Shortest Edit Path Crossover: A Theory-driven Solution to the Permutation Problem in Evolutionary Neural Architecture Search

Xin Qiu [1] **Risto Miikkulainen** [1 2]

Abstract

Population-based search has recently emerged as a possible alternative to Reinforcement Learning (RL) for black-box neural architecture search (NAS). It performs well in practice even though it is not theoretically well understood. In particular, whereas traditional population-based search methods such as evolutionary algorithms (EAs) draw much power from crossover operations, it is difficult to take advantage of them in NAS. The main obstacle is believed to be the permutation problem: The mapping between genotype and phenotype in traditional graph representations is many-to-one, leading to a disruptive effect of standard crossover. This paper presents the first theoretical analysis of the behaviors of mutation, crossover and RL in black-box NAS, and proposes a new crossover operator based on the shortest edit path (SEP) in graph space. The SEP crossover is shown theoretically to overcome the permutation problem, and as a result, have a better expected improvement compared to mutation, standard crossover and RL. Further, it empirically outperform these other methods on state-of-the-art NAS benchmarks. The SEP crossover therefore allows taking full advantage of population-based search in NAS, and the underlying theory can serve as a foundation for deeper understanding of black-box NAS methods in general.

1. Introduction

Neural architecture search (NAS), a technique for automatically designing architectures for neural networks, outperforms human-designed models in many tasks (Zoph et al., 2018; Chen et al., 2018; Miikkulainen et al., 2021). One major branch of NAS approaches are the black-box NAS methods, which require only zeroth-order information about the objectives. While reinforcement learning (RL) contributed to the early success of black-box NAS methods (Zoph & Le, 2017), population-based search has emerged recently as a popular and empirically more powerful alternative (Real et al., 2017; 2019; Ying et al., 2019), achieving SOTA performance in various benchmarks and real-world domains (Real et al., 2017; Elsken et al., 2019; Real et al., 2019; So et al., 2021; Gao et al., 2022). Population-based NAS is usually based on evolutionary algorithms (EAs) (Liu et al., 2021), which mimic natural evolution by maintaining a population of solutions and evolving them through mutation and crossover. Mutation provides for local search (i.e. refinement), while crossover implements a directed global search, and thus constitutes the engine behind evolutionary discovery. However, most recent evolutionary NAS methods are limited to mutation only (Real et al., 2017; Fernando et al., 2017; Liu et al., 2018; Elsken et al., 2019; Real et al., 2019; So et al., 2021; Co-Reyes et al., 2021; Gao et al., 2022), which has also been used extensively in simple hill-climbing/local search methods (White et al., 2021a;b).

The main obstacle in applying crossover to NAS is the *permutation problem* (Radcliffe, 1992; 1993), also known as the *competing conventions problem* (Montana & Davis, 1989; Schaffer et al., 1992). This problem is due to isomorphisms in graph space, i.e., functionally identical architectures are mapped to different encodings/representations, making crossover operations disruptive. A number of possible solutions to this problem have been proposed in the neuroevolution community (Thierens, 1996; Stanley & Miikkulainen, 2002; Dragoni et al., 2014; Mahmood et al., 2007; Wang et al., 2018; Uriot & Izzo, 2020). However, they either only work on fixed or constrained network topologies, or are limited to one particular algorithm or search space; none of them generalize to arbitrary graphs or architectures such as those that might arise from NAS. Moreover, prior work has focused only on empirical verification without a theoretical analysis of potential solutions. Theoretical understanding of search efficiency of mutation, crossover, and RL is still lacking in black-box NAS.

[1]Cognizant AI Labs, San Francisco, USA [2]The University of Texas at Austin, Austin, USA. Correspondence to: Xin Qiu <qiuxin.nju@gmail.com>, Risto Miikkulainen <risto@cognizant.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

To meet the above challenges, this paper first proposes a new crossover operator based on shortest edit path (SEP) in the original graph space. The SEP crossover does not impose any constraints on other algorithmic components or application scope, thereby forming a simple and generalizable solution to the permutation problem. Second, a theory is derived for analyzing mutation, standard crossover, RL, and the proposed SEP crossover in the NAS domain. The SEP crossover is shown to have the best expected improvement in terms of graph edit distance (GED) between the found architecture and the global optimum. Third, empirical results on SOTA NAS benchmarks further verify the theoretical analysis, demonstrating that the SEP approach is effective. It thus allows taking full advantage of population-based search, and serves as a theoretical foundation for further research on methods for NAS and similar problems. All source codes for reproducing the experimental results are provided at: (https://github.com/cognizant-ai-labs/sepx-paper).

2. Related Work

NAS NAS approaches can generally be categorized into two groups: one-shot methods and black-box methods (Mehta et al., 2022). In one-shot approaches (Liu et al., 2019; Dong & Yang, 2019; Chen et al., 2021), a supernet is trained to represent the entire search space. The overall training cost is reduced significantly; however, these approaches can only be run on small cell-based search spaces with a complete graph (Mehta et al., 2022; Zela et al., 2020) and the search objectives must be differentiable. In contrast, although computationally more expensive, black-box methods have no restrictions on the search space or objectives, making them a more general solution to NAS. Thus, this paper will focus on black-box NAS.

Black-box NAS Black-box NAS methods, also called zeroth-order methods, iteratively generate architectures for evaluation, and then use the outcome to update the search strategy. There are four main types of search strategies in black-box NAS approaches: random search (Li & Talwalkar, 2020; Yu et al., 2020), RL (Zoph & Le, 2017; Zoph et al., 2018), evolutionary search (Real et al., 2017; 2019), and local search (White et al., 2021a;b). Local search, whether used alone or together with neural predictors (e.g., Bayesian models), is based on operations that are essentially the same as mutation (although different terminology may be used) (White et al., 2021a;b). They can therefore be seen as equivalent to mutation-only evolutionary search with a population size of one. The one search strategy that is significantly different from evolutionary methods is RL, and will thus be included in the theoretical analysis in this paper. The theory developed in this paper thus covers most of the search strategies in Black-box NAS.

RL-based black-box NAS RL-based methods work by iteratively sampling architectures using a RL agent, then collecting the prediction accuracies as the reward for updating the policy. Zoph & Le (2017) successfully generated well-performing convolutional networks and recurrent cells using a specially designed recurrent neural network as the agent. Zoph et al. (2018) further showed that the approach finds architectures that transfer well between different datasets. In a recent empirical study (Ying et al., 2019), a simple RL agent based on multinomial probability distributions was found to perform significantly better on NAS-bench-101 than previous RL-based NAS methods. This RL controller is analyzed in this paper as well.

Evolutionary Black-box NAS Evolutionary NAS methods work by improving a population of architectures over time (Liu et al., 2021). To generate new offspring architectures, two operators can be used: a random edge/node mutation applied to an existing architecture, and crossover to recombine two existing architectures. Architectures that do not perform well are removed periodically from the population, and the best-performing architecture returned in the end. While crossover is a powerful operator, most existing methods rely on mutation only because of the permutation problem. It is this problem that this work aims to solve, in order to take full advantage of the evolutionary approch in NAS.

The permutation problem and existing solutions The permutation problem has been discussed in the Neuroevolution community for many years. One simple but common solution is simply to get rid of crossover completely during evolution (Angeline et al., 1994; Yao & Liu, 1998). Indeed, almost all newly developed evolutionary NAS methods avoid using a crossover operator (Real et al., 2017; Fernando et al., 2017; Liu et al., 2018; Elsken et al., 2019; Real et al., 2019; So et al., 2021; Co-Reyes et al., 2021; Gao et al., 2022). For instance, Real et al. (2017) reported that crossover operators were included in their initial experiments, but no performance improvement was observed, and therefore only mutation was deployed in their final AmoebaNet algorithm.

A number of principled solutions have been proposed to overcome the permutation problem as well. Many of them require that the network topologies are fixed. For instance, Thierens (1996) proposed a non-redundant encoding for matching neurons during crossover, Uriot & Izzo (2020) developed a safe crossover through a neural alignment mechanism, and Gangwani & Peng (2018) used genetic distillation to improve crossover. Further, Dragoni et al. (2014) proposed a generalization where the population can include different topologies, but only parents with a similar topology can be crossed over.

Other solutions have been developed for special cases, making them non-applicable to arbitrary architectures. For instance, the unit-alignment method (Sun et al., 2020) utilizes a special encoding that is only for CNN-based architectures. A graph matching recombination operator (Mahmood et al., 2007) only applies to parents with very different qualities. It mimics the behaviors of mutating the weaker parent towards the stronger parent, so the offspring does not differ from parents greatly. A modular inheritable crossover (He et al., 2021) is developed for a specific cell-based structure, and the default order of the nodes is preserved when performing crossover, without any node matching or reordering. As a result, the permutation problem still remains. The historical markings in NEAT-based algorithms (Stanley & Miikkulainen, 2002; Miikkulainen et al., 2019) are intended to be used together with other mechanisms in NEAT, and cannot be directly applied to any given architectures.

In contrast to these existing solutions, the proposed SEP crossover does not have any constraints on the encoding or other algorithmic components, and can be directly applied to any arbitrary architectures.

3. The Shortest Edit Path Crossover

In this section, the permutation problem is first described and a solution to it proposed in the form of Shortest Edit Path Crossover.

Given two neural architectures as parents, a crossover operator generates an offspring architecture by recombining the two parents. The crossover design consists of the encoding (i.e. genotype) and the recombination strategy, with the goal of properly integrating the information in both parents. The permutation problem arises when the same architecture (i.e. phenotype) can have multiple distinct genotypes. As a result, crossover on these genotypes has a disruptive effect on the information encoded in the parents, leading to damaged offspring (Stanley & Miikkulainen, 2002).

In order to propose a solution, let us first define a representation of the neural network architecture and a distance metric between two architectures. A neural architecture is a computation graph that can always be represented by an attributed directed graph, defined as:

Definition 3.1 (Directed graph)**.** A directed graph $\mathcal{G}$ consists of a set of vertices $\mathbb{V} = \{v_i | i = 1, 2, \ldots, n\}$, where n is the number of vertices and each v_i denotes a vertex (node), and a set of directed edges $\mathbb{E} = \{e_{i,j} | i, j \in 1, 2, \ldots, n\}$, where $e_{i,j}$ denotes a directed edge from v_i to v_j. The order of a directed graph $\mathcal{G}$ equals the number of its vertices, represented by $|\mathcal{G}|$. For an attributed directed graph, a function γ_v assigns an attribute (e.g., an integer) to each vertex, and a funtion γ_e assigns an attribute to each edge.

In the context of NAS, each vertex with an attribute denotes an operation in a neural architecture, and the directed edges denote data flows. The similarity between two architectures can then be measured by the graph edit distance (GED) between their corresponding graphs, defined as:

Definition 3.2 (Graph edit distance)**.** A graph edit operation is defined as a function $\delta : \mathcal{G} \to \mathcal{G}'$ that applies an elementary graph edit to transform $\mathcal{G}$ to $\mathcal{G}'$. In standard neural architecture search, the set of elementary graph edits typically includes vertex deletion/insertion, edge deletion/insertion, and vertex attribute substitution. An edit path is defined as a sequence of graph edit operations $\overline{\delta} = \delta_1, \delta_2, \ldots, \delta_d$, where d is the length of the edit path. Application of $\overline{\delta}$ to a graph is equivalent to applying each edit sequentially: $\overline{\delta}(\mathcal{G}) = \delta_d \circ \ldots \circ \delta_2 \circ \delta_1(\mathcal{G})$. Graph edit distance between two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is then defined as $\mathrm{GED}(\mathcal{G}_1, \mathcal{G}_2) = \min_{\overline{\delta} \in \Delta(\mathcal{G}_1, \mathcal{G}_2)} \sum_{i=1}^{d} c(\delta_i)$, where $\Delta(\mathcal{G}_1, \mathcal{G}_2)$ denotes the set of all edit paths that transform $\mathcal{G}_1$ to an isomorphism of $\mathcal{G}_2$ (including $\mathcal{G}_2$ itself), $\overline{\delta} = \delta_1, \delta_2, \ldots, \delta_d$, and $c(\delta_i)$ is the cost of edit δ_i. In this work, all types of edit operations are defined to have the same cost of 1. As a result, the edit path that minimizes the total edit cost, $\overline{\delta}^*_{\mathcal{G}_1,\mathcal{G}_2}$, equals the shortest edit path between $\mathcal{G}_1$ and $\mathcal{G}_2$. Thus, $\mathrm{GED}(\mathcal{G}_1, \mathcal{G}_2) = d^*_{\mathcal{G}_1,\mathcal{G}_2}$, where $d^*_{\mathcal{G}_1,\mathcal{G}_2}$ is the length of this shortest edit path. Note that $\overline{\delta}^*_{\mathcal{G}_1,\mathcal{G}_2}$ may not be unique, and thus there may exist multiple shortest edit paths that have the same length.

The proposed SEP crossover is then defined as

Definition 3.3 (Shortest edit path (SEP) crossover)**.** Given two attributed directed graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, suppose $\overline{\delta}^*_{\mathcal{G}_1,\mathcal{G}_2} = \delta^*_1, \delta^*_2, \ldots, \delta^*_{d^*_{\mathcal{G}_1,\mathcal{G}_2}}$. SEP crossover generates an offspring graph $\mathcal{G}_{\text{new}}$ by

$$\begin{aligned}\mathcal{G}_{\text{new}} =& \delta^*_{\pi_r(\lceil \frac{d^*_{\mathcal{G}_1,\mathcal{G}_2}}{2} \rceil)} \circ \delta^*_{\pi_r(\lceil \frac{d^*_{\mathcal{G}_1,\mathcal{G}_2}}{2} \rceil - 1)} \circ \delta^*_{\pi_r(\lceil \frac{d^*_{\mathcal{G}_1,\mathcal{G}_2}}{2} \rceil - 2)} \\ & \circ \ldots \circ \delta^*_{\pi_r(2)} \circ \delta^*_{\pi_r(1)}(\mathcal{G}_1),\end{aligned}$$

where π_r is a random permutation of the $d^*_{\mathcal{G}_1,\mathcal{G}_2}$ indices: $\pi_r : 1, 2, \ldots, d^*_{\mathcal{G}_1,\mathcal{G}_2} \to \pi(1), \pi(2), \ldots, \pi(d^*_{\mathcal{G}_1,\mathcal{G}_2})$, and $\lceil \cdot \rceil$ denotes the ceiling function. In other words, the SEP crossover shuffles the edits randomly in the SEP between parents, then selects half of them randomly, and applies them to one of the parents to obtain the offspring.

This operator is motivated by a common observation in the literature (e.g. Ying et al., 2019; White et al., 2021a; Mehta et al., 2022) that the differences in predictive performance between two architectures are positively correlated with their GEDs. This observation suggests that the edits in the SEP encode fundamental differences between two architectures that matter to predictive performance. An offspring that lies in the middle of this SEP can explore the search regions where the parents have fundamental discrepancies. At the same time, the offspring can automatically preserve those common substructures between parents, avoiding unnecessary disruptive behaviors, and thus avoiding the permutation problem. A visual demo showing how the SEP crossover resolves the permutation problem is provided in Appendix A.3.

4. Theoretical Analysis

In this section, the SEP crossover, standard crossover, mutation, and RL approaches to NAS will be analyzed theoretically, showing that the SEP crossover has an advantage in improving the expected quality of generated graphs. The fundamental concepts are defined first in Section 4.1, leading to new interpretations of graph edit distance, crossover and mutation based on *attributed adjacency matrices*. Feasibility assumptions are then declared, and theorems derived for expected improvement for SEP, standard crossover, and mutation. Section 4.2 focuses on RL: It interprets RL in terms of the same fundamental concepts, defines two extreme cases whose combinations span the possible states of the RL process, and derives theorems for expected improvement for both. Section 4.3 then brings these theorems together, showing that the SEP crossover results in more improvement than the other methods in common NAS setups. Section 4.4 further verifies the robustness of the SEP crossover under inaccurate GED calculations. All proofs and lemmas are included in Appendix A.1. For clarify, a full list of mathematical symbols is provided in Appendix A.2.

4.1. Expected Improvement with Crossover and Mutation

First, let us define the basic concepts:

Definition 4.1 (Attributed adjacency matrix)**.** An attributed adjacency matrix (AA-matrix) $\boldsymbol{A}_{\mathcal{G}}$ is a representation of an attributed directed graph. It is a $n \times n$ matrix, where n is the number of vertices in $\mathcal{G}$. The entry in ith row and jth column is represented by $A^{\mathcal{G}}_{i,j}$. $A^{\mathcal{G}}_{i,j} = 0$ if there is no edge from v_i to v_j, and $A^{\mathcal{G}}_{i,j} = \gamma_e(e_{i,j})$ if there exists an edge from v_i to v_j, for $i, j \in 1, 2, \ldots, n$ and $i \neq j$. $A^{\mathcal{G}}_{i,i} = \gamma_v(v_i)$, for $i \in 1, 2, \ldots, n$.

Definition 4.2 (Permutation matrix)**.** Given a permutation π of n elements: $\pi : 1, 2, \ldots, n \to \pi(1), \pi(2), \ldots, \pi(n)$, a permutation matrix $\boldsymbol{P}_\pi$ can be constructed by permuting the columns or rows of an $n \times n$ identity matrix $\boldsymbol{I}_n$ according to π. In this work, a column permutation of $\boldsymbol{I}_n$ is performed to obtain $\boldsymbol{P}_\pi$, The entry in ith row and jth column is represented by $P^\pi_{i,j}$, and $P^\pi_{i,j} = 1$ if $j = \pi(i)$, and $P^\pi_{i,j} = 0$ otherwise.

Definition 4.3 (Null vertex)**.** A null vertex has no connections to other existing vertices in a graph. It is assigned with a special "null" attribute, which means that it does not have any impact on the original graph. Null vertices are added to an existing graph only for convenience of theoretical analysis, and they do not affect the calculation of GEDs.

Based on the above definitions, GED, crossover and mutation can be interpreted from the AA-matrix perspective:

Definition 4.4 (AA-matrix-based interpretation of GED)**.** Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ can both be extended to have the same order $n = \max(|\mathcal{G}_1|, |\mathcal{G}_2|)$ by adding null vertices. The extended $\mathcal{G}_1$ and $\mathcal{G}_2$ are denoted as $\hat{\mathcal{G}}_1$ and $\hat{\mathcal{G}}_2$. Calculating the GED between $\mathcal{G}_1$ and $\mathcal{G}_2$ can then be defined as

$$\mathrm{GED}(\mathcal{G}_1, \mathcal{G}_2) = \min_{\pi \in S_n} d(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_\pi \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}_\pi^\top),$$

where $d(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{A_{i,j} \neq B_{i,j}}$, $m \times n$ is the order of both $\boldsymbol{A}$ and $\boldsymbol{B}$, $\mathbf{1}_{\text{condition}}$ is 1 if the condition is true, 0 otherwise (i.e., $d(\boldsymbol{A}, \boldsymbol{B})$ counts the number of different entries between two matrices with same shape), S_n denotes the set of all permutations of $\{1, 2, 3, \ldots, n\}$. The permutation that minimizes $d(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_\pi \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}_\pi^\top)$ is denoted as $\pi^*_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2}$, and the permuted AA-matrix of $\hat{\mathcal{G}}_2$ is denoted as $\boldsymbol{A}_{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1} = \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2}} \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2}}$.

Remark 4.5. In the context of standard neural architecture search, assume $\gamma_e(\cdot)$ always assigns 1 to any existing edge, and $\gamma_v(\cdot)$ assigns 0 to "null" vertex and positive integers for other types of vertex attributes (each type of attribute has its own unique integer). Then the differences between $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}$ correspond to the shortest edit path that transforms $\mathcal{G}_1$ to $\mathcal{G}_2$ in the following way: $\delta :=$ (1) add a vertex with attribute $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,i}$, if $A^{\hat{\mathcal{G}}_1}_{i,i} = 0$ and $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,i} > 0$; (2) delete vertex v_i from $\mathcal{G}_1$, if $A^{\hat{\mathcal{G}}_1}_{i,i} > 0$ and $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,i} = 0$; (3) change attribute of vertex v_i to $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,i}$, if $A^{\hat{\mathcal{G}}_1}_{i,i} > 0$ and $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,i} > 0$; (4) add an edge from v_i to v_j, if $A^{\hat{\mathcal{G}}_1}_{i,j} = 0$ and $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,j} = 1, i \neq j$; (5) delete the edge from v_i to v_j, if $A^{\hat{\mathcal{G}}_1}_{i,j} = 1$ and $A^{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}_{i,j} = 0, i \neq j$. Note that when adding an edge, the origin v_i and/or destination v_j may be newly added vertices.

Definition 4.6 (AA-matrix-based interpretation of crossover)**.** Assume two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are extended to have the same order by adding null vertices, resulting $\hat{\mathcal{G}}_1$ and $\hat{\mathcal{G}}_2$. A crossover between $\mathcal{G}_1$ and $\mathcal{G}_2$ is defined as the process of generating an offspring graph $\mathcal{G}_{\text{new}}$ by recombining $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$: $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_\pi \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}_\pi^\top)$, where function $r(\boldsymbol{A}, \boldsymbol{B})$ returns a matrix that inherits each entry from $\boldsymbol{A}$ or $\boldsymbol{B}$ with probability 0.5. That is, if $\boldsymbol{C} = r(\boldsymbol{A}, \boldsymbol{B})$, then $p(C_{i,j} = A_{i,j}) = p(C_{i,j} = B_{i,j}) = 0.5$ for any valid i, j. $\boldsymbol{P}_\pi$ is a permutation matrix based on permutation π, which is decided by the specific crossover operator utilized. For the SEP crossover, $\pi = \pi^*_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2}$, which minimizes the GED between $\mathcal{G}_1$ and $\mathcal{G}_2$. For the standard crossover, since the vertices may be in any order in the original AA-matrix representation and there is no particular vertex/edge matching mechanisms during crossover, a purely random permutation π_{rand} is used to represent this randomness. The result, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}$, is the AA-matrix of the generated new graph with null vertices. By removing all null vertices from $\hat{\mathcal{G}}_{\text{new}}$, the offspring graph $\mathcal{G}_{\text{new}}$ is obtained.

Definition 4.7 (AA-matrix-based interpretation of mutation)**.** Given a graph $\mathcal{G}_1$, a mutation operation is defined as the process of generating an offspring graph $\mathcal{G}_{\text{new}}$ by mutating $\mathcal{G}_1$. In standard NAS, allowed mutations to $\mathcal{G}_1$ include vertex deletion/insertion, edge deletion/insertion, and vertex attribute substitution. In the AA-matrix representation, a mutation operation is then defined as $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}} = m(\boldsymbol{A}_{\hat{\mathcal{G}}_1})$, where function $m(\boldsymbol{A})$ alters each element of $\boldsymbol{A}$ with an equal probability p_m. and p_m is usually selected so that on average one element is altered during each mutation operation. The $\hat{\mathcal{G}}_1$ is the extended graph of $\mathcal{G}_1$ with null vertices, so that node additions can be performed in $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ (by changing a null vertex to a vertex with a valid attribute). An element $A^{\hat{\mathcal{G}}_1}_{i,j}$ can be altered in order to randomly resample an allowed value that is different from the original $A^{\hat{\mathcal{G}}_1}_{i,j}$. The result, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}$, is the AA-matrix of the generated new graph with null vertices. By removing all null vertices from $\hat{\mathcal{G}}_{\text{new}}$, the mutated offspring graph $\mathcal{G}_{\text{new}}$ is obtained.

Next, in order to define a performance metric for comparing different crossover and mutation operators, a realistic assumption needs to be made about the search space:

Locality in NAS search spaces means that close architectures (in terms of GED) tend to have similar performance. Random-walk autocorrelation (RWA; Weinberger, 1990) is a commonly used metric to measure such locality. Strong autocorrelation of prediction accuracies of architectures during a random walk, in which each move is a graph edit operation, has been consistently observed in many existing NAS benchmarks or studies (Ying et al., 2019; White et al., 2021a; Mehta et al., 2022). This observation leads to the following assumption:

Assumption 4.8 (Positive correlation between GED and fitness/reward difference)**.** If $\text{GED}(\mathcal{G}_i, \mathcal{G}_j) < \text{GED}(\mathcal{G}_i, \mathcal{G}_k)$, then $\mathbb{E}(|f(\mathcal{G}_i) - f(\mathcal{G}_j)|) < \mathbb{E}(|f(\mathcal{G}_i) - f(\mathcal{G}_k)|)$, where $f(\mathcal{G})$ returns the fitness/reward of $\mathcal{G}$, i.e., the prediction accuracy.

Suppose $\mathcal{G}_{\text{opt}}$ is the global optimal graph (i.e. the target of the evolutionary search), $\mathcal{G}_1$ and $\mathcal{G}_2$ are the two parents to undergo crossover or mutation, and $\mathcal{G}_{\text{new}}$ is the generated offspring. For convenience of theoretical analysis, $\mathcal{G}_{\text{opt}}$, $\mathcal{G}_1$, and $\mathcal{G}_2$ are extended to have the same order $n = \max(|\mathcal{G}_{\text{opt}}|, |\mathcal{G}_1|, |\mathcal{G}_2|)$ by adding null vertices. The extended $\mathcal{G}_{\text{opt}}$ is denoted as $\hat{\mathcal{G}}_{\text{opt}}$, and $\hat{\mathcal{G}}_1$, $\hat{\mathcal{G}}_2$ and $\hat{\mathcal{G}}_{\text{new}}$ have the same meaning as in Definitions 4.6 and 4.7.

Given assumption 4.8, a direct measurement of the progress of the entire search is $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$, and the ultimate goal is to minimize it so that a good solution can be generated. $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}}) = d^*_{\mathcal{G}_{\text{opt}},\mathcal{G}_{\text{new}}} = d(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}})$ can be decomposed to $d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}}) + d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}})$, where $d_v(\boldsymbol{A}, \boldsymbol{B}) = \sum_i \mathbf{1}_{A_{i,i} \neq B_{i,i}}$ counts only the number of different diagonal entries, i.e., the differences in vertex attributes, and $d_e(\boldsymbol{A}, \boldsymbol{B}) = \sum_i \sum_{j \neq i} \mathbf{1}_{A_{i,j} \neq B_{i,j}}$ counts the number of different non-diagonal entries, i.e., the differences in edges/connections, thereby measuring the topological similarity.

In order to derive a performance metric, let's consider two factors. First, $d_v(\cdot)$ only covers n elements, whereas $d_e(\cdot)$ covers $n \cdot (n-1)$ elements. We have $n \cdot (n-1) \gg n$ when n increases, so $d_e(\cdot)$ is a dominant factor in deciding $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$. Second, modeling of vertex attributes varies a lot across different NAS spaces, e.g., they have different numbers of usable attributes and different constraints on vertex attribute assignments. In contrast, $\gamma_e(\cdot) = 1$ can simply be used for all valid edges in most NAS spaces, leading to generality of any theoretical conclusions. These two factors suggest that $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}})$ is a representative quantitative metric when comparing different crossover and mutation operators theoretically. For simplicity, we will use $d^*_{e,\mathcal{G}_1,\mathcal{G}_2}$ to denote $d_e(\boldsymbol{A}_{\mathcal{G}_1}, \boldsymbol{A}_{\mathcal{G}_2 \to \mathcal{G}_1})$.

Accordingly, the main performance metric for crossover and mutation can now be defined as follows:

Definition 4.9 (Expected improvement of crossover and mutation)**.** This work focuses on the expected improvement in terms of topological similarity to the global optimal graph. More specifically, expected improvement refers to $\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1 \to \hat{\mathcal{G}}_{\text{opt}}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}}), 0))$, which compares offspring graph $\mathcal{G}_{\text{new}}$ with one parent graph $\mathcal{G}_1$ in terms of the expected edge/connection differences to $\mathcal{G}_{\text{opt}}$. The $\max(\cdot, 0)$ part takes into account the selection pressure in standard EAs; that is, only the offspring that is better than its parent can survive and become the next parent.

As the penultimate step, three lemmas are derived in Appendix A.1 to assist the proofs regarding expected improvement. According to Lemmas A.1 and A.2, any π' can be chosen to analyze the behaviors of SEP crossover, standard crossover, and mutation, without affecting the result of $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}} \to \hat{\mathcal{G}}_{\text{opt}}})$. Lemma A.3 further derives the lower bound for common parts in $\mathcal{G}_{\text{opt}}$, $\mathcal{G}_1$ and $\mathcal{G}_2$. Now, choose $\pi' = \pi_1 = \pi^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ and $\pi_2 = \pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}$ so that there are at least $n_s = \max(n^2 - d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$ common entries among $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{A}_{\hat{\mathcal{G}}_1 \to \hat{\mathcal{G}}_{\text{opt}}}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}'_1}$, where $\boldsymbol{A}_{\hat{\mathcal{G}}_1 \to \hat{\mathcal{G}}_{\text{opt}}} = \boldsymbol{A}_{\hat{\mathcal{G}}'_1}$. Regarding the remaining entries, the following assumption is made:

Assumption 4.10 (Uniform distribution of differences)**.** The entries that are different between $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}'_1}$ are assumed to be uniformly distributed on the positions other than those n_s common entries.

With these lemmas and assumption, the expected improvement of SEP crossover, standard crossover and mutation can be derived:

Theorem 4.11 (Expected improvement of SEP crossover)**.**

Following Assumption 4.10, let $n_{se} = \max(n \cdot (n-1) - d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$*. and suppose* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}'_1}, \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}} \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}})$*. Then we have*

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0))\\ &\geq \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} \cdot d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n \cdot (n-1) - n_{se}} - \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5), 0))\\ &= \mathrm{LBEI}_{\rm SEPX},\end{aligned}$$

where $\mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5)$ *denotes the number of successful trials after sampling from a binomial distribution with* $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ *trials and success probability of 0.5, and* $\mathrm{LBEI}_{\rm SEPX}$ *denotes the lower bound of expected improvement of the SEP crossover.*

Theorem 4.12 (Expected improvement of standard crossover)**.** *Suppose* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}'_1}, \boldsymbol{P}_{\pi_{\rm rand}} \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}^\top_{\pi_{\rm rand}})$*. Then we have*

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0))\\ &\geq \mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - \mathcal{B}(\frac{n_1^1 \cdot n_2^0 + n_1^0 \cdot n_2^1}{n \cdot (n-1)}, 0.5) -\\ &\frac{(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + n_1^1 - n_{\rm opt}^1) \cdot n_2^1 + (d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + n_1^0 - n_{\rm opt}^0) \cdot n_2^0}{2n \cdot (n-1)}\\ &, 0)) = \mathrm{LBEI}_{\rm STDX},\end{aligned}$$

where $n^1_{\rm opt}$*,* n^1_1 *and* n^1_2 *denote the number of ones in* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}$*,* $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ *and* $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$ *(excluding diagonal entries), respectively,* $n^0_{\rm opt}$*,* n^0_1 *and* n^0_2 *denote the number of zeros in* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}$*,* $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ *and* $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$ *(excluding diagonal entries), respectively, and* $\mathrm{LBEI}_{\rm STDX}$ *denotes the lower bound of expected improvement of the standard crossover.*

Theorem 4.13 (Expected improvement of mutation)**.** *Suppose* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}} = m(\boldsymbol{A}_{\hat{\mathcal{G}}'_1})$*. Then we have*

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0))\\ &\geq \mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - \mathcal{B}(n \cdot (n-1) - d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1}, p_m)\\ &- \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1}, 1 - p_m), 0)) = \mathrm{LBEI}_{\rm MUTA},\end{aligned}$$

where p_m *is the mutation rate usually chosen to be* $p_m = \frac{1}{n\cdot(n-1)}$*, and* $\mathrm{LBEI}_{\rm MUTA}$ *denotes the lower bound of expected improvement of mutation.*

4.2. Expected Improvement with RL

First, let us interpret the RL approach using concepts established in Section 4.1. The setup follows the implementation of Ying et al. (2019), which provides good performance in NAS-bench-101 dataset.

Definition 4.14 (AA-matrix-based interpretation of RL)**.** RL invokes an agent that generates architectures following a probability distribution $\boldsymbol{Q}_\theta$ defined in AA-matrix space. For $\boldsymbol{A}_\theta \sim \boldsymbol{Q}_\theta$, each entry $A^\theta_{i,j}$ is sampled from a separate categorical distribution defined by $Q^\theta_{i,j}$. The $\theta = \{z^k_{i,j} | k \in 0, 1, \cdots, k^{\max}_{i,j}, \text{for } i, j \in 1, 2, \cdots, n\}$ is the parameter set that contains the logits for defining the categorical distributions through softmax functions $p(A^\theta_{i,j} = k) = \frac{\mathrm{e}^{z^k_{i,j}}}{\Sigma_k \mathrm{e}^{z^k_{i,j}}}$, for $k \in 0, 1, \cdots, k^{\max}_{i,j}$, and $i, j \in 1, 2, \cdots, n$. The learning process of θ follows the standard REINFORCE rule (Williams, 1992). The resulting scaled policy gradient is calculated as $\mathbb{E}_{\boldsymbol{A}_\theta\sim\boldsymbol{Q}_\theta}(\Sigma_{i,j} \bigtriangledown_\theta \log p(A^\theta_{i,j}) \cdot (R - b))$, where R is the reward for the currently sampled architecture (usually the validation accuracy) and b is a baseline to reduce the variance of gradient estimate.

The expected improvement of a policy update can then be defined. It is based on Lemma A.4 in Appendix A.1 that defines $\boldsymbol{Q}^*_\theta$ as the optimal permutation of $\boldsymbol{Q}_\theta$ and establishes an upper bound of expected GED to optimal.

Definition 4.15 (Expected improvement of a policy update)**.** Suppose the RL policy parameters are updated from θ_t to θ_{t+1}, where t indicates the current time step. The expected improvement is defined as $\Sigma_{i,j} p(A^{\theta_t *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_{\theta_t} \sim \boldsymbol{Q}^*_{\theta_t}) - \Sigma_{i,j} p(A^{\theta_{t+1} *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_{\theta_{t+1}} \sim \boldsymbol{Q}^*_{\theta_{t+1}})$ for $i, j \in 1, 2, \cdots, n$ and $i \neq j$. That is, it is the change in the upper bound of expected GED to optimal after policy update, considering only the edge/connection differences (similar to that of crossover and mutation).

Next, expected improvement can be derived in two extreme cases:

Definition 4.16 (Unbiased RL agent and oracle RL agent)**.** Given a pre-defined value for $\Sigma_{i,j} p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta)$ ($i, j \in 1, 2, \cdots, n$ and $i \neq j$), an unbiased agent is one that has the same $p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta)$ value for any $i, j \in 1, 2, \cdots, n$ and $i \neq j$, and an oracle agent is one that has the maximum number of non-diagonal entries in $\boldsymbol{A}^*_\theta$ satisfying $p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) = 0$, while all the remaining non-diagonal entries have the same and positive value for $p(A^{\theta *}_{i,j} = A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta)$.

In practical NAS experiments, the RL agent is usually initially unbiased, and converges towards the oracle agent during learning. Therefore, it is possible to interpolate between these two cases to span the entire RL search process. Next, expected improvement in RL is derived for the two cases:

Theorem 4.17 (Expected improvement of unbiased agent and oracle agent)**.** *Suppose* $\Sigma_{i,j} p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) = b^*_{e,\theta}$*, and further suppose* $R - b = \alpha \cdot (\Sigma_{i,j} p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$ *for* $i, j \in 1, 2, \cdots, n$ *and* $i \neq j$*, where* α *is a positive scaling factor and* $\mathcal{G}_{\theta_t}$ *is a graph sampled at time step* t *to obtain the empirical approximation of the policy gradient. With all* $z^k_{i,j}$ *initialized to 0, the expected improvement after one policy update with* *learning rate η is no less than*

$$\mathrm{LBEI_{RLU}} = b^*_{e,\theta} - (n_w \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta} - n_w)(1-p_w)}} + (n(n-1) - n_w) \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta} - n_w)\cdot p_w}})$$

for unbiased agent, where $p_w = \frac{b^*_{e,\theta}}{n(n-1)}, n_w = \mathcal{B}(n(n-1), \frac{b^*_{e,\theta}}{n(n-1)})$*, and no less than*

$$\mathrm{LBEI_{RLO}} = b^*_{e,\theta} - (n_w \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta} - n_w)(1-p_w)}} + (\lfloor b^*_{e,\theta} \rfloor + 1 - n_w) \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta} - n_w)\cdot p_w}})$$

for oracle agent, where $p_w = \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1}, n_w = \mathcal{B}(\lfloor b^*_{e,\theta} \rfloor + 1, \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1})$*, and $\lfloor \cdot \rfloor$ is the floor function.*

4.3. Comparisons based on Theory

As Theorems 4.11–4.13 and 4.17 indicate, expected improvement with the different methods depends on several factors, making problem-agnostic comparisons in closed form infeasible. It is, however, possible to compare these theoretical constructs numerically in specific representative settings, such as the various NAS benchmark domains.

To this end, $\mathrm{LBEI_{SEPX}}$, $\mathrm{LBEI_{MUTA}}$, $\mathrm{LBEI_{STDX}}$, $\mathrm{LBEI_{RLU}}$ and $\mathrm{LBEI_{RLO}}$ were compared in NAS-bench-101 benchmark (Ying et al., 2019). A numerical comparison requires instantiating the methods with specific parameter values. The standard NAS-bench-101 setup was used for $n = 7$, $n^1_{\mathrm{opt}} = 9$, $n^1_1 = 9$, and $n^1_2 = 9$, and for $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$ and $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ different combinations within a reasonable range were evaluated (the validity of these ranges will be verified in Section 5.1). The expected improvement in each case was then estimated through a Monte Carlo simulation with 10^6 trials. For RL, $b^*_{e,\theta} \equiv d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$, and $\alpha \cdot \eta = 0.1$ was used because this value provides the best tradeoff between unbiased and oracle agents (Figure A.4).

Figure 1 shows the main results: $\mathrm{LBEI_{SEPX}}$ is larger than $\mathrm{LBEI_{MUTA}}$, $\mathrm{LBEI_{RLU}}$, and $\mathrm{LBEI_{RLO}}$ in almost all cases. In contrast, Figure A.3 and Figure A.6 in Appendix A.5 show that the standard crossover leads to worse LBEI compared to mutation and RL. This numerical analysis thus illustrates the theoretical advantage of SEP crossover compared to mutation, standard crossover, and RL. More comparisons, as well as another benchmark (NAS-bench-NLP; Klyuchnikov et al., 2022), are included in Appendix A.5, reinforcing these conclusions.

4.4. Effect of Errors during GED Calculation

Finding the shortest edit path between two graphs requires calculating the GED between them, which is a NP-hard problem if an exact optimal solution is desired. Several fast approximation methods exist for GED calculation (Riesen, 2016; Serratosa, 2015). They can be run in polynomial time, at the cost of slightly reduced accuracy of the returned GED. To verify that SEP crossover is robust against such a loss of accuracy, a theoretical analysis was conducted. First, a corollary was derived to quantify the resulting expected improvement of SEP crossover with errors in the GED calculation. Second, a numerical analysis based on this corollary was run under three different levels of error.

Following Theorem 4.11, an error in calculating GED between two architectures $\hat{\mathcal{G}}_1$ and $\hat{\mathcal{G}}_2$ can be expressed as

$$d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} = d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \cdot (1 + \epsilon),$$

where $\epsilon > 0$ is the error ratio and $d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ is the expectation of GED calculation result. Assuming the resulting GED is either $\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor$ or $\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1$ following a Bernoulli distribution, the following corollary can be obtained:

Corollary 4.18 (Effect of GED errors on $\mathrm{LBEI_{SEPX}}$)**.** *With error ratio ϵ in calculating $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$, $\mathrm{LBEI_{SEPX}}$ becomes*

$$\begin{aligned}\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}} = {} & (d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} - \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor) \\ & \cdot \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} \cdot (\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1)}{n \cdot (n-1) - \lfloor n^{\epsilon}_{se} \rfloor} - \mathcal{B}(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1, 0.5), 0)) \\ & + (\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1 - d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}) \\ & \cdot \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} \cdot \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor}{n \cdot (n-1) - \lceil n^{\epsilon}_{se} \rceil} - \mathcal{B}(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor, 0.5), 0)),\end{aligned}$$

where $n^{\epsilon}_{se} = \max(n \cdot (n-1) - d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} - d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$.

As in Section 4.3, Monte Carlo simulations with 10^6 trials each were performed to estimate the values of $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ under different error ratios ϵ. Figure A.7 in Appendix A.6 compares $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ with the LBEI values for other methods under error ratios $\epsilon = 0.1$, 0.2, and 0.3.

The conclusion is that the SEP crossover has a theoretical advantage in expected improvement compared to mutation, standard crossover, and RL even with a very high error ratio of 30% in the GED calculations. Thus, approximation methods can be used for GED if the computational cost of the SEP crossover needs to be reduced.

5. Empirical Verification

This section first verifies that the parameter values used in the numerical analysis indeed apply to real-world problems. It then demonstrates that the SEP crossover is effective in real NAS problems under both noise-free and noisy environments. Experiment setup is provided in Appendix A.4.

5.1. Applicability of the Theory

Figures 1 (and Figure A.2 in Appendix A.5) demonstrate the theoretical advantage of SEP crossover numerically. How-

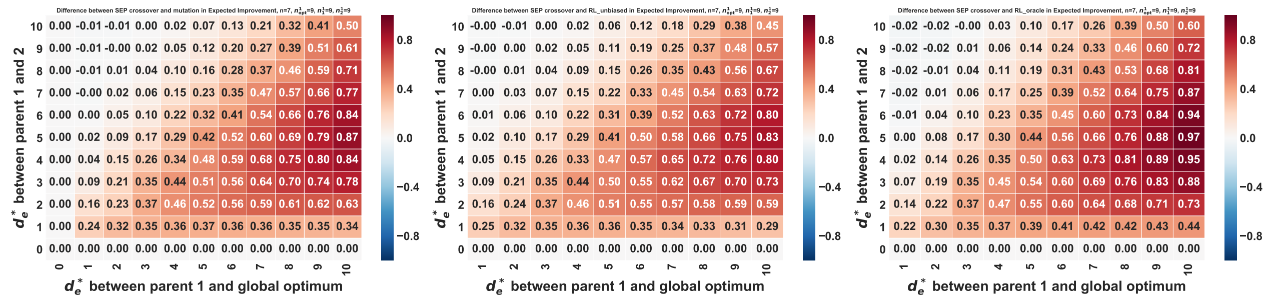


Figure 1. **Comparison of expected improvement between SEP crossover, mutation, and RL in NAS-bench-101.** (Left) Differences between $\mathrm{LBEI}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. (Middle) Differences between $\mathrm{LBEI}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{RLU}}$. (Right) Differences between $\mathrm{LBEI}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{RLO}}$. $\mathrm{LBEI}_{\mathrm{SEPX}}$ is larger (i.e. more red) than $\mathrm{LBEI}_{\mathrm{MUTA}}$, $\mathrm{LBEI}_{\mathrm{RLU}}$, and $\mathrm{LBEI}_{\mathrm{RLO}}$ almost everywhere. Thus, the SEP crossover has a theoretical advantage over mutation and RL.

ever, it is important to verify that the parameter values used in the Monte Carlo simulation indeed lie within the favorable regions in real NAS problems. In particular, the values used for $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$, $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$, n_1^1, and n_2^1 are critical to the expected improvement and need to be verified in standard benchmarks and with a standard NAS algorithm.

A NAS benchmark is said to be queryable if it directly returns the predictive performance of any architecture in the search space. While NAS-bench-101 has the most flexible graph search space among all queryable NAS benchmarks, NAS-bench-NLP (which is not queryable) has the largest search space among all existing NAS benchmarks (Mehta et al., 2022). They were both thus used to evaluate the parameter ranges. In order to evaluate the SEP crossover with a standard NAS algorithm, it was incorporated into the state-of-the-art Regularized Evolution method (RE; Real et al., 2019). RE employs only a mutation operator; SEP crossover was integrated into it by alternating crossover with mutation. To measure the parameter ranges, RE was run on both benchmarks, and the relative frequency distributions of the above parameters recorded (see Appendix A.7).

The results indeed show that the parameters lie within the range of the numerical analysis in Section 4.3. Moreover, they are within the subrange where the SEP crossover has a theoretical advantage (Figure 1). The results thus verify that the theory applies to NAS in real-world problems.

5.2. Performance in Noise-free Environments

The evaluation step in NAS, i.e. the training and testing of an architecture, can be very noisy (White et al., 2021a). To evaluate the search efficiency of the SEP crossover without the confounding effects of such noise, a noise-free evaluation function was first employed as the GED between the candidate architecture and the target architecture. The global optimum was selected as the target in NAS-bench-101, while the GRU (Cho et al., 2014) and LSTM (Hochreiter & Schmidhuber, 1997) models were used as targets in NAS-bench-NLP. Because the NAS-bench-NLP is not queryable, the global optimum is unknown. However, GRU and LSTM are two known top-performing models in this search space, and can therefore be used as a proxy for the global optimum. Since the RL method discussed in this work is only applicable to NAS-bench-101 space, it is not included in experiments on other benchmarks.

Plots (a) and (b) in Figure 2 compare the performance of random search, the original RE with mutation only, a modified RE augmented with standard crossover, RL (Ying et al., 2019), and a modified RE augmented with the SEP crossover. The SEP crossover performs significantly better than the other methods, demonstrating its value in practical NAS in noise-free environments. The experiments using LSTM as the target on NAS-bench-NLP is shown in Figure A.10 in Appendix A.8, and a similar advantage of the SEP crossover can be observed. Note that the standard crossover also performs better than mutation; the population is not very diverse in these experiments and thus most parent graphs are already well aligned for crossover.

5.3. Performance in Noisy Environments

In the third experiment, the robustness of the SEP crossover was evaluated by applying it to NAS problems with noisy evaluations. Noise arises from two sources: (1) the direct fitness/reward, e.g., the validation accuracy, used for search strategy is noisy; (2) The mapping between the final objective, e.g., the test accuracy, and direct fitness/reward is noisy. The validation accuracy in NAS-bench-101, which consists of random sampling of three real-world training trials was used as the direct fitness/reward. The average test accuracy in NAS-bench-101 was used as the final objectives.

Plots (c) and (d) of Figure 2 again compare the performance of random search, RE with mutation-only, RE augmented with standard crossover, RL (Ying et al., 2019), and RE augmented with the SEP crossover. The SEP crossover consistently outperforms other variants in this setup as well.

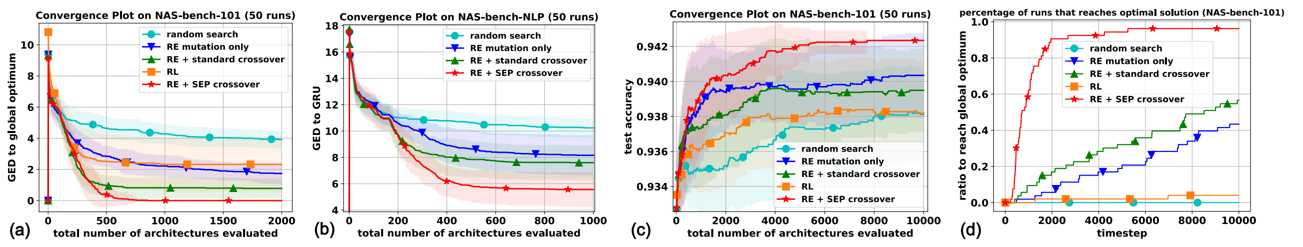


Figure 2. **Convergence in noise-free ((a) and (b)) and noisy environments ((c) and (d)).** (a) GED to global optimum in NAS-bench-101. (b) GED to GRU in NAS-bench-NLP. (c) Average testing accuracy in NAS-bench-101. (d) Percentage of runs that reach the global optimal architecture in NAS-bench-101. In all experiments, the SEP crossover performs consistently better than the other methods in both noise-free and noisy environments. The SEP crossover also reaches the global optimum significantly more efficiently than the other methods in NAS-bench-101. Together the experiments show that SEP consistently improves evolutionary NAS in practice.

Its performance is superior to the others in reaching the global optimal architecture (in terms of direct fitness/reward) in NAS-bench-101. Appendix A.9 and A.10 show more comparisons to two Bayesian optimization (BO) methods, namely BOHB (Falkner et al., 2018) and SMAC (Hutter et al., 2011), and crossover based on path encoding (White et al., 2021b). The SEP crossover significantly outperforms all these approaches. The empirical results thus demonstrate that the proposed SEP crossover is robust and effective in realistic noisy environments as well. Note that the standard crossover performs worse than mutation on NAS-bench-101 in both test accuracy and validation accuracy (see Figure A.10 in Appendix A.8). The population converges slower in these noisy environments, and the parent graphs are not as well aligned. Supplementary experiments using the surrogate predictions on NAS-bench-301 (Zela et al., 2022) are included in Figure A.10 of Appendix A.8; the SEP crossover shows consistently better search ability.

6. Discussion and Future Work

To the best of our knowledge, this paper presents the first theoretical analysis on evolutionary NAS. In addition to the SEP crossover operator itself, the definitions, assumptions, lemmas and theorems can form a foundation for future theory development. The work thus deepens our understanding of the behaviors of EAs and provides useful insights toward developing better evolutionary NAS methods.

Although the advantage of the SEP crossover over mutation is demonstrated both theoretically and empirically, it does not mean that mutation should be avoided. To make any crossover operators work, diversity in the population is important. Mutation is critical in introducing new architectures into the population, thereby increasing and maintaining diversity. Search that takes advantage of both a proper crossover and mutation, such as RE augmented with SEP, is likely to be the most effective.

The theoretical results show that the standard crossover is not as good as mutation in terms of expected improvement. This conclusion is consistent with observations in prior literature: Applying crossover without resolving the permutation problem may simply make search less efficient. On the other hand, the advantage of the SEP crossover demonstrates that crossover can indeed help evolutionary search in NAS problems if the permutation problem can be avoided.

The computational cost of the SEP crossover depends on the calculation of GED between two parent graphs. Appendix A.11 reports the computation time for exact GED calculation in the NAS experiments. This cost is still negligible compared to the training and evaluation of an architecture, which may take several GPU hours or even days. GED calculation is therefore not the computation bottleneck for existing NAS problems. Moreover, analysis in Section 4.4 suggests that the SEP crossover is robust to inaccurate GED calculation, and that if needed, approximate methods can be used to further reduce its computational costs.

Future directions include: (1) Developing a generative model that can output the offspring architecture for SEP crossover given two parents directly without a GED calculation; (2) applying the SEP crossover to more evolutionary NAS approaches and large-scale real-world NAS problems; and (3) applying the SEP crossover to other types of graph search/optimization problems, thus evaluating it as a general solution to optimization problems that involve graph search.

7. Conclusion

The SEP crossover is proposed as a solution the permutation problem in evolutionary NAS. Its advantage over standard crossover, mutation and RL was first shown theoretically, with a focus on the expected improvement of GED to global optimal. Empirical studies were then performed to verify the applicability of the theoretical results, and demonstrate the superior performance of the SEP crossover in both noise-free and noisy environments. The SEP crossover therefore allows taking full advantage of evolution in NAS, and potentially other similar design problems as well.

References

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., and Martineau, P. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1*, ICPRAM 2015, pp. 271278, Setubal, PRT, 2015. SCITEPRESS - Science and Technology Publications, Lda. ISBN 9789897580765. doi: 10.5220/0005209202710278. URL https://doi.org/10.5220/0005209202710278.

Angeline, P., Saunders, G., and Pollack, J. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994. doi: 10.1109/72.265960.

Chen, L.-C., Collins, M. D., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., and Shlens, J. Searching for efficient multi-scale architectures for dense image prediction. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 87138724, Red Hook, NY, USA, 2018. Curran Associates Inc.

Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C.-J. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=9FWas6YbmB3.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012.

Co-Reyes, J. D., Miao, Y., Peng, D., Real, E., Le, Q. V., Levine, S., Lee, H., and Faust, A. Evolving reinforcement learning algorithms. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0XXpJ4OtjW.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society. doi: 10.1109/CVPR.2019.00186.

Dragoni, M., Azzini, A., and Tettamanzi, A. G. B. Simba: A novel similarity-based crossover for neuro-evolution. *Neurocomput.*, 130:108122, apr 2014. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.03.042. URL https://doi.org/10.1016/j.neucom.2012.03.042.

Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ByME42AqK7.

Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/falkner18a.html.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017. URL http://arxiv.org/abs/1701.08734.

Gangwani, T. and Peng, J. Genetic policy optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByOnmlWC-.

Gao, J., Xu, H., Shi, H., Ren, X., Yu, P. L. H., Liang, X., Jiang, X., and Li, Z. Autobert-zero: Evolving bert backbone from scratch. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10): 10663–10671, Jun. 2022. doi: 10.1609/aaai.v36i10.21311. URL https://ojs.aaai.org/index.php/AAAI/article/view/21311.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J. (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

He, C., Tan, H., Huang, S., and Cheng, R. Efficient evolutionary neural architecture search by modular inheritable crossover. *Swarm and Evolutionary Computation*, 64:100894, 2021. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2021.100894. URL https://www.sciencedirect.com/science/article/pii/S2210650221000559.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In Coello, C. A. C. (ed.), *Learning and Intelligent Optimization*, pp. 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., and Burnaev, E. Nas-bench-nlp: Neural architecture search benchmark for natural language processing. *IEEE Access*, 10:45736–45747, 2022.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In Adams, R. P. and Gogate, V. (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 367–377. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/li20c.html.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G. G., and Tan, K. C. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021. doi: 10.1109/TNNLS.2021.3100554.

Mahmood, A., Sharmin, S., Barua, D., and Islam, M. M. Graph matching recombination for evolving neural networks. In Liu, D., Fei, S., Hou, Z., Zhang, H., and Sun, C. (eds.), *Advances in Neural Networks – ISNN 2007*, pp. 562–568, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72393-6.

Mehta, Y., White, C., Zela, A., Krishnakumar, A., Zabergja, G., Moradian, S., Safari, M., Yu, K., and Hutter, F. NAS-bench-suite: NAS evaluation is (now) surprisingly easy. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0DLwqQLmqV.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., and Hodjat, B. Chapter 15 - evolving deep neural networks. In Kozma, R., Alippi, C., Choe, Y., and Morabito, F. C. (eds.), *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pp. 293–312. Academic Press, 2019. ISBN 978-0-12-815480-9. doi: https://doi.org/10.1016/B978-0-12-815480-9.00015-3. URL https://www.sciencedirect.com/science/article/pii/B9780128154809000153.

Miikkulainen, R., Meyerson, E., Qiu, X., Sinha, U., Kumar, R., Hofmann, K., Yan, Y. M., Ye, M., Yang, J., Caiazza, D., and Brown, S. M. Evaluating medical aesthetics treatments through evolved age-estimation models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pp. 10091017, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3459378. URL https://doi.org/10.1145/3449639.3459378.

Montana, D. J. and Davis, L. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pp. 762767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

Radcliffe, N. J. *Genetic Neural Networks on MIMD Computers*. PhD thesis, GBR, 1992. UMI Order No. GAXD-94195.

Radcliffe, N. J. Genetic set recombination and its application to neural network topology optimisation. *Neural Computing & Applications*, 1(1):67–90, 1993. doi: 10.1007/BF01411376. URL https://doi.org/10.1007/BF01411376.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 29022911. JMLR.org, 2017.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, Jul. 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://ojs.aaai.org/index.php/AAAI/article/view/4405.

Riesen, K. *Structural Pattern Recognition with Graph Edit Distance: Approximation Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319272519.

Schaffer, J., Whitley, D., and Eshelman, L. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp. 1–37, 1992. doi: 10.1109/COGANN.1992.273950.

Serratosa, F. Computation of graph edit distance: Reasoning about optimality and speed-up. *Image and Vision Computing*, 40:38–48, 2015. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2015.06.005. URL https://www.sciencedirect.com/science/article/pii/S0262885615000736.

So, D., Mańke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. Searching for efficient transformers for language modeling. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 6010–6022. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/2f3c6a4cd8af177f6456e7e51a916ff3-Paper.pdf.

Stanley, K. O. and Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, 06 2002. ISSN 1063-6560. doi: 10.1162/106365602320169811. URL https://doi.org/10.1162/106365602320169811.

Sun, Y., Xue, B., Zhang, M., and Yen, G. G. Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 24(2): 394–407, 2020. doi: 10.1109/TEVC.2019.2916183.

Thierens, D. Non-redundant genetic coding of neural networks. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 571–575, 1996. doi: 10.1109/ICEC.1996.542662.

Uriot, T. and Izzo, D. Safe crossover of neural networks through neuron alignment. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, pp. 435443, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3390197. URL https://doi.org/10.1145/3377930.3390197.

Wang, B., Sun, Y., Xue, B., and Zhang, M. Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2018. doi: 10.1109/CEC.2018.8477735.

Weinberger, E. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990. doi: 10.1007/BF00202749. URL https://doi.org/10.1007/BF00202749.

White, C., Neiswanger, W., and Savani, Y. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a.

White, C., Nolen, S., and Savani, Y. Exploring the loss landscape in neural architecture search. In de Campos, C. and Maathuis, M. H. (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 654–664. PMLR, 27–30 Jul 2021b. URL https://proceedings.mlr.press/v161/white21a.html.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Yao, X. and Liu, Y. Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90, 1998. ISSN 0096-3003. doi: https://doi.org/10.1016/S0096-3003(97)10005-4. URL https://www.sciencedirect.com/science/article/pii/S0096300397100054.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. NAS-bench-101: Towards reproducible neural architecture search. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/ying19a.html.

Yu, K., Sciuto, C., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020.

Zela, A., Siems, J., and Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020.

Zela, A., Siems, J. N., Zimmer, L., Lukasik, J., Keuper, M., and Hutter, F. Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OnpFa95RVqs.

Zoph, B. and Le, Q. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. Learning transferable architectures for scalable image recognition. pp. 8697–8710, 06 2018. doi: 10.1109/CVPR.2018.00907.

A. Appendix

A.1. Lemmas and Theorems with Proof Details

Lemma A.1 (Invariance of SEP and standard crossover to parent permutation)**.** *For any permutation π', suppose graph $\hat{\mathcal{G}}_1'$ has the corresponding AA-matrix $\boldsymbol{A}_{\hat{\mathcal{G}}_1'} = \boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi'}^\top$, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'} = r(\boldsymbol{A}_{\hat{\mathcal{G}}_1'}, \boldsymbol{P}_{\pi_a}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_a}^\top)$, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_{\pi_b}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_b}^\top)$, and $\mathcal{G}_{\text{new}}'$, $\mathcal{G}_{\text{new}}$ are the graphs after removing all null vertices from $\hat{\mathcal{G}}_{\text{new}}'$ and $\hat{\mathcal{G}}_{\text{new}}$, respectively. If $\pi_a = \pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}$, $\pi_b = \pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$, or $\pi_a = \pi_{\text{rand}}$, $\pi_b = \pi_{\text{rand}}$ (π_a and π_b are sampled independently), then $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}}') = \text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$, $d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$, and $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$.*

Proof. Since a permutation of nodes (without changing their attributes and connections) simply generates an isomorphism of the original graph, $\hat{\mathcal{G}}_1'$ is an isomorphism of $\hat{\mathcal{G}}_1$. Calculations of the graph edit distance between two graphs are invariant to isomorphisms of either graph, so we have $\boldsymbol{A}_{\hat{\mathcal{G}}_2\to\hat{\mathcal{G}}_1'} = \boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_2\to\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi'}^\top \Rightarrow \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}}^\top = \boldsymbol{P}_{\pi'}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}^\top\boldsymbol{P}_{\pi'}^\top \Rightarrow \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}} = \boldsymbol{P}_{\pi'}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}$. Because $r(\boldsymbol{A}, \boldsymbol{B})$ is an element-wise operation that randomly chooses each entry either from $\boldsymbol{A}$ or $\boldsymbol{B}$, we have $r(\boldsymbol{PAP}^\top, \boldsymbol{PBP}^\top) = \boldsymbol{P}r(\boldsymbol{A}, \boldsymbol{B})\boldsymbol{P}^\top$ for any $\boldsymbol{P}$. Given $\pi_a = \pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}$, $\pi_b = \pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$, we have $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'} = r(\boldsymbol{A}_{\hat{\mathcal{G}}_1'}, \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}}^\top) = r(\boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi'}^\top, \boldsymbol{P}_{\pi'}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}^\top\boldsymbol{P}_{\pi'}^\top) = \boldsymbol{P}_{\pi'}r(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}^\top)\boldsymbol{P}_{\pi'}^\top = \boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}\boldsymbol{P}_{\pi'}^\top$, which shows $\mathcal{G}_{\text{new}}'$ is an isomorphism of $\mathcal{G}_{\text{new}}$. Therefore, calculating $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}}')$ is equivalent to calculating $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$, and $d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$, $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$.

For the situation where $\pi_a = \pi_{\text{rand}}$, $\pi_b = \pi_{\text{rand}}$ (π_a and π_b are sampled independently), since any permutation of a randomly generated sequence is equivalent to directly generating a random sequence, we have $\boldsymbol{P}_{\pi'}\boldsymbol{P}_{\pi_{\text{rand}}} = \boldsymbol{P}_{\pi_{\text{rand}}}$ for any π'. We can then derive the same conclusion as we did with $\boldsymbol{P}_{\pi'}\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}} = \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}}$. □

Lemma A.2 (Invariance of mutation to parent permutation)**.** *For any permutation π', suppose graph $\hat{\mathcal{G}}_1'$ has the corresponding AA-matrix $\boldsymbol{A}_{\hat{\mathcal{G}}_1'} = \boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi'}^\top$, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'} = m(\boldsymbol{A}_{\hat{\mathcal{G}}_1'})$, $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}} = m(\boldsymbol{A}_{\hat{\mathcal{G}}_1})$, and $\mathcal{G}_{\text{new}}'$, $\mathcal{G}_{\text{new}}$ are the graphs after removing all null vertices from $\hat{\mathcal{G}}_{\text{new}}'$ and $\hat{\mathcal{G}}_{\text{new}}$, respectively, then $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}}') = \text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$, $d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$, and $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$.*

Proof. Since $m(\boldsymbol{A})$ is an element-wise operation, we have $m(\boldsymbol{PAP}^\top) = \boldsymbol{P}m(\boldsymbol{A})\boldsymbol{P}^\top$ for any $\boldsymbol{P}$. We then have $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'} = m(\boldsymbol{A}_{\hat{\mathcal{G}}_1'}) = m(\boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi'}^\top) = \boldsymbol{P}_{\pi'}m(\boldsymbol{A}_{\hat{\mathcal{G}}_1})\boldsymbol{P}_{\pi'}^\top = \boldsymbol{P}_{\pi'}\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}\boldsymbol{P}_{\pi'}^\top$, so $\mathcal{G}_{\text{new}}'$ is an isomorphism of $\mathcal{G}_{\text{new}}$. Therefore, we have $GED(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}}') = GED(\mathcal{G}_{\text{opt}}, \mathcal{G}_{\text{new}})$, $d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$, and $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}'\to\hat{\mathcal{G}}_{\text{opt}}}) = d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})$. □

Lemma A.3 (Lower bound for common parts in $\mathcal{G}_{\text{opt}}$, $\mathcal{G}_1$ and $\mathcal{G}_2$)**.** *Suppose* $\text{GED}(\mathcal{G}_{\text{opt}}, \mathcal{G}_1) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}}) + d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}}) = d^*_{v,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} + d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} = d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$, $\text{GED}(\mathcal{G}_1, \mathcal{G}_2) = d_v(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{A}_{\hat{\mathcal{G}}_2\to\hat{\mathcal{G}}_1}) + d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{A}_{\hat{\mathcal{G}}_2\to\hat{\mathcal{G}}_1}) = d^*_{v,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} + d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} = d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$*, there exist π_1 and π_2 so that $s(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n^2 - d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$, $s_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n - d^*_{v,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{v,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$, $s_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n \cdot (n-1) - d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$, where $s(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \sum_i \sum_j \mathbf{1}_{A_{i,j}=B_{i,j}=C_{i,j}}$, $s_v(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \sum_i \mathbf{1}_{A_{i,i}=B_{i,i}=C_{i,i}}$, $s_e(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \sum_i \sum_{j\neq i} \mathbf{1}_{A_{i,j}=B_{i,j}=C_{i,j}}$.*

Proof. Let's choose $\pi_1 = \pi^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ and $\pi_2 = \pi^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2}$, where $\hat{\mathcal{G}}_1'$ has the corresponding AA-matrix $\boldsymbol{A}_{\hat{\mathcal{G}}_1'} = \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top$, then we will have $d(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top) = d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ and $d(\boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) = d^*_{\hat{\mathcal{G}}_1',\hat{\mathcal{G}}_2} = d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$. In the worst case that the $d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ entries and $d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ entries have the least overlaps in positions, the number of same entries in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top$ and $\boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top$ will be no less than $n^2 - d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (if it is not negative). As a result, we have $s(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n^2 - d^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$. When we decompose $s(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top)$ into $s_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top)$ and

$s_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top)$, we can easily obtain $s_v(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n - d^*_{v,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - d^*_{v,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$ and $s_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{P}_{\pi_1}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}_{\pi_1}^\top, \boldsymbol{P}_{\pi_2}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}_{\pi_2}^\top) >= \max(n\cdot(n-1) - d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$. □

Lemma A.4 (Upper bound of expected GED to optimal)**.** *Given an RL agent as defined in Definition 4.14, its expected GED to optimal is defined as* $\mathbb{E}_{\boldsymbol{A}_\theta\sim\boldsymbol{Q}_\theta}(\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}_\theta))$*, where* $\mathcal{G}_\theta$ *is the corresponding graph of* $\boldsymbol{A}_\theta$*. Suppose* $\mathcal{G}_{\rm opt}$ *is within the sample space of the RL agent, and* $\boldsymbol{Q}_\theta$ *is permuted to be* $\boldsymbol{Q}^*_\theta = \boldsymbol{P}_{\pi^*_{\mathcal{G}_{\rm opt},\theta}}\boldsymbol{Q}_\theta\boldsymbol{P}^\top_{\pi^*_{\mathcal{G}_{\rm opt},\theta}}$ *such that for any permutation* π', $\Sigma_{i,j}p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) \leq \Sigma_{i,j}p(A^{\theta\prime}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}'_\theta \sim \boldsymbol{Q}'_\theta)$*, where* $\boldsymbol{Q}'_\theta = \boldsymbol{P}_{\pi'}\boldsymbol{Q}_\theta\boldsymbol{P}^\top_{\pi'}$*, we have* $\mathbb{E}_{\boldsymbol{A}_\theta\sim\boldsymbol{Q}_\theta}(\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}_\theta)) \leq \Sigma_{i,j}p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta)$*, for* $i,j \in 1,2,\cdots,n$*.*

Proof. $\boldsymbol{Q}^*_\theta$ is one of the permutations of $\boldsymbol{Q}_\theta$ that minimizes the expected number of different entries between $\boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta$ and $\boldsymbol{A}_{\mathcal{G}_{\rm opt}}$, i.e., $\Sigma_{i,j}p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) = \mathbb{E}_{\boldsymbol{A}^*_\theta\sim\boldsymbol{Q}^*_\theta} d(\boldsymbol{A}^*_\theta, \boldsymbol{A}_{\mathcal{G}_{\rm opt}})$. Since for every sampled $\boldsymbol{A}^*_\theta$, we have $\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}^*_\theta) \leq d(\boldsymbol{A}^*_\theta, \boldsymbol{A}_{\mathcal{G}_{\rm opt}})$, which leads to $\mathbb{E}_{\boldsymbol{A}^*_\theta\sim\boldsymbol{Q}^*_\theta}(\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}^*_\theta)) \leq \mathbb{E}_{\boldsymbol{A}^*_\theta\sim\boldsymbol{Q}^*_\theta} d(\boldsymbol{A}^*_\theta, \boldsymbol{A}_{\mathcal{G}_{\rm opt}})$. Because $\boldsymbol{Q}^*_\theta$ is a permutation of $\boldsymbol{Q}_\theta$, we have $\mathbb{E}_{\boldsymbol{A}_\theta\sim\boldsymbol{Q}_\theta}(\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}_\theta)) = \mathbb{E}_{\boldsymbol{A}^*_\theta\sim\boldsymbol{Q}^*_\theta}(\mathrm{GED}(\mathcal{G}_{\rm opt}, \mathcal{G}^*_\theta)) \leq \mathbb{E}_{\boldsymbol{A}^*_\theta\sim\boldsymbol{Q}^*_\theta} d(\boldsymbol{A}^*_\theta, \boldsymbol{A}_{\mathcal{G}_{\rm opt}}) = \Sigma_{i,j}p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta)$, for $i,j \in 1,2,\cdots,n$. □

Theorem 4.11 (Expected improvement of SEP crossover)**.** *Following Assumption 4.10, let* $n_{se} = \max(n\cdot(n-1) - d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$*. and suppose* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}'_1}, \boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}})$*. Then we have*

$$\begin{aligned}
&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0)) \\
&\geq \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1}\cdot d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}} - \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5), 0)) = \mathrm{LBEI}_{\mathrm{SEPX}},
\end{aligned}$$

where $\mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5)$ *denotes a binomial distribution with* $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ *trials and success probability of 0.5, and* $\mathrm{LBEI}_{\mathrm{SEPX}}$ *denotes the lower bound of expected improvement of the SEP crossover.*

Proof. Following Assumption 4.10, since n_{se} elements are shared by $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}}$, the $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ different elements among them are uniformly distributed within the remaining $n\cdot(n-1)-n_{se}$ entries. As a result, the chance for any one of these $n\cdot(n-1)-n_{se}$ entries to have the same values in both parents equals $1 - \frac{d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}}$, then the number of entries in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ that is originally different from $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}$ and stay intact after crossover is $(1 - \frac{d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}})\cdot d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1}$. Since all the non-diagonal elements in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}'_1,\hat{\mathcal{G}}_2}}$ are either 0 or 1 (indicating whether there is an edge between two nodes), the number of remaining entries that one of the parents is correct while the other is incorrect equals $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$. Therefore, $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}}) = (1 - \frac{d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}})\cdot d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5)$. Considering the fact that $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}) \leq d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}})$, we have

$$\begin{aligned}
&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0)) \geq \mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}}), 0)) \\
&= \mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - ((1 - \frac{d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}})\cdot d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5)) \\
&= \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1}\cdot d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}{n\cdot(n-1)-n_{se}} - \mathcal{B}(d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0.5), 0)).
\end{aligned}$$

□

Theorem 4.12 (Expected improvement of standard crossover)**.** *Suppose* $\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}} = r(\boldsymbol{A}_{\hat{\mathcal{G}}'_1}, \boldsymbol{P}_{\pi_{\rm rand}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi_{\rm rand}})$*. Then we have*

$$\begin{aligned}
&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\rm opt}}) - d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\rm opt}}, \boldsymbol{A}_{\hat{\mathcal{G}}_{\rm new}\to\hat{\mathcal{G}}_{\rm opt}}), 0)) \\
&\geq \mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} - \frac{(d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + n^1_1 - n^1_{\rm opt})\cdot n^1_2 + (d^*_{e,\hat{\mathcal{G}}_{\rm opt},\hat{\mathcal{G}}_1} + n^0_1 - n^0_{\rm opt})\cdot n^0_2}{2n\cdot(n-1)} - \mathcal{B}(\frac{n^1_1\cdot n^0_2 + n^0_1\cdot n^1_2}{n\cdot(n-1)}, 0.5), 0)) \\
&= \mathrm{LBEI}_{\mathrm{STDX}},
\end{aligned}$$

where n^1_{opt}, n^1_1 and n^1_2 denote the number of ones in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$ (excluding diagonal entries), respectively, n^0_{opt}, n^0_1 and n^0_2 denote the number of zeros in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$ (excluding diagonal entries), respectively, and $\text{LBEI}_{\text{STDX}}$ *denotes the lower bound of expected improvement of the standard crossover.*

Proof. The resulting corresponding graph of $\boldsymbol{P}_{\pi_{\text{rand}}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi_{\text{rand}}}$ is equivalent to an isomorphism that randomly shuffles the order of vertices of $\hat{\mathcal{G}}_2$, therefore any non-diagonal entries in $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$, which represents the connection status between two vertices, has the same chance to be moved to any non-diagonal positions in $\boldsymbol{P}_{\pi_{\text{rand}}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi_{\text{rand}}}$ after the vertices shuffling. The number of different non-diagonal entries between $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{P}_{\pi_{\text{rand}}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi_{\text{rand}}}$ then equals to $n^1_1\cdot\frac{n^0_2}{n\cdot(n-1)}+n^0_1\cdot\frac{n^1_2}{n\cdot(n-1)}=\frac{n^1_1\cdot n^0_2+n^0_1\cdot n^1_2}{n\cdot(n-1)}$. The number of non-diagonal entries that are same in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{P}_{\pi_{\text{rand}}}\boldsymbol{A}_{\hat{\mathcal{G}}_2}\boldsymbol{P}^\top_{\pi_{\text{rand}}}$ but are different from $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$ equals $n^1_{\text{w}}\cdot\frac{n^1_2}{n\cdot(n-1)}+n^0_{\text{w}}\cdot\frac{n^0_2}{n\cdot(n-1)}$, where n^1_{w} and n^0_{w} denotes the number of 1s and 0s in the non-diagonal entries where $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$ are different (we treat these entries as "wrong" entries, so we use the subscript "w"), respectively. To calculate n^1_{w}, we need to consider two cases: (1) if $n^1_1\geq n^1_{\text{opt}}$, then n^1_{w} consists of two parts, namely $n^1_1-n^1_{\text{opt}}$, which represents the number of extra 1s in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ that $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$ can never match, and $\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_1-n^1_{\text{opt}})}{2}$, which is derived from the fact that in the remaining entries where $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$ have the same number of 1s, one misplace (compared to $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$) of 1 in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ also leads to one misplace of 0 in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ (otherwise the number of 1s will be unequal in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$), so exactly half of these $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_1-n^1_{\text{opt}})$ mismatched entries will be 1 in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$. After summing these two parts up, we obtain $n^1_1-n^1_{\text{opt}}+\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_1-n^1_{\text{opt}})}{2}=\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_{\text{opt}}-n^1_1)}{2}$. (2) if $n^1_1<n^1_{\text{opt}}$, we only need to consider the entries that excluding those extra 1s in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$ that cannot be matched by $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$, that is, half of the remaining $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_{\text{opt}}-n^1_1)$ mismatched entries. We then have $n^1_{\text{w}}=\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^1_{\text{opt}}-n^1_1)}{2}$, which also equals to the result of the first case. Similarly, we can get $n^0_{\text{w}}=\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-(n^0_{\text{opt}}-n^0_1)}{2}$.

Given the above intermediate results, we can obtain $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}})=n^1_{\text{w}}\cdot\frac{n^1_2}{n\cdot(n-1)}+n^0_{\text{w}}\cdot\frac{n^0_2}{n\cdot(n-1)}+\mathcal{B}(\frac{n^1_1\cdot n^0_2+n^0_1\cdot n^1_2}{n\cdot(n-1)},0.5)=\frac{(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}+n^1_1-n^1_{\text{opt}})\cdot n^1_2+(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}+n^0_1-n^0_{\text{opt}})\cdot n^0_2}{2n\cdot(n-1)}+\mathcal{B}(\frac{n^1_1\cdot n^0_2+n^0_1\cdot n^1_2}{n\cdot(n-1)},0.5)$. Since $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})\leq d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}})$, we have

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}})-d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}}),0))\geq\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}})-d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}),0))\\&=\mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-\frac{(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}+n^1_1-n^1_{\text{opt}})\cdot n^1_2+(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}+n^0_1-n^0_{\text{opt}})\cdot n^0_2}{2n\cdot(n-1)}-\mathcal{B}(\frac{n^1_1\cdot n^0_2+n^0_1\cdot n^1_2}{n\cdot(n-1)},0.5),0)).\end{aligned}$$

□

Theorem 4.13 (Expected improvement of mutation)**.** *Suppose $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}=m(\boldsymbol{A}_{\hat{\mathcal{G}}'_1})$. Then we have*

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}})-d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}}),0))\\&\geq\mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-\mathcal{B}(n\cdot(n-1)-d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},p_m)-\mathcal{B}(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},1-p_m),0))=\text{LBEI}_{\text{MUTA}},\end{aligned}$$

where p_m is the mutation rate usually chosen to be $p_m=\frac{1}{n\cdot(n-1)}$, and $\text{LBEI}_{\text{MUTA}}$ *denotes the lower bound of expected improvement of mutation.*

Proof. Since $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}=\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}}\boldsymbol{A}_{\hat{\mathcal{G}}_1}\boldsymbol{P}^\top_{\pi^*_{\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}}$, there are $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ non-diagonal elements in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ that are different from $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$. Because all the non-diagonal elements in $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$ are either 0 or 1, and $m(\boldsymbol{A}_{\hat{\mathcal{G}}'_1})$ has p_m probability to flip each non-diagonal element of $\boldsymbol{A}_{\hat{\mathcal{G}}'_1}$, we have $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}})=\mathcal{B}(n\cdot(n-1)-d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},p_m)+\mathcal{B}(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},1-p_m)$. Since $d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}})\leq d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}})$, we have

$$\begin{aligned}&\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}})-d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}\to\hat{\mathcal{G}}_{\text{opt}}}),0))\geq\mathbb{E}(\max(d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_1\to\hat{\mathcal{G}}_{\text{opt}}})-d_e(\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}},\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{new}}}),0))\\&=\mathbb{E}(\max(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}-\mathcal{B}(n\cdot(n-1)-d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},p_m)-\mathcal{B}(d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1},1-p_m),0)).\end{aligned}$$

□

Theorem 4.17 (Expected improvement of unbiased agent and oracle agent)**.** *Suppose* $\Sigma_{i,j} p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) = b^*_{e,\theta}$ *and assume* $R - b = \alpha \cdot (\Sigma_{i,j} p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$ *for* $i, j \in 1, 2, \cdots, n$ *and* $i \neq j$*, where* α *is a positive scaling factor and* $\mathcal{G}_{\theta_t}$ *is a graph sampled at time step t for obtaining the empirical approximation of policy gradient. With all* $z^k_{i,j}$ *initialized to 0, the expected improvement after one policy update with learning rate* η *is no less than*

$$\mathrm{LBEI}_{\rm RLU} = b^*_{e,\theta} - (n_w \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta} - n_w)(1-p_w)}} + (n(n-1) - n_w) \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta} - n_w)\cdot p_w}})$$

for unbiased agent, where $p_w = \frac{b^*_{e,\theta}}{n(n-1)}, n_w = \mathcal{B}(n(n-1), \frac{b^*_{e,\theta}}{n(n-1)})$*, and no less than*

$$\mathrm{LBEI}_{\rm RLO} = b^*_{e,\theta} - (n_w \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta} - n_w)(1-p_w)}} + (b^*_{e,\theta} + 1 - n_w) \cdot \frac{1}{1 + (\frac{1}{p_w} - 1) \cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta} - n_w)\cdot p_w}})$$

for oracle agent, where $p_w = \frac{b^*_{e,\theta}}{b^*_{e,\theta}+1}, n_w = \mathcal{B}(b^*_{e,\theta} + 1, \frac{b^*_{e,\theta}}{b^*_{e,\theta}+1})$.

Proof. Under the REINFORCE rule, the policy gradient based on one sample is $\Sigma_{i,j} \bigtriangledown_\theta \log p(A^\theta_{i,j}) \cdot (R - b)$ for $i, j \in 1, 2, \cdots, n$, and the constraint $i \neq j$ can be added to only consider edges/connections. Since only two values 0 and 1 are allowed for each entry that denotes an edge connection, they can be mapped to "correct" and "wrong" by comparing entries between $\boldsymbol{A}^*_\theta$ and $\boldsymbol{A}_{\mathcal{G}_{\rm opt}}$: "correct" means $A^{\theta*}_{i,j} = A^{\mathcal{G}_{\rm opt}}_{i,j}$ and "wrong" means $A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j}$. For an entry in $\boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta$, let p_c be the probability for it to be correct and p_w the probability for it to be wrong, and let z_c and z_w be the logits for generating p_c and p_w, respectively. Then

$$\begin{aligned}\frac{\partial \log p_c}{\partial z_c} =& \frac{\partial}{\partial z_c} \log(\frac{\mathrm{e}^{z_c}}{\mathrm{e}^{z_c} + \mathrm{e}^{z_w}}) = \frac{\partial}{\partial z_c}(z_c - \log(\mathrm{e}^{z_c} + \mathrm{e}^{z_w})) \\ =& 1 - \frac{1}{\mathrm{e}^{z_c} + \mathrm{e}^{z_w}} \cdot (\frac{\partial}{\partial z_c}(\mathrm{e}^{z_c} + \mathrm{e}^{z_w})) = 1 - \frac{\mathrm{e}^{z_c}}{\mathrm{e}^{z_c} + \mathrm{e}^{z_w}} = 1 - p_c.\end{aligned}$$

Similarly, $\frac{\partial \log p_c}{\partial z_w} = -p_w$, $\frac{\partial \log p_w}{\partial z_w} = 1 - p_w$ and $\frac{\partial \log p_w}{\partial z_c} = -p_c$. For the entries that sample correctly, the policy gradient for updating z_c and z_w is $\frac{\partial \log p_c}{\partial z_c} \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}}) = (1 - p_c) \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$ and $-p_w \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$, respectively. For the entries that sample wrong, the policy gradient for updating z_c and z_w is $\frac{\partial \log p_w}{\partial z_c} \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}}) = -p_c \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$ and $(1 - p_w) \cdot \alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})$, respectively. Since $p_c = 1 - p_w$, the policy gradients are always opposite but the same magnitude for z_c and z_w. Because all the $z^k_{i,j}$ are initialized to 0, thus $z_c = -z_w$ for each entry. Given $p_w = \frac{\mathrm{e}^{z_w}}{\mathrm{e}^{z_c} + \mathrm{e}^{z_w}}$ and $z_c = -z_w$, then $\mathrm{e}^{z_c} = \sqrt{\frac{1-p_w}{p_w}}$ and $\mathrm{e}^{z_w} = \sqrt{\frac{p_w}{1-p_w}}$. Therefore, for the entries that sample correctly, p_c is updated as

$$p'_c = \frac{\mathrm{e}^{z_c} \cdot \mathrm{e}^{(1-p_c)\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta}}{\mathrm{e}^{z_c} \cdot \mathrm{e}^{(1-p_c)\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta} + \mathrm{e}^{z_w} \cdot \mathrm{e}^{-p_w\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta}}.$$

For the entries that sample wrong, p_w is updated as

$$p'_w = \frac{\mathrm{e}^{z_w} \cdot \mathrm{e}^{(1-p_w)\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta}}{\mathrm{e}^{z_w} \cdot \mathrm{e}^{(1-p_w)\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta} + \mathrm{e}^{z_c} \cdot \mathrm{e}^{-p_c\cdot\alpha(b^*_{e,\theta} - d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}})\cdot\eta}}.$$

For the unbiased agent, in order to have $\Sigma_{i,j} p(A^{\theta*}_{i,j} \neq A^{\mathcal{G}_{\rm opt}}_{i,j} | \boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta) = b^*_{e,\theta}$, every entry should have the same $p_w = \frac{b^*_{e,\theta}}{n(n-1)}$. The number of different non-diagonal entries between $\boldsymbol{A}^*_\theta$ and $\boldsymbol{A}_{\mathcal{G}_{\rm opt}}$ is then $n_w = d^*_{e,\mathcal{G}_{\rm opt},\mathcal{G}_{\theta_t}} = \mathcal{B}(n(n-1), \frac{b^*_{e,\theta}}{n(n-1)})$. Since there are n_w entries sampled wrong and $n(n-1) - n_w$ sampled correctly, the expected number of different non-diagonal entries between $\boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta$ and $\boldsymbol{A}_{\mathcal{G}_{\rm opt}}$ after updating every p_c and p_w becomes $n_w \cdot p'_w + (n(n-1) - n_w) \cdot (1 - p'_c)$. Considering the possibility of further permuting the updated entries in $\boldsymbol{Q}^*_{\theta_t}$ to obtain $\boldsymbol{Q}^*_{\theta_{t+1}}$, and supposing that $\boldsymbol{Q}^*_{\theta_t} = \boldsymbol{Q}^*_\theta$, then $n_w \cdot p'_w + (n(n-1)-n_w)\cdot(1-p'_c) \geq \Sigma_{i,j} p(A^{\theta_{t+1}*}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta_{t+1}} \sim \boldsymbol{Q}^*_{\theta_{t+1}})$. As a result, the expected improvement is

$$\begin{aligned}&\Sigma_{i,j} p(A^{\theta_t *}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta_t} \sim \boldsymbol{Q}^*_{\theta_t}) - \Sigma_{i,j} p(A^{\theta_{t+1}*}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta_{t+1}} \sim \boldsymbol{Q}^*_{\theta_{t+1}})\\ &\geq b^*_{e,\theta} - (n_w \cdot p'_w + (n(n-1)-n_w)\cdot(1-p'_c))\\ &= b^*_{e,\theta} - (n_w \cdot \frac{1}{1+(\frac{1}{p_w}-1)\cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta}-n_w)(1-p_w)}} + (n(n-1)-n_w)\cdot \frac{1}{1+(\frac{1}{p_w}-1)\cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta}-n_w)\cdot p_w}}),\end{aligned}$$

where $p_w = \frac{b^*_{e,\theta}}{n(n-1)}, n_w = \mathcal{B}(n(n-1), \frac{b^*_{e,\theta}}{n(n-1)})$.

For the oracle agent, if $b^*_{e,\theta}$ is an integer and there are exactly $n(n-1) - b^*_{e,\theta}$ entries that have $p_c = 1.0$, the remaining $b^*_{e,\theta}$ entries can only have $p_c = 0$, due to the pre-condition that $\Sigma_{i,j} p(A^{\theta *}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta} \sim \boldsymbol{Q}^*_{\theta}) = b^*_{e,\theta}$. This setup results in a stuck agent that can no longer explore and update itself, and it does not satisfy Definition 4.16. Therefore, the maximum number of entries with $p_c = 1.0$ can only be $n(n-1) - b^*_{e,\theta} - 1$ when $b^*_{e,\theta}$ is an integer, and $n(n-1) - \lceil b^*_{e,\theta} \rceil$ otherwise. Since $n(n-1) - \lceil b^*_{e,\theta} \rceil = n(n-1) - \lfloor b^*_{e,\theta} \rfloor - 1$ always holds true and $n(n-1) - b^*_{e,\theta} - 1 = n(n-1) - \lfloor b^*_{e,\theta} \rfloor - 1$ is true when $b^*_{e,\theta}$ is an integer, $n(n-1) - \lfloor b^*_{e,\theta} \rfloor - 1$ can always be used to describe the maximum number of entries with $p_c = 1.0$. The remaining $\lfloor b^*_{e,\theta} \rfloor + 1$ entries would have $p_w = \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1}$, within which n_w entries are sampled wrong, and $\lfloor b^*_{e,\theta} \rfloor + 1 - n_w$ entries are sampled correctly, with $n_w = \mathcal{B}(\lfloor b^*_{e,\theta} \rfloor + 1, \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1})$. The expected number of different non-diagonal entries between $\boldsymbol{A}^*_\theta \sim \boldsymbol{Q}^*_\theta$ and $\boldsymbol{A}_{\mathcal{G}_{\text{opt}}}$ after updating every p_c and p_w thus becomes $n_w \cdot p'_w + (\lfloor b^*_{e,\theta} \rfloor + 1 - n_w)\cdot(1-p'_c)$. Similar to the analysis of the unbiased agent, we have

$$\begin{aligned}&\Sigma_{i,j} p(A^{\theta_t *}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta_t} \sim \boldsymbol{Q}^*_{\theta_t}) - \Sigma_{i,j} p(A^{\theta_{t+1}*}_{i,j} \neq A^{\mathcal{G}_{\text{opt}}}_{i,j} | \boldsymbol{A}^*_{\theta_{t+1}} \sim \boldsymbol{Q}^*_{\theta_{t+1}})\\ &\geq b^*_{e,\theta} - (n_w \cdot \frac{1}{1+(\frac{1}{p_w}-1)\cdot \mathrm{e}^{-2\alpha\eta(b^*_{e,\theta}-n_w)(1-p_w)}} + (\lfloor b^*_{e,\theta} \rfloor + 1 - n_w)\cdot \frac{1}{1+(\frac{1}{p_w}-1)\cdot \mathrm{e}^{2\alpha\eta(b^*_{e,\theta}-n_w)\cdot p_w}})\end{aligned}$$

for the oracle agent, where $p_w = \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1}, n_w = \mathcal{B}(\lfloor b^*_{e,\theta} \rfloor + 1, \frac{b^*_{e,\theta}}{\lfloor b^*_{e,\theta} \rfloor + 1})$. □

Corollary 4.18 (Effect of GED errors on $\text{LBEI}_{\text{SEPX}}$). *With error ratio ϵ in calculating $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$, $\text{LBEI}_{\text{SEPX}}$ becomes*

$$\begin{aligned}\text{LBEI}^\epsilon_{\text{SEPX}} = (d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} - \lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor)\cdot \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}\cdot(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1)}{n\cdot(n-1) - \lfloor n^\epsilon_{se} \rfloor} - \mathcal{B}(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1, 0.5), 0))\\ + (\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1 - d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2})\cdot \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}\cdot\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor}{n\cdot(n-1) - \lceil n^\epsilon_{se} \rceil} - \mathcal{B}(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor, 0.5), 0)),\end{aligned}$$

where $n^\epsilon_{se} = \max(n\cdot(n-1) - d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}, 0)$.

Proof. Given the assumption that the resulting GED can only be either $\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor$ or $\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1$ following a Bernoulli distribution, and the expectation is $d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} = d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}\cdot(1+\epsilon)$, the probabilities for getting the two results can be derived as $p(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor) = \lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1 - d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ and $p(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1) = d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} - \lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor$. In case of getting $\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor$ as the GED calculation result, $\text{LBEI}_{\text{SEPX}}$ becomes

$$\text{LBEI}_{\text{SEPX}} | \lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor = \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}\cdot\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor}{n\cdot(n-1) - \lceil n^\epsilon_{se} \rceil} - \mathcal{B}(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor, 0.5), 0)),$$

where $\lceil n^\epsilon_{se} \rceil = \max(n\cdot(n-1) - d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1} - \lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor, 0)$. Similarly,

$$\text{LBEI}_{\text{SEPX}} | (\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1) = \mathbb{E}(\max(\frac{d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}\cdot(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1)}{n\cdot(n-1) - \lfloor n^\epsilon_{se} \rfloor} - \mathcal{B}(\lfloor d^\epsilon_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1, 0.5), 0)),$$

if $\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1$ is not an integer. If $\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1$ is an integer, then $p(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1) = d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} - \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor = 0$, so this case does not need to be considered. By combining the two cases,

$$\begin{aligned}
\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}} &= p(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor) * \mathrm{LBEI}_{\mathrm{SEPX}} | \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + p(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1) \cdot \mathrm{LBEI}_{\mathrm{SEPX}} | (\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1) \\
&= (d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} - \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor) \cdot \mathbb{E}(\max(\frac{d^{*}_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} \cdot (\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1)}{n \cdot (n-1) - \lfloor n^{\epsilon}_{se} \rfloor} - \mathcal{B}(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1, 0.5), 0)) \\
&+ (\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor + 1 - d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}) \cdot \mathbb{E}(\max(\frac{d^{*}_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} \cdot \lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor}{n \cdot (n-1) - \lceil n^{\epsilon}_{se} \rceil} - \mathcal{B}(\lfloor d^{\epsilon}_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} \rfloor, 0.5), 0)).
\end{aligned}$$

□

A.2. List of Mathematical Symbols

This section provides a list of all mathematical symbols used in this paper.

$\mathbb{V}$	A set of vertices
v_i	Vertex(node) with index i
$\mathbb{E}$	A set of directed edges
$e_{i,j}$	A directed edge from vertex i to vertex j
$\mathcal{G}$	A directed graph
$\lvert\mathcal{G}\rvert$	The order of a directed graph $\mathcal{G}$, which equals the number of its vertices
γ_v	A function that assigns an attribute (e.g., an integer) to each vertex of a directed graph
γ_e	A function that assigns an attribute (e.g., an integer) to each edge of a directed graph
$\delta : \mathcal{G} \to \mathcal{G}'$	A function that applies an elementary graph edit to transform $\mathcal{G}$ to $\mathcal{G}'$
$\overline{\delta} = \delta_1, \delta_2, \ldots, \delta_d$	A sequence of graph edit operations, and d is the length of the resulting edit path
$\mathrm{GED}(\mathcal{G}_1, \mathcal{G}_2)$	The graph edit distance (GED) between $\mathcal{G}_1$ and $\mathcal{G}_2$
$\Delta(\mathcal{G}_1, \mathcal{G}_2)$	The set of all edit paths that transform $\mathcal{G}_1$ to an isomorphism of $\mathcal{G}_2$ (including $\mathcal{G}_2$ itself)
$c(\delta_i)$	The cost of edit δ_i (in this work, all types of edit operations are defined to have the same cost of 1)
$\overline{\delta}^{*}_{\mathcal{G}_1,\mathcal{G}_2}$	The edit path that minimizes the total edit cost to transform $\mathcal{G}_1$ to an isomorphism of $\mathcal{G}_2$ (including $\mathcal{G}_2$ itself)
$d^{*}_{\mathcal{G}_1,\mathcal{G}_2}$	The length of the shortest edit path that transforms $\mathcal{G}_1$ to an isomorphism of $\mathcal{G}_2$ (including $\mathcal{G}_2$ itself)
π	A permutation of multiple elements/indices
$\lceil \cdot \rceil$	The ceiling function
$\lfloor \cdot \rfloor$	The floor function
$\boldsymbol{A}_{\mathcal{G}}$	The attributed adjacency matrix (AA-matrix) for graph $\mathcal{G}$
$A^{\mathcal{G}}_{i,j}$	The entry in ith row and jth column of matrix $\boldsymbol{A}_{\mathcal{G}}$

$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{P}_\pi$	A permutation matrix based on permutation π
$P^{\pi}_{i,j}$	The entry in ith row and jth column of matrix $\boldsymbol{P}_\pi$
$d(\boldsymbol{A}, \boldsymbol{B})$	A function that returns the number of different entries between two matrices ($\boldsymbol{A}$ and $\boldsymbol{B}$ here) with same shape
$\hat{\mathcal{G}}$	The extended graph of $\mathcal{G}$ after adding null vertices
S_n	The set of all permutations of $\{1, 2, 3, \ldots, n\}$
$\mathbf{1}_{\text{condition}}$	A function that returns 1 if the condition is true, 0 otherwise
$\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$	The permutation that minimizes $d(\boldsymbol{A}_{\hat{\mathcal{G}}_1}, \boldsymbol{P}_\pi \boldsymbol{A}_{\hat{\mathcal{G}}_2} \boldsymbol{P}_\pi^\top)$
$\boldsymbol{A}_{\hat{\mathcal{G}}_2 \to \hat{\mathcal{G}}_1}$	The permuted AA-matrix of $\hat{\mathcal{G}}_2$ using permutation matrix $\boldsymbol{P}_{\pi^*_{\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}}$
$r(\boldsymbol{A}, \boldsymbol{B})$	A function that returns a matrix inheriting each entry from $\boldsymbol{A}$ or $\boldsymbol{B}$ with probability 0.5 (that is, if $\boldsymbol{C} = r(\boldsymbol{A}, \boldsymbol{B})$, then $p(C_{i,j} = A_{i,j}) = p(C_{i,j} = B_{i,j}) = 0.5$ for any valid i, j)
$m(\boldsymbol{A})$	A function that alters each element of $\boldsymbol{A}$ with an equal probability
p_m	Mutation probability
$f(\mathcal{G})$	The fitness/reward of $\mathcal{G}$
$\mathcal{G}_{\text{opt}}$	The global optimal graph
$\max(\cdot, \cdot, \ldots, \cdot)$	A function that returns the maximum value among all inputs
$d_v(\boldsymbol{A}, \boldsymbol{B})$	A function that returns the number of different diagonal entries between two matrices ($\boldsymbol{A}$ and $\boldsymbol{B}$ here) with same shape
$d_e(\boldsymbol{A}, \boldsymbol{B})$	A function that returns the number of different non-diagonal entries between two matrices ($\boldsymbol{A}$ and $\boldsymbol{B}$ here) with same shape
$d^*_{e,\mathcal{G}_1,\mathcal{G}_2}$	A simplified symbol to denote $d_e(\boldsymbol{A}_{\mathcal{G}_1}, \boldsymbol{A}_{\mathcal{G}_2 \to \mathcal{G}_1})$
n_s	Number of common entries among multiple matrices
n_{se}	Number of common non-diagonal entries among multiple matrices
$\mathcal{B}(n, p)$	The number of successful trials after sampling from a binomial distribution with n trials and success probability of p
$\text{LBEI}_{\text{SEPX}}$	The lower bound of expected improvement of the SEP crossover
$\text{LBEI}_{\text{STDX}}$	The lower bound of expected improvement of the standard crossover
$\text{LBEI}_{\text{MUTA}}$	The lower bound of expected improvement of mutation
LBEI_{RLU}	The lower bound of expected improvement of the unbiased agent
LBEI_{RLO}	The lower bound of expected improvement of the oracle agent
n^1_{opt}, n^1_1 and n^1_2	The number of ones in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$, excluding diagonal entries
n^0_{opt}, n^0_1 and n^0_2	The number of zeros in $\boldsymbol{A}_{\hat{\mathcal{G}}_{\text{opt}}}$, $\boldsymbol{A}_{\hat{\mathcal{G}}_1}$ and $\boldsymbol{A}_{\hat{\mathcal{G}}_2}$, excluding diagonal entries

$\boldsymbol{Q}_\theta$	A matrix in which each entry $Q^\theta_{i,j}$ defines a separate categorical distribution
θ	The parameter set that contains the logits for defining the categorical distributions in $\boldsymbol{Q}_\theta$
$z^k_{i,j}$	The logits used to defining a categorical distribution through softmax functions, where k denotes the class label
R	The reward for the currently sampled architecture in a RL run
b	A baseline reward to reduce the variance of gradient estimate
$\boldsymbol{Q}^*_\theta$	The optimal permutation of $\boldsymbol{Q}_\theta$ as defined in Lemma A.4 in Appendix A.1
t	The current time step
θ_t	Policy parameter at time step t
α	A positive scaling factor
η	Learning rate
ϵ	Error ratio
$d^\epsilon_{e,\mathcal{G}_1,\mathcal{G}_2}$	The resulting $d^*_{e,\mathcal{G}_1,\mathcal{G}_2}$ with error ratio ϵ in GED calculation
$\mathrm{LBEI}^\epsilon_{\mathrm{SEPX}}$	The resulting $\mathrm{LBEI}_{\mathrm{SEPX}}$ with error ratio ϵ in GED calculation
n^ϵ_{se}	The resulting n_{se} with error ratio ϵ in GED calculation

A.3. Example for Demonstrating the Permutation Problem and the SEP Crossover Solution

Figure A.1 provides a visual example of the permutation problem and how the SEP crossover solves it.

A.4. Experimental Setup Details

For experiments in Section 5.1, all the RE-based variants used a population size of 100 and tournament selection with size 10. For NAS-bench-101, GED to the global optimal architecture was used as the fitness, and 50 independent runs were performed, each with a maximum number of evaluations of 10^3. The allowed mutation operations were the same as in the original NAS-bench-101 example code (https://github.com/google-research/nasbench). For RL, the same implementation as in https://github.com/automl/nas_benchmarks was used. For experiments in Section 5.3, the learning rate was 0.5, as recommended by Ying et al. (2019).

For NAS-bench-NLP, GED to the GRU architecture was used as fitness, and 50 independent runs were performed, each with a maximum number of evaluations of 10^3. The mutation operation was the same as in https://github.com/automl/NASLib. In both benchmarks, for each crossover operation, the offspring was evaluated only if it was a valid architecture in the benchmark space and different from both parents. The maximum number of trials was 50, i.e. the current crossover was skipped after reaching this limit.

For experiments on NAS-bench-101 in Section 5.2, the experimental setups was the same as in Section 5.1 except the maximum number of evaluations was 2×10^3.

For experiments in Section 5.3, the setup for the NAS-bench-101 experiments was the same as in Section 5.1, except validation accuracy was used as the fitness during evolution and test accuracy as the final performance of each architecture. The maximum number of evaluations was 10^4. In the experiments on NAS-bench-301, all the RE-based variants had a population size of 100 and tournament size of 10, and 30 independent runs were performed, each with a maximum number of evaluations of 2×10^3. The mutation operation followed the standard strategy in https://github.com/automl/NASLib.

For experiments regarding BO methods (Appendix A.9), the same setup as in https://github.com/automl/nas_benchmarks is

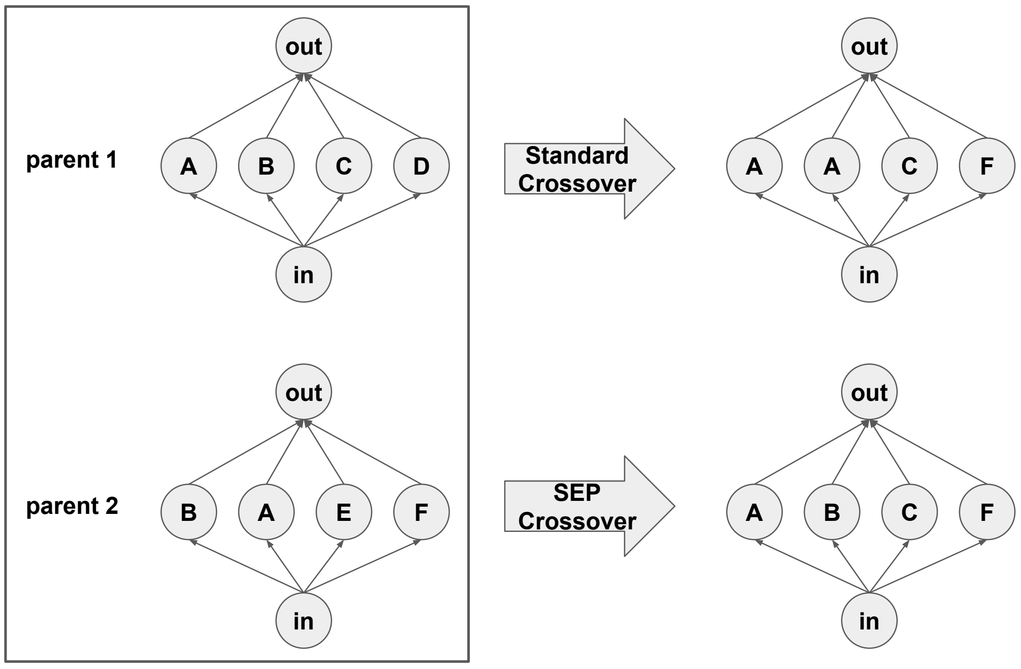


Figure A.1. **The permutation problem and the SEP crossover solution.** The two parent architectures share vertices A and B. Although these two vertices appear in a different order, together they implement the same function, and this function should not be disrupted during crossover. However, standard crossover cannot identify the subgraph isomorphism, and it loses this substructure. In contrast, the shortest edit path calculation recognizes the isomorphism, and as a result, the SEP crossover preserves this substructure. Thus, the SEP crossover only explores the parts that are functionally inconsistent between the two parents.

used.

For experiments regarding path encoding (Appendix A.10), the default setup without cutoff as in https://github.com/naszilla/naszilla is used.

A.5. Additional Figures for Section 4.3

This section includes the rest of the comparisons between the SEP crossover, standard crossover, mutation, and the two RL variants. Note that the color scales differ between figures to make the conclusions more clear.

Figure A.2 compares $\text{LBEI}_{\text{SEPX}}$ vs. $\text{LBEI}_{\text{MUTA}}$ and $\text{LBEI}_{\text{STDX}}$ vs. $\text{LBEI}_{\text{MUTA}}$ for different combinations of $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ and $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ in NAS-bench-NLP (Klyuchnikov et al., 2022). The standard setup was used: $n = 12$, $n^1_{\text{opt}} = 14$, $n^1_1 = 11$, and $n^1_2 = 11$. Although the standard crossover is slightly worse than mutation in most cases, the SEP crossover has a considerable theoretical advantage.

Figure A.3 compares $\text{LBEI}_{\text{STDX}}$ vs. $\text{LBEI}_{\text{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ combinations. $\text{LBEI}_{\text{STDX}}$ is smaller than $\text{LBEI}_{\text{MUTA}}$ in most cases.

Figure A.4 shows LBEI_{RLU}, LBEI_{RLO} and LBEI_{RLU} vs. LBEI_{RLO} under different $\alpha \cdot \eta$ values. A $\alpha \cdot \eta$ value of 0.1 provides the best tradeoff between unbiased agent and oracle agent.

Figure A.5 compares LBEI_{RLU} vs. $\text{LBEI}_{\text{MUTA}}$ and LBEI_{RLO} vs. $\text{LBEI}_{\text{MUTA}}$ under different $\alpha \cdot \eta$ values. Whereas unbiased agent is generally worse than mutation, the oracle agent is better in some cases and worse in others.

Figure A.6 compares $\text{LBEI}_{\text{STDX}}$ vs. LBEI_{RLU} and $\text{LBEI}_{\text{STDX}}$ vs. LBEI_{RLO} under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ combinations. The standard crossover is slightly worse than both RL agents in most cases.

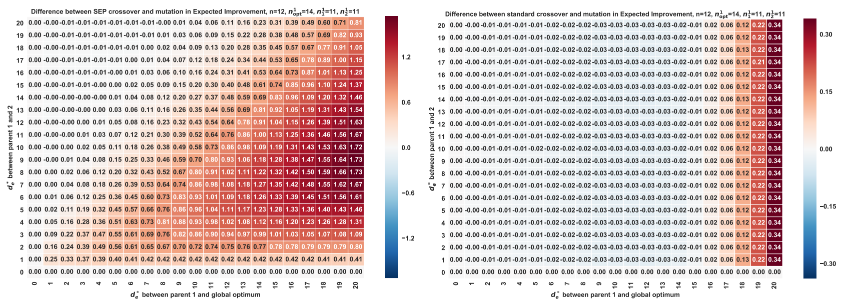


Figure A.2. **Comparison of expected improvement in NAS-bench-NLP.** (Left) Differences between $\text{LBEI}_{\text{SEPX}}$ and $\text{LBEI}_{\text{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. $\text{LBEI}_{\text{SEPX}}$ is larger than $\text{LBEI}_{\text{MUTA}}$ in most situations. (Right) Differences between $\text{LBEI}_{\text{STDX}}$ and $\text{LBEI}_{\text{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. $\text{LBEI}_{\text{STDX}}$ is slightly smaller than $\text{LBEI}_{\text{MUTA}}$ in most situations. These two observations lead to the same conclusion for NAS-bench-NLP as for NAS-bench-101 in Figure 1: Although the standard crossover has a slightly worse expected improvement than mutation under most circumstances, the SEP crossover has a considerable theoretical advantage.

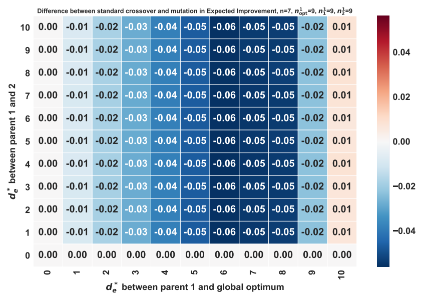


Figure A.3. **Comparison of expected improvement between standard crossover and mutation in NAS-bench-101.** Differences between $\text{LBEI}_{\text{STDX}}$ and $\text{LBEI}_{\text{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. $\text{LBEI}_{\text{STDX}}$ is smaller than $\text{LBEI}_{\text{MUTA}}$ in most cases.

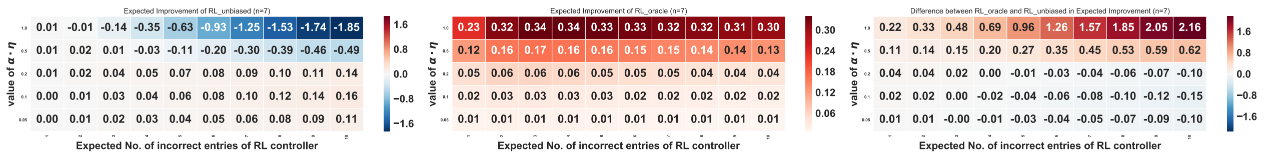


Figure A.4. **Expected improvement of RL** (Left) LBEI_{RLU} under different $\alpha \cdot \eta$ values. (middle) LBEI_{RLO} under different $\alpha \cdot \eta$ values. (right) $\text{LBEI}_{\text{RLO}} - \text{LBEI}_{\text{RLU}}$ under different $\alpha \cdot \eta$ values. A $\alpha \cdot \eta$ value of 0.1 provides the best tradeoff between unbiased agent and oracle agent.

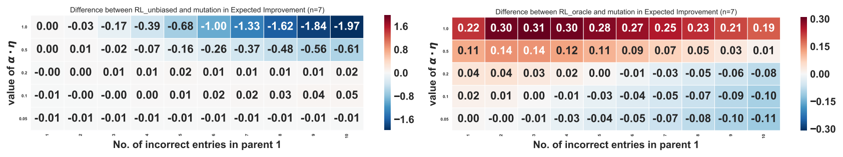


Figure A.5. **Comparison of expected improvement between RL and mutation** (Left) $\text{LBEI}_{\text{RLU}} - \text{LBEI}_{\text{MUTA}}$ under different $\alpha \cdot \eta$ values. (right) $\text{LBEI}_{\text{RLO}} - \text{LBEI}_{\text{MUTA}}$ under different $\alpha \cdot \eta$ values. Whereas unbiased agent is generally worse than mutation, the oracle agent is better in some cases and worse in others.

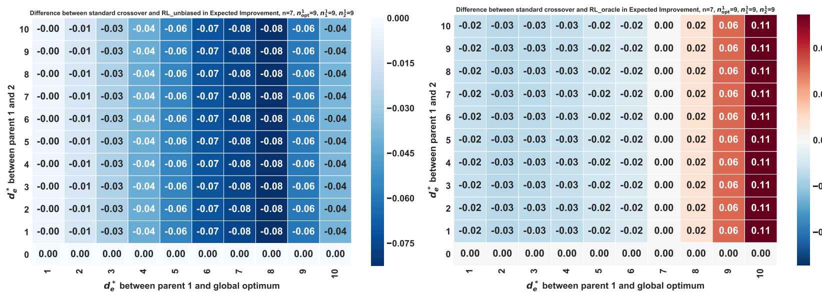


Figure A.6. **Comparison of expected improvement between standard crossover and RL** (Left) $\mathrm{LBEI}_{\mathrm{STDX}} - \mathrm{LBEI}_{\mathrm{RLU}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. (right) $\mathrm{LBEI}_{\mathrm{STDX}} - \mathrm{LBEI}_{\mathrm{RLO}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. A $\alpha \cdot \eta$ value of 0.1 is used for RL. The standard crossover is slightly worse than both RL agents in most cases.

A.6. Additional Figures for Section 4.4

As in Section 4.3, Monte Carlo simulations with 10^6 trials each were performed to estimate the values of $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ under different error ratios ϵ. Figure A.7 compares $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ with $\mathrm{LBEI}_{\mathrm{MUTA}}$, $\mathrm{LBEI}_{\mathrm{RLU}}$ and $\mathrm{LBEI}_{\mathrm{RLO}}$ under error ratios $\epsilon =$ 0.1, 0.2, and 0.3. Because Figure A.3 and A.6 already show that $\mathrm{LBEI}_{\mathrm{STDX}}$ is worse than $\mathrm{LBEI}_{\mathrm{MUTA}}$, $\mathrm{LBEI}_{\mathrm{RLU}}$ and $\mathrm{LBEI}_{\mathrm{RLO}}$ in most cases, $\mathrm{LBEI}_{\mathrm{STDX}}$ is not included in these comparisons.

The conclusion is that the SEP crossover has a theoretical advantage in expected improvement compared to mutation, standard crossover, and RL even with a very high error ratio of 30% in the GED calculations. Thus, if the computational cost of the SEP crossover needs to be reduced, approximation methods can be used to calculate GED.

A.7. Additional Figures for Section 5.1

Figures A.8 and A.9 show relative frequencies of different parent combinations in NAS-bench-101 and NAS-bench-NLP, respectively. Note that the high relative frequencies of $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1} = 0$ and $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2} = 0$ are due to the convergence of the search algorithm, i.e. no further improvement can be made from them. The high-frequency areas match the assumptions of the theory, and thus the theoretical conclusions apply to real-world NAS.

A.8. Additional Results for Section 5.3

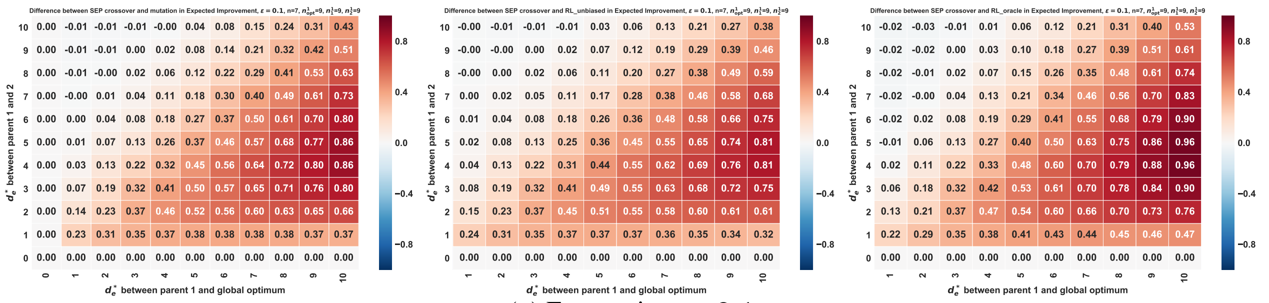
Figure A.10 shows that SEP crossover converges consistently faster than the other methods in all three benchmarks, i.e. NAS-bench-101, NAS-bench-NLP, and NAS-bench-301.

Note that since NAS-bench-NLP is not queryable, it is not possible to measure the prediction accuracy of each architecture in this benchmark the same way as that architecture is trained; the computational cost would be prohibitive given the scale of the experiments (i.e. multiple evaluations of multiple algorithms trained many times in each evaluation). Instead, the GED to GRU/LSTM is used as a noise-free fitness/reward for evaluating the performance of different methods in this complex search space.

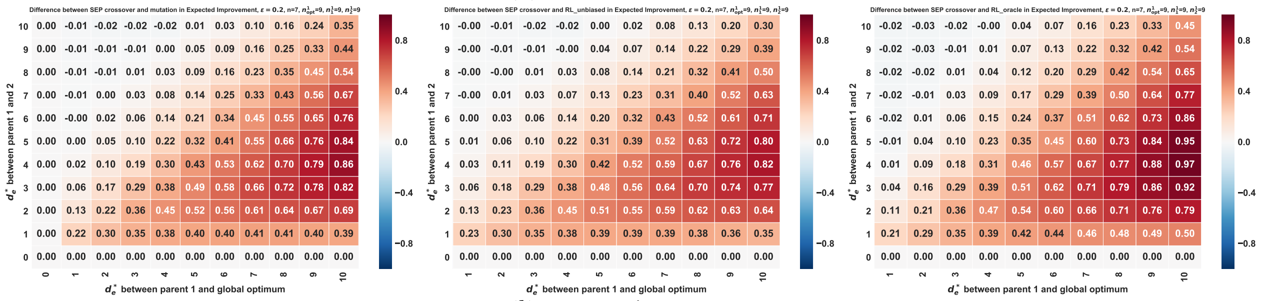
Note also that NAS-bench-301 was not used in GED-related experiments because there are two separate graphs for each architecture (the normal cell and the reduction cell); currently the theory does not apply to pairs of graphs. Such an extension is left for future work.

A.9. Comparison with Bayesian optimization

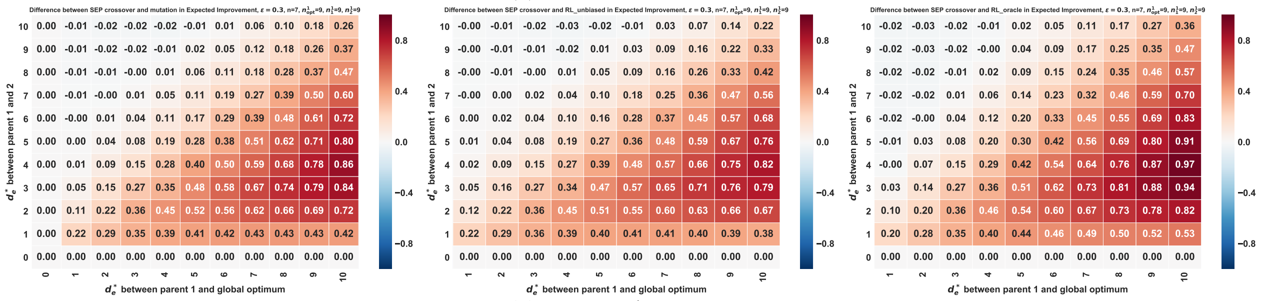
Figure A.11 compares the SEP crossover with two Bayesian optimization (BO) methods, namely BOHB (Falkner et al., 2018) and SMAC (Hutter et al., 2011). The parameter setup of BOHB and SMAC follows the guidelines in Ying et al. (2019). The SEP crossover consistently outperforms both BOHB and SMAC in NAS-bench-101.



(*a*) Error ratio $\epsilon = 0.1$



(*b*) Error ratio $\epsilon = 0.2$



(*c*) Error ratio $\epsilon = 0.3$

Figure A.7. **Comparison of expected improvement between SEP crossover, mutation, and RL in NAS-bench-101 at various level of GED calculation error.** (Left) Differences between $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{MUTA}}$ under different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\mathrm{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. (Middle) Differences between $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{RLU}}$. (Right) Differences between $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ and $\mathrm{LBEI}_{\mathrm{RLO}}$. $\mathrm{LBEI}^{\epsilon}_{\mathrm{SEPX}}$ is larger (i.e. more red) than $\mathrm{LBEI}_{\mathrm{MUTA}}$, $\mathrm{LBEI}_{\mathrm{RLU}}$, and $\mathrm{LBEI}_{\mathrm{RLO}}$ in almost all cases. Thus, the SEP crossover has a theoretical advantage over mutation and RL even at a very high level of error in the GED calculations. Therefore, if needed, approximation methods can be used to reduce the computational cost of the SEP crossover.

A.10. Comparison with path encoding

Figure A.12 compares the SEP crossover with a crossover operator based on path encoding (White et al., 2021b). During a path encoding crossover, the offspring inherits the path with 100% probability if this path is in both parents, and with 50% if it is only in one of them. The SEP crossover significantly outperforms the path encoding crossover in NAS-bench-101.

A.11. Computational time of GED calculation

The GED calculations in the experiments are based on the NetworkX library (Hagberg et al., 2008), which implements an exact GED calculation method (Abu-Aisheh et al., 2015) with reasonable computational efficiency. In the experiments, the calculation time of GED was found to depend not only on the size of the two graphs, but also on the distance between them: If the two graphs are similar, the GED is calculated faster. As a characterization of the computational cost of GED calculations during SEP crossover, their average computation times under different parent distances and sizes are shown in Table A.2. These cases cover more than 99.9% of the cases encountered in the experiments (according to Figure A.8 and A.9). All the GED computations ran on a single Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz. According to Table A.2, the GED calculation time is almost negligible in NAS-bench-101 search space, and acceptable even in the largest NAS benchmark, i.e. NAS-bench-NLP.

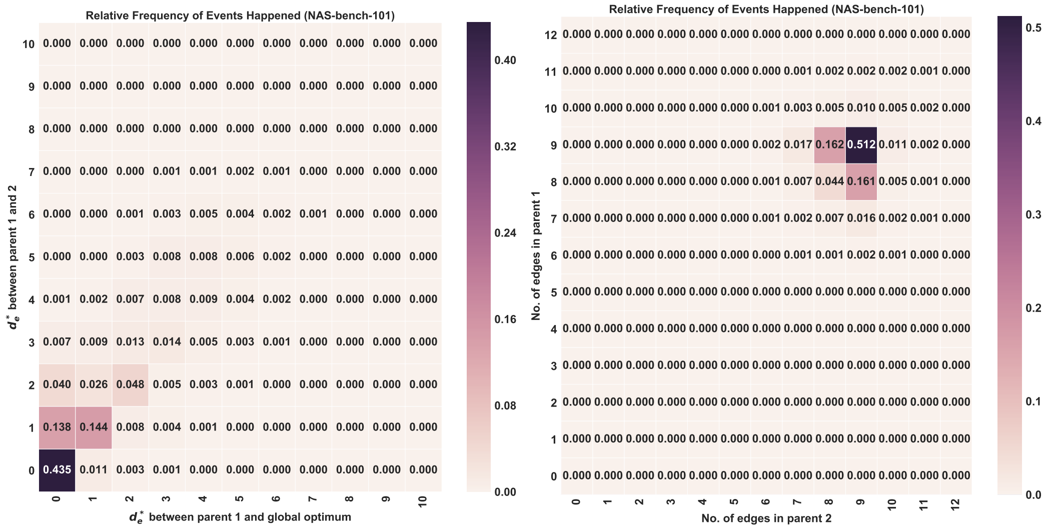


Figure A.8. **Relative frequencies of different parent combinations in NAS-bench-101 experiments.** (Left) Relative frequencies of different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. All events happen in the regions where the SEP crossover has a theoretical advantage in terms of expected improvement (as seen in Figure 1). (Right) Relative frequencies of different n_1^1 and n_2^1 combinations. The event $n_1^1 = 9$ and $n_2^1 = 9$ happens most frequently during the experiments, and this setup is indeed used in the theoretical analysis. Thus, the theoretical analysis applies to situations that arise in NAS-bench-101.

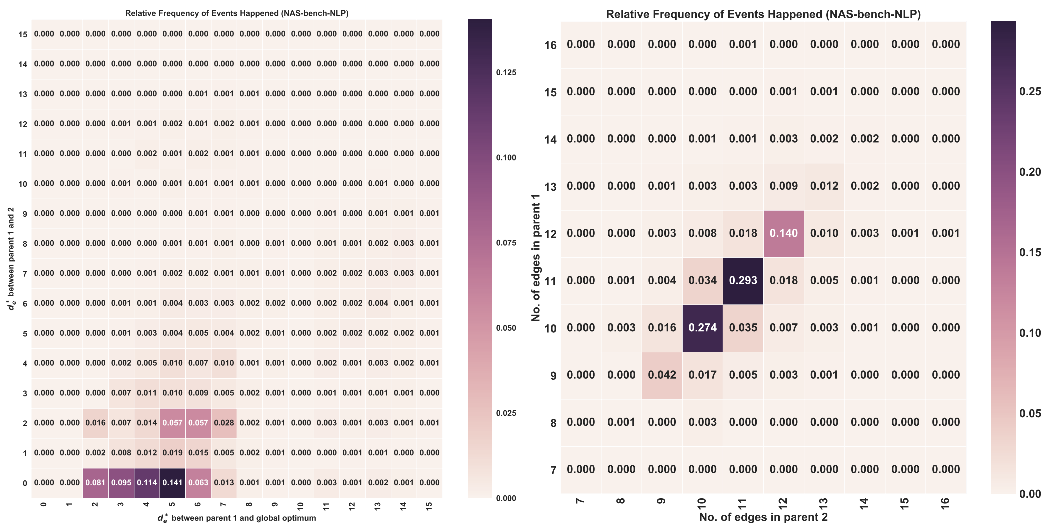


Figure A.9. **Relative frequency of different parent combinations in NAS-bench-NLP experiments.** (Left) Relative frequencies of different $d^*_{e,\hat{\mathcal{G}}_1,\hat{\mathcal{G}}_2}$ (y-axis) and $d^*_{e,\hat{\mathcal{G}}_{\text{opt}},\hat{\mathcal{G}}_1}$ (x-axis) combinations. All the events happen in the regions where the SEP crossover has a theoretical advantage in terms of expected improvement (as seen in Figure A.2). (Right) Relative frequencies of different n_1^1 and n_2^1 combinations. Most events happen around the $n_1^1 = 11$ and $n_2^1 = 11$ combination, which is the setup used in the theoretical analysis. Together Figures A.8 and A.9 show that the theoretical analysis applies to the actual experimental settings.

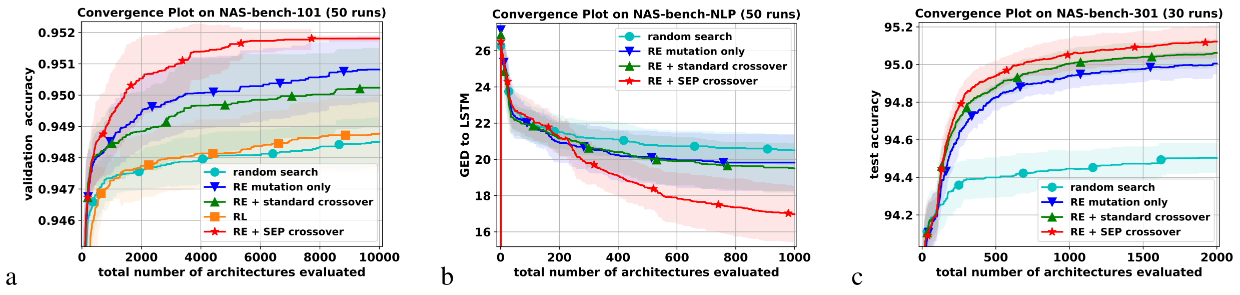


Figure A.10. **Performance of the search methods in three different NAS benchmarks.** (a) Convergence in NAS-bench-101. The plot shows the validation accuracy used as the direct fitness/reward for search strategies. The SEP crossover performs significantly better than the other approaches. (b) Convergence in NAS-bench-NLP. This benchmark is a noise-free environment, and the plot shows the convergence of GED to LSTM. Again, the SEP crossover performs better than the other methods. (c) Convergence in NAS-bench-301. The plot shows the surrogate-returned noise-free test accuracy. The noisy version of accuracy predicted by the surrogate model was used as the direct fitness/reward, and the noise-free accuracy (shown) as the final objective. The SEP crossover has consistently better search ability in this benchmark as well.

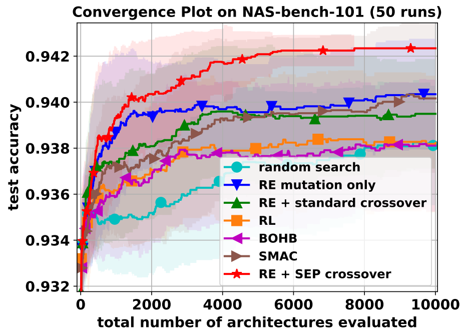


Figure A.11. **Average test accuracy in NAS-bench-101.** The SEP crossover performs consistently better than the two Bayesian optimiation (BO) methods BOHB and SMAC.

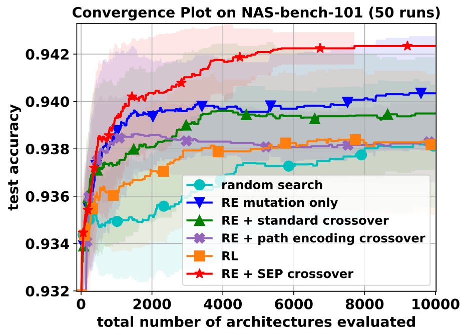


Figure A.12. **Average test accuracy in NAS-bench-101.** The SEP crossover performs significantly better than the path encoding crossover.

Table A.2. Computation Time of GED Calculation

NAS-Bench-101 (7 nodes)							
GED between parents	1	2	3	4	5	6	7
computation time (s)	0.009	0.012	0.019	0.029	0.041	0.060	0.084
NAS-Bench-NLP (12 nodes)							
GED between parents	1	3	5	7	9	11	13
computation time (s)	0.015	0.046	0.281	1.156	3.190	10.374	21.957