Generating Robust Portfolios of Optimization Models using Large Language Models

Eleni Straitouri [1] **Cheol Woo Kim** [2] **Milind Tambe** [2]

Abstract

Mathematical optimization is a powerful tool for structured decision-making across domains such as resource allocation and planning. Formulating optimization models faithful to reality, though, remains a significant bottleneck as it typically demands both domain expertise and optimization knowledge that are often scarce. Recent advances in large language models (LLMs) promise to bridge this gap, enabling the generation of candidate optimization models from natural language descriptions. However, there is no guarantee that any single LLM-generated model is reliable, and existing approaches that output only one model are therefore risky. In this work, we propose a novel algorithm that generates a portfolio of optimization models, designed to be robust to the limitations of LLMs. Our method exploits the observation that a single LLM can play two distinct roles—as a stochastic generator and as a reasoning evaluator—and proposes a unified framework that leverages both capabilities in a complementary manner. We provide theoretical guarantees showing that, as long as either the generator or the evaluator is well-aligned with human preferences, the portfolio is guaranteed to contain high-quality candidates, enabling a principled human-in-the-loop process in which a decision-maker can review multiple candidates before committing to one. We further validate our approach empirically, demonstrating strong performance across a range of optimization modeling tasks.

1. Introduction

Formalizing a structured decision-making task as a mathematical optimization problem plays a pivotal role in designing optimal decision policies in domains such as resource allocation and planning (Brill Jr, 1979; Katoh & Ibaraki, 1998; Vercellis, 2011). However, defining an optimization model that accurately reflects all the real-world requirements and constraints of the decision-making task can be rather challenging; typically, it requires not only exhaustive manual tuning, but also a combination of domain expertise and deep optimization knowledge.

To address this challenge, there has been a growing interest in leveraging Large Language Models (LLMs) to automate the definition of optimization models given a task description in natural language. Recent lines of work typically focus on either automating the entire model definition (Yang et al., 2023; 2024; Ahmaditeshnizi et al., 2024; Astorga et al., 2024; Zhang et al., 2024; Jiang et al., 2024; Ahmed & Choudhury, 2024; Huang et al., 2025; Xiao et al., 2025) or a partial model definition limited to the design of the objective (reward) function in (Icarte et al., 2022; Yu et al., 2023; Shinn et al., 2023; Ma et al., 2024; Hwang et al., 2024; Xie et al., 2024; Behari et al., 2024; Verma et al., 2025; Sun et al., 2026).

Prior work, though, typically proposes computationally-heavy approaches, requiring additional training or fine-tuning of language models, with the objective to generate a single optimization model, while lacking guarantees on its quality. In this work, we introduce a lightweight, training-free algorithm to generate a portfolio of optimization models with guarantees over the quality of generated models while being robust to limitations of language models. To achieve this, our algorithm leverages a dual perspective on the capabilities of language models that we describe next.

Language models can operate as stochastic *generators* (Verma et al., 2025; Cardenoso & Caarls, 2025) that can provide diverse models through repeated stochastic sampling, accounting for different trade-offs present in the optimization task at hand. In addition, language models, can also operate as judges or reasoning evaluators (Verma et al., 2025; Saccon et al., 2025) arguing about the quality of given inputs, based on their world knowledge and reasoning capabilities. By unifying these modes of operation, our approach leverages their complementary strengths to provide robustness guarantees over the quality of models comprising our portfolios, allowing a decision-maker to review multiple

[1]Max Planck Institute for Software Systems, Kaiserslautern, Germany [2]Harvard University, Cambridge, United States. Correspondence to: Eleni Straitouri <estraitouri@mpi-sws.org>.

Preprint. May 26, 2026.

high quality candidates before committing to one.

Contributions. We propose a method that first uses a language model, namely a *generator*, to generate a distribution of candidate optimization models through repeated stochastic sampling. Next, our method uses a judge agent, namely an *evaluator* to rank these optimization models based on how well they align with the optimization task description in natural language. We construct a portfolio comprising the candidate optimization models with the highest evaluator ranks that have a total generation probability above a user-specified threshold. We show that in this way our portfolio guarantees to include high quality candidates if either of the generator or the evaluator satisfy a certain level of alignment with respect to human preferences. We empirically verify the strong performance of our portfolio in experiments with both synthetic and real data.

2. Building a Portfolio through a Generator and Evaluator

Let d be the natural language description of an optimization problem, g be a stochastic generator model and e an evaluator model.

