KeeA*: Epistemic Exploratory A* Search via Knowledge Calibration

Dengwei Zhao[1], Shikui Tu[1]*, Yanan Sun[2] Lei Xu[1,3]*
[1]School of Computer Science, Shanghai Jiao Tong University
[2]School of Integrated Circuits, Shanghai Jiao Tong University
[3]Guangdong Laboratory of Artificial Intelligence and Digital Economy(SZ)
{zdwccc, tushikui, yannasun, leixu}@sjtu.edu.cn

Abstract

In recent years, neural network-guided heuristic search algorithms, such as Monte-Carlo tree search and A* search, have achieved significant advancements across diverse practical applications. Due to the challenges stemming from high state-space complexity, sparse training datasets, and incomplete environmental modeling, heuristic estimations manifest uncontrolled inherent biases towards the actual expected evaluations, thereby compromising the decision-making quality of search algorithms. Sampling exploration enhanced A* (SeeA*) was proposed to improve the efficiency of A* search by constructing an dynamic candidate subset through random sampling, from which the expanded node was selected. However, uniform sampling strategy utilized by SeeA* facilitates exploration exclusively through the injection of randomness, which completely neglects the heuristic knowledge relevant to open nodes. Moreover, the theoretical support of cluster sampling remains ambiguous. Despite the existence of potential biases, heuristic estimations still encapsulate certain valuable information. In this paper, epistemic exploratory A* search (KeeA*) is proposed to integrate heuristic knowledge for calibrating the sampling process. We first theoretically demonstrate that SeeA* with cluster sampling outperforms uniform sampling due to the distribution-aware selection with higher variance. Building on this insight, cluster scouting and path-aware sampling are introduced in KeeA* to further exploit heuristic knowledge to increase the sampling mean and variance, respectively, thereby generating higher-quality extreme candidates and enhancing overall decision-making performance. Finally, empirical results on retrosynthetic planning and logic synthesis demonstrate superior performance of KeeA* compared to state-of-the-art heuristic search algorithms.

1 Introduction

With the rise of deep learning, neural networks are utilized as guiding heuristic functions within Monte Carlo Tree Search (MCTS), which has contributed crucially to the successes of AlphaGo [41] and also sparked the renaissance of classical search algorithms such as A* search [18]. Empowered by the remarkable representational capabilities of neural networks, heuristic search algorithms have demonstrated superhuman performance across a broad spectrum of real-world applications, including games[39], *de novo* drug design [37], functional protein generation and optimization[48], retrosynthetic planning of organic molecules [4, 40, 59], logic synthesis in very large scale integration (VLSI) design [7], and combinatorial optimization tasks such as traveling salesman problem [6, 53] and bin packing problem [21]. Moreover, heuristic search plays a critical role in state-of-the-art

*Correspondence authors are Shikui Tu and Lei Xu.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

large language models by facilitating the efficient traversal of expansive solution spaces and enabling multi-step decision-making through structured reasoning [45, 58].

A* search [18] is a best-first search algorithm that selects the node with the maximum estimated evaluation reward f at each iteration for expanding. For a given node n, $f(n)$ is defined as the summation of $g(n)$, the cumulative reward from the initial node n_0 to n, and $h(n)$, the estimated reward from n to the desired goal, i.e.,

$$f(n) = g(n) + h(n). \tag{1}$$

$g(n)$ is derived from the historically traversed path and thus exactly corresponds to the true cumulative reward $g^*(n)$. In contrast, $h(n)$ is a heuristic estimation of the unobserved future reward, which typically diverges from the true remaining reward $h^*(n)$. A* is theoretically guaranteed to identify an optimal solution if $h(n)$ is admissible, i.e., $h(n) \geq h^*(n)$, even in the presence of estimation bias. Nonetheless, inaccuracies in the heuristics may still substantially degrade search efficiency through inducing suboptimal node expansions. In practical applications, constructing an accurate heuristic function $h(n)$ is challenging due to the high complexity of the state space and the sparsity of available training data. Moreover, since search algorithms are inherently model-based optimization methods, inaccuracies in state transitions and immediate reward feedback, stemming from incomplete environment modeling, further compromise search performance.

SeeA* [60] introduces a selective sampling process to construct a dynamic subset, incorporating exploratory behavior into A* search to escape from local optimal branches. Although theoretical analysis demonstrates that even uniform sampling can improve the efficiency of A* search, it entirely disregards heuristic information associated with open nodes, which, despite inherent potential biases, still encapsulate certain valuable guidance for the search process. Furthermore, while experimental results suggest that advanced strategies, such as cluster sampling, increase the likelihood of selecting higher-quality nodes and thereby contributing to enhanced decision performance, the theoretical benefits remain unclear.

In this paper, KeeA*, an epistemic exploratory A* search algorithm enhanced by knowledge calibration, is proposed to address the aforementioned limitations of SeeA*. The main contributions are summarized from three aspects[2]:

- We theoretically demonstrate that cluster sampling produces a candidate set with more favorable extrema than uniform sampling by embedding distributional knowledge into the cluster structure, and increasing both the expected mean and variance of sampled nodes contributes to improved decision-making performance.
- Cluster scouting and path-aware sampling are introduced by KeeA* to enhance the quality of the sampled candidate set by injecting epistemic knowledge into the sampling. We first show that for any two nodes, a greater disparity in their f-values corresponds to a higher likelihood that the node with the larger f also exhibits a greater true reward f^*, despite the presence of prediction errors. Accordingly, cluster scouting is proposed by first evaluating clusters based on f-statistics and subsequently allocating a larger sampling budget to clusters with higher estimated quality to improve the expected mean of candidate nodes. Furthermore, path-aware sampling mechanism is proposed to incorporate historical path dependencies into the intra-cluster sampling process, thereby increase the variance of selected nodes.
- Experiments are conducted on two real-world applications: retrosynthetic planning in organic chemistry and logic synthesis in VLSI design. KeeA* demonstrates superior performance over SeeA* and other state-of-the-art heuristic search algorithms, achieving higher success rates in problem-solving and superior solution quality.

2 Related work

A* search [18] is a foundational algorithm in heuristic search and has been extensively applied across diverse application domains, including route planning [46, 47], combinatorial games such as Rubik's Cube and sliding puzzles [1], motion planning in robotics [14], and so on. Monte Carlo Tree Search (MCTS) [3, 10] integrates stochastic node sampling with a tree-based search framework to efficiently explore large search spaces. Upper Confidence bounds applied to Trees with predictor

[2]The source code is publicly available at `https://github.com/CMACH508/KeeA`.

implemented via deep neural networks (PUCT) has been adopted in AlphaGo [41] and its successors [43, 42, 39], achieving superhuman performance and also sparking the renaissance of classical search algorithms. However, constructing accurate heuristic functions for complex real-world applications remains a significant challenge due to the combinatorial explosion of the search space and the limited availability of high-quality training data. For example, heuristic estimations in retrosynthetic planning exhibit substantial overfitting [59], thereby impairing search effectiveness by misdirecting expansion priorities. Furthermore, in applications such as functional protein design [22, 56] and *de novo* drug design [36], reward functions employed for learning are typically obtained through in silico simulations, which intrinsically diverge from real-world ground truth. The inherent complexities of environment modeling further hinder the effective development of robust heuristic functions.

Exploration has been incorporated into A* search to escape local optima induced by biased heuristic functions. ϵ-greedy node selection has been shown to improve the coverage of search algorithms, even when multiple enhancements have already been integrated into LAMA [44]. Type-WA* [9] augments Weighted A* with type-based exploration [51], randomly selecting one of T node types within the focal list [34] for expansion. LevinTS [33] incorporates node depth as a penalty term to encourage exploring alternative branches. Gumbel MuZero [12] employs Gumbel-Top-k sampling [25] by combining Gumbel noise $gumbel(a)$ with policy logits to generate a candidate action set for subsequent selection. MENST [50] combines MCTS and entropy regularization for the first time to explicitly promote exploration. By employing relative entropy and Tsallis entropy as regularization terms, RENTS and TENTS [11] significantly enhance the convergence efficiency of MCTS. SeeA* [59, 60] introduces exploration into A* search by stochastically sampling a subset of open nodes and selecting the expansion node from this candidate set.

Retrosynthetic planning aims to identify a feasible synthetic route by recursively decomposing a target molecule into simpler and commercially available precursors. To efficiently explore the vast combinatorial space of chemical reactions, search algorithms such as MCTS [19, 40, 57], A* search [4, 16, 23, 27, 52, 60] and depth-first proof-number search [24], in conjunction with a single-step predictor, are extensively utilized to address this challenging problem. Logic synthesis (LS) is a critical step in very large scale integration (VLSI) design, transforming high-level abstract representations of logic circuits into gate-level implementations by optimizing area, delay, and power consumption. Recent developments utilizing reinforcement learning [8, 20, 28, 35, 61, 26] and search algorithms [7, 32] have demonstrated substantial improvements in both the efficiency and scalability of solving LS tasks.

3 Preliminaries on A* search

In A* search, nodes in the search tree are divided into an OPEN set $\mathcal{O}$, which consists of unexpanded leaf nodes, and a CLOSED set $\mathcal{C}$, containing nodes that have already been explored. At each iteration, the node $n \in \mathcal{O}$ with the maximum path reward $f(n)$ is selected, i.e., $n = \arg\max_{n' \in \mathcal{O}} f(n')$. If n corresponds to the target goal, the search process terminates successfully. Otherwise, n is moved to the CLOSED set $\mathcal{C}$, and its successors $CH(n)$ are added to the OPEN set $\mathcal{O}$ for subsequent exploration, i.e., $\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$, $\mathcal{O} \leftarrow \mathcal{O} \setminus \{n\} \cup CH(n)$. The greedy best-first expansion strategy of A* search makes it highly susceptible to getting trapped in local optima due to the biases in the estimated f-values. SeeA* introduces exploratory capabilities to the A* search through the dynamic construction of a candidate set $\mathcal{D}$, achieving superior efficiency even with uniform sampling when the estimation bias of f is substantial. In SeeA*, node selection involves first constructing $\mathcal{D}$ from the OPEN set $\mathcal{O}$ using a sampling strategy, and then selecting the node n with the highest f-value from $\mathcal{D}$ for expansion. The expansion probability of the optimal node n^* in $\mathcal{O}$ by SeeA* is given by

$$P(n^* \text{ is expanded}) = P(n^* \in \mathcal{D}) \times P(n^* = \arg\max_{n \in \mathcal{D}} f(n) | n^* \in \mathcal{D}). \tag{2}$$

Advanced sampling strategies not only increase the probability of including n^* in $\mathcal{D}$, but also enhance the likelihood of its selection from $\mathcal{D}$ for expansion due to the reduced candidate set size, thus enhancing the efficiency of the search process.

