed inequality gives:

$$\Phi(w') - \Phi(w) \geq \text{weight}_c(u, r, v) + (c + 1)\Delta\text{OPT}(w, r) = \nabla(w, r)$$

Moreover, since V is finite we can set $B = \max_{v \in V} \Psi(\widehat{w_v})$, then since w is reachable we have:

$$\Phi(w) = \Psi(\widehat{w}) + (c + 1) \min_{X \in \mathcal{C}} w(X) \leq (c + 1) \min_{X \in \mathcal{C}} w(X) + B$$

□

Therefore combining all these facts we get the finiteness of the BFS procedure to enumerate all possible normalized work functions, the sufficient of computing potential on the normalized work functions. So the candidate satisfying all inequalities in the Work Function Graph of the finite metric space with integer values constitutes the competitiveness of the Work Function Algorithm on this metric space.

A.5. Canonical Potentials

Let $n \in \mathbb{N}$, an index matrix $I \in \mathbb{Z}^{h \times k}$, and a symmetric coefficient matrix $C \in \mathbb{R}^{n \times n}$. We define the *canonical potential* parameterized by auxiliary points $a_1, \ldots, a_n \in M$ as

$$\Phi(w; a_1, \ldots, a_n) = \sum_{i=1}^{h} w\big(a_{I[i,1]}, \ldots, a_{I[i,k]}\big) - \sum_{1 \leq i < j \leq n} C[i, j]\, d(a_i, a_j),$$

where:

- $I[i,j]$ denotes the (i,j)-th entry of the index matrix,
- $w(a_{i_1},\dots,a_{i_k})$ denotes the value of the work function at configuration $(a_{i_1},\dots,a_{i_k})$,
- negative indices are interpreted via the convention $a_{-i} = \overline{a_i}$ (the antipode of a_i on the circle).

The final potential is defined by minimizing over the auxiliary points:

$$\Phi(w) = \min_{a_1,\dots,a_n \in M} \Phi(w; a_1,\dots,a_n).$$

Due to the 1-Lipschitz property of the work function, any canonical potential of the above form satisfies the upper bound

$$\Phi(w) \;\le\; h \cdot \min_{X\in\mathcal{C}} w(X) + \text{const},$$

where h is the number of rows in the index matrix I, and the constant depends only on the metric space and the chosen coefficient matrix C (but not on the request sequence). Thus, the number of rows h effectively controls the achievable competitiveness bound encoded by the potential.

We also note that this general framework was introduced in Huang & Zhang (2022) in a slightly more restricted form. In that formulation, the index matrix is required to contain a row of the form $(-1,-1,\dots,-1)$, and in all other rows the index 1 appears explicitly, with all distance coefficients involving point 1 set to zero. Our definition relaxes these structural constraints while preserving the same underlying minimization template.

In our implementation, instead of storing the full symmetric matrix C, we use a coefficient vector of size $\binom{n}{2}$.

Examples from the Literature. The unifying potential of Coester & Koutsoupias (2021) fits into this framework with

$$n := k, \qquad I := \begin{pmatrix} 1 & 2 & \dots & k \\ -1 & 2 & \dots & k \\ -2 & -2 & \dots & k \\ \vdots & \vdots & \ddots & \vdots \\ -k & -k & \dots & -k \end{pmatrix}, \qquad C := 0.$$

For $k=3$, the potential of Huang & Zhang (2022) is given by

$$n := 4, \qquad I := \begin{pmatrix} -1 & -1 & -1 \\ 1 & 2 & -3 \\ 1 & 3 & -4 \\ 1 & 4 & -2 \end{pmatrix}, \qquad C := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Equivalently, these potentials can be written in the original form:

$$\Phi(w) = \min_{x_1,\dots,x_k \in M} \sum_{i=0}^{k} w(\overline{x_i},\dots,\overline{x_i},x_{i+1},\dots,x_k)$$

and

$$\Phi(w) = \min_{u,x,y,z} w(\overline{u},\overline{u},\overline{u}) + w(u,x,\overline{y}) + w(u,y,\overline{z}) + w(u,z,\overline{x}) - \\ - d(x,y) - d(x,z) - d(y,z)$$

correspondingly.

B. Benchmark Specification, Verifier and Implementation Details

B.1. Work-Function Graph Construction

For each finite metric instance, the verifier first constructs an explicit work-function graph. Let M be the finite metric space, let k be the number of servers, and let

$$\mathcal{C} = \{\text{multisets of } k \text{ points of } M\}$$

be the configuration space. A work function is represented as a vector $w \in \mathbb{R}^{|\mathcal{C}|}$, where the coordinate $w(X)$ is indexed by a configuration $X \in \mathcal{C}$. Given a request $r \in M$, the work-function update operator is

$$T_r(w)(X) := \min_{\substack{Y \in \mathcal{C} \\ r \in Y}} \big\{ w(Y) + d(X, Y) \big\}, \qquad X \in \mathcal{C},$$

where $d(X, Y)$ denotes the minimum matching distance between the two configurations.

The graph is built over normalized work functions. For any vector w, set

$$\widehat{w} := w - \min_{X \in \mathcal{C}} w(X) \cdot \mathbf{1}.$$

The instance builder starts from the normalized initial work function and runs a breadth-first exploration over reachable normalized work functions. When a state u, represented by w_u, is popped from the queue, the builder applies every request in the instance request alphabet. For an ordinary k-server request $r \in M$, it computes

$$w' = T_r(w_u), \qquad w_v = \widehat{w'}.$$

If w_v has not been seen before, it is assigned a new vertex id and added to the queue. The builder then emits the directed edge

$$e = (u, r, v).$$

The edge stores the right-hand side of the normalized potential inequality. For a target competitiveness parameter c, define

$$\mathrm{ext}(w_u, r) := \max_{X \in \mathcal{C}} \big(T_r(w_u)(X) - w_u(X) \big)$$

and

$$\Delta_{\mathrm{opt}}(w_u, r) := \min_{X \in \mathcal{C}} T_r(w_u)(X) - \min_{X \in \mathcal{C}} w_u(X).$$

The edge weight used by the verifier is

$$\gamma_c(u, r, v) := \mathrm{ext}(w_u, r) - (c + 1)\Delta_{\mathrm{opt}}(w_u, r).$$

Note that, in the main text, we write γ without the subscript c; this denotes γ_k, since the default competitive ratio considered in the k-server literature is $c = k$.

Thus a submitted normalized potential Ψ is accepted on the edge if

$$\Psi(w_v) - \Psi(w_u) \geq \gamma_c(u, r, v).$$

In implementation, each metric instance is serialized as a precomputed graph: the work-function context, the array of reachable nodes, and the edge list with its stored weights. The context supplies the configuration-indexing maps, configuration distances, and work-function update routines, while the evaluator only needs to instantiate the submitted potential, compute $\Psi(w)$ once per visited node, and scan the stored edge list. This separation keeps candidate evaluation fast and makes the benchmark deterministic: all candidates are checked against exactly the same finite system of inequalities.

For the circle instances, distances are scaled to integers. Since the update operator preserves integrality, every reachable normalized work function and every stored edge weight is integer-valued. This also makes the breadth-first exploration finite: normalized reachable work functions are integer vectors bounded by the diameter of the configuration metric. The verifier may expose a floating-point interface to candidate programs for convenience, but the underlying instance generation is based on exact integer-valued work-function updates.

B.2. k-taxi augmentation construction

The ordinary work-function graph contains one outgoing transition for each point request $r \in M$. To obtain a richer finite approximation of the circle problem, we augment this graph with transitions induced by k-taxi requests. A k-taxi request is a pair (s, t), interpreted as a passenger request with start s and destination t. In the "easy" k-taxi model used for this augmentation, serving (s, t) charges the total distance traveled by the chosen server. Coester & Koutsoupias (2021) observed that such a request can be simulated by a sequence of ordinary k-server requests along the shortest path from s to t. Accordingly, we view a taxi transition as a compressed representation of a limiting sequence of ordinary request transitions.

Formally, let $r_1, \ldots, r_m$ be equally spaced points along a shortest path from $s = r_1$ to $t = r_m$. For a work function w, define

$$T^{\text{taxi}}_{s\to t}(w) := \lim_{m\to\infty} T_{r_m} \circ T_{r_{m-1}} \circ \cdots \circ T_{r_1}(w).$$

On the circle the shortest path is unique unless s and t are antipodal; in the antipodal case the implementation fixes a deterministic orientation, e.g. clockwise, to make the transition canonical.

The limiting update has a useful closed description. Let $\operatorname{supp}(w)$ denote the support configurations of a work function. After first issuing the request at s, the taxi movement from s to t replaces s by t in every support configuration and adds the path length $d(s, t)$ to the corresponding support value:

$$\operatorname{supp}\big(T^{\text{taxi}}_{s\to t}(w)\big) = \{\, S - s + t : S \in \operatorname{supp}(T_s(w)) \,\},$$

and

$$T^{\text{taxi}}_{s\to t}(w)(S - s + t) = T_s(w)(S) + d(s, t), \qquad S \in \operatorname{supp}(T_s(w)).$$

The full work-function vector is then recovered by the standard lower-envelope extension from its support:

$$T^{\text{taxi}}_{s\to t}(w)(X) = \min_{S\in\operatorname{supp}(T_s(w))} \Big\{ T_s(w)(S) + d(s, t) + d(S - s + t, X) \Big\}, \qquad X \in \mathcal{C}.$$

In the augmented instance builder, the request alphabet is extended from ordinary point requests to

$$\mathcal{R}_{\text{aug}} = M \,\cup\, \mathcal{R}_{\text{taxi}},$$

where $\mathcal{R}_{\text{taxi}}$ is the finite set of taxi pairs selected for the circle discretization. For every reachable node u and every $(s, t) \in \mathcal{R}_{\text{taxi}}$, the builder computes

$$w' = T^{\text{taxi}}_{s\to t}(w_u), \qquad w_v = \widehat{w'},$$

adds w_v as a vertex if it is new, and emits the augmented edge

$$e = (u, (s, t), v).$$

The stored edge weight is the accumulated normalized inequality weight of the ordinary request sequence represented by the taxi transition:

$$\gamma^{\text{taxi}}_c(u, (s, t), v) := \lim_{m\to\infty} \sum_{i=1}^{m} \Big[\operatorname{ext}(w_{i-1}, r_i) - (c + 1)\big(\min_X w_i(X) - \min_X w_{i-1}(X)\big) \Big],$$

where $w_i = T_{r_i}(w_{i-1})$ and $w_0 = w_u$. The verifier treats this edge exactly like an ordinary graph edge and checks

$$\Psi(w_v) - \Psi(w_u) \geq \gamma^{\text{taxi}}_c(u, (s, t), v).$$

The purpose of the augmentation is not to change the candidate-potential interface, but to enlarge the finite inequality system with constraints that approximate long, structured request sequences on the circle. This makes the finite verifier more stringent while preserving the same evaluation loop: compute potentials on nodes once, scan edges, and count violated inequalities. For the largest taxi-augmented circle instances, the builder additionally uses the rotational and reflection symmetries of the circle to reduce redundant states; when exact visited-set storage becomes too large, a Bloom-filter-based visited structure can be used during graph exploration, in which case reported node and edge counts should be interpreted as a lower bound.

B.3. Notes on the largest instance, $k = 4, m = 8$ with k-taxi

Enumerating the largest instance, namely $k = 4$, $m = 8$ with k-taxi augmentation, via a straightforward BFS required prohibitive memory. To make enumeration feasible, we used Bloom filters to maintain the set of visited nodes. As a result, the counts reported in Table 1 are not exact; they should be interpreted as lower bounds.

In addition, for instances with k-taxi requests we applied an extra node-filtering procedure exploiting the rotational and reflection symmetries of the circle, which further reduced the effective number of *distinct* nodes.

In addition, explicit lookup-table memorization is practically unattractive at the scale of the hardest instances. For the $k = 4, m = 8$ k-taxi instance, constructing a Bellman–Ford certificate would require running shortest-path relaxation on a graph with roughly 7×10^7 nodes and 7×10^9 edges. A naive Bellman–Ford computation has $O(|V||E|)$ complexity, which corresponds to on the order of 5×10^{17} edge relaxations for this instance, before accounting for memory bandwidth, graph storage, or Python overhead. Thus, the benchmark is not designed to reward agents for recovering a dense tabular potential by exhaustive graph computation.

This approximation affects only the completeness of the generated finite graph, not the soundness of any reported violation. A violated inequality on an included edge is still a valid refutation of the submitted potential for that included subinstance. However, the converse is not true: zero violations on the Bloom-filter-generated graph would not certify the exact finite $k = 4, m = 8$ k-taxi instance, since false positives may cause reachable nodes and their incident constraints to be omitted.

This is consistent with the broader role of our finite benchmarks. Even exactly enumerated finite discretizations are sound but incomplete relaxations of the infinite circle problem: they can refute candidate potentials by exhibiting violations, but satisfying all inequalities on a finite discretization does not by itself prove the infinite case. The approximate $k = 4, m = 8$ k-taxi instance introduces an additional, explicitly marked layer of incompleteness relative to the exact finite graph, and we therefore treat it as exploratory rather than as an official zero-violation certification target.

B.4. Potential Definition

Our reference implementation expects submissions to define a `Potential` object with a lightweight callable interface:

```
class Potential:
   def __init__(self, context: WFContext, **kwargs):
      pass

   def __call__(self, wf: np.ndarray) -> float:
      pass
```

Here, `WFContext` is a helper class that encapsulates the metric instance (via the distance matrix) and provides utilities for working with work functions and configurations:

```
class WFContext:
   def __init__(self, k: int, distance_matrix: np.ndarray):
      pass

   def config_to_idx(self, config: Tuple[int, ...]) -> int:
      """
      Maps `config` (a tuple of point indices) to its index in the work-function
      vector, i.e., w(C) = wf[idx] where config_to_idx(C) = idx.
      """
      pass

   def idx_to_config(self, idx: int) -> Tuple[int, ...]:
      """
      Inverse mapping of `config_to_idx`.
      """
      pass

   def update_wf(self, wf: np.ndarray, r: int) -> np.ndarray:
      """
      Returns the updated work function after serving request r.
      """
```

```
        pass

    ...
```

We support precomputation via `__init__` so that a submission can amortize expensive setup and keep `__call__` fast. This is particularly important because evaluation requires many repeated calls to the potential over large work-function graphs. For example, our `CanonicalPotential` implementation precomputes indices of relevant configurations:

```
class CanonicalPotential:
    def __init__(
        self, context, n: int,
        index_matrix: np.ndarray,
        coefs: np.ndarray
    ):
        # shape: [N_CANDIDATES x k]
        self.candidate_config_idxes = ...

        # shape: [N_CANDIDATES]
        self.penalties = ...

    def __call__(self, wf: np.ndarray) -> float:
        wf_part = wf[self.candidate_config_idxes].sum(axis=-1)
        return (wf_part - self.penalties).min()
```

B.5. Metrics Definition

Normalized scores and violation diagnostics. In some experiments we report the normalized violation score

$$1 - \frac{\#\text{violations}}{\#\text{edges}},$$

which lies in $[0, 1]$ and is useful as a smooth optimization signal, especially when combining multiple instances into a single objective during search. For example, an agent optimizing simultaneously over several discretizations may use normalized scores to balance instances with very different numbers of edges. However, normalized scores should not be interpreted as the primary correctness criterion. On very large graphs, they can be misleadingly close to one even when a candidate is mathematically invalid: a potential with thousands of violated inequalities among billions of edges may have an apparently near-perfect normalized score while still failing the zero-violation target. For this reason, our main evaluation emphasizes raw violation counts and zero-violation success rates. In the open $k = 4$ regimes, where the graphs are particularly large, we report absolute violation counts rather than relying on normalized scores.

In addition to raw violation counts, the verifier records diagnostic information intended to make failures more interpretable. These diagnostics include violation magnitudes, the number of violated symmetry orbits when symmetry reduction is available, hard-edge statistics, and special cases such as violations on edges with $\Delta\mathrm{OPT} = 0$. For held-out or private evaluation instances, we also report whether candidates generalize across discretization levels or hidden perturbations. These diagnostics are not substitutes for the zero-violation criterion, but they help distinguish candidates that fail by many unrelated constraints from near-feasible candidates whose remaining failures are localized or structurally informative.

Core Metrics.

`violations_k`. Number of edges violating the normalized inequality

$$\Phi(\hat{w_v}) - \Phi(\hat{w_u}) + (k+1)\,\Delta\mathrm{OPT}(w_u, r) \;\geq\; \nabla(w_u, r).$$

Equivalently, this corresponds to the step inequality

$$\Phi(w_v) - \Phi(w_u) \;\geq\; \nabla(w_u, r_{t+1})$$

in the normalized work-function graph. This is the *primary optimization target*: the smaller `violations_k`, the closer the potential is to satisfying all required inequalities. The optimal potential achieves `violations_k` $= 0$.

violations_k_l{1,2,inf}. The ℓ_1, ℓ_2, and ℓ_∞ norms of violation magnitudes. These quantify the *severity* of violations. For example, a potential may have many small violations (large ℓ_1 but small ℓ_∞), suggesting it is close to feasibility and may be repairable via small structural changes.

violations_dmin_0. Number of violated edges with $\Delta\mathrm{OPT} = 0$. For such edges the inequality reduces to

$$\Phi(\hat{w_v}) - \Phi(\hat{w_u}) \;\geq\; \nabla(w_u, r),$$

with no contribution from the OPT increase term. These transitions are particularly demanding and measure how well the potential handles *zero-OPT-gap* edges.

detected_dmin_0_score. Fraction of "hard" edges (those with $\Delta\mathrm{OPT} = 0$ and $\nabla(w_u, r) > 0$) for which $\Phi(w_v) \neq \Phi(w_u)$. This measures *sensitivity*: whether the potential meaningfully distinguishes difficult transitions. For instance, a trivial constant potential may incur few violations under relaxed metrics, yet will typically fail to detect any such hard edges.

violations_renorm. Number of violations in the *renormalized* graph, evaluated as

$$\Phi\big(\hat{w_v} + \mathbf{1} \cdot \Delta\mathrm{OPT}\big) - \Phi\big(\hat{w_u}\big) \;\geq\; \nabla(w_u, r),$$

where $+\mathbf{1}\Delta\,\mathrm{OPT}$ means adding the same constant to all work function values. This metric serves as a consistency check under normalization.

For example, a simple sum-type potential often performs very well under renormalized evaluation, since normalization removes scaling artifacts. However, such potentials typically perform poorly on `violations_k`, as they implicitly correspond to a very large competitive ratio. Because `violations_k` balances both feasibility and competitive ratio, a low `violations_renorm` alone does *not* indicate a strong potential.

Competitiveness Metrics.

strong_hypothesis_rho. The smallest value ρ such that all edges with $\Delta\mathrm{OPT} > 0$ satisfy

$$\Phi(w_v) - \Phi(w_u) + (\rho + 1)\,\Delta\mathrm{OPT}(w_u, r) \;\geq\; \nabla(w_u, r).$$

This estimates the *tightest attainable competitive ratio* for the given potential structure.

opt_upper_bound. A structural proxy for the competitive ratio; it estimates the growth rate of $\Phi(\cdot)$ with adding same constant value to all work function values.

Interpretation.

- If `opt_upper_bound` $< k+1$, then `violations_k` cannot reach zero since every algorithm is at least k competitive.
- If `strong_hypothesis_rho` $<$ `opt_upper_bound` and `violations_dmin_0` $= 0$, the competitive ratio can potentially be reduced.
- If `strong_hypothesis_rho` $>$ `opt_upper_bound`, the targeted competitive ratio must be increased.

Thus, `strong_hypothesis_rho` reflects the best competitive ratio compatible with the current potential form.

Bellman-Based Guidance Metrics. For diagnostics, we compute the Bellman potential (via Bellman–Ford) in the normalized graph with edge weights

$$(k + 1)\,\Delta\mathrm{OPT} - \nabla.$$

Although such numeric potentials do not yield symbolic proofs, they provide valuable guidance for constructing structured potentials.

`bellman_{node/edge}_{r2,corr}`. Coefficient of determination (R^2) or correlation between:

- proposed potential values Φ, and
- Bellman–Ford potential values,

either across nodes (`node`) or across edges (`edge`, comparing $\Phi(w_v) - \Phi(w_u)$).

High correlation indicates that the candidate potential resembles the Bellman–Ford solution, though this does not guarantee validity.

B.6. Feasibility Tests

Because the zero-violation target is only meaningful if the corresponding finite inequality system is feasible, we include an explicit feasibility diagnostic in the released code:

`tests/test_networkx_bellman_generated_instances.py`.

For a fixed finite work-function graph, the benchmark constraints form a system of difference inequalities

$$\Psi(v) - \Psi(u) \geq \text{weight}_c(u, r, v) \qquad \forall (u, r, v) \in E.$$

Equivalently, after reversing signs, this is a standard shortest-path feasibility problem. The system is feasible if and only if the graph contains no cycle whose total required increase is positive; in the sign-reversed formulation, this is the usual absence of negative cycles. Thus, Bellman–Ford or an equivalent shortest-path oracle gives a complete feasibility test for the finite tabular problem.

The purpose of this test is diagnostic rather than competitive. When feasible, Bellman–Ford returns a numerical potential value for every reachable work function. Such a certificate shows that zero violations are achievable by a finite-state tabular potential, but it is tied to one enumerated graph and is not the kind of compact functional potential sought by the benchmark. In particular, a Bellman–Ford lookup table is not useful for subsequent mathematical analysis and is precisely the kind of instance-specific object that the robustness and anti-memorization protocol is designed to discourage. Nevertheless, the oracle is important because it rules out the possibility that an official finite instance is silently infeasible as a graph-potential problem.

In our released tests, we run this Bellman–Ford/negative-cycle diagnostic on the generated instances for which exact graph enumeration is tractable. The test checks that the official exactly enumerated instances used for zero-violation claims are feasible as raw finite-state systems. Thus, for these instances, failure to find zero violations reflects the difficulty of finding a compact programmatic potential, not infeasibility of the finite inequality system itself. For the open $k = 4$ regime, this distinction is essential: the exactly enumerated $k = 4, m = 6$ k-taxi instance is feasible as a tabular graph system, but no known compact or canonical potential satisfies all of its inequalities. Our best compact candidates still leave a small number of violations, so zero violations in this regime are known to be achievable by a finite-state oracle but remain open within the proof-relevant functional representations targeted by the benchmark.

For the approximate $k = 4, m = 8$ k-taxi instance, we do not claim a complete Bellman–Ford certificate for the full finite graph. As discussed in subsection B.3, this instance is too large for straightforward complete enumeration and is used as a large-scale held-out stress test. Violations found on checked edges remain sound refutations for the checked subinstance, but zero violations on an approximate or subsampled graph should be interpreted only relative to the checked constraints, not as certification of the exact full finite $m = 8$ k-taxi graph.

B.7. Robustness and Overfitting Prevention

The notion of overfitting in k-server-bench differs from standard statistical overfitting in machine learning. A submission is not trained to generalize from samples drawn from an unknown data distribution. Rather, the main concern is that a candidate potential might exploit artifacts of a particular finite enumeration, for example by using a numerical table, a Bellman–Ford-style lookup potential, or case-dependent logic tied to the public work-function graph. Such objects may satisfy many finite inequalities, but they are unlikely to be useful for subsequent mathematical analysis, where the goal is to identify a compact functional potential that can plausibly extend across discretizations and eventually to the infinite circle.

Our current robustness protocol is designed to rule out the most direct forms of representation-level memorization. Candidate potentials are evaluated under mathematically valid transformations of the same instance, including global rescaling of the metric, permutation of the configuration indexing, and additive shifts of the work function. These transformations preserve the underlying mathematical inequalities while invalidating simple submissions that depend on absolute coordinate conventions, array indices, or unnormalized work-function values.

This protocol should not be interpreted as a complete anti-memorization guarantee. In particular, it does not by itself prevent overfitting to the finite public graph, to a cached set of hard edges or hard work-function values, or to a particular discretization level, such as a specific value of m. Therefore, for both $k = 3$ and $k = 4$, we hold out the $m = 8$ instances with k-taxi augmentations as private evaluation instances that should not be used during search. For $k = 4$, this instance is extremely large, so we use a ladder of private subsamples for intermediate evaluation before running the largest held-out evaluation on the $k = 4, m = 8$ k-taxi instance.

Finally, we emphasize that these safeguards are pragmatic rather than foundational. The benchmark is already a sound but incomplete relaxation of the infinite circle problem: a violated inequality is a valid refutation of a candidate on the checked instance, while zero violations on any finite discretization do not by themselves prove the infinite case. The private held-out evaluations play the complementary role of making finite success more informative by reducing the risk that a candidate is tailored only to public constraints, cached hard edges, or a single discretization level. If future systems find ways to exploit the verifier without producing proof-relevant functional potentials, we will treat those cases as benchmark failures and add corresponding prevention mechanisms. Conversely, a new *compact* potential that passes the verifier and survives held-out checks remains scientifically interesting, even if it is specialized to the finite benchmark, because no known compact potential currently solves the hardest $k = 4$ regimes.

C. Agentic Search Protocol

C.1. Search Framework

Legacy evaluator. The following description presents the evaluator used for the $k = 3$ and $k = 4$ experiments reported in the paper. The next paragraph describes the updated evaluator used in the released benchmark. Using the updated evaluator, we were able to reproduce the paper's experimental claims while resolving several implementation issues present in the legacy version.

Our *search* formulation introduces three interfaces: `Potential`, `PotentialFamily`, and `SearchEvaluator`. The `Potential` interface remains unchanged from the non-search setting. The `PotentialFamily` interface is responsible for driving optimization over the space of candidate potentials, typically through an ask/tell interaction loop. Finally, `SearchEvaluator` provides a *proxy* evaluation signal for proposed candidates. In the simplest implementation, it computes the potential on all nodes and returns the total number of violated inequalities; however, this can become prohibitively expensive on larger instances.

The search process proceeds as follows. Within a fixed time budget, `PotentialFamily` is periodically *asked* to propose candidate potentials for evaluation. Each proposed candidate is sent to a worker pool and scored by `SearchEvaluator`. The resulting feedback is then *told* back to `PotentialFamily`, which may use it to update its internal search state and propose improved candidates in subsequent rounds.

At the end of the allotted time, `PotentialFamily` returns a set of final candidates, which are then evaluated on the full benchmark instances using the complete scoring procedure. The score of the search method is defined by the best candidate found during this process.

We provide default implementations for all three components: a naive `Potential`, a canonical `Potential`, a naive `SearchEvaluator` that exhaustively computes all violations, and a naive `PotentialFamily` that simply returns the supplied hyperparameters without adaptation.

Recommended evaluator. We also provide a less restrictive evaluation protocol in which participants are only required to define a `Potential` and output its hyperparameters after a fixed timeout. Under this protocol, submissions are executed as standalone programs with a specified time budget and CPU allocation.

The legacy evaluator was more restrictive and, in our implementation, could lead to instabilities that required frequent experiment restarts. The recommended evaluator avoids these issues while preserving the same final scoring procedure:

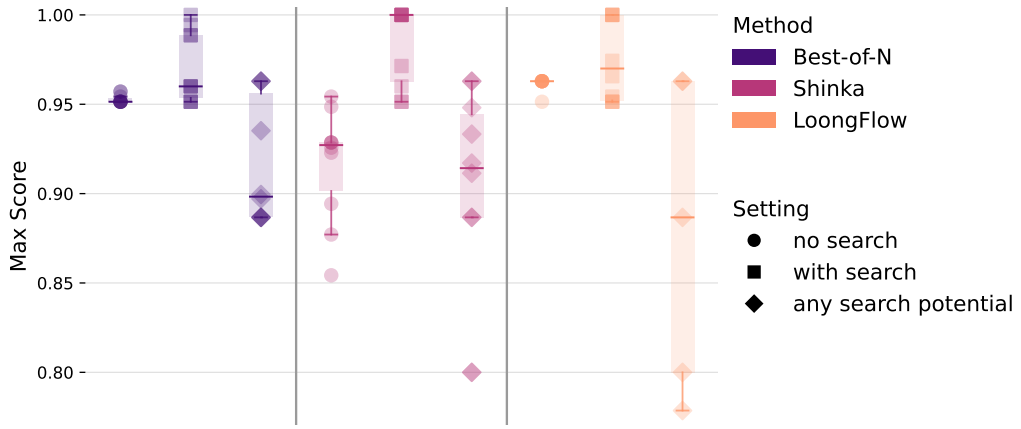

Figure 1. 10 independent runs of SHINKAEVOLVE, BEST-OF-N, LOONGFLOW powered by `gpt-oss-120b` per task design: (i) no search, *canonical hint*, (ii) with search, *canonical-hint*, (iii) with search, *no-hint*. All methods had the same starting conditions per task design and had 100 verifier calls cap. See Table 4 in the Appendix for the alternative representation and additional statistics.

after the search phase, the returned candidates are evaluated by the full benchmark verifier.

C.2. Methods Descriptions and Used Hyperparameters

The hyperparameters for SHINKAEVOLVE and LOONGFLOW were selected based on the example configurations provided in their public repositories. In our preliminary ablations, moderate changes to these parameters did not lead to substantial differences in performance.

Best-of-N. In each Best-of-N iteration, we allow up to two sampling attempts to obtain a parseable response from the LLM. We use temperature $\tau = 0.8$ and cap responses at `max_tokens`=16,000.

SHINKAEVOLVE. SHINKAEVOLVE[2] emphasizes *sample efficiency* through mechanisms that balance exploration and exploitation (e.g., adaptive parent sampling), encourage diversity (e.g., novelty-based rejection sampling of code proposals), and adaptively allocate generation effort across models, while iteratively improving program quality.

In the $k = 3$ testbed experiments, we use the same model for coding, novelty, and meta tasks. We employ *weighted* parent sampling with `parent_selection_ratio`=10. Each prompt includes 4 archive inspirations and 2 top-k inspirations. The archive size is set to 40, and the elite selection ratio is 0.3. Migration is performed every 10 iterations with migration rate 0.1. We use 4 islands. Generations are capped at 80,000 tokens, and patch types are sampled with probabilities 0.6, 0.3, and 0.1 for `diff`, `full`, and `cross`, respectively.

LOONGFLOW. LOONGFLOW replaces "blind" mutation with a more deliberative *Plan–Execute–Summarize* loop, and combines this with an evolutionary memory system and diversity-preserving selection to support longer-horizon improvement and reduce premature convergence.

We use the same model for the planner, executor, and summarizer. The planner is allocated 8 rounds; the executor uses 4 rounds with branching factor 4; and the summarizer is allowed up to 4 ReAct steps. The database uses 3 islands with population size 40, and the parent-sampling parameter is set to 2.

D. $k = 3$ testbed experiments

To study the effect of task design, we consider three settings: (i) *no search* with a canonical potential hint, (ii) *search* with a canonical hint, and (iii) *search* without a canonical hint. Figure 1 shows box plots of the best scores achieved across runs and Table 4 shows summary statistics of these runs.

[2]We used a fork of SHINKAEVOLVE from October 2025. The current version has since received several major updates, so experiments with the latest version may yield different results.

Table 4. Support summary for the Figure 1. "-search" means no search with canonical hint setting, "+search" means with search setting with canonical hint and "any" means with search and without canonical hint setting. The evaluations were done on one $k = 3, m = 6$ without k-taxi augmentations instance.

Metric	Best-of-N			Shinka			LoongFlow		
	-search	+search	any	-search	+search	any	-search	+search	any
Perfect	0	1	0	0	**6**	0	0	4	0
Mean Final	0.953	0.971	0.948	0.916	**0.983**	0.901	0.962	0.975	0.866
Median Final	0.951	0.960	0.963	0.927	**1.000**	0.914	0.963	0.970	0.845
Min Final	**0.951**	**0.951**	0.897	0.854	**0.951**	0.800	**0.951**	**0.951**	0.779
Max Final	0.957	**1.000**	0.963	0.954	**1.000**	0.963	0.963	**1.000**	0.963
Mean Attempt	0.796	0.764	0.778	0.790	**0.916**	0.785	0.424	0.621	0.438
Median Attempt	0.786	0.731	0.800	0.789	**0.951**	0.777	0.649	0.771	0.649

The *search with canonical hint* setting consistently outperforms the alternatives across all methods. In this regime, SHINKAEVOLVE slightly outperforms LOONGFLOW, while Best-of-N never achieves a perfect score, although it comes close. In the *no search* setting, Best-of-N outperforms its SHINKAEVOLVE counterpart and performs only slightly worse than LOONGFLOW, suggesting that without search the differences between agentic scaffolds become marginal. Finally, the *search without canonical hint* setting exhibits the highest variance; however, the best discovered potentials are comparable to those obtained in the no-search setting, indicating difficulty in navigating the unrestricted hypothesis space.

We hypothesize that this behavior is driven by the structure of good potentials in the k-server domain. High-quality solutions tend to exhibit strong symmetry and global structure, making them difficult to discover via incremental, locally guided improvements. This distinction between tasks that favor global construction versus iterative refinement has been noted in Novikov et al. (2025). In particular, when the target is a symmetric mathematical object, large-scale sampling methods such as FunSearch (Romera-Paredes et al., 2024) can be effective due to their ability to explore many candidates, while iterative methods struggle to extract useful signal from the evaluator. Conversely, approaches such as AlphaEvolve (Novikov et al., 2025) and other LLM-driven evolutionary systems are better suited for problems where evaluation is expensive, heuristic guidance is informative, and incremental improvements are possible.

In our setting, the search paradigm remains advantageous despite these challenges, as it enables scaling through additional evaluations and avoids wasting LLM calls on trivial hyperparameter variations, which can instead be handled within the generated programs. This allows frontier models to focus their capacity on proposing structurally meaningful candidates, making search-based methods a natural fit for this benchmark.