Generator. Given a natural language description $d \in \mathcal{D}$, where $\mathcal{D}$ is the space of any natural language prompt, we consider the generator g as a probability distribution p over the space of candidate optimization models $\mathcal{O}$. [1]

Evaluator. Given the optimization problem description d, we consider the evaluator as a ranking policy π_e over the space of candidate optimization models $o \in \mathcal{O}$, inducing a ranking

$$\pi_e(d) = (o_{(1)^e}, o_{(2)^e}, \dots), \tag{1}$$

where the subscript $(\cdot)$ denotes that rank of candidate o. We assume that the lower the rank, the better the candidate given the description d, according to world knowledge of the evaluator, and ties are broken randomly.

We use the probability distribution p induced by the generator values $g(d)$ and the ranking $\pi_e(d)$ induced by the evaluator e to construct our portfolio $\mathcal{P}$ of candidate optimization models as follows

$$\mathcal{P}(d;\alpha) = \left\{o_{(i)^e}\right\}_{i=1}^{k^*(\alpha)}, \tag{2}$$

where $\alpha \in (0,1)$ is a user-defined parameter and

$$k^*(\alpha) = \inf\left\{k \in \mathbb{N} : \sum_{i=1}^{k} p(o_{(i)^e}) \geq 1-\alpha\right\}.$$

[1] We assume that $\mathcal{O} = \mathcal{O}_v \cup \perp$, where $\mathcal{O}_v$ is the space of text representations of any valid optimization model, and $\perp = \mathcal{D} \setminus \mathcal{O}_v$. Note that $\mathcal{O}$ can be finite or infinite.

By constructing our portfolio in this way, we make sure that, for a small enough α our portfolio will include candidate models either because they have a low (good) evaluator rank or because they have a high probability of being generated This means that our portfolio will include high-quality models as long as either the evaluator ranks them low, or the generator produces them with high probability, *i.e.*, whenever, either of them is well-aligned with human preferences. Motivated by this observation, in what follows, we prove that our portfolio guarantees to include high-quality candidates if either of the generator or evaluator are human-aligned.

3. Robust Portfolio Generation

We begin by defining human-alignment of the generator and evaluator under the following assumption

Assumption 3.1. Given the natural description d there is a human ranking policy π^*, inducing a ranking $\pi^*(d)$ over $o \in \mathcal{O}$, with

$$\pi^*(d) = (o_{(1)^*}, o_{(2)^*}, \dots), \tag{3}$$

where the lower the rank, the higher the quality of the candidate optimization model o according to human preferences.

Based on the above, we provide the following definitions.

Definition 3.2 (Evaluator Alignment)**.** An evaluator is human-aligned given d, if the induced ranking

$$\pi_e(d) = \pi^*(d).$$

Definition 3.3 (Generator Alignment)**.** A generator model inducing a probability p over candidate optimization models $o \in \mathcal{O}$ is human-aligned given d, if for any $o_{(i)^*}, o_{(j)^*} \in \pi^*(d)$ with $i \leq j$, it holds

$$p(o_{(i)^*}) \geq p(o_{(j)^*}). \tag{4}$$

Intuitively, a human-aligned generator generates candidates judged as high quality according to human preferences with high probability.

We show that under either evaluator or generator alignment our portfolio is guaranteed to include high quality candidates or more formally achieve positive *coverage*, where we define coverage as follows.

Definition 3.4 (Portfolio coverage)**.** A portfolio $\mathcal{P}$ of k candidates has coverage

$$c(\mathcal{P}) = \frac{\sum_{i=1}^{k} \mathbb{I}\{o_{(i)^*} \in \mathcal{P}\}}{k} \tag{5}$$

Under evaluator alignment and any generator, we prove: [2]

[2] Proofs are deferred to Appendix A.

Corollary 3.5 (Evaluator robustness)**.** *Assume an aligned evaluator under Definition 3.2 for a description d. For any value of $\alpha \in (0,1)$ and any generator, the portfolio $\mathcal{P}(d;\alpha)$ constructed using Eq. 2 has coverage $c(\mathcal{P}(d;\alpha)) = 1$.*

Under generator alignment and any evaluator, we prove

Proposition 3.6. *Assume an aligned generator under Definition 3.3 for a description d. For any value of $\alpha \in (0, 1/2)$ and any evaluator, any non-empty portfolio $\mathcal{P}(d;\alpha)$ constructed using Eq. 2 has coverage*

$$c(\mathcal{P}(d;\alpha)) > \frac{1-2\alpha}{k^*(\alpha)} > 0.$$

4. Experiments

We begin by simulating the generator and evaluator under different levels of human-alignment in a synthetic data setup, where we i) empirically verify the robustness of our portfolio, and ii) investigate the effect of human-alignment on the portfolio coverage and size. We proceed with a real data implementation of our portfolios on optimization modeling, where we show that the optimization models comprising our portfolios are qualitatively superior compared to optimization models provided by randomly sampled portfolios.

4.1. Simulated Portfolios

In our synthetic data setup, we consider a finite space of candidate optimization models $\mathcal{O}$, such that $|\mathcal{O}| = K$, for $K \in \{10, 20, 50, 100\}$, with human ranking $(1, 2, ..., K)$. We simulate several generator and evaluator types, each with a different level of human alignment as described below. For each generator-evaluator pair, we use Eq. 2 to construct a portfolio for each α value from 0 to 1 with step 0.02 and repeat the experiment with 40 different seeds.

Generators. We implement the following generator types (refer to Appendix B.1 for more details).

- ALIGNED. A generator satisfying Definition 3.3.
- WEAKLY ALIGNED. A generator violating Definition 3.3, for which though Proposition 3.6 holds for any portfolio of $K/2$ generated candidates.
- UNIFORM. A generator where each generated candidate $i \in [K]$ has probability $p(i) = 1/K$.
- MISALIGNED. A reversely aligned generator.

Evaluators. We characterize each evaluator by the fraction of incorrectly ranked candidates, which we denote as the evaluator error $\epsilon = \frac{\sum_{i=1}^{K} \mathbb{I}\{o_{(i)^*} \neq o_{(i)^e}\}}{K}$. We implement the evaluator for $\epsilon \in \{0, 0.3, 0.5, 0.7, 1\}$, where $\epsilon = 0$ characterizes a human-aligned evaluator with $\pi^* = \pi_e$, and $\epsilon = 1$, an evaluator e such that $\forall i \in [K], o_{(i)^e} = o_{(K+1-i)^*}$.

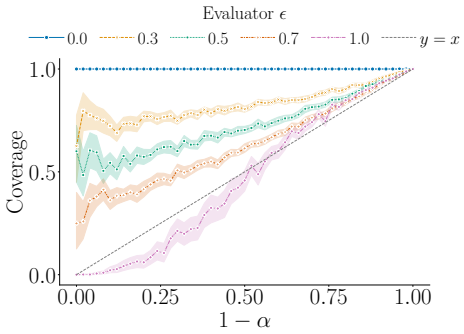


Figure 1. Portfolio mean coverage against the value of $1-\alpha$ for the WEAKLY ALIGNED generator paired with each evaluator for $K = 100$. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.

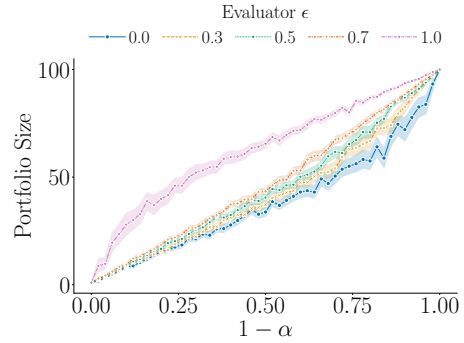


Figure 2. Portfolio mean size against the value of $1-\alpha$ for the WEAKLY ALIGNED generator paired with each evaluator for $K = 100$. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.

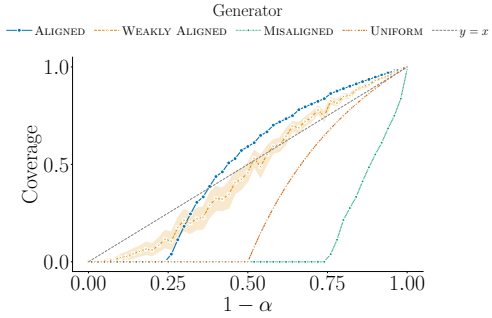


Figure 3. Portfolio mean coverage against the value of $1-\alpha$ for the evaluator with $\epsilon = 1.0$ paired with each generator for $K = 100$. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.

Results. Figure 1 and 3 show—in consistence with Proposition 3.6— that for $\alpha < 0.5$ our portfolio achieves positive coverage even under the WEAKLY ALIGNED generator inde-

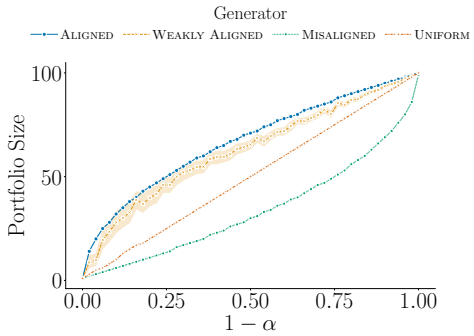


Figure 4. Portfolio mean size against the value of $1-\alpha$ for the evaluator with $\epsilon = 1.0$ paired with each generator for $K = 100$. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.

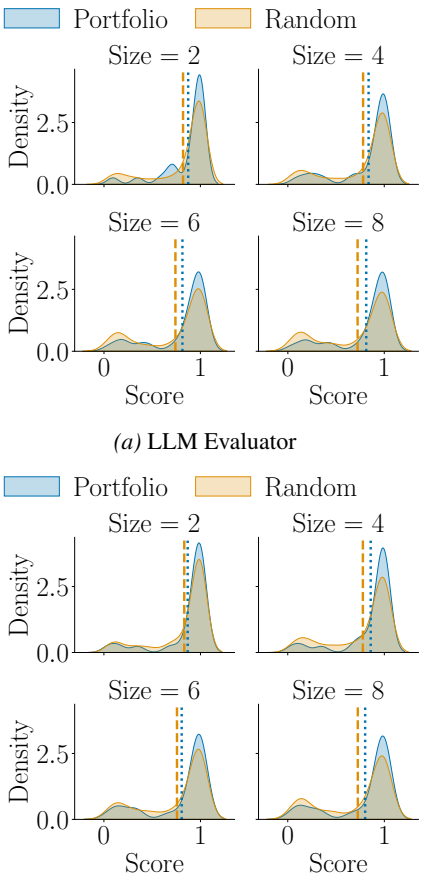
pendently of the evaluator error ϵ. In fact, Figure 1 shows, in practice, that coverage is lower bounded by $1-\alpha$—a tighter lower bound compared to Proposition 3.6—as coverage is always above the diagonal for $\alpha < 0.5$. Further, Figure 1– 4 reveal notable insights on how the level of human alignment controls the trade-off between portfolio coverage and its size; Figure 1 and 2 show that under a Weakly Aligned generator, the lower the evaluator error, the higher the coverage and the lower the portfolio size for the same value of α; Figure 3 and 4 show that under the worst evaluator, the more aligned the generator, the higher the coverage, at the price though of larger portfolios.

4.2. Portfolios of Optimization Models

We construct portfolios of optimization models for 25 optimization problems given in natural language from the dataset NL4LP (Ahmaditeshnizi et al., 2024). We use a strong LLM-as-a-judge to evaluate the quality of the candidate optimization models in our portfolios and compare it against portfolios constructed by random sampling (i.e., prompting multiple times). Refer to Appendix B.2 for all LLM prompts and further details.

Generator. For each optimization problem, we generate 50 candidate optimization models with `gpt-5.4-nano`, where each model comprises a natural language description followed by the corresponding python code. For each candidate model o we compute the generator probability $p(o)$ using the normalized log probabilities of the tokens of the text representation of the candidate model.

Evaluator. For each generated model, we execute the code and use the output—together with the model and problem—to assign a scalar score from 1 to 100 to each generated model using `gpt-5.4-nano` with repeated sampling. For each problem, we rank the generated models with the average score over 4 samples and use this as the evaluator ranking. Further, we implement an additional evaluator type, where we rank the generated models using the probability distribution induced by the generator—essentially using the generator as evaluator too.



(a) LLM Evaluator

(b) Evaluator using the Generator Probabilities

Figure 5. Kde plot of scores assigned by `gpt-5.4`-as-a-judge to portfolios of size $s \in \{2, 4, 6, 8\}$ against s randomly selected candidates for two evaluator types. The score distributions are over 25 problems and 30 samplings of the random candidates. Dashed and dotted lines represent the mean score value over the randomly chosen and the portfolio candidates respectively.

Results using LLM-as-a-judge. We use `gpt-5.4` as a judge under the same scoring regime as for the evaluator for each generated model, except that in the scoring input prompt, we also add the ground truth solution of the optimization model as provided in the NL4LP dataset. As the final score of each generated model we consider the average normalized LLM-as-a-judge score over the repeated samplings. We characterize the quality of each portfolio with the minimum among the scores of the models it contains, assuming a worst-case scenario, where a human would select the worst model in the portfolio. Figure 5 shows that our portfolio for both the evaluator types consistently outperforms random portfolios under this worst-case scenario.

Further, it also shows that our portfolios using the reasoning capabilities of the LLM evaluator achieve higher scores compared to portfolios using the generator probabilities. These results demonstrate in practice the competitive advantage of our approach on unifying the generative and reasoning capabilities of LLMs.

Acknowledgments

Straitouri acknowledges support from a Google PhD Fellowship. Kim and Tambe acknowledge support by ONR MURI N00014-24-1-2742.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahmaditeshnizi, A., Gao, W., and Udell, M. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. 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. 577–596. PMLR, 21–27 Jul 2024.

Ahmed, T. and Choudhury, S. Lm4opt: Unveiling the potential of large language models in formulating mathematical optimization problems. *INFOR: Information Systems and Operational Research*, 62(4):559–572, 2024.

Astorga, N., Liu, T., Xiao, Y., and Van Der Schaar, M. Autoformulation of mathematical optimization models using llms. *arXiv preprint arXiv:2411.01679*, 2024.

Behari, N., Zhang, E., Zhao, Y., Taneja, A., Nagaraj, D., and Tambe, M. A decision-language model (dlm) for dynamic restless multi-armed bandit tasks in public health. *Advances in Neural Information Processing Systems*, 37: 3964–4002, 2024.

Brill Jr, E. D. The use of optimization models in public-sector planning. *Management Science*, 25(5):413–422, 1979.

Cardenoso, F. and Caarls, W. Leveraging llms for reward function design in reinforcement learning control tasks. *arXiv preprint arXiv:2511.19355*, 2025.

Huang, C., Tang, Z., Hu, S., Jiang, R., Zheng, X., Ge, D., Wang, B., and Wang, Z. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025.

Hwang, M., Weihs, L., Park, C., Lee, K., Kembhavi, A., and Ehsani, K. Promptable behaviors: Personalizing multi-objective rewards from human preferences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16216–16226, June 2024.

Icarte, R. T., Klassen, T. Q., Valenzano, R., and McIlraith, S. A. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Jiang, C., Shu, X., Qian, H., Lu, X., Zhou, J., Zhou, A., and Yu, Y. Llmopt: Learning to define and solve general optimization problems from scratch. *arXiv preprint arXiv:2410.13213*, 2024.

Katoh, N. and Ibaraki, T. Resource allocation problems. *Handbook of Combinatorial Optimization: Volume1–3*, pp. 905–1006, 1998.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Saccon, E., De Martini, D., Saveriano, M., Lamon, E., Palopoli, L., and Roveri, M. Automated generation of mdps using logic programming and llms for robotic applications. *IEEE Robotics and Automation Letters*, 11(2): 1770–1777, 2025.

Shinn, N., Cassano, F., Labash, B., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning, 2023. *URL https://arxiv. org/abs/2303.11366*, 1, 2023.

Sun, S., Liu, R., Lyu, J., Yang, J.-W., Zhang, L., and Li, X. A large language model-driven reward design framework via dynamic feedback for reinforcement learning. *Know.-Based Syst.*, 326(C), January 2026. ISSN 0950-7051. doi: 10.1016/j.knosys.2025.114065. URL https://doi.org/10.1016/j.knosys.2025.114065.

Vercellis, C. *Business intelligence: data mining and optimization for decision making*. John Wiley & Sons, 2011.

Verma, S., Boehmer, N., Kong, L., and Tambe, M. Balancing act: prioritization strategies for llm-designed restless bandit rewards. In *International Conference on Game Theory and AI for Security*, pp. 376–394. Springer, 2025.

Xiao, Z., Xie, J., Xu, L., Guan, S., Zhu, J., Han, X., Fu, X., Yu, W., Wu, H., Shi, W., et al. A survey of optimization modeling meets llms: Progress and future directions. *arXiv preprint arXiv:2508.10047*, 2025.

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Yang, Z., Wang, Y., Huang, Y., Guo, Z., Shi, W., Han, X., Feng, L., Song, L., Liang, X., and Tang, J. Optibench meets resocratic: Measure and improve llms for optimization modeling. *arXiv preprint arXiv:2407.09887*, 2024.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., Ichter, B., Xiao, T., Xu, P., Zeng, A., Zhang, T., Heess, N., Sadigh, D., Tan, J., Tassa, Y., and Xia, F. Language to rewards for robotic skill synthesis. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 374–404. PMLR, 06–09 Nov 2023.

Zhang, J., Wang, W., Guo, S., Wang, L., Lin, F., Yang, C., and Yin, W. Solving general natural-language-description optimization problems with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pp. 483–490, 2024.

A. Proofs

A.1. Proof of Corollary 3.5

Proof. Let π_e be the ranking policy of the aligned evaluator. By Definition 3.2, we have that $\pi_e(d) = \pi^*(d)$, thus $o_{(i)^e} = o_{(i)^*}$ for any $i \in \mathbb{N}$. For any generator model and $\alpha \in (0, 1)$ we use Eq. 2, to construct a portfolio $\mathcal{P}(d; \alpha) = \{o_{(j)^e}\}_{i=1}^{k*(\alpha)}$, where $k^*(\alpha)$ is given by Eq. 3. The portfolio $\mathcal{P}(d; \alpha)$ has coverage

$$c(\mathcal{P}(d;\alpha)) = c\left(\{o_{(i)^e}\}_{i=1}^{k^*(\alpha)}\right) = c\left(\{o_{(i)^*}\}_{i=1}^{k^*(\alpha)}\right) = \frac{\sum_{i=1}^{k^*(\alpha)} \mathbb{I}\left\{o_{(i)^*} \in \{o_{(i)^*}\}_{i=1}^{k^*(\alpha)}\right\}}{k^*(\alpha)} = 1.$$

□

A.2. Proof of Proposition 3.6

Proof. With a slight abuse of notation, we will write $\mathcal{P}$ to denote $\mathcal{P}(d; \alpha)$ for ease of exposition. Let $\mathcal{X} \subseteq \mathcal{P}$ such that $\mathcal{X} = \mathcal{P} \cap \mathcal{P}^*$, where $\mathcal{P}^*$ is a portfolio of size $|\mathcal{P}^*| = |\mathcal{P}|$ with coverage $c(\mathcal{P}^*) = 1$. For $\mathcal{X} \subseteq \mathcal{P}$ we have

$$\sum_{o \in \mathcal{X}} p(o) + \sum_{o' \in \mathcal{P} \setminus \mathcal{X}} p(o') \geq 1 - \alpha. \tag{6}$$

Since the generator model is human aligned, it must hold that $\forall o \in \mathcal{P} \setminus \mathcal{X}$ and $\forall o' \in \mathcal{P}^* \setminus \mathcal{X}$, $rank(o) \geq rank(o')$ given the human ranking $\pi^*(d)$ and as a result, $p(o) \leq p(o')$. Using this and the fact that $|\mathcal{P} \setminus \mathcal{X}| = |\mathcal{P}^* \setminus \mathcal{X}| = k^*(\alpha) - |\mathcal{X}|$, Eq. 6 becomes

$$\sum_{o \in \mathcal{X}} p(o) + \sum_{o'' \in \mathcal{P}^* \setminus \mathcal{X}} p(o'') \geq \sum_{o \in \mathcal{X}} p(o) + \sum_{o' \in \mathcal{P} \setminus \mathcal{X}} p(o') \geq 1 - \alpha. \tag{7}$$

We will now use that $\sum_{o'' \in \mathcal{P}^* \setminus \mathcal{X}} p(o'') \leq \alpha$. This is because $\mathcal{P}^* \setminus \mathcal{X} \subseteq \mathcal{O} \setminus \mathcal{P}$ and by Eq. 6

$$1 - \sum_{o \in \mathcal{O} \setminus \mathcal{P}} p(o) = \sum_{o \in \mathcal{X}} p(o) + \sum_{o' \in \mathcal{P} \setminus \mathcal{X}} p(o') \geq 1 - \alpha.$$

As a result,

$$\sum_{o \in \mathcal{O} \setminus \mathcal{P}} p(o) \leq \alpha$$

and since $\mathcal{P}^* \setminus \mathcal{X} \subseteq \mathcal{O} \setminus \mathcal{P}$ it also holds

$$\sum_{o'' \in \mathcal{P}^* \setminus \mathcal{X}} p(o'') \leq \sum_{o \in \mathcal{O} \setminus \mathcal{P}} p(o) \leq \alpha.$$

Using the above and Eq. 7, we have

$$\sum_{o \in \mathcal{X}} p(o) + \alpha \geq \sum_{o \in \mathcal{X}} p(o) + \sum_{o'' \in \mathcal{P}^* \setminus \mathcal{X}} p(o'') \geq 1 - \alpha$$

and therefore

$$\sum_{o \in \mathcal{X}} p(o) \geq 1 - 2\alpha.$$

We will now use that $\sum_{o \in \mathcal{X}} p(o) < |\mathcal{X}| = k^*(\alpha) \cdot \frac{|\mathcal{X}|}{k^*(\alpha)}$ to rewrite the above as

$$k^*(\alpha) \cdot \frac{|\mathcal{X}|}{k^*(\alpha)} > \sum_{o \in \mathcal{X}} p(o) \geq 1 - 2\alpha.$$

Remember that $\mathcal{X} = \mathcal{P} \cap \mathcal{P}^*$, so $c(\mathcal{P}) = \frac{|\mathcal{X}|}{k^*(\alpha)}$. Therefore the above becomes

$$k^*(\alpha) \cdot c(\mathcal{P}) > 1 - 2\alpha$$

and, thus

$$c(\mathcal{P}) > \frac{1 - 2\alpha}{k^*(\alpha)}.$$

□

B. Additional Experimental Details

B.1. Simulated Portfolios

Generator Types.

- ALIGNED. Each generated candidate $i \in [K]$ has probability $p(i) = \frac{K+1-i}{\sum_{i'=1}^{K} K+1-i'}$; therefore $\forall i < j \in [K], p(i) > p(j)$ satisfying Definition 3.3.
- WEAKLY ALIGNED. Each generated candidate $i \in [K]$ has a randomly selected probability value such that $\sum_{i \in [K/2]} p(i) > \sum_{j \in [K/2]} p(K/2 + j)$, where we select $\sum_{i \in [K/2]} p(i) \in [0.5, 0.99)$ uniformly at random. Note that Proposition 3.6 holds for any portfolio of $K/2$ generation candidates constructed using this generator type.
- UNIFORM. Each generated candidate $i \in [K]$ has probability $p(i) = 1/K$.
- MISALIGNED. Each generated candidate $i \in [K]$ has probability $p(i) = \frac{i}{\sum_{i'=1}^{K} i'}$.

B.2. Portfolios of Optimization Models

We prompt all LLMs with temperature 1.0, while setting verbosity to '`low`' and the reasoning effort to '`none`'. Below we provide the prompt templates and system prompts we used for the generator, evaluator and LLM-as-a-judge models. For the templates we follow the templates used by Huang et al. (2025).

Prompt template:

Below is an operations research question. Build a mathematical model and corresponding python code using `gurobi` that appropriately addresses the question.

Question:
{Question}

Response:

System prompt:

You are an expert in operations research and optimization. You are given a question that can be answered by building a mathematical model and writing code to solve it. Be concise and to the point.

Figure 6. Generator prompt and system prompt.

Evaluator prompt template:

Below is an operations research question followed by a proposed mathematical model, the corresponding python code as well as the code output. Score the mathematical model, code and code output based on how well they address the question with a score from 1 to 100.

Question:
{Question}

Proposed Mathematical Model and Python Code:
{Response}

Code Output:
{Solution}

Score (1-100):

Evaluator system prompt:

You are an expert in operations research and optimization. You are given a question with a proposed mathematical model, python code and the code output. Reply with only a single integer score from 1 to 100. A score of 100 means the model, code and code output are perfect and optimally address the question, while a score of 1 means they are very poor and do not address the question at all. Do not provide any explanations or comments, only the score.

Figure 7. Evaluator prompt and system prompt.

LLM-as-a-judge prompt template:

You are an expert in operations research and optimization. You are given a research question followed by a proposed mathematical model, the corresponding python code as well as the code output. The proposed mathematical model, code, and code outputs are followed by the ground truth solution, given in JSON format. Score the mathematical model, code and code output based on how well they address the question given the ground truth with a score from 1 to 100.

Question:
{Question}

Proposed Mathematical Model and Python Code:
{Response}

Code Output:
{Solution}

Ground Truth Solution:
{GroundTruth}

Score (1-100):

LLM-as-a-judge system prompt:

You are an expert in operations research and optimization. You are given a question with a proposed mathematical model, python code and the code output, as well as the ground truth solution in JSON format. Reply with only a single integer score from 1 to 100. A score of 100 means the model, code and code output perfectly and optimally address the question given the ground truth, while a score of 1 means they are very poor and do not address the question at all. Do not provide any explanations or comments, only the score.

Figure 8. LLM-as-a-judge prompt and system prompt.

C. Additional Experimental Results on Simulated Portfolios

Below we present results over several generator-evaluator pairs and values of K.

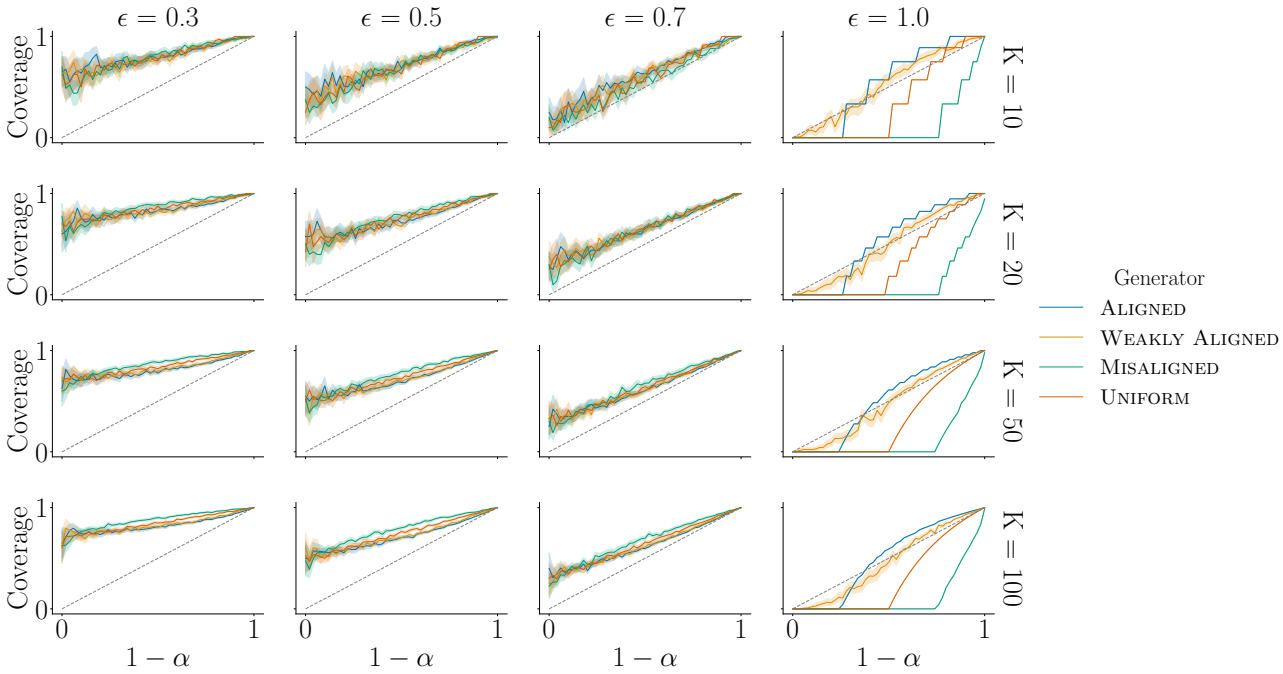


Figure 9. Portfolio coverage against $1 - \alpha$ for several generator evaluator pairs and value of K. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.

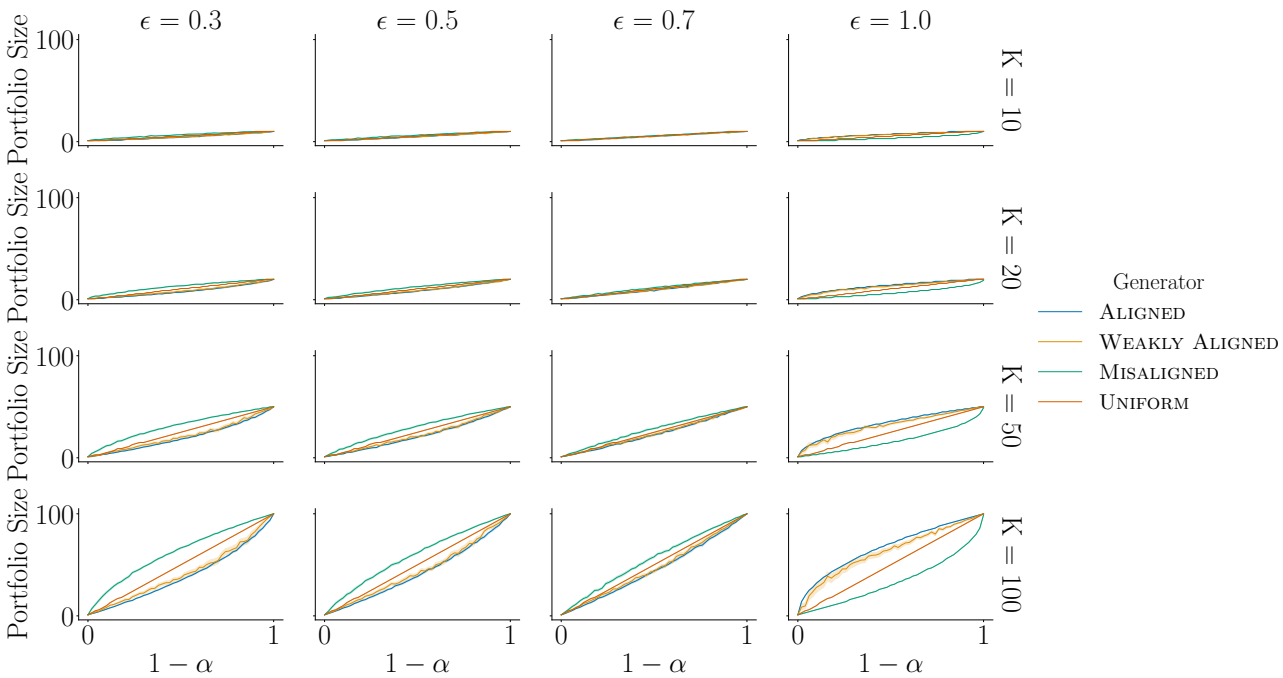


Figure 10. Portfolio size against $1 - \alpha$ for several generator evaluator pairs and value of K. The mean is over 40 iterations and shaded areas represent 95% confidence intervals.