Uniform sampling employed in SeeA* is a non-differentiated sampling strategy that does not incorporate any node-specific knowledge, treating all nodes equally without prioritizing more promising candidates. Cluster sampling first partitions the nodes in $\mathcal{O}$ into multiple clusters, from which an equal number of nodes are uniformly sampled from each cluster to construct the candidate set $\mathcal{D}$. Experimental results demonstrate that cluster sampling outperforms uniform sampling, yet the

underlying mechanism remains unexplored. This paper first theoretically establishes that cluster structure introduces node-specific knowledge to calibrate the epistemic process, thereby enhancing the quality of the constructed $\mathcal{D}$. Then, KeeA* is proposed to leverage domain-independent node knowledge for constructing a more efficient search algorithm.

4 The superiority of cluster sampling

For the cluster sampling strategy in SeeA*, N nodes in the OPEN set $\mathcal{O}$ are partitioned into K disjoint clusters, each containing N_i open nodes:

$$\{\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_K | \cup_{i=1}^{K} \mathcal{O}_i = \mathcal{O} \textbf{ and } \forall i \neq j, \mathcal{O}_i \cap \mathcal{O}_j = \emptyset\}. \tag{3}$$

The probability of a node belonging to the i-th cluster is given by $p_i^c = N_i/N$. The mean and variance of the true reward f^* for nodes in $\mathcal{O}_i$ are assumed to be μ_i and σ_i^2, respectively. For uniform sampling, N^c nodes are sampled from $\mathcal{O}$ to construct the candidate set $\mathcal{D}_u$, while in cluster sampling, N_i^c nodes are sampled from each cluster $\mathcal{O}_i$ to constitute $\mathcal{D}_c$ together. The probability of nodes in $\mathcal{D}$ being sampled from $\mathcal{O}_i$ is given by $p_i^s = N_i^c/N^c$, and M denotes the extreme value of f^* among the N^c candidate nodes:

$$M_a = \max_{n \in \mathcal{D}_a} f^*(n), a \in \{u, c\}. \tag{4}$$

It is worth noting that a node with a larger f^* is closer to the optimal solution, whose path reward is $f^*_{\max}$. Given the preceding assumptions, the following theorem is established.

Theorem 4.1 *The candidate set constructed by cluster sampling is closer to the optimal solution compared to uniform sampling, exhibiting a larger expected extreme value of f^*:*

$$E_{\mathcal{D}_c}[M_c] > E_{\mathcal{D}_u}[M_u] \tag{5}$$

To prove Theorem 4.1, two lemmas are provided as follows.

Lemma 4.2 *Suppose there are K component distributions, each with a probability weight p_i, mean μ_i, and variance σ_i^2. Then, the mean and variance of the overall distribution are:*

$$\mu = \sum_i p_i \mu_i, \quad \sigma^2 = \sum_i p_i \sigma_i^2 + \sum_i p_i \mu_i^2 - \left(\sum_i p_i \mu_i \right)^2. \tag{6}$$

Lemma 4.3 *Suppose a population consists of N samples with mean μ and variance σ^2. A subset $\mathcal{D}$ with n samples is drawn without replacement. Let $\bar{X}$ and S^2 denote the mean and variance of $\mathcal{D}$, respectively. Then:*

$$E[\bar{X}] = \mu, \quad Var[\bar{X}] = \frac{N-n}{N-1} \frac{\sigma^2}{n}, \quad E[S^2] = \sigma^2 \tag{7}$$

Leveraging Lemma 4.2 and 4.3, we demonstrate that $\mathcal{D}_c$, constructed through cluster sampling with $p_i^s = p_i^c$, has the same expected mean but a larger expected variance compared to $\mathcal{D}_u$, which is obtained via uniform sampling:

$$E_{\mathcal{D}_c}(\tilde{\mu}_c) = E_{\mathcal{D}_u}(\tilde{\mu}_u), \quad E_{\mathcal{D}_c}(\tilde{\sigma}_c^2) > E_{\mathcal{D}_u}(\tilde{\sigma}_u^2), \tag{8}$$

where $\tilde{\mu}_c$ and $\tilde{\mu}_u$ denote the means of real path reward f^* for nodes in $\mathcal{D}_c$ and $\mathcal{D}_u$, respectively, and $\tilde{\sigma}_c^2$ and $\tilde{\sigma}_u^2$ are the corresponding variance. The detailed proof can be found in Appendix A. According to the Gumbel extreme value theorem, the expected value of M for n samples with a mean of μ and a variance of σ^2 is

$$E[M] = \mu + \sigma \times \frac{2\gamma + \ln\left(n^4/(4\pi \ln n)\right)}{2\sqrt{2 \ln n}}, \tag{9}$$

where γ is the Euler–Mascheroni constant. Therefore, combining with Equation 8, we obtain that $E_{\mathcal{D}_c}[M_c]$ is larger than $E_{\mathcal{D}_u}[M_u]$ because cluster sampling entails a higher expected variance, and Theorem 4.1 is proved. Let $\mathcal{Q}$ denote the set of near-optimal nodes, defined as,

$$\mathcal{Q} = \{n | f^*(n) \geq (1-\varepsilon) f^*_{max}\}, a \in \{u, c\}, \tag{10}$$

where $\varepsilon < 1$ is a hyperparameter that controls the gap from optimality. Based on Theorem 4.1, the likelihood that $\mathcal{D}_c$ contains at least one near-optimal node exceeds that of $\mathcal{D}_u$, i.e., $P(\exists\, n \in \mathcal{Q}$ such that $n \in \mathcal{D}_c) > P(\exists\, n \in \mathcal{Q}$ such that $n \in \mathcal{D}_u)$. According to Equation 2, cluster sampling increases the probability of including near-optimal nodes, thereby promoting broader exploration of high-quality regions in the search space and leading to improved solution quality. Under the framework of SeeA*, drawing m samples from N nodes with uniform sampling yields a probability of m/N for including the optimal node. Under cluster sampling, m/K nodes are sampled from each of the K clusters uniformly. If the optimal node resides in a large cluster with more than N/K nodes, the probability of selecting the optimal node under cluster sampling will be lower than m/N in uniform sampling, leading to reduced search efficiency. Theorem 4.1 offers a complementary perspective by first showing that cluster sampling increases the variance of collected candidates. Leveraging the extreme value theorem, a higher expected maximum is achieved by cluster sampling, providing a theoretical justification for the advantage of cluster sampling in SeeA*.

5 KeeA* search algorithm

In SeeA* search [60], cluster sampling leverages the underlying node distribution as prior knowledge to calibrate the epistemic process, thereby improving search efficiency over uniform sampling. However, the strategy of equally allocating candidates across clusters and performing uniform sampling within each cluster remains suboptimal. What's more, the optimal sampling policy is inherently dependent on the distribution of the true path reward f^* of nodes within each cluster, which is typically inaccessible due to estimation bias. A detailed analysis is provided in Appendix C. To address these limitations, KeeA* is proposed to incorporate distributional knowledge for more effective guidance of the epistemic process. Specifically, cluster scouting and path-aware sampling are introduced to respectively improve the mean and variance of the sampled nodes, thereby enabling the construction of higher-quality candidate sets.

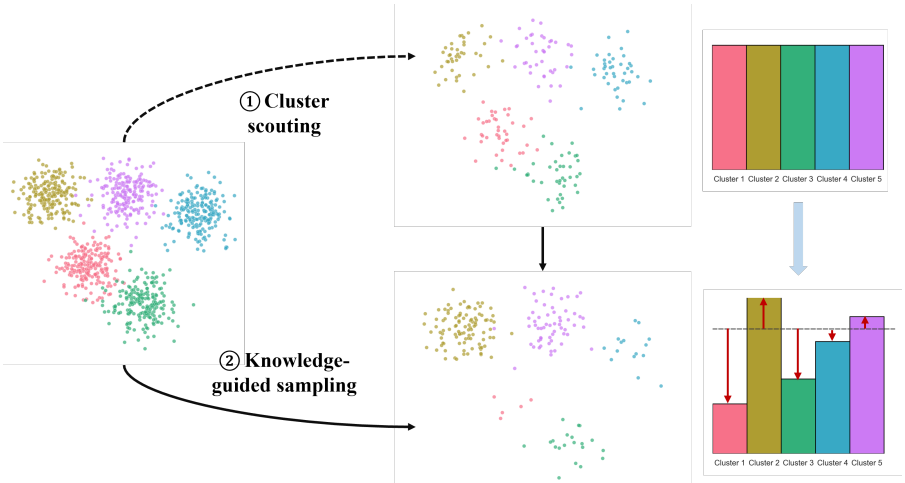

Figure 1: The sampling procedure of cluster scouting sampling.

5.1 Cluster scouting

Despite the inherent bias of f, it still encapsulates certain valuable information. Let $\mathcal{N}(\cdot,\cdot)$ denote a Gaussian distribution. Theorem 5.1 is established, and a detailed proof is provided in Appendix D.

Theorem 5.1 *Let the true path reward f^* be independently and identically sampled from $\mathcal{N}(\mu_g, \sigma_g^2)$, and assume that the prediction errors of f follow a Gaussian distribution, i.e., $f(n) \sim \mathcal{N}(f^*(n), \sigma_p^2)$. For any two nodes n_1 and n_2, the likelihood that $f^*(n_1) > f^*(n_2)$ is given by:*

$$P(f^*(n_1) > f^*(n_2)) = 1 - \frac{1}{2}\exp\left\{-\frac{\sigma_g^2(f(n_1)-f(n_2))^2}{4\sigma_p^2(\sigma_p^2+\sigma_g^2)\cos^2\xi}\right\}, \tag{11}$$

where $0 < \xi < \pi/2$ is a constant, and the probability $P(f^*(n_1) > f^*(n_2))$ increases monotonically with the magnitude of the discrepancy between $f(n_1)$ and $f(n_2)$.

In fact, only the derivation of Equation 11 explicitly relies on the Gaussian noise assumption. For general noise distributions, the probability $P(f^*(n_1) > f^*(n_2))$ remains a monotonic function of the difference $f_1 - f_2$. More details are provided in Appendix B.

According to Theorem 5.1, nodes with larger f-values exhibit a higher likelihood of being closer to the optimal solution relative to others. Motivated by this insight, cluster scouting is proposed by initially refining cluster identification through scout sampling, after which the sampling proportion is strategically increased within near-optimal clusters to enable the construction of a superior $\mathcal{D}$. As depicted in Figure 1, the sampling procedure consists of two stages:

- An equal number of nodes, denoted as $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$ are uniformly sampled from each cluster $\mathcal{O}_i$ to perform scouting and evaluate the quality of each cluster according to:

$$\text{score}(\mathcal{O}_i) = \frac{\mathbb{E}[f; \mathcal{D}_i]}{\mathbb{SD}[f; \mathcal{D}_i] + \kappa}, \tag{12}$$

  where $\mathbb{E}[\cdot]$ and $\mathbb{SD}[\cdot]$ denote the mean and standard variance of the estimated f-values of the nodes in $\mathcal{D}_i$, respectively. κ is introduced to avoid a zero variance. $\mathbb{E}[\cdot]$ is utilized to assess the quality of cluster, while $\mathbb{SD}[\cdot]$ mitigate the impact of within-cluster variance.

- Cluster sampling is performed by selecting nodes from K clusters, with the number of nodes selected proportional to the following probability:

$$p_i^s = \alpha \times \frac{\exp\{\text{score}(\mathcal{O}_i)\}}{\sum_{j=1,\cdots,K} \exp\{\text{score}(\mathcal{O}_j)\}} + (1-\alpha) \times \frac{1}{K}, \tag{13}$$

  where $\alpha \in [0, 1]$ is a hyperparameter balancing exploitation of $\text{score}(\mathcal{O}_i)$ and uniform exploration across clusters.

As higher f-values indicate a greater likelihood of superior f^*, allocating larger sampling probabilities p_i^s to such clusters increases the expected mean of $\tilde{\mu}_c$ compared to the uniform allocation strategy used in SeeA*, consequently yielding a higher expected extreme value M_c as shown in Equation 9.

5.2 Path-aware sampling

Beyond optimizing inter-cluster sampling distributions, conducting more effective intra-cluster sampling is also instrumental. While cluster scouting improves the expected mean of the sampled $\mathcal{D}_c$, path-aware sampling is proposed to leverage path information to enhance intra-cluster exploration, resulting in greater sampling variance than uniform strategies.

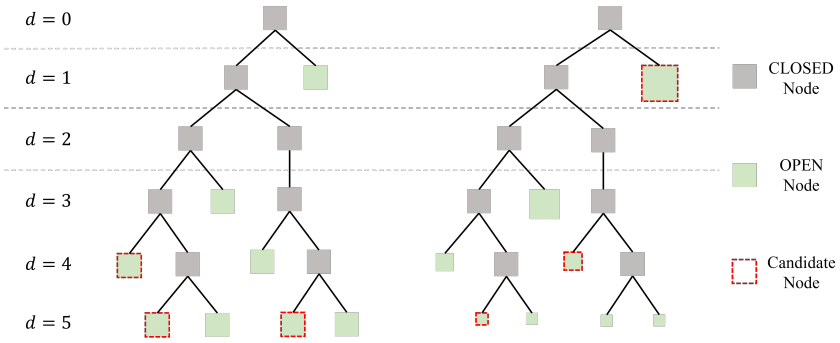

Figure 2: Comparison between uniform sampling and path-aware sampling within a cluster $\mathcal{O}_i$.

As the number of nodes grows exponentially with depth, uniform sampling disproportionately selects nodes from deeper branches as candidates, reducing exploration of shallower regions. When heuristics learned by neural networks violate the admissible condition due to their black-box nature, the search process is more susceptible to being trapped in this local optima. As illustrated in Figure 2, node depth $d(n)$ is employed as path information to bias sampling toward shallower nodes, mitigating the concentration of samples within limited deeper branches. Enhanced exploration facilitates the generation of more diverse candidate set $\mathcal{D}$, thereby increasing the sampling variance σ_i^2 and

consequently improving the expected extreme value M_c, according to Equation 9. The sampling probability of node n in cluster $\mathcal{O}_i$ is calculated by

$$p_i(n) = \beta \times \frac{\exp d(n)}{\sum_{j=1}^{N_i} \exp d(n_j)} + (1-\beta) \times \frac{1}{N_i}, \tag{14}$$

where $\beta \in [0, 1]$ is a hyperparameter that controls the trade-off between depth-aware exploration and uniform sampling.

In summary, KeeA* improves the mean and variance of candidate nodes through cluster scouting and path-aware sampling, respectively, which leads to higher extreme values of the real path reward f^* and yields solutions that are closer to the optimum. Algorithmic details are provided in Algorithm 1.

Algorithm 1: KeeA* search algorithm

Input: root node n_0, the number of cluster K, candidate size N^c, and heuristic function f.
Initialize CLOSED set $\mathcal{C} \leftarrow \emptyset$, and OPEN set $\mathcal{O} \leftarrow \cup_{i=1}^{K} \mathcal{O}_i$ with K empty clusters $\mathcal{O}_i \leftarrow \emptyset$, and cluster centers $\{w_i \mid i = 1, \cdots, K\}$
Assign root to the nearest cluster $i = \arg\min_{j=1,\cdots,K} ||w(n_0) - w_j||^2$
Update cluster: $\mathcal{O}_i \leftarrow \mathcal{O}_i \cup \{n_0\}$, $w_i \leftarrow w_i + \eta \times (w(n_0) - w_i)$
repeat
  Compute cluster-level sampling distribution $\{p_1^s, p_2^s, \cdots, p_K^s\}$ via Equation 13.
  Sample $p_i^s \times N^c$ nodes from each cluster $\mathcal{O}_i$ using Equation 14 to form candidate set $\mathcal{D}$.
  Node selection: $n \leftarrow \arg\max_{n \in \mathcal{D}} f(n)$.
  if n is the goal node **then**
    return n
  else
    Move n from $\mathcal{O}$ to $\mathcal{C}$: $\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$, $\mathcal{O} \leftarrow \mathcal{O} \setminus \{n\}$.
    for each child node $n_c \in CH(n)$ **do**
      Assign n_c to the nearest cluster: $i = \arg\min_{j=1,\cdots,K} ||w(n_c) - w_j||^2$
      Update cluster: $\mathcal{O}_i \leftarrow \mathcal{O}_i \cup \{n_c\}$, $w_i \leftarrow w_i + \eta \times (w(n_c) - w_i)$.
    end for
  end if
until $\mathcal{O}$ is empty
return False

6 Experiments

To evaluate the effectiveness of KeeA*, we consider two real-world applications: retrosynthetic planning in organic chemistry and logic synthesis in VLSI design. In both domains, heuristic functions are susceptible to significant overfitting due to the large state-space complexity and limited high-quality training data, posing significant challenges for search algorithms [59, 60]. Additional application-specific details are provided in Appendix E and F. All experiments are conducted on NVIDIA Tesla V100 GPUs and an Intel(R) Xeon(R) Gold 6238R CPU. Guiding heuristic models are consistent with SeeA* for fair comparison.

6.1 Results on retrosynthetic planning

Retrosynthetic planning aims to iteratively decompose a target molecule through predicted chemical reactions until all intermediate molecules are identified as commercially available building blocks. A single-step retrosynthetic prediction model is employed as the policy to generate candidate reactions for synthesizing input molecules, and only top 50 reaction templates ranked by predicted probability are maintained to form the legal action space. A heuristic value function is employed to estimate the synthesis cost. Each molecule is represented as a 2048-dimensional Morgan fingerprint vector [38], which serves as the input to the heuristic functions, and the last hidden state in the value network is employed as the embedding feature $w(n)$ for clustering. Both the single-step model and the cost estimator are adopted from Retro*+ [23], and are consistently used to guide the baseline algorithms.

Experiments are conducted on the widely used USPTO benchmark, which comprises 190 target molecules [4], and additional 4719 molecules collected from logP [5], logS [54], Toxicity LD50

Table 1: Success rate on seven test dataset for retrosynthetic planning problem (%).

Algorithm	USPTO	logP	logS	Toxicity LD50	Ames	BBBP	ClinTox	Mean
Retro*	86.84	53.96	67.08	55.39	57.40	47.87	38.69	54.66
Retro*+	91.05	61.14	69.29	59.98	63.51	52.46	43.15	59.93
A*	88.42	58.71	68.55	59.17	62.98	51.80	42.04	58.73
WA*	84.21	58.43	68.30	59.52	62.89	52.30	44.59	58.87
MCTS	89.47	58.15	67.08	58.26	63.42	52.95	46.34	59.20
LevinTS	96.84	61.14	70.76	60.32	64.84	54.92	43.63	61.01
PHS	87.37	55.45	65.60	57.00	59.96	50.98	39.01	56.16
SeeA*(Uniform)	96.84	63.37	71.00	62.73	**67.32**	56.39	45.70	62.97
SeeA*(Cluster)	98.42	64.12	72.73	63.53	66.08	**57.54**	47.77	63.56
KeeA* w/o CS	**98.94**	63.93	72.48	**63.76**	66.52	57.05	**48.25**	63.74
KeeA*	**98.94**	**64.21**	**72.97**	63.30	66.60	57.21	**48.25**	**63.76**

[49], Ames [17], BBBP [29], and ClinTox [15] dataset. Molecules from the *eMolecules* database[3] are used as the set of commercially available building blocks. As the majority of computational overhead arises from invoking the single-step retrosynthetic prediction model, all search algorithms are constrained to a maximum of 500 calls or 10 minutes of wall-clock time, following prior works [4, 23]. To avoid redundant computation, predictions from the single-step model are cached and reused when the same molecule is revisited [30]. The candidate size is fixed at $N^c = 50$, and the number of clusters is $K = 5$, which are consistent with SeeA*. The hyperparameters α and β in KeeA* are set as 0.5 and 0.8, respectively.

Synthesis success rate and average solution length are employed as valuation metrics for model comparison. The success rate is defined as the percentage of target molecules for which a complete and valid synthesis route is successfully identified. To penalize failures, molecules for which no valid route is found are assigned a fixed path length of 32. Experiment results are summarized in Table 1 and Table 2[4]. KeeA* achieves a success rate of 63.76% across seven benchmark datasets, slightly outperforming SeeA*, which attains 63.56%. A McNemar test is conducted to compare the success rates of KeeA* and SeeA* on paired binary outcomes. The chi-square statistic is 3.615 with a p-value of $0.0286 (< 0.05)$, indicating that KeeA* achieves a statistically significant improvement in success rate over SeeA*. See Appendix G for additional details. The average path length generated by KeeA* is 14.14, shorter than SeeA*'s 14.31. When considering only successfully synthesized molecules, KeeA* achieves an average path length of 3.95, significantly shorter than SeeA*'s 4.17. KeeA* highlighting its overall superiority by not only increasing the likelihood of finding a valid synthesis route but also identifying shorter and more efficient reaction pathways.

In terms of computational cost, SeeA* with cluster sampling requires an average runtime of 37.98 seconds per molecule on the USPTO benchmark, while KeeA* costs a comparable 37.40 seconds, demonstrating that KeeA* improves synthesis quality without introducing significant computational overhead. Ablation studies reveal that removing cluster scouting module degrades both success rate and solution quality, with the average solution length of successful cases increasing to 4.01, though still surpassing SeeA*. Both cluster scouting and path-aware sampling contribute to the overall effectiveness of synthesis planning.

6.2 Results on logic synthesis

For the logic synthesis task, an And-Inverter Graph (AIG) is optimized to minimize the area-delay product (ADP) via a sequence of functionality-preserving transformations. 7 legal transformations are allowed, and the action sequence is constrained to be 10 steps. Due to diverse functionalities in VLSI design, the logic synthesis problem exhibits substantial combinatorial complexity. The widely used *resyn2* transformation script is adopted as the baseline [7, 8, 32], and the solution S is evaluated by $1 - ADP(S)/ADP(resyn2)$, where ADP is approximately estimated using the ABC synthesis tool [2]. 12 MCNC benchmark circuits $\{C_1 \sim C_{12}\}$ [55] are used for evaluation. The guiding heuristics

[3] `http://downloads.emolecules.com/free/2023-12-01/`

[4] Values marked in red and blue correspond to the best and second-best performance, respectively. CS is short for cluster scouting.

Table 2: Solution length on seven dataset for retrosynthetic planning problem.

Algorithm	USPTO	logP	logS	ToxicityLD50	Ames	BBBP	ClinTox	Mean
Retro*	9.71	16.67	12.63	16.24	15.91	18.29	21.11	16.58
Retro*+	8.74	15.01	12.26	15.23	14.67	17.37	20.06	15.44
A*	9.27	15.64	12.44	15.49	14.94	17.56	20.26	15.78
WA*	10.16	15.62	12.46	15.39	14.90	17.36	19.43	15.66
MCTS	8.23	16.27	13.00	15.99	15.05	17.35	19.15	15.91
LevinTS	7.45	15.55	12.48	15.74	15.02	17.25	20.24	15.74
PHS	10.19	16.56	13.29	16.11	15.72	17.79	21.09	16.51
ε-Greedy	43.78	23.21	12.76	16.70	16.32	18.43	23.82	19.88
SeeA*(Uniform)	7.34	14.64	11.81	14.76	14.00	16.62	19.41	14.85
SeeA*(Cluster)	6.48	14.05	11.20	14.21	13.79	15.85	18.65	14.31
KeeA* w/o CS	6.15	13.95	11.10	**14.03**	13.70	**15.66**	18.45	14.16
KeeA*	**5.89**	**13.93**	**10.96**	14.19	**13.55**	15.82	**18.41**	**14.14**

remain consistent across the different search algorithms, and the final hidden embedding vector is employed for clustering. The candidate size is fixed at $N^c = 10$, with the number of clusters $K = 5$. Five nodes are sampled from each cluster for scouting. Hyperparameters α and β are set to 0.5 and 0.8, respectively.

Experimental results are summarized in Table 3. KeeA* achieves a 25.2% ADP improvement, surpassing SeeA* with cluster sampling, which attains a 23.5% improvement. SeeA* (Cluster) outperforms KeeA* on only 1 of the 12 benchmark circuits, highlighting the strong generalization of KeeA* across diverse design instances. With only the path-aware sampling module enabled, KeeA* still achieves a 24.0% performance improvement, exceeding SeeA* and highlighting the complementary contributions of both proposed components.

Table 3: The ADP reduction (%) rates against *resyn2* on MCNC testing datasets.

Algorithm	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	$C9$	$C10$	$C11$	$C12$	Mean
DRiLLS	18.9	6.7	8.0	13.0	38.4	19.1	5.4	18.0	14.3	18.6	6.6	11.0	14.8
Online-RL	20.6	6.6	8.1	13.5	39.4	21.0	5.0	17.9	16.2	20.2	4.7	11.4	15.4
SA+Pred.	17.6	17.0	15.6	13.0	46.5	18.2	8.5	23.6	19.9	17.6	10.0	20.3	19.0
MCTS	17.1	15.9	13.1	13.0	46.9	14.9	6.5	23.2	17.7	20.5	13.1	19.7	18.5
ABC-RL	19.9	19.6	16.8	15.0	46.9	19.1	12.1	24.3	21.3	21.1	13.6	21.6	20.9
A* search	18.3	16.6	19.7	15.7	43.6	15.2	13.3	25.5	19.4	20.8	7.5	18.8	19.5
PV-MCTS	17.3	20.0	27.9	**20.1**	27.3	20.7	13.5	24.7	14.3	14.1	14.7	20.0	19.5
PHS	21.4	17.1	11.7	8.4	47.9	5.2	8.7	10.2	20.5	12.0	7.3	20.8	15.9
SeeA*(Uniform)	21.9	18.7	21.9	16.5	37.2	13.8	12.3	25.5	21.5	24.1	**21.5**	24.0	21.6
SeeA*(Cluster)	23.2	20.8	22.7	16.2	45.9	22.6	13.4	24.8	**22.4**	24.2	20.3	25.1	23.5
KeeA* w/o CS	21.0	21.6	24.5	15.5	50.8	**26.5**	12.1	24.8	**22.4**	**27.2**	17.5	23.6	24.0
KeeA*	**23.7**	**24.6**	**28.3**	16.2	**53.6**	24.2	**14.7**	**26.0**	**22.4**	24.5	16.9	**26.7**	**25.2**

6.3 The impact of the hyperparameters

The hyperparameter α in cluster scouting controls the balance between exploration and exploitation of the heuristic function f in the allocation of candidate nodes across clusters. When $\alpha = 1.0$, the number of candidates sampled from each cluster is fully determined by the evaluations of scouted nodes, which may lead to suboptimal performance due to heuristic bias. Conversely, setting $\alpha = 0$ results in uniform allocation across clusters, failing to exploit epistemic knowledge of cluster quality and thereby reducing search efficiency. Similarly, β controls the trade-off between exploration and exploitation during intro-cluster sampling. When $\beta = 1.0$, the sampling process is dominated by biased path information, misguiding the search toward a breadth-first strategy. Conversely, $\beta = 0$ results in uniform sampling, disregarding potentially informative knowledge. As presented in Appendix H, excessively large or small values of α and β lead to performance degradation, emphasizing the necessity of tuning the hyperparameters to ensure performance robustness.

K represents the number of clusters, which is a critical parameter of KeeA*. An excessively small K induces under-fitting, leading to a model of limited flexibility, while an overly large K risks

overfitting to collected states. Both extremes impair the efficacy of decision-making during search, highlighting the importance of appropriate cluster number K selection to optimize the efficiency of KeeA*. Details are provided in Appendix I

To quantify KeeA*'s variability induced by randomness, we executed three runs for each of ten random seeds on the USPTO test set. A T-test with the significance level is used to compute the error intervals. SeeA* not only exhibits a larger variance across multiple runs but also shows greater performance fluctuations under different random seeds, highlighting the superiority of KeeA* over SeeA* (Cluster) with respect to stability. More details are provided in Appendix J.

7 Conclusion

In this paper, we first theoretically demonstrate the superiority of cluster sampling over uniform sampling within the SeeA* framework and reveal that the optimal sampling strategy is intrinsically determined by the underlying distribution of the real path reward f^*. Cluster scouting and path-aware sampling is proposed by KeeA* to exploit node distributional knowledge for epistemic calibration, enhancing the mean and variance of sampled candidate nodes, respectively. The likelihood of incorporating nodes closer to the optimal solution into the candidate set is increased, thereby enhancing the decision-making efficiency of search algorithms. Due to the dynamic construction of candidate set, KeeA* exhibits slight randomness. Nevertheless, extensive empirical experiments across thousands of test instances and statistical hypothesis testing results validate the effectiveness and robustness of KeeA*.

Beyond improving the predictive accuracy of the heuristic function, increasing the likelihood of sampling the optimal solution within a smaller $\mathcal{D}$ can also significantly improve search efficiency. Besides raising the mean reward of candidate nodes, enhancing variance improves the likelihood of sampling near-optimal nodes, providing a promising direction for future research. This work is still in the nascent stages and has not yet been applied to real-world scenarios directly impacting everyday human activities. Substantial ethical risks or detrimental social impacts are not anticipated.

8 Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 62172273, 62174110, by the Science and Technology Commission of Shanghai Municipality under Grant 24510714300, and by the Shanghai Municipal Science and Technology Major Project under Grant No. 2021SHZDZX0102, Natural Science Foundation of Shanghai (23ZR1433200).

References

[1] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 2019.

[2] Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *Computer Aided Verification: 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*, pages 24–40. Springer, 2010.

[3] Hyeong Soo Chang, Michael C Fu, Jiaqiao Hu, and Steven I Marcus. An adaptive sampling algorithm for solving markov decision processes. *Operations Research*, 53(1):126–139, 2005.

[4] Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: learning retrosynthetic planning with neural guided A* search. In *International Conference on Machine Learning*, pages 1608–1616. PMLR, 2020.

[5] Tiejun Cheng, Yuan Zhao, Xun Li, Fu Lin, Yong Xu, Xinglong Zhang, Yan Li, Renxiao Wang, and Luhua Lai. Computation of octanol- water partition coefficients by guiding an additive model with knowledge. *Journal of chemical information and modeling*, 47(6):2140–2148, 2007.

[6] Jinho Choo, Yeong-Dae Kwon, Jihoon Kim, Jeongwoo Jae, André Hottung, Kevin Tierney, and Youngjune Gwon. Simulation-guided beam search for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:8760–8772, 2022.

[7] Animesh Basak Chowdhury, Marco Romanelli, Benjamin Tan, Ramesh Karri, and Siddharth Garg. Retrieval-guided reinforcement learning for boolean circuit minimization. In *The Twelfth International Conference on Learning Representations*, 2023.

[8] Animesh Basak Chowdhury, Benjamin Tan, Ryan Carey, Tushit Jain, Ramesh Karri, and Siddharth Garg. Bulls-eye: Active few-shot learning guided logic synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[9] Eldan Cohen, Richard Anthony Valenzano, and Sheila A McIlraith. Type-wa*: Using exploration in bounded suboptimal planning. In *IJCAI*, pages 4047–4053, 2021.

[10] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[11] Tuan Q Dam, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Convex regularization in monte-carlo tree search. In *International Conference on Machine Learning*, pages 2365–2375. PMLR, 2021.

[12] Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with gumbel. In *International Conference on Learning Representations*, 2022.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[14] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia engineering*, 96:59–69, 2014.

[15] Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology*, 23(10):1294–1301, 2016.

[16] Peng Han, Peilin Zhao, Chan Lu, Junzhou Huang, Jiaxiang Wu, Shuo Shang, Bin Yao, and Xiangliang Zhang. Gnn-retro: Retrosynthetic planning with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 4014–4021, 2022.

[17] Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius Ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Muller. Benchmark data set for in silico prediction of ames mutagenicity. *Journal of chemical information and modeling*, 49(9):2077–2081, 2009.

[18] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[19] Siqi Hong, Hankz Hankui Zhuo, Kebing Jin, Guang Shao, and Zhanwen Zhou. Retrosynthetic planning with experience-guided monte carlo tree search. *Communications Chemistry*, 6(1):120, 2023.

[20] Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda. Drills: Deep reinforcement learning for logic synthesis. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 581–586. IEEE, 2020.

[21] Zhiming Hu, James Tu, and Baochun Li. Spear: Optimized dependency-aware task scheduling with deep reinforcement learning. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*, pages 2037–2046. IEEE, 2019.

[22] Hyeonah Kim, Minsu Kim, Taeyoung Yun, Sanghyeok Choi, Emmanuel Bengio, Alex Hernández-García, and Jinkyoo Park. Improved off-policy reinforcement learning in biological sequence design. In *NeurIPS Workshop on AI for New Drug Modalities*, 2024.

[23] Junsu Kim, Sungsoo Ahn, Hankook Lee, and Jinwoo Shin. Self-improved retrosynthetic planning. In *International Conference on Machine Learning*, pages 5486–5495. PMLR, 2021.

[24] Akihiro Kishimoto, Beat Buesser, Bei Chen, and Adi Botea. Depth-first proof-number search with heuristic edge cost and application to chemical synthesis planning. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.

[26] Xihan Li, Xing Li, Lei Chen, Xing Zhang, Mingxuan Yuan, and Jun Wang. Circuit transformer: A transformer that preserves logical equivalence. In *The Thirteenth International Conference on Learning Representations*, 2025.

[27] Guoqing Liu, Di Xue, Shufang Xie, Yingce Xia, Austin Tripp, Krzysztof Maziarz, Marwin Segler, Tao Qin, Zongzhang Zhang, and Tie-Yan Liu. Retrosynthetic planning with dual value networks. In *International Conference on Machine Learning*, pages 22266–22276. PMLR, 2023.

[28] Chenyang Lv, Ziling Wei, Weikang Qian, Junjie Ye, Chang Feng, and Zhezhi He. Gpt-ls: Generative pre-trained transformer with offline reinforcement learning for logic synthesis. In *2023 IEEE 41st International Conference on Computer Design (ICCD)*, pages 320–326. IEEE, 2023.

[29] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[30] Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gaiński, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. *arXiv preprint arXiv:2310.19796*, 2023.

[31] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[32] Walter Lau Neto, Yingjie Li, Pierre-Emmanuel Gaillardon, and Cunxi Yu. Flowtune: End-to-end automatic logic optimization exploration via domain-specific multi-armed bandit. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[33] Laurent Orseau, Levi Lelis, Tor Lattimore, and Théophane Weber. Single-agent policy tree search with guarantees. *Advances in Neural Information Processing Systems*, 31, 2018.

[34] Judea Pearl and Jin H Kim. Studies in semi-admissible heuristics. *IEEE transactions on pattern analysis and machine intelligence*, (4):392–399, 1982.

[35] Yasasvi V Peruvemba, Shubham Rai, Kapil Ahuja, and Akash Kumar. Rl-guided runtime-constrained heuristic exploration for logic synthesis. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[36] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[37] Hao Qian, Cheng Lin, Dengwei Zhao, Shikui Tu, and Lei Xu. Alphadrug: protein target specific de novo molecular generation. *PNAS nexus*, 1(4):pgac227, 2022.

[38] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[39] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[40] Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610, 2018.

[41] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[42] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[43] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[44] Richard Valenzano, Nathan Sturtevant, Jonathan Schaeffer, and Fan Xie. A comparison of knowledge-based gbfs enhancements and knowledge-free exploration. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 375–379, 2014.

[45] Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *Forty-first International Conference on Machine Learning*, 2024.

[46] Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q-H Meng. Neural rrt*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, 17(4):1748–1758, 2020.

[47] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. Empowering a* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 539–547, 2019.

[48] Yi Wang, Hui Tang, Lichao Huang, Lulu Pan, Lixiang Yang, Huanming Yang, Feng Mu, and Meng Yang. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, 5(8):845–860, 2023.

[49] Kedi Wu and Guo-Wei Wei. Quantitative toxicity prediction using topology based multitask deep neural networks. *Journal of chemical information and modeling*, 58(2):520–531, 2018.

[50] Chenjun Xiao, Ruitong Huang, Jincheng Mei, Dale Schuurmans, and Martin Müller. Maximum entropy monte-carlo planning. *Advances in Neural Information Processing Systems*, 32, 2019.

[51] Fan Xie, Martin Müller, Robert Holte, and Tatsuya Imai. Type-based exploration with multiple search queues for satisficing planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[52] Shufang Xie, Rui Yan, Peng Han, Yingce Xia, Lijun Wu, Chenjuan Guo, Bin Yang, and Tao Qin. Retrograph: Retrosynthetic planning with graph search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2129, 2022.

[53] Zhihao Xing and Shikui Tu. A graph neural network assisted monte carlo tree search approach to traveling salesman problem. *IEEE Access*, 8:108418–108428, 2020.

[54] Guoli Xiong, Zhenxing Wu, Jiacai Yi, Li Fu, Zhijiang Yang, Changyu Hsieh, Mingzhu Yin, Xiangxiang Zeng, Chengkun Wu, Aiping Lu, et al. Admetlab 2.0: an integrated online platform for accurate and comprehensive predictions of admet properties. *Nucleic Acids Research*, 49(W1):W5–W14, 2021.

[55] Saeyang Yang. *Logic synthesis and optimization benchmarks user guide: version 3.0*. Citeseer, 1991.

[56] Michael Yao, Yimeng Zeng, Hamsa Bastani, Jacob Gardner, James Gee, and Osbert Bastani. Generative adversarial model-based optimization via source critic regularization. *Advances in Neural Information Processing Systems*, 37:44009–44039, 2024.

[57] Yemin Yu, Ying Wei, Kun Kuang, Zhengxing Huang, Huaxiu Yao, and Fei Wu. Grasp: Navigating retrosynthetic planning with goal-driven policy. *Advances in Neural Information Processing Systems*, 35:10257–10268, 2022.

[58] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: LLM self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

[59] Dengwei Zhao, Shikui Tu, and Lei Xu. Efficient retrosynthetic planning with MCTS exploration enhanced A* search. *Communications Chemistry*, 7(1):52, 2024.

[60] Dengwei Zhao, Shikui Tu, and Lei Xu. Seea*: Efficient exploration-enhanced A* search by selective sampling. *Advances in Neural Information Processing Systems*, 37:104138–104179, 2024.

[61] Keren Zhu, Mingjie Liu, Hao Chen, Zheng Zhao, and David Z Pan. Exploring logic optimizations with reinforcement learning and graph convolutional network. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pages 145–150, 2020.

A Proof of the sampling expectation

In the section, we will provide the detailed proof of comparing the uniform sampling and cluster sampling with $p_i^s = p_i^c = N_i/N$. In this situation, the number of nodes selected by cluster sampling from $\mathcal{O}_i$ is $p_i^c \times N^c$, and the number of nodes selected from cluster $\mathcal{O}_i$ by uniform sampling, denoted as N_i^u, is also expected to be proportional to the cluster size N_i, i.e., $E[N_i^u] = p_i^c \times N^c$. According to Lemma 4.2, the mean and variance of the global distribution encompassing K clusters are:

$$\mu_g = \sum_{i=1}^{K} p_i^c \mu_i, \quad \sigma_g^2 = \sum_{i=1}^{K} p_i^c \sigma_i^2 + \sum_{i=1}^{K} p_i^c \mu_i^2 - \mu_g^2. \tag{15}$$

In each sampling iteration, N^c nodes are directly acquired from the global distribution, exhibiting empirical mean and variance $\tilde{\mu}_u$ and $\tilde{\sigma}_u^2$, respectively. Building upon Lemma 4.3, the expected values of $\tilde{\mu}_u$ and $\tilde{\sigma}_u^2$ of the sampled nodes using uniform sampling are:

$$E_{\mathcal{D}_u}[\tilde{\mu}_u] = \mu_g = \sum_{i=1}^{K} p_i^c \mu_i, \quad E_{\mathcal{D}_u}[\tilde{\sigma}_u^2] = \sigma_g^2 = \sum_{i=1}^{K} p_i^c \sigma_i^2 + \sum_{i=1}^{K} p_i^c \mu_i^2 - \mu_g^2 \tag{16}$$

For cluster sampling, let $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$ denote the empirical mean and variance, respectively, of the N_i^c nodes drawn from $\mathcal{O}_i$. According to Lemma 4.3, the following relations hold:

$$E_{\mathcal{D}_c}[\tilde{\mu}_i] = \mu_i, \quad Var_{\mathcal{D}_c}[\tilde{\mu}_i] = \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c}, \quad E_{\mathcal{D}_c}[\tilde{\sigma}_i^2] = \sigma_i^2 \tag{17}$$

Based on Lemma 4.2, the mean and variance of the candidate set $\mathcal{D}$ are:

$$\tilde{\mu}_c = \sum_{i=1}^{K} p_i^s \tilde{\mu}_i, \quad \tilde{\sigma}_c^2 = \sum_{i=1}^{K} p_i^s \tilde{\sigma}_i^2 + \sum_{i=1}^{K} p_i^s \tilde{\mu}_i^2 - \tilde{\mu}_c^2 \tag{18}$$

Considering that $p_i^s = p_i^c$, we have:

$$E_{\mathcal{D}_c}[\tilde{\mu}_c] = \sum_{i=1}^{K} p_i^s E[\tilde{\mu}_i] = \sum_{i=1}^{K} p_i^s \mu_i = E[\tilde{\mu}_u] \tag{19}$$

$$E_{\mathcal{D}_c}[\tilde{\sigma}_c^2] = \sum_{i=1}^{K} p_i^s \sigma_i^2 + \sum_{i=1}^{K} p_i^s \left(\frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c} + \mu_i^2 \right) - \left[\sum_{i=1}^{K} (p_i^s)^2 \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c} + E_{\mathcal{D}_c}[\tilde{\mu}_c]^2 \right] \tag{20}$$

$$\begin{aligned} E_{\mathcal{D}_c}[\tilde{\sigma}_c^2] - E_{\mathcal{D}_u}[\tilde{\sigma}_u^2] &= \sum_{i=1}^{K} p_i^c \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c} - \sum_{i=1}^{K} (p_i^c)^2 \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c} \\ &= \sum_{i=1}^{K} p_i^c \frac{N_i - N_i^c}{N_i - 1} \frac{1 - p_i^c}{N_i^c} \sigma_i^2 > 0 \end{aligned} \tag{21}$$

Therefore, for cluster sampling proportional to the cluster size, the expected value of the mean $E_{\mathcal{D}_c}[\tilde{\mu}_c]$ remains consistent with $E_{\mathcal{D}_u}[\tilde{\mu}_u]$, but the expected variance $E_{\mathcal{D}_c}[\tilde{\sigma}_c^2]$ is larger than $E_{\mathcal{D}_u}[\tilde{\sigma}_u^2]$.

B Error assumption of Theorem 5.1

Due to the presence of prediction errors, for two nodes n_1 and n_2, we have that $f_1 = f_1^* + \epsilon_1$ and $f_2 = f_2^* + \epsilon_2$, where ϵ is the prediction error of f to the real evaluation f^*. Define $\Delta = f_2 - f_1 > 0$ and $\eta = \epsilon_2 - \epsilon_1$, we have that

$$P(f_2^* > f_1^* | f_2 - f_1 = \Delta) = P(f_2^* - f_1^* > 0 | \Delta) = P(\Delta > \eta | \Delta). \tag{22}$$

Since $P(\Delta > \eta|\Delta)$ is increasing with Δ, $P(f_2^* > f_1^*|f_2 - f_1 = \Delta)$ also increases with the gap between f_1 and f_2. The monotonicity property established in Theorem 5.1 remains valid even when the noise distribution ϵ is non-Gaussian and node-dependent.

C Suboptimal of cluster sampling in SeeA*

We further delve into the sampling strategies within cluster sampling by decomposing Equation 20 into

$$E_{\mathcal{D}_c}\left[\tilde{\sigma}^2\right] = \phi(\mu_1, \cdots, \mu_K) + \psi(\sigma_1^2, \cdots, \sigma_K^2), \tag{23}$$

where

$$\phi = \sum_{i=1}^{K} p_i^s \mu_i^2 - \left(\sum_{i=1}^{K} p_i^s \mu_i\right)^2, \tag{24}$$

$$\psi = \sum_{i=1}^{K} p_i^s \sigma_i^2 + \sum_{i=1}^{K} p_i^s \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c} - \sum_{i=1}^{K} \left(p_i^s\right)^2 \frac{N_i - N_i^c}{N_i - 1} \frac{\sigma_i^2}{N_i^c}. \tag{25}$$

ϕ is the variance of the mean of different clusters, while ψ evaluates the uncertainty of the sampling distribution. Given that the true node path reward f^* is unobservable, the corresponding mean μ_i and variance σ_i^2 for cluster $\mathcal{O}_i$ are inherently difficult to estimate. Assume that μ_i for cluster $\mathcal{O}_i$ is drawn from a distribution with $E[\mu_i] = \mu_{m*}$ and $Var[\mu_i] = \sigma_{m*}^2$, and that variance σ_i^2 is also sampled from a distribution with $E[\sigma_i^2] = \mu_{v*}$ and $Var[\sigma_i^2] = \sigma_{v*}^2$. Therefore,

$$E_\mu[E_{\mathcal{D}_c}[\tilde{\mu}]] = \sum_{i=1}^{K} p_i^s E[\mu_i] = \mu_{m*} \tag{26}$$

$$\begin{aligned}
E_\mu[\phi] &= \sum_{i=1}^{K} p_i^s E[\mu_i^2] - E\left[\sum_{i=1}^{K} \left(p_i^s\right)^2 \mu_i^2 + \sum_{i \neq j} p_i^s p_j^s \mu_i \mu_j\right] \\
&= \sigma_{m*}^2 + \mu_{m^*}^2 - \sum_{i=1}^{K} \left(p_i^s\right)^2 \left(\sigma_{m*}^2 + \mu_{m^*}^2\right) - \sum_{i \neq j} p_i^s p_j^s \mu_{m^*}^2 \\
&= \sigma_{m*}^2 + \mu_{m^*}^2 - \sum_{i=1}^{K} \left(p_i^s\right)^2 \sigma_{m*}^2 - \left(\sum_{i=1}^{K} p_i\right)^2 \mu_{m*}^2 \\
&= \sigma_{m*}^2 \left(1 - \sum_{i=1}^{K} (p_i^s)^2\right) \leq \frac{K-1}{K} \sigma_{m*}^2,
\end{aligned} \tag{27}$$

where the equality holds if and only if $\forall i$, $p_i^s = 1/K$. Therefore, $E_{\mu,\sigma^2}^{eq}[\phi]$, which sampling equal number of nodes from each cluster, is larger than $E_{\mu,\sigma^2}^{prop}[\phi]$, which proportional sampling from each cluster. What's more,

$$\begin{aligned}
E_\mu[\psi] &= \mu_{v*} \left[1 + \sum_{i=1}^{K} (1 - p_i^s) \frac{N_i - p_i^s N^c}{(N_i - 1) N^c}\right] \approx \mu_{v*} \left[1 + \sum_{i=1}^{K} (1 - p_i^s) \left(\frac{1}{N^c} - \frac{p_i^s}{N_i}\right)\right] \\
&= \mu_{v*} \left[1 + \frac{K-1}{N^c} + \sum_{i=1}^{K} \frac{p_i^s (p_i^s - 1)}{N_i}\right] \leq \mu_{v*} \left(1 + \frac{K-1}{N^c}\right),
\end{aligned} \tag{28}$$

where the equality holds if and only if $\exists\, i$, $p_i^s = 1$ and $\forall j \neq i$, $p_j^s = 0$. What's more, if each cluster has the same number of nodes, i.e., $N_1 = N_2 = \cdots = N_K$, $p_i = 1/K$ will minimize $E_\mu[\psi]$. In summary, sampling from each cluster evenly maximizes $E_\mu[\phi]$, while sampling from single cluster maximizes $E_\mu[\psi]$. Considering both Equation 27 and 28, Lagrange multiplier method is employed to maximize $E_\mu[E_{\mathcal{D}}[\tilde{\sigma}^2]]$:

$$\begin{aligned}
\max \quad & \sigma_{m*}^2 \left(1 - \sum_{i=1}^{K} (p_i^s)^2\right) + \mu_{v*} \left[1 + \frac{K-1}{N^c} + \sum_{i=1}^{K} \frac{p_i^s (p_i^s - 1)}{N_i}\right] \\
\text{subject to} \quad & \sum_{i=1}^{K} p_i^s = 1, \quad 1 \leq p_i^s \leq 1 \quad \forall i.
\end{aligned}$$

The optimal sampling distribution is $p_i^s = \min\left(1, \max\left(0, \frac{\lambda N_i + \mu_{v*}}{2(\mu_{v*} - \sigma_{m*}^2 N_i)}\right)\right)$, where λ is a constant to ensure $\sum_{i=1}^{K} p_i^s = 1$. Based on Equation 26, the expectation of the mean of sampled nodes is independent of the sampling distribution p_i^s, and the optimal p_i^s to maximize the expectation of the sampling variance is related to the unknown σ_{m*}^2 and σ_{v*}^2. Therefore, considering Equation 9, selecting nodes from each cluster evenly employed in SeeA* is not guaranteed to be the optimal for cluster sampling, and the optimal sampling distribution still needs further investigation to achieve a trade-off between $E_\mu[\phi]$ and $E_\mu[\psi]$.

D Proof of Theorem 5.1

We begin by establishing a lemma that supports the proof of Theorem 5.1.

Lemma D.1 *Assume x is random variable from distribution $\mathcal{N}(\mu_0, \sigma_0^2)$, y from $\mathcal{N}(\mu_1, \sigma_1^2)$. If x and y are independent of each other and $\mu_1 > \mu_0$, then*

$$P(x > y) = \frac{1}{2} \exp\left\{ -\frac{1}{2} \frac{\left[(\mu_1 - \mu_0)/\sqrt{\sigma_0^2 + \sigma_1^2}\right]^2}{\cos^2 \xi} \right\} \tag{29}$$

where $0 < \xi < \pi/2$ is a constant.

According to Bayes' theorem,

$$P(f^*|f) \propto P(f|f^*) \times P(f^*) = \mathcal{N}(f^*, \sigma_p^2) \times \mathcal{N}(\mu_g, \sigma_g^2) = \mathcal{N}\left(\frac{\sigma_g^2 f + \sigma_p^2 \mu_g}{\sigma_p^2 + \sigma_g^2}, \frac{\sigma_p^2 \sigma_g^2}{\sigma_p^2 + \sigma_g^2}\right) \tag{30}$$

According to Lemma D.1, for two nodes n_1 and n_2 with $f(n_1) > f(n_2)$:

$$P(f^*(n_1) > f^*(n_2)) = 1 - P(f^*(n_1) < f^*(n_2)) = 1 - \frac{1}{2} \exp\left\{ -\frac{\sigma_g^2 (f(n_1) - f(n_2))^2}{4\sigma_p^2 (\sigma_p^2 + \sigma_g^2) \cos^2 \xi} \right\}. \tag{31}$$

According to Equation 31, likelihood $P(f^*(n_1) > f^*(n_2))$ increases with the difference $f(n_1) - f(n_2)$. Theorem 5.1 is proved.

E Introduction of retrosynthesis planning

Retrosynthetic planning aims to identify a synthetic route from available molecules to a target compound. It can be formulated as a Markov Decision Process and solved using heuristic tree search algorithms:

- **State**: The current set of molecules obtained through a sequence of retrosynthetic steps from the target molecule.
- **Action**: The reaction provided by the single-step retrosynthesis model to synthesize the first non-building-block intermediate in the current retrosynthetic state
- **Reward**: The cost of a chemical reaction is defined as the negative logarithm of its predicted probability by the single-step retrosynthesis model, and the reward is given as the negative of this cost.
- **Transition function**: A new molecular state is obtained by replacing the product in the current molecular set with the reactants of the applied chemical reaction.

A terminal state is reached when all molecules are identified as available building blocks, indicating a complete synthetic route. The single-step retrosynthesis model is designed for a multi-class task based on $381,302$ chemical reaction templates. The input is a 2048-dimensional Morgan Fingerprint vector [38], and the architecture of the policy network is:

- A fully connected layer with dimensions $[2048, 512]$.

- A batch normalization layer.
- A dropout layer with a dropout rate of 0.3.
- A fully connected layer with dimensions $[512, 381302]$.
- A softmax layer.

The synthesis cost of a state is defined as the sum of the synthesis costs of all molecules in the set, where the cost of available building blocks is 0, and the synthesis costs of other molecules are estimated by a well-trained value network. The input is a 2048-dimensional molecular vector, and the architecture is as follows:

- A fully connected layer with dimensions $[2048, 128]$.
- A ReLU activation layer.
- A dropout layer with a dropout rate of 0.1.
- A fully connected layer with dimensions $[128, 1]$.
- Normalize the output y with $log(1 + e^y)$.

The parameters of the policy and value networks are identical to those of Retro*+ [23], and also consistent with SeeA* [60]. The set of molecules used for testing is derived from the following seven datasets:

- **USPTO** [4]: 190 molecules are collected from the United States Patent and Trademark Office (USPTO) used primarily for retrosynthetic planning and related research. Only molecules for which reactions in the synthesis route are all covered by the top-50 predictions by the one-step model are kept.
- **logP** [5]: The logarithm of the partition coefficient quantifies the solubility of a molecule in a particular solvent, which is crucial for understanding the molecule's pharmacokinetics and pharmacodynamics. 1073 molecules are collected.
- **logS** [54]: This property is used to assess the solubility of molecules, which significantly impacts the absorption, distribution, metabolism, and excretion of drug candidates. 407 molecules are contained in the logS.
- **Toxicity LD50** [49]: The LD50 value measures the toxicity of a substance, playing a critical role in evaluating the safety and efficacy of pharmaceutical compounds. 872 molecules are considered.
- **Ames** [17]: The Ames test is widely employed in drug development to assess the mutagenic potential of drug candidates and other chemicals used in drug formulation or as excipients. Ames benchmark consists of 1129 molecules.
- **BBBP** [29]: The Blood-Brain Barrier Penetration dataset evaluates the ability of molecules to cross the blood-brain barrier, a critical factor in the development of central nervous system drugs. 610 molecules are included.
- **ClinTox** [15]: A dataset comprising 628 FDA-approved drugs and those that failed clinical trials due to toxicity concerns, providing insight into the safety profiles of pharmaceutical compounds.

F Introduction of logic synthesis

Logic synthesis refers to the process of converting a hardware design described at the register-transfer level (RTL) into a gate-level Boolean representation, typically modeled as an And-Inverter Graph (AIG)—a netlist composed solely of AND and NOT gates. An optimized AIG is then produced by applying a series of semantics-preserving transformations aimed at improving circuit performance, area, and power. The logic synthesis problem can also be formulated as a Markov Decision Process and addressed using tree search algorithms:

- **State**: The current state is encoded as an And-Inverter Graph, which captures the structural and functional characteristics of the circuit.

- **Action**: Following ABC [2] and recent reinforcement learning approaches [7, 20], seven logic optimization operations are permitted: balance, re-substitution, re-substitution -z, rewrite, rewrite -z, refactor, and refactor -z.
- **Reward**: *resyn2* synthesis recipe is used as the baseline during the evaluation. The immediate reward for non-termination state is 0, and the area-delay product (ADP) reduction for an action sequence S is used as the reward for termination state:

$$R(S,t) = \begin{cases} \max\{-1, 1 - \frac{ADP(S)}{ADP(resyn2)}\}, & \text{if } t = |S| \\ 0, & \text{Otherwise} \end{cases} \tag{32}$$

  where $ADP(\cdot)$ is estimated by the ABC library [2].
- **Transition model**: Each action is designed to perform a global structural transformation on the AIG. The state transition function is implemented using ABC [2], resulting in a newly transformed AIG.

A value network is employed to estimate the expected reduction in ADP, thereby guiding the search process. The input to the value estimator consists of the initial AIG and the sequence of actions. Each node in the AIG is encoded as a two-dimensional vector that captures both types and the number of inverted predecessors, while the adjacency matrix encodes the structural connectivity of AIG. A Graph Convolutional Network (GCN) is utilized to extract the AIG embedding, with the architecture specified as follows:

- A GCN layer with a hidden size of 32.
- A batch normalization layer.
- A LeakyReLU activation layer.
- A second GCN layer with a hidden size of 32.
- A batch normalization layer.
- Mean pooling and max pooling are applied independently, and their outputs are concatenated to yield a 64-dimensional graph embedding.

The action sequence, along with the number of steps, is encoded as a string and passed to a BERT model [13], which generates a 768-dimensional sequence embedding. The final input to the value estimator is formed by concatenating the 64-dimensional AIG embedding with the 768-dimensional BERT embedding. The architecture of the value estimator is as follows:

- A fully connected layer with dimensions $[832, 256]$.
- A LeakyReLU activation layer.
- A fully connected layer with dimensions $[256, 256]$.
- A LeakyReLU activation layer.
- A fully connected layer with dimensions $[256, 1]$.
- A Tanh activation layer.

The parameters of the value networks are identical to those of SeeA* [60]. Twelve benchmark circuits from the MCNC dataset are used for evaluation, each exhibiting varying numbers of components and structure connections. Detailed specifications are provided in Table 4.

G McNemar test on success rate

To assess the statistical significance of the difference in success rates between two models on paired binary outcomes, we employ the McNemar test [31]. This non-parametric test is employed to evaluate classification performance differences on matched samples. Given a 2×2 contingency table:

	Model B Correct	Model B Incorrect
Model A Correct	n_{11}	n_{10}
Model A Incorrect	n_{01}	n_{00}

Table 4: Characterization of testing circuits from MCNC dataset.

Circuit	# Inputs	# Outputs	# Nodes	Depth
alu4	10	6	735	42
apex1	45	45	2655	27
apex2	39	3	445	29
apex4	9	19	3452	21
b9	41	21	105	10
c880	60	26	327	24
c7552	207	108	2074	29
i9	88	63	889	14
m4	8	16	760	14
pair	173	137	1500	24
max1024	10	6	1021	20
prom1	9	40	7803	24

The test statistic is defined as:

$$\chi^2 = \frac{(n_{01} - n_{10})^2}{n_{01} + n_{10}} \tag{33}$$

Under the null hypothesis of no difference, χ^2 approximately follows a chi-square distribution with one degree of freedom. A p-value less than 0.05 indicates a statistically significant difference in performance.

The synthesis outcomes of $4,909$ molecules across seven datasets are used for hypothesis testing. The resulting chi-square statistic was 3.615 with a p-value of $0.0286 (< 0.05)$, leading to the rejection of the null hypothesis. These results indicate that KeeA* achieves a statistically significant improvement over SeeA*.

H An investigation of hyperparameters

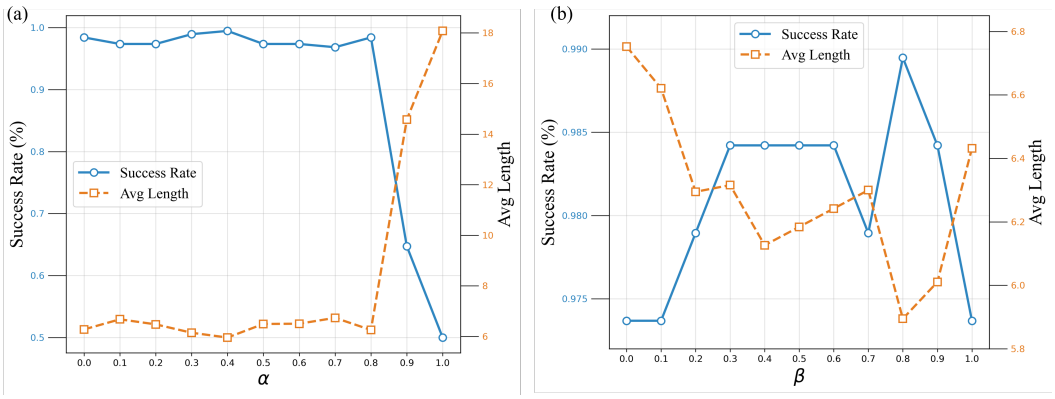

Figure 3: Success rate and average solution length on the USPTO benchmark with different (a) α and (b) β.

The comparative experiments under different hyperparameter settings on the USPTO and ClinTox test benchmark are presented in Figure 3 and 4, respectively. The best success rate and average solution length are achieved when $\alpha = 0.4$. As α approaches 0, performance slightly degrades; however, a more significant drop is observed when $\alpha \to 1.0$, highlighting the adverse impact of an overly imprecise heuristic on the search process. KeeA* performs best when $\beta = 0.8$, while both excessively large and small values of β lead to performance degradation. These results indicate that appropriate choices of α and β effectively balance exploration and exploitation, thereby enhancing the overall search performance. The comparative experimental results on the b9 circuit for logic synthesis task are shown in Figure 5, yielding a similar conclusion: appropriate values of α and β lead to improved performance.

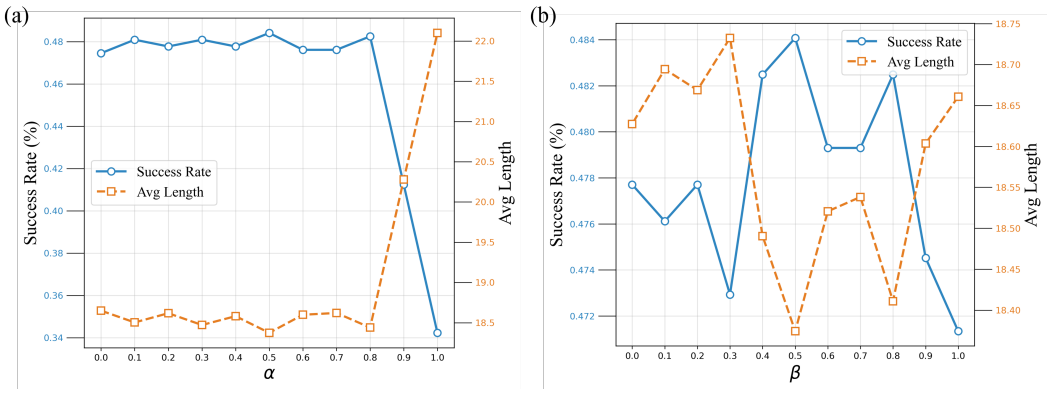

Figure 4: Success rate and average solution length on the ClinTox benchmark with different (a) α and (b) β.

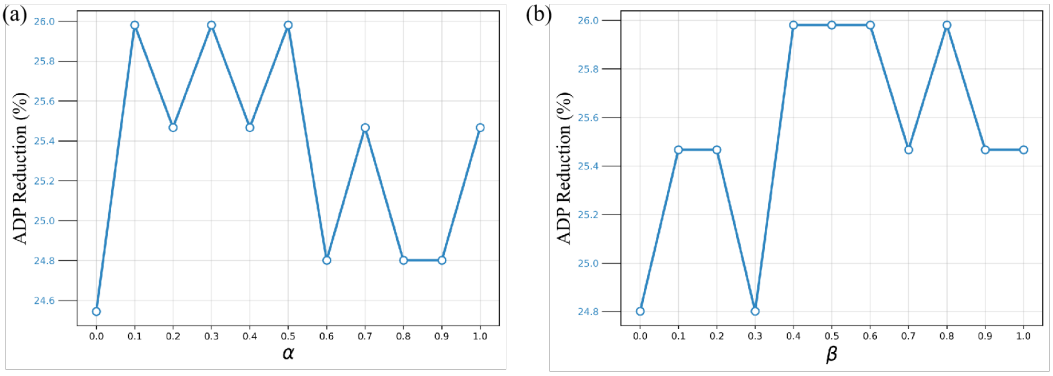

Figure 5: ADP reduction on the b9 circuit on logic synthesis with different (a) α and (b) β.

To demonstrate the robustness of KeeA*, an exhaustive grid search is conducted with three independent runs per configuration, and a T-test at the 0.05 significance level is used to compute the error intervals. Because only three runs are conducted, the deviation is relatively large. Both the mean and the error interval are reported in Table 5. KeeA* demonstrates robust performance, consistently outperforming SeeA*'s $97.30 \pm 2.36\%$ across a wide range of hyperparameter settings, with the highest success rate reaching $98.77 \pm 0.62\%$.

	$\beta = 0.1$	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.7$	$\beta = 0.9$
$\alpha = 0.1$	97.72 ± 2.46	98.07 ± 2.68	97.54 ± 1.64	97.72 ± 1.64	97.72 ± 1.24
$\alpha = 0.3$	97.89 ± 2.83	97.72 ± 1.64	98.60 ± 2.21	98.60 ± 2.68	97.54 ± 1.24
$\alpha = 0.5$	97.54 ± 2.46	98.77 ± 2.46	98.07 ± 0.62	97.37 ± 1.84	97.89 ± 0.00
$\alpha = 0.7$	98.25 ± 0.62	98.07 ± 1.64	98.07 ± 0.62	98.77 ± 0.62	97.89 ± 1.07
$\alpha = 0.9$	64.74 ± 7.70	62.81 ± 8.57	64.04 ± 6.43	60.88 ± 4.32	64.74 ± 2.14

Table 5: Success rate of KeeA* under different α and β on the USPTO benchmark (%).

I Investigation of cluster number K

To assess the influence of the number of clusters K on KeeA*, three runs are performed for various K, and a T-test with the 0.05 significance level is used to compute the error intervals. Because only three runs are conducted, the error interval is relatively large. KeeA* consistently outperforms SeeA* (Cluster) on the USPTO test set. An excessively small K induces under-fitting, leading to a model of limited flexibility, while an overly large K risks overfitting to collected states. Both extremes impair

Table 6: Success rate of KeeA* and SeeA* (Cluster) with different cluster number K on the USPTO benchmark (%).

K	KeeA*	SeeA* (Cluster)
3	$\mathbf{95.44 \pm 0.66}$	94.91 ± 1.74
4	$\mathbf{97.54 \pm 0.50}$	97.37 ± 0.86
5	$\mathbf{98.53 \pm 0.60}$	97.30 ± 0.95
6	$\mathbf{97.36 \pm 1.55}$	96.32 ± 0.43
7	$\mathbf{95.26 \pm 0.86}$	94.04 ± 0.89
8	$\mathbf{94.91 \pm 1.74}$	92.98 ± 2.21

the efficacy of decision-making during search, highlighting the importance of appropriate cluster number K selection to optimize the efficiency of SeeA* and KeeA*.

J Variability of KeeA* due to randomness.

To quantify KeeA*'s variability induced by randomness, we executed three runs for each of ten random seeds on the USPTO test set. A T-test with the significance level is used to compute the error intervals. The average success rate over all 30 runs is 98.53%, which is comparable to the 98.84% reported in the paper. KeeA* exhibits robust performance across various random seeds, achieving more stable results compared to SeeA* (Cluster). SeeA* not only exhibits a larger variance across multiple runs but also shows greater performance fluctuations under different random seeds, highlighting the superiority of KeeA* over SeeA* (Cluster) with respect to stability.

Table 7: Success rate of KeeA* and SeeA* (Cluster) with different random seed on the USPTO benchmark (%).

Seed	KeeA*	SeeA* (Cluster)
0	$\mathbf{98.07 \pm 0.25}$	96.49 ± 0.66
1	97.89 ± 0.86	$\mathbf{98.07 \pm 0.66}$
2	$\mathbf{98.60 \pm 0.25}$	96.84 ± 0.00
3	$\mathbf{98.60 \pm 0.50}$	97.89 ± 0.43
4	98.25 ± 0.25	$\mathbf{98.60 \pm 0.25}$
5	$\mathbf{98.95 \pm 0.43}$	97.37 ± 0.00
6	$\mathbf{99.30 \pm 0.50}$	96.67 ± 1.08
7	$\mathbf{98.95 \pm 0.00}$	96.67 ± 0.89
8	$\mathbf{98.07 \pm 0.25}$	96.84 ± 1.14
9	$\mathbf{98.60 \pm 0.25}$	97.54 ± 0.25
All	$\mathbf{98.53 \pm 0.60}$	97.30 ± 0.95

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Conclusion section

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Assumptions and detailed proof are provided in the main text and appendix materials.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The algorithmic procedure, network architecture, and hyperparameter settings are all detailed in the paper, and both the code and datasets will be made publicly available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided in the supplemental material and will be publicly available once acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All details are summarized in the Experiment section and appendix

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Although error bars are not provided, the proposed method was evaluated on over $4,909$ test instances — far exceeding the 190 molecules used in previous studies. The McNemar test is employed to statistically validate the robustness of KeeA*.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computer resources employed in this work is summarized in Experiments section

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that this paper conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussion of societal impacts is provided in the Conclusion section.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work focuses on heuristic search algorithms and currently poses no risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All materials used in this study are publicly available and well-cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets, but the associated code will be made publicly available upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are not involved as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.