Fitness Distance Correlation and Search Space Analysis for Permutation Based Problems

Botond Draskoczy

University of Stuttgart, FMI,
Universitätstr. 38, 70569 Stuttgart,Germany
draskoczy@fmi.uni-stuttgart.de

Abstract. The fitness distance correlation (FDC) as a measure for problem difficulty was first introduced by Forrest and Jones. It was applied to many binary coded problems. This method is now applied to permutation based problems. As demanded by Schiavinotto and Stützle, the distance in a search space is calculated by regarding the steps of the (neighborhood) operator. In this paper the five most common operators for permutations are analyzed on symmetric and asymmetric TSP instances. In addition a local quality measure, the point quality (PQ) is proposed as a supplement to the global FDC. With this local measure more characteristics and differences can be investigated. Some experimental results are illustrating these concepts.

1 Introduction

Permutation based problems like TSP are well analyzed and successfully optimized by evolutionary algorithms (EA) and other optimization heuristics. The common way is to find well adapted operators and heuristic specific parameters by extensive experiments. Another way is to predict good operators by analyzing positive characteristics of the operator specific search space. The fitness distance correlation [6] was successfully applied to this task [3]. By classification of Thierens the FDC is more an analysis tool than a predictive tool [14]. Usually it is not straightforward to calculate the correct operator specific distances between solutions, therefore approximations are used ([8] and [10]). But Schiavinotto and Stützle pointed out the importance of correct distance algorithms in [12], but didn't applied them. One goal of our study is to apply the FDC analysis to permutation based problems with precisely calculated distances for five common operators. An example for such an analysis can be found in [5], but there only symmetric TSP instances for two operators are analyzed. One problem of the FDC is its oversimplifying nature, one scalar value is sufficient to compare operators, but not for analyzing the search space.

After the preliminary definitions in section two, the point quality (PQ) is defined. As a local quality measure, the PQ helps to analyze and visualize a search space. Section three specifies the implemented distance algorithms. Data generation and the inspected test problems are described in section four. Results of the FDC analysis and examples of PQ diagrams are finally presented in section five.

2 Definitions and Operators

Important for analysis of search space (synonymously used for landscape or search graph) is the concept of distance. This pairwise distance depends on the used operator and defines a neighborhood for each point in the search space.

Definition 1. *A (neighborhood) operator* OP *is defined over a set S_n (in this paper the set of all permutations of length n) by*

$$\mathrm{OP} : S_n \to 2^{S_n}$$

The (k-th) neighborhood of a subset $s \subseteq S_n$ we denote by

$$\mathcal{N}^0_{\mathrm{OP}}(s) = s, \quad \mathcal{N}^k_{\mathrm{OP}}(s) = \mathcal{N}_{\mathrm{OP}}(\mathcal{N}^{k-1}_{\mathrm{OP}}(s)) \quad \textit{and} \quad \mathcal{N}_{\mathrm{OP}}(s) = \bigcup_{\pi \in s} \mathrm{OP}(\pi) = \mathcal{N}^1_{\mathrm{OP}}(s)$$

The operator specific size of the neighborhood is denoted as $|OP(\pi)|$ or $|OP|$ (if it has the same size for all $\pi \in S_n$).

Definition 2. *The distance $d_{\mathrm{OP}}(\pi, \varphi)$ between two points (solutions) $\pi, \varphi \in S_n$ in search space is*

$$d_{\mathrm{OP}}(\pi, \varphi) = \min\left\{k \mid \varphi \in \mathcal{N}^k_{\mathrm{OP}}(\{\pi\})\right\}$$

Definition 3. *The operator specific diameter of the search space is*

$$\Phi_{\mathrm{OP}} = \max\left\{d_{\mathrm{OP}}(\pi, \varphi) | \pi, \varphi \in S_n\right\}$$

2.1 Operators

For the formal definition of the operators for permutations (from now S_n is the set of all permutations of length n) we denote permutations in common notation. For example $\pi = (3, 4, 1, 2)$ is a permutation of length $n = 4$ and is identical to the permutation $\pi = (1, 3)(2, 4)$ in cycle notation. The composition is calculated from right to left, for example $(4, 3, 2, 1) \circ (4, 3, 1, 2) = (1, 2, 4, 3)$.

Definition 4 (Neighbor Swap (NbrSwap)). *For a permