Profiling the Distance Characteristics of Mutation Operators for Permutation-Based Genetic Algorithms

Vincent A. Cicirello and **Robert Cernera**
Computer Science and Information Systems
School of Business, Richard Stockton College
101 Vera King Farris Drive
Galloway, NJ 08205

Abstract

In this paper, we consider the permutation representation of genetic algorithms, and more generally, local search algorithms. We use a variety of permutation distance measures to profile the behavior of the most commonly used mutation operators for permutation-based genetic algorithms. Our operator profiles are also applicable to other local search algorithms, such as simulated annealing, as the most common permutation mutation operators are also commonly found as neighborhood operators for other metaheuristics in a search of the space of permutations. In addition to using several existing distance measures, we introduce two specific instances of the edit distance measure. Our aim is to offer the GA, and local search practitioner, guidance in the selection of mutation and neighborhood operators.

1 Introduction

Genetic algorithms are among a broad class of problem solving algorithm, known as evolutionary computation, that are motivated by concepts from natural genetics and evolution (Mitchell 1998). A genetic algorithm (GA) evolves a population of candidate solutions to the problem through operations that mimic natural selection, mutation, and crossover. GAs have been applied successfully to a wide variety of problems from diverse domains such as engineering, the sciences, entertainment, robotics, software engineering, and many others (Soule 2012; Krasnogor 2011).

GAs rely on problem independent operators that randomly perturb candidate solutions (mutation operators) and that recombine parts of multiple parents to produce offspring for subsequent generations (crossover operators). Aside from the unusual case of a specialized problem-dependent operator, the most common bitstring-based GA has a limited set of mutation and crossover operators. This is part of the basis for the GA's claimed power as a problem-independent problem solver—regardless of the problem, the operators remain the same. In the so-called Simple GA, as well as its most common variations, the candidate solutions in the population are represented as bitstrings. Knowledge of the specific problem is limited to the fitness function which interprets a bitstring as a candidate solution to the problem and provides an evaluation of the quality of that solution. This mapping of bitstring to its problem dependent meaning is similar to the mapping of a genotype to a phenotype. For the common bitstring representation, mutation typically involves random "flipping" of one or more bits. For crossover (the breeding of a pair of offspring from a pair of parents), there are a variety of commonly employed operators that exchange subsets of the bitstrings between the parents.

We do not concern ourselves here with the bitstring representation of a GA; rather, we examine the operators of the permutation-based GA. For some problems, the bitstring representation does not lend itself well to solution representation; and thus, other representations have been developed. One such representation is the permutation, which enables a more natural representation to problems that involve sequencing or ordering. In the permutation representation, a candidate solution to the problem is represented as a permutation over the elements of a set—e.g., for the Traveling Salesperson Problem, an individual member of the GA population is represented as a permutation of the cities. Although this seems to move away from the problem independent nature of a GA, the operators that are employed by a permutation-based GA do not rely on any problem-specific knowledge for mutation and recombination—the operators do not care what the permutation itself represents.

The biggest challenge with applying a permutation-based GA to a problem is choosing mutation and crossover operators. The operators of the Simple GA do not have direct corresponding operators for permutations as the obvious adaptations would create invalid permutations. Instead, a catalog of mutation and crossover operators has been developing over the years by the practitioners of permutation-based GAs. Some of these operators are claimed as good choices for problems where the important aspect of the permutation is an element's absolute position within the permutation independent of other elements, others when the relative ordering of the elements is critical. One issue, however, is that not all problems are easily categorized into one or the other category (Martí, Laguna, and Campos 2005). Additionally, not all of the available operators are easily categorized as strictly appropriate for absolute or relative ordering properties.

In this paper, we focus on delivering to the GA or local search practitioner guidance in selecting operators for permutation-based problems. Specifically, we focus on the most commonly employed mutation operators. Mutation is

meant to be a *small* random perturbation of a member of the population, relying on the GA assumption (and more generally, the assumption of local search algorithms) that nearby solutions are of similar fitness. What is a *small* random perturbation of a permutation? This depends on whether the problem is what has been termed an "A-permutation problem" or an "R-permutation problem" (Campos, Laguna, and Martí 2005; Martí, Laguna, and Campos 2005), depending on whether absolute or relative positioning is most important to the problem. We introduce two distance measures, *Reinsertion Distance* and *Interchange Distance*, for permutations that are instances of the concept of edit distance. Using our distance measures and a variety of other permutation distances (Ronald 1998; Sörensen 2007) that are available in the literature, we have developed profiles of the most common mutation operators. As these mutation operators are also often used as the neighborhood function for other local search algorithms (e.g., simulated annealing), our operator profiles are relevant more generally to other permutation-based metaheuristics. Our aim is to offer the GA practitioner guidance in the design of a permutation-based GA.

Our paper is organized as follows. We begin, in Section 2, with a discussion of background on fitness landscapes and existing measures of permutation distance. In Section 3, we discuss the concept of Edit Distance, and introduce two instances of edit distance that we call Reinsertion Distance and Interchange Distance. We overview our profiling methodology in Section 4. Next, in Section 5, we present our profiles of the common mutation operators for permutation-based GAs. Finally, we wrap-up in Section 6.

2 Background

2.1 Fitness Landscapes

We can conceptualize the space of candidate solutions to an optimization problem as points on a surface where the height of the point corresponds to the fitness of that solution. This surface is known as the fitness landscape (Mitchell 1998), and likely contains a variety of peaks and valleys which represent local optima. Our optimization problem is to find an optimal point on that landscape. Local search algorithms and GAs tend to perform better when fitness landscapes are smooth with small numbers of local optima; and are especially challenged when faced with a fitness landscape with plateaus and large numbers of local optima.

GA mutation operators serve the same purpose as a neighborhood operator for a local search algorithm such as simulated annealing—namely, enabling locally improving modifications to a current candidate solution; whereas crossover operators serve the purpose of escaping from local optima.

An important characteristic of fitness landscapes is fitness-distance correlation (Jones and Forrest 1995), which is an application of the Pearson correlation coefficient, used to measure the correlation between the fitness of a solution and its distance from the optimal solution. If fitness (i.e., solution quality) improves the nearer you are to the optimal solution (as distance to the optimal solution decreases), then searching locally (i.e., via mutation) around the current population of solutions will be effective.

The structure of this conceptual fitness landscape depends strongly on how we define the neighborhood of a candidate solution (e.g., (Czogalla and Fink 2009)). What does it mean for a solution to be "near" another? The shape of the fitness landscape, therefore, depends upon more than the problem alone. It also depends upon the operator used for local improvement, which in the case of the GA is the mutation operator. In order to define a relevant fitness landscape that characterizes the problem solving behavior of a GA, one needs to carefully consider the choice of distance measure.

2.2 Permutation Distance Measures

We need a mechanism for determining if a permutation operator produces small modifications to a permutation. Thus, we need to have a notion of distance between permutations. Many distance measures have been proposed (Ronald 1998; Sörensen 2007; Shapira and Storer 2007). We have chosen a set of distance measures that capture the essence of the so-called "A-Permutation" and "R-Permutation" type problems. In all of the formalization of distance measures that follow, we use p_1 and p_2 to refer to permutations, $p_1(i)$ refers to the element in position i of the permutation, and N is the length of a permutation.

Absolute Position Based Distance Measures: When profiling the mutation operators, we consider the following distance measures as representative of the characteristics of problems where "absolute" position within the permutation is most critical to the permutation's fitness as a solution:

- **Exact Match:** The exact match distance (Ronald 1998) is an extension of the concept of Hamming distance to permutations. It is the count of the number of positions containing different elements.

$$\delta(p_1, p_2) = \sum_{i=1\ldots N} \begin{cases} 1 & \text{if } p_1(i) \neq p_2(i) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