E. $k = 4$

E.1. Potentials from the Literature

The canonical-potential framework in Appendix A.5 provides a useful way to compare known constructions. A canonical potential with h work-function terms has the form

$$\Phi(w) = \min_{a_1,\ldots,a_n \in M} \left[\sum_{i=1}^{h} w\big(a_{I[i,1]}, \ldots, a_{I[i,k]}\big) - \sum_{1 \le p < q \le n} C[p,q]\, d(a_p, a_q) \right].$$

By the Lipschitz property of the work function, such a potential is bounded above by

$$h \min_{X \in \mathcal{C}} w(X) + O(1).$$

Consequently, in the standard potential argument, a canonical potential using h work-function terms can certify competitiveness at most $h - 1$. Thus, for the k-server target competitiveness k, the canonical form must use at most $k + 1$ work-function terms. This term count is therefore not merely a matter of notation: it is part of the competitive-ratio certificate.

This observation explains why the $k = 3$ circle potential of Huang & Zhang (2022) does not directly apply to the $k = 4$ case. For $k = 3$, their construction uses exactly four work-function terms, matching the $k + 1$ budget needed for a 3-competitive certificate. However, the construction is not a uniform formula that can be substituted into the $k = 4$ setting without changing its structure.

E.2. Scaling Challenges

Token Consumption Analysis. To better understand the cost side of proxy-based search, we inspected a matched pair of SHINKA runs. The two runs used the same task family, the same prompt construction logic, and the same search scaffold (same numbers of inspirations, archive settings, mutation modes, and proxy-evaluation interface), but differed in the model pool: one run used only `gpt-oss-120b`, while the other used a frontier mixture of `gpt-5.2`, `gpt-5-mini`, and `gpt-oss-120b` (with `gpt-5.2` also handling meta-summaries). Importantly, the prompt only specified the generic k-server potential-search task on the four $k = 4$ circle / circle-taxi instances.

Despite this matched setup, the frontier run consumed far more tokens. The frontier run used 28.8M total tokens across 399 model-generated iterations, compared with 11.5M tokens across 395 iterations for the OSS-only run, a factor of about $2.5\times$. The frontier run did achieve a better best score, improving from 0.4819 (OSS-only) to 0.5937, but this gain came at substantially worse token efficiency.

The main reason was not a higher retry rate or a different interface contract. In fact, retry statistics were comparable, and in some cases slightly worse for the OSS-only run. Instead, the frontier models quickly produced much larger candidate programs, and SHINKA's prompt builder recursively pastes the current program together with several previously discovered programs back into later prompts. As a result, average candidate code size nearly doubled (about 28.4k versus 14.2k characters), and average prompt size more than doubled (about 203k versus 90.5k characters). This creates a self-amplifying long-context loop: larger code leads to larger prompts, which in turn encourages still larger edits.

Search Summary. We carried out a broad $k = 4$ search campaign across multiple regimes, ranging from early canonical-structured searches to later structurally hinted and interface-ablated setups. In total, the saved outputs contain 44 launches, 139 individual runs, and approximately $20,429$ LLM-generated evaluated candidates ($9,048$ stored SHINKA programs and $11,381$ LOONGFLOW evaluation artifacts). The largest share of this effort was spent in the structurally hinted family, which alone accounts for roughly $10,577$ evaluated candidates. Despite this scale, no experiment produced a complete $k = 4$ solution. The best saved result reached combined score 0.9941 with violation vector [0, 96, 9427, 663], so even the strongest candidate still incurred about 9.4k violations on the hardest taxi-augmented $m = 6$ instance.

The remaining regimes support the same conclusion. In the early canonical / fixed-structure family, the best saved run reached combined score 0.9844 with violation vector [0, 368, 10524, 2020], while one intermediate candidate briefly reduced the total violation count to 7249 with vector [0, 32, 515, 6702]. Later interface-ablation runs also improved incumbents but did not change the qualitative picture: the best PotentialFamily (without SearchEvaluator and fixed CanonicalPotential) run reached 0.9092 with [114, 11688, 65128, 3294], and the best run with simplified evaluation interface follow-up reached 0.8579 with [210, 15752, 38282, 7762].

Most of these experiments were run with 'gpt-oss' models rather than frontier models. The reason is the cost pathology discussed in the previous paragraph: frontier models tend to generate substantially larger candidate programs, and both SHINKA and LOONGFLOW construct new prompts by pasting in multiple complete prior solutions. As solution size grows, prompt size grows with it, so token usage increases superlinearly over the course of a run. In practice, this made frontier-model sweeps much more expensive without changing the qualitative $k = 4$ outcome, and therefore most of the large-scale search effort reported here relied on OSS models.

E.3. Structural Hints

Unifying Hint. In addition to the standard task definition, canonical-potential description, and interface prompt, we provided the following hint:

Unifying Hint

The particularly strong 'index_matrix' is the following:
[[1, 2, 3, 4], [-1, 2, 3, 4], [-2, -2, 3, 4], [-3, -3, -3, 4], [-4, -4, -4, -4]]
Search for matrices around this matrix and coefficients that are suited for this matrix.

One suggestion produced by ShinkaEvolve was to consider $n = 5$ instead of $n = 4$ while keeping the same `index_matrix`. Intuitively, this introduces an additional auxiliary point into the minimization, which interacts with the existing points only through the distance coefficients. More details are given in subsection E.5.

We also exhaustively checked all coefficient vectors with support in $\{-1, 0, 1\}$ for the given matrix, and confirmed that 14 was indeed the minimum achievable value in this restricted search space.

Symmetry Hint.

Symmetric Hint

Try satisfying the following requirements for your search over matrices and coefficients:
- Fix n = 5
- One row should be all -1's
- Every other row should contain 1
- Every other index should be balanced: appear the same number of times with positive and negative signs

Do the search around these matrices and suiting coefficients.

Under this hint, we additionally explored several matrices satisfying these structural requirements, including:

$$\begin{pmatrix} -5 & -5 & -5 & -5 \\ 5 & -1 & -1 & -1 \\ 5 & 1 & 2 & -3 \\ 5 & 1 & 3 & -4 \\ 5 & 1 & 4 & -2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -5 & -5 & -5 & -5 \\ 5 & -1 & -2 & -2 \\ 5 & 1 & 3 & 4 \\ 5 & 2 & -4 & -4 \\ 5 & 2 & 4 & -3 \end{pmatrix}.$$

Here the index 5 is simply a relabeling of the distinguished point previously denoted by 1. Among these candidates, the second matrix yielded better empirical performance.

E.4. Codex-found Solution

In this phase, the search was organized around the hypothesis that the main structural difficulty lay in choosing the right canonical $n = 7$ index matrix, after which the remaining improvement could be obtained by coefficient optimization alone. The search therefore first stabilized on a highly symmetric five-row $n = 7$ ansatz with one distinguished anchor variable and a cyclic arrangement of the remaining variables, and then froze this matrix while focusing entirely on the pair-coefficient vector. Early coefficient searches were deliberately conservative: they explored sparse local perturbations with coefficients essentially restricted to small $\{-1, 0, 1\}$-type moves, which was already enough to reduce the taxi instance to 6 violations and then 5 violations. The next step was to enlarge the coefficient search neighborhood by allowing larger signed updates and a wider integer range, so that the search was no longer limited to single-step local corrections; this richer coefficient dynamics enabled the jump from the first near-solutions to the first 3-violation hit.

From an engineering point of view, the `Potential` and `SearchEvaluator` were designed to make very short search iterations informative. For a fixed $n = 7$ matrix, the `Potential` implementation precomputed the configuration-index tables induced by the rows, as well as all pair-distance values over the full assignment space, so that evaluating a candidate potential reduced to summing a few preindexed work-function entries, subtracting a precomputed linear penalty, and taking a minimum. The `SearchEvaluator` then exploited this by evaluating candidates on sampled edge sets rather than full scans during search. It memoized node potentials within each evaluation so that each node was scored at most once, mixed random edge samples with a cache of previously discovered hard edges, and stopped early after a prescribed number of violations. In this way, each failed candidate not only produced a score, but also exposed a small set of particularly informative violated edges; these were recycled into subsequent evaluations, so the search increasingly concentrated on the difficult parts of the taxi instance instead of repeatedly spending effort on easy edges.

The `PotentialFamily` acted as a lightweight staged local-search controller on top of this evaluator. It maintained a queue of candidate coefficient vectors on the fixed symmetric matrix, separated into a cheap "quick" stage and a more reliable "confirm" stage. Candidates were seeded from a small family of structured sparse coefficient patterns and then expanded by repeated local mutations, initially with single-step sign changes and later with larger jumps and larger allowable magnitudes. Only coefficients not involving the distinguished auxiliary variable were actively mutated, which reduced the search dimension and preserved the symmetry assumptions built into the matrix ansatz. Candidates that performed well under quick sampled evaluation were promoted to confirm runs with larger edge budgets, while hard violated edges collected from these runs were added back into the evaluator's cache. Thus the overall method was not a black-box optimizer, but a deliberately engineered loop: first identify a strong symmetric $n = 7$ matrix, then run an increasingly expressive coefficient

search around it, with cached hard-edge feedback and staged evaluation guiding the progression from 6 to 5 and finally to 3 violations.

E.5. Discovered Potentials

see subsection A.1 for discussion on the competitiveness and what does k and $k+1$-competitive means and why $k+1$-competitiveness is the relaxed setting.

Below we report representative canonical potentials discovered with human-in-the-loop assistance.

k-competitive candidate.

$$n := 5, \qquad I := \begin{pmatrix} -5 & -5 & -5 & -5 \\ 5 & -1 & -2 & -2 \\ 5 & 1 & 3 & 4 \\ 5 & 2 & -4 & -4 \\ 5 & 2 & 4 & -3 \end{pmatrix}, \qquad C := \begin{pmatrix} 0 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Note that in the previous section we referred to the index matrix as an $n = 7$ matrix, whereas here we present the case $n = 5$. This simplification does not alter the potential. The originally discovered coefficient vector contained several large-magnitude coefficients that effectively merged two pairs of points, causing them to take identical values in every minimization. We therefore present the simplified formulation here.

$(k+1)$-competitive candidate.

$$n := 4, \qquad I := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ -1 & 2 & 3 & 4 \\ -2 & -2 & 3 & 4 \\ -3 & -3 & -3 & 4 \\ -4 & -4 & -4 & -4 \end{pmatrix}, \qquad C := 0.$$

This construction closely resembles the unifying potential for $k = 4$. The main difference is that one work-function term (e.g., $(1, 2, 3, 4)$ or equivalently $(-1, 2, 3, 4)$ up to symmetry) appears with multiplicity two.

Shinka's modification of Unifying Potential. The index matrix is the same as in the Unifying Potential case, but it increased the n from 4 to 5 by effectively introducing additional point to the minimization and connecting it to the rest of the points only via distance coefficients:

$$C := \begin{pmatrix} 0 & 1 & -1 & -1 & -1 \\ 1 & 0 & -1 & -1 & -1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 & 0 \end{pmatrix}$$

So in the non-matrix form:

$$\begin{aligned} \Phi(w) = \min_{a_1,\ldots,a_5} \Big(& w(a_1, a_2, a_3, a_4) + w(\overline{a_1}, a_2, a_3, a_4) + w(\overline{a_2}, \overline{a_2}, a_3, a_4) \\ & + w(\overline{a_3}, \overline{a_3}, \overline{a_3}, a_4) + w(\overline{a_4}, \overline{a_4}, \overline{a_4}, \overline{a_4}) \Big) \\ - \Big(& d(a_1, a_2) - d(a_1, a_3) - d(a_1, a_4) - d(a_1, a_5) - d(a_2, a_3) - d(a_2, a_4) - d(a_2, a_5) \\ & + d(a_3, a_4) + d(a_3, a_5) + d(a_4, a_5) \Big). \end{aligned}$$

E.6. Coefficient Brute-Force Analysis

After discovering the SHINKA and CODEX potentials, we performed an additional brute-force sweep over their coefficient vectors while keeping the corresponding canonical matrices fixed. Specifically, we enumerated all coefficient assignments in the restricted range $\{-1, 0, 1\}$ for each coefficient. This search did not find any candidate improving on the discovered SHINKA or CODEX potentials within the tested templates. Thus, at least in this local coefficient neighborhood, the remaining violations do not appear to be removable by a simple coefficient-only adjustment.

E.7. Analysis of the Violations

We also compared the sets of violated inequalities for the unifying potential, the SHINKA modification, and the CODEX candidate. On the augmented $k = 4, m = 6$ instance, the 14 violations of the SHINKA potential form a strict subset of the 17 violations incurred by the unifying potential. Thus, in this case, the modification appears to repair some of the original obstructions without introducing new ones.

The 3-violation CODEX candidate behaves differently. Its violation set is not a subset of the unifying potential's violation set: two of its violations coincide with violations of the unifying potential, while one is new. This suggests that the candidate is not merely a local repair of the unifying potential, but shifts the obstruction pattern. The distinction becomes more pronounced on the augmented $m = 8$ instance, where the overlap between violation sets provides additional evidence that the discovered candidates occupy different regions of the potential space.

F. Compute Resources and Runtime Details

This section summarizes the compute resources needed to reproduce the main experimental results. The benchmark consists of two types of computation: (i) generating and storing finite k-server instances, and (ii) evaluating candidate potentials during agentic search.

All verifier-based experiments were run on CPU workers. The main machine used 96 logical CPUs.

Evaluation hardware. All reported verifier-based experiments were run on CPU workers. The evaluator is parallelized across candidate potentials and/or edge batches, and does not require GPUs. The main resource requirements are CPU time and memory for storing pre-generated work-function graph instances and associated edge data. The exact hardware used in our experiments consisted of 96 Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz, 512GB, running Ubuntu 22.1. Our estimates for the minimum resources are ≈ 5 cores and 8GB RAM. The released repository includes scripts for running the evaluator with configurable numbers of workers.

Benchmark instance size. The computational cost is dominated by the number of inequalities checked. The resolved $k = 3$ calibration instances are small enough for rapid iteration: for example, full evaluation of a canonical potential on the $k = 3, m = 6$ instance takes less than 10 seconds on our hardware. In contrast, the augmented $k = 4, m = 6$ instance contains roughly 7 million inequalities, and full evaluation of a canonical potential takes more than a minute with parallelization. The largest generated instance, $k = 4, m = 8$ with k-taxi augmentation, is substantially larger (around one week) and is used only for limited validation because exhaustive evaluation and storage become prohibitive.

$k = 3$ **calibration experiments.** For the experiments in Figure 1 and Table 2, each method was run for 10 independent repetitions. Each run used a budget of 100 verifier calls. The evaluated methods were Best-of-N, ShinkaEvolve, and LoongFlow, all powered by `gpt-oss-120b`. Since these experiments use the $k = 3$ calibration instances, they can be reproduced on a multicore CPU machine with 8GB. The total wall-clock time per run was approximately 4-12 hours, depending on the setting and number of parallel workers.

$k = 4$ **open-regime experiments.** The $k = 4$ experiments are substantially more expensive and are not intended as a controlled leaderboard of agentic methods. Full verification on the augmented $k = 4, m = 6$ instance takes over a minute per candidate on our hardware, making large sweeps with full evaluation at every step impractical. To make search feasible, we used proxy evaluation during search and reserved full verification for selected candidates. Proxy strategies included hard-edge caches, staged evaluation, and early stopping. The best-found candidates reported in Table 3 were then evaluated with the full verifier.

LLM inference cost. For self-hosted experiments with `gpt-oss-120b`, inference was run on 2×A100 80GB. For hosted frontier-model experiments, we used API-based coding-agent workflows. Because the $k = 4$ prompts include substantial mathematical context, proxy-evaluation instructions, and long code states, token usage is significantly higher than in the $k = 3$ calibration experiments. We therefore report the $k = 4$ experiments as construction-search results rather than as a systematic comparison of all methods under equal frontier-model budgets.

Reproducibility notes. All scripts, prompts, model configurations, benchmark instances, and evaluation commands are provided in the released repository. Users can reduce compute requirements by starting with the $k = 3, m = 6$ calibration instance, limiting the number of verifier workers, or using the normalized violation score on smaller discretizations before running full verification on larger instances.

G. Future Work

A central direction for future work is to move beyond the canonical potential classes explored here. Huang & Zhang (2022) suggest that purely canonical forms may not be expressive enough for the open $k = 4$ circle case, so a promising next step is to let agents search over richer non-canonical structures, such as nested min–max expressions. This raises a methodological challenge: to scale such exploration, we need automatic tests for whether a proposed expression is genuinely non-canonical or merely canonical in disguise. One possible diagnostic is convexity. Since canonical potentials in our setting impose convex structure, detecting non-convex behavior of a candidate potential can help identify when the search has escaped the canonical class. This is especially important for nested min–max formulas, which may collapse to an effectively canonical form on the tested instances if the optimizing branches always follow the same pattern.

H. Extended Related Works

Recent AI systems have shown growing promise in formal scientific domains, including both physics (Brenner et al., 2026; Guevara et al., 2026) and mathematics (Georgiev et al., 2025; Dobriban, 2025; Bubeck et al., 2025; Feng et al., 2026a;b; Sothanaphan, 2026; Chen et al., 2026c). Alongside these application-oriented results, there is a substantial line of work on AI systems and search algorithms for mathematical and scientific discovery (Romera-Paredes et al., 2024; Novikov et al., 2025; Sharma, 2025; Lange et al., 2025; Surina et al., 2025; Jiang et al., 2026; Wan et al., 2025), as well as on applying such systems to concrete open mathematical tasks (Novikov et al., 2025; Georgiev et al., 2025; Nagda et al., 2025; 2026). A large fraction of these works focus on open-ended construction tasks, such as improving upper or lower bounds, where progress is measured by small numerical improvements. While this paradigm is valuable, it can be difficult to interpret scientifically and often yields benchmarks that do not clearly distinguish methods. In contrast, our setting has a sharp and meaningful target—zero violations on the generated instances—together with a direct connection to a longstanding open problem in competitive analysis. In particular, success on our benchmark would constitute progress toward the k-server conjecture, and the closely related $k = 3$ circle case has already led to a substantial theoretical result published in a reputable venue (Huang & Zhang, 2022).

A second line of work develops benchmark suites built directly around research-level mathematical problems (Glazer et al., 2025; Epoch AI, 2026; Abouzaid et al., 2026; Feng et al., 2026a;b). FrontierMath (Glazer et al., 2025) focuses primarily on difficult mathematical problems with known answers, while its more recent Open Problems track (Epoch AI, 2026) extends this paradigm to open questions with verifiable final answers. These benchmarks are philosophically close to our design in that they aim for objective verification, but they also have important limitations for agent research: parts of the evaluation stack remain closed or paywalled, which restricts independent experimentation, and some verifiers can certify correctness only with high confidence rather than absolute certainty. Our benchmark shares the general challenge that verification may be incomplete in the infinite setting, but differs in being fully executable and openly inspectable on finite instances. Related proof-oriented benchmarks, such as First Proof (Abouzaid et al., 2026) and Erdős Bench (Feng et al., 2026b;a), evaluate the ability of models to produce complete mathematical proofs. Compared with these settings, our task is more interactive: agents receive dense, instance-specific feedback, can iteratively improve candidate constructions, and do not require human judgment in the evaluation loop. This makes large-scale automated experimentation substantially easier and, we hope, more conducive to rapid progress on the underlying mathematical problem.

Finally, within competitive analysis itself, there has been recent work using AI to study online algorithms, for example in online matching where reinforcement learning is used to tighten competitive bounds (Zhang et al., 2024). Our setting is qualitatively different. In online matching, the emphasis is often on finding improved algorithms or counterexamples,

whereas for the k-server conjecture there is already a canonical candidate algorithm—the Work Function Algorithm—and the central challenge is to prove its competitiveness. Accordingly, our benchmark is aimed not at algorithm discovery, but at discovering proof-relevant potential functions that could ultimately contribute to resolving a central conjecture in online algorithms.

Our work is also related to the rapidly growing literature on AI-assisted proof-oriented mathematics and formal reasoning (Biswal et al., 2026; Angdinata et al., 2026a; Chen et al., 2026a;c;b; Angdinata et al., 2026b). Much of this line focuses on theorem proving, formalization, or discovering objects that can be verified in a proof assistant or by a mathematical argument. Our approach is complementary: we do not ask the agent to produce a complete formal proof of the infinite-circle case. Instead, we use automatic finite verification to search for compact functional potentials that could become the central object of a subsequent proof. This places our work between executable program search and proof-oriented mathematical discovery.

I. Released Code and Artifacts

Code can be found at: https://anonymous.4open.science/r/k-server-bench-8384/ and generated circle instances can be found at https://figshare.com/s/8ef50b29231d34e7cf9f?file=64334772.