- **Deviation Distance:** Deviation distance (Ronald 1998) is the normalized sum of the positional deviations of the elements from one permutation to the other, and is formalized as follows:

$$\delta(p_1, p_2) = \frac{1}{N-1} \sum_{e \in p_1} |i-j|, \text{where } p_1(i) = p_2(j) = e. \quad (2)$$

Relative Position Based Distance Measures: We consider the following distance measure as representative of the characteristics of problems where "relative" position among the elements of the permutation is most critical to its fitness:

- **R-Type Distance:** "R-Type" distance (Campos, Laguna, and Martí 2005; Martí, Laguna, and Campos 2005) was developed for permutation problems where relative ordering primarily influences solution fitness, and is the count of the number of adjacent element pairs of p_1 that appear as adjacent element pairs in p_2 and can be defined as:

$$\delta(p_1, p_2) = \sum_{i=1}^{N-1} \begin{cases} 0 & \text{if } \exists j, p_1(i) = p_2(j) \text{ and} \\ & \quad p_1(i+1) = p_2(j+1) \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

Table 1: Pearson correlation, R, between pairs of distance measure for permutations of length 10. Distances between all 3628800 permutations of length 10 to a reference permutation were used in computing the coefficients, R.

	Exact Match	Deviation	R-Type
Exact Match	1.000	0.515	0.024
Deviation	0.515	1.000	0.020
R-Type	0.024	0.020	1.000

Table 2: Pearson correlation, R, between the reinsertion and interchange distances and the other distance measures for permutations of length 10. For reinsertion distance and interchange distance, $R = 0.182$.

	Reinsertion	Interchange
Exact Match	0.301	0.766
Deviation	0.650	0.395
R-Type	0.422	0.009

In selecting a set of distance measures to use in our study, we wanted to include a representative set of distances that can be used in characterizing a variety of fitness landscapes. Using all permutations of length 10 ($N = 3628800$), we computed the distances to a reference permutation from the set. We then examined the Pearson correlation coefficient for pairs of distance measure. The data is summarized in Table 1. You will discover that although both Exact Match and Deviation distance are absolute-position distance measures, they are only moderately correlated ($R = 0.515$). The relative position based distance measure, R-Type, is not correlated to either of the absolute-position measures.

3 Edit Distance for Permutations

Edit distance between two structures (e.g., permutations) is the minimum cost of the "edit operations" required to transform one structure into the other. The concept originated in the string pattern matching community (Wagner and Fischer 1974) and is easily extended to permutations (Sörensen 2007). We consider two "edit distance" measures.

3.1 Reinsertion Distance

We introduce a permutation distance measure that we call *Reinsertion Distance* as the minimum number of removal/reinsertion operations needed to transform one permutation into the other. Edit distance for permutations and strings (Wagner and Fischer 1974; Sörensen 2007), typically allows three different operations (element removal, element insertion, and relabeling an element), although variations exist that include additional operations—e.g., moving or deleting substrings (Shapira and Storer 2007). To apply edit distance, we need to assign costs for each of the 3 operations.

In our Reinsertion Distance, we allow a single type of operation which removes an element and reinserts it someplace else in the permutation. To compute the minimum number of removal/reinsertion operations, we use Wagner and Fischer's dynamic programming algorithm for string edit distance (Wagner and Fischer 1974) where we assign costs of 0.5 for removals, 0.5 for insertions, and ∞ for element replacements. The latter assures us that the edit distance calculation will not consider relabelings. The costs of 0.5 for each of removals and reinsertions has the effect of counting the minimum number of removal/reinsertion operations needed to transform one permutation into the other.

This application of edit distance does not neatly fit into either the absolute or relative distance categories. In Table 2, you will find the correlation coefficient between reinsertion distance and each of exact match, deviation distance, and R-type distance. Reinsertion distance is moderately correlated to all three, with strongest correlation to deviation distance ($R = 0.65$), an absolute-position based measure followed by R-type (0.422), a relative-position based measure. A single removal/reinsert operation could potentially change the absolute locations of a large number of elements (all elements in the most extreme case) while retaining their relative ordering, but the resulting permutation would be a distance of only 1 away from its parent. This seems to imply that this is a relative ordering distance measure. Yet, it is also possible to create a pair of permutations with almost identical relative positions, for which this application of edit distance will give a rather large distance. Consider the permutations: $p_1 = \{1, 2, \ldots N/2, N/2+1, N/2+2, \ldots, N\}$ and $p_2 = \{N/2+1, N/2+2, \ldots, N, 1, 2, \ldots N/2\}$. These 2 permutations are a reinsertion distance apart of $N/2$, yet all but one adjacent pair of elements are held in common.

3.2 Interchange Distance

We introduce another edit distance for permutations. The interchange distance is the minimum number of element by element interchanges (or swaps) needed to transform one permutation into the other—by "swap" we refer to general swaps and not strictly adjacent element exchanges. In Table 2, we demonstrate this to be an absolute-position based distance measure, correlating most strongly with exact match ($R = 0.766$) and moderately correlating with deviation distance ($R = 0.395$). This is another edit distance. However, it does not use Wagner and Fischer's edit distance operations. Instead, it considers a single edit distance operation, general element interchanges–i.e., swapping the locations of 2 elements within the permutation. This is surprisingly easy to compute. Our algorithm for computing interchange distance is inspired by the permutation crossover operator known as cycle crossover (Oliver, Smith, and Holland 1987), namely on its concept of a "cycle." Each "cycle" of k elements in length, contributes $k - 1$ to the number of interchanges needed to transform one permutation into the other. Our algorithm computes Interchange distance by finding all such "cycles." Pseudocode can be found in Algorithm 1.

4 Profiling Methodology

4.1 Permutation Mutation Operators

With the more traditional bitstring-based GA, mutation is almost always limited to a random bit flip—the most obvious operation to produce a small random perturbation of a bitstring. With the permutation representation, the choice of

Algorithm 1 Interchange Distance

Input: $X = \{x_1, x_2, \ldots, x_N\}, Y = \{y_1, y_2, \ldots, y_N\}$
Output: $\delta(X, Y)$ = minimum number of swaps to transform permutation X to Y

```
D ← 0
elements ← {e|e ∈ X}
while elements ≠ ∅ do
    cycle ← ∅
    current ← any member of the set elements
    i ← IndexOf(current, X)
    while current ∉ cycle do
        cycle ← cycle ∪ {current}
        current ← Y(i)
        i ← IndexOf(current, X)
    end while
    D ← D + |cycle| − 1
    elements ← elements − cycle
end while
return D                                  ▷ D = δ(X, Y)
```

operator is not as obvious. Here, we consider the most common choices of mutation operator for permutation-based GAs. Our aim is to shed light on when their behavior can be considered to produce *small* random perturbations. The mutation operators that we profile are as follows:

- **Insertion**: Insertion mutation removes one randomly selected element from the permutation, and reinserts it into a different randomly selected position.
- **Swap**: Swap mutation exchanges the positions of 2 randomly selected elements from the permutation. All other elements remain in their current positions.
- **Scramble**: Scramble mutation selects 2 different random indices, and then randomly shuffles all elements between the 2 indices, inclusive, with all possible reorderings of the selected region equally likely.
- **Reversal**: Reversal mutation selects 2 different random indices, and reverses the selected sub-permutation.

These are the most common permutation mutation operators and are also widely used as neighborhood operators for other local search algorithms (Serpell and Smith 2010; Cicirello 2006; 2007; Valenzuela 2001).

4.2 Generating Profiling Data

Our process for generating the data used to profile the mutation operators is as follows. We consider the following permutation lengths: $\{16, 32, 64, 128, 256, 512, 1024\}$. For each permutation length, and for each combination of mutation operator and distance measure, we generated 10000 random parent permutations and created a child of each via one random mutation. Therefore, in the profiles that follow, each point on each graph is the average distance between the parent and child of 10000 random mutations.

5 Mutation Operator Distance Profiles

We first profile the mutation operators from the perspective of the absolute distance measures, Exact Match Distance (Figure 1) and Deviation Distance (Figure 2). In Figure 1, we see that a single swap mutation produces a child that is a constant exact match distance of 2 from its parent, independent of permutation length. All other mutation operators produce children that are approximately an exact match distance of $N/3$ from the parents. In Figure 2, we see that both Scramble and Reversal are disruptive under deviation distance, producing children that are unlike the parents, especially for larger permutations. However, both the Insertion and Swap operators make very small changes to the parent permutations. The average deviation distance between parent and child for both of these operators is around 0.67, independent of permutation length.

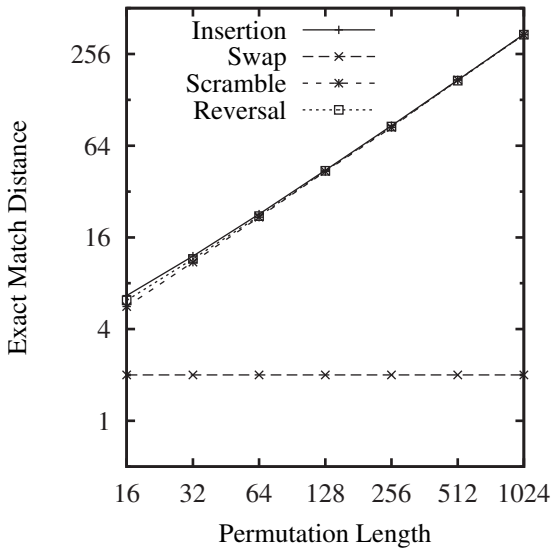

Figure 1: Average Exact Match Distance between parent and child permutations produced by 1 random mutation.

For so-called "A-permutation" problems, Swap mutation should be preferred as it produces small perturbations of the candidate permutation, independent of which absolute position based distance measure we consider. However, Insertion mutation should not be overlooked as a candidate operator for these problems as well. The exact match distance simply fails to recognize the small positional movement made to those elements that are shifted, while deviation distance captures that behavior. The Scramble and the Reversal operators are very disruptive with respect to "A-permutation" problems. However, as such, they might be useful to other local search algorithms to "kick" the search out of a local optima.

We now profile the operators for relative position based distances, namely for the R-Type distance measure (see Figure 3). For the Scramble and Reversal operators, distances increase linearly with the length of the permutation. Insertion mutation consistently produces children on average a smaller distance from the parents as compared to swap—average distances of 3 and 4, respectively, independent of permutation length. Both operators are reasonable choices

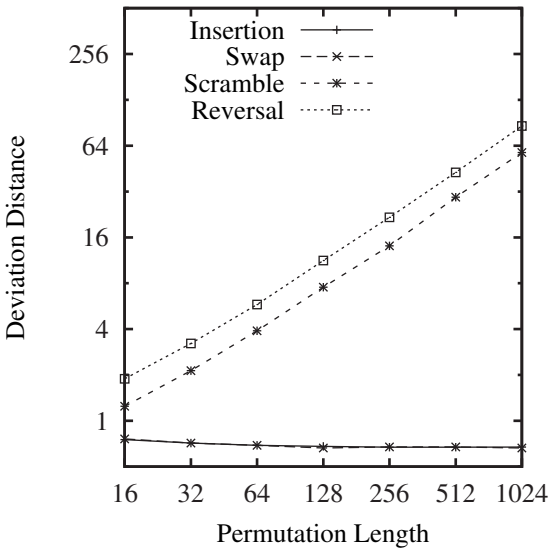

Figure 2: Average Deviation Distance between parent and child permutations produced by 1 random mutation.

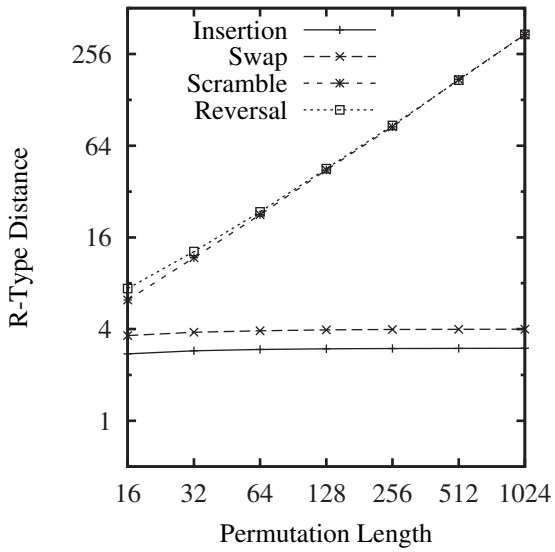

Figure 3: Average R-Type Distance between parent and child permutations produced by 1 random mutation.

for "R-permutation" problems, but in terms of consistent lower distance which implies the smaller random perturbation that mutation operators are meant to provide, insertion mutation is superior to swap. Again, the scramble and reversal operators may be applicable to local search algorithms to "kick" the search out of a locally optimal solution.

In Figure 4, we profile the mutation operators for the Reinsertion Distance. Recall that this edit distance measure is defined as the minimum number of removal/reinsertion operations needed to transform (or edit) one permutation to the other. Thus, the average distance of a child produced via the reinsertion mutation operator is a constant distance of 1 from its parent. Thus, reinsertion mutation is an ideal candidate for permutation problems whose fitness landscape can be characterized by this distance measure. Swap mutation produces children whose average distance from the parents approaches 2 with increasing permutation length. Thus, swap mutation is also a reasonable candidate. The average distances produced by Scramble and Reversal mutations grow linearly with permutation length, and are thus too disruptive to serve as an effective GA mutation operator.

In Figure 5, we profile the mutation operators for Interchange Distance. Recall that this is another edit distance. However, unlike Reinsertion Distance, the Interchange Distance is primarily an absolute position based distance measure. Like the other absolute position based measures, we find that the swap operator stands out from the others, consistently producing children a distance of 1 from the parents, independent of permutation length. Though recall that this measure is defined as the minimum interchanges needed to transform one permutation into the other. As the swap mutation randomly exchanges the positions of 2 elements, by definition only 1 interchange is needed to undo that operation.

6 Conclusions

We have profiled the most common mutation operators for the permutation representation of a GA. These operators are also commonly used as local neighborhood operators for other local search algorithms such as simulated annealing when searching the space of permutations. Thus, our results are generally applicable to metaheuristics beyond the GA.

We have found that for "A-permutation" problems, where absolute position within the permutation is most indicative of solution fitness, that the Swap mutation operator is the most applicable. Regardless of which absolute distance measure we consider (Exact Match, Deviation Distance, or Interchange Distance), the swap operator produces a *small* random change to the permutation, which is the essence of what mutation is meant to be. Depending upon which specific absolute distance measure leads to the "friendliest" fitness landscape (smoother landscape, less local optimas, etc) for the problem we are solving, we may also consider Insertion mutation (see analysis of deviation distance).

For "R-permutation" problems where relative positioning within the permutation has the biggest impact on solution fitness, we have found that Insertion mutation, as well as Swap mutation, produces the smallest random changes to the candidate solutions, and are thus likely to be the most effective choices as mutation operator. Insertion mutation also has proven to be the best choice for problems where the edit distance known as Reinsertion Distance leads to a promising fitness landscape for local improvement algorithms.

Although Scramble mutation, and in most cases Reversal mutation, appear too disruptive to be effective mutators, our profiles can be used to argue that they can be effectively used within a local search algorithm to "kick" the search out of local optimas to prevent or mitigate search stagnation.

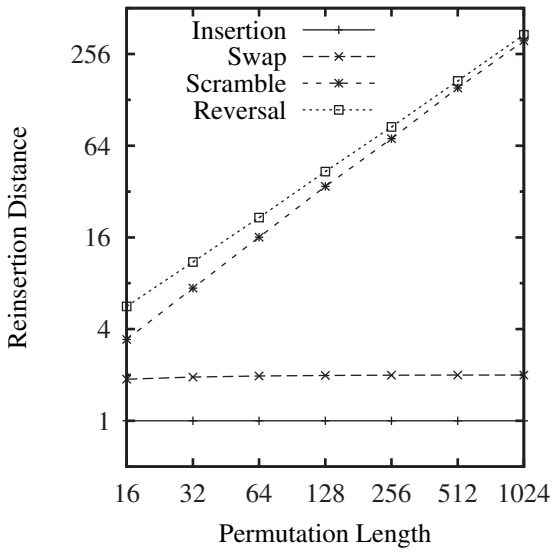

Figure 4: Average Reinsertion Distance between parent and child permutations produced by 1 random mutation.

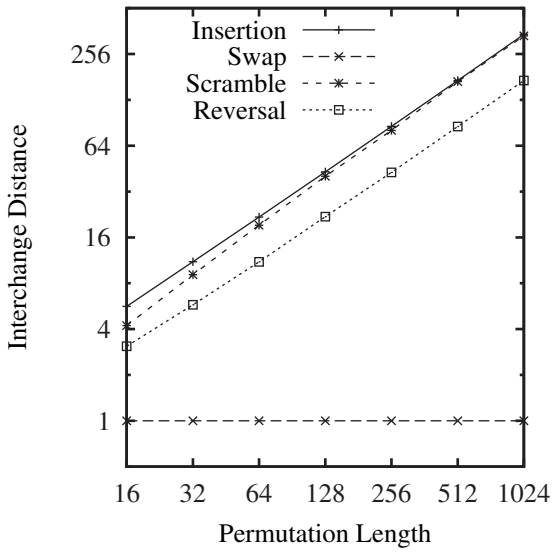

Figure 5: Average Interchange Distance between parent and child permutations produced by 1 random mutation.

Acknowledgments

This work was supported by the National Science Foundation under grant DUE-1059934. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Campos, V.; Laguna, M.; and Martí, R. 2005. Context-independent scatter and tabu search for permutation problems. *INFORMS Journal on Computing* 17(1):111–122.

Cicirello, V. A. 2006. Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. In M. Keijzer et al., ed., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'06)*, volume 2, 1125–1131. ACM Press.

Cicirello, V. A. 2007. On the design of an adaptive simulated annealing algorithm. In *International Conference on Principles and Practice of Constraint Programming: First Workshop on Autonomous Search*. AAAI Press.

Czogalla, J., and Fink, A. 2009. Fitness landscape analysis for the resource constrained project scheduling problem. In Stützle, T., ed., *Learning and Intelligent Optimization*. Springer-Verlag. 104–118.

Jones, T., and Forrest, S. 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 184–192. Morgan Kaufmann Publishers Inc.

Krasnogor, N., ed. 2011. *GECCO '11: Proceedings of the 13th International Conference on Genetic and Evolutionary Computation*. ACM.

Martí, R.; Laguna, M.; and Campos, V. 2005. Scatter search vs. genetic algorithms: An experimental evaluation with permutation problems. In *Metaheuristic Optimization via Memory and Evolution*. Springer. 263–282.

Mitchell, M. 1998. *An Introduction to Genetic Algorithms*. MIT Press.

Oliver, I.; Smith, D. J.; and Holland, J. R. C. 1987. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms*, 224–230.

Ronald, S. 1998. More distance functions for order-based encodings. In *Proceedings of the IEEE Conference on Evolutionary Computation*, 558–563. IEEE Press.

Serpell, M., and Smith, J. E. 2010. Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation* 18(3):491–514.

Shapira, D., and Storer, J. A. 2007. Edit distance with move operations. *Journal of Discrete Algorithms* 5(2):380–392.

Sörensen, K. 2007. Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics* 13(1):35–47.

Soule, T., ed. 2012. *GECCO '12: Proceedings of the 14th International Conference on Genetic and Evolutionary Computation*. ACM.

Valenzuela, C. L. 2001. A study of permutation operators for minimum span frequency assignment using an order based representation. *Journal of Heuristics* 7(1):5–21.

Wagner, R. A., and Fischer, M. J. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery* 21(1):168–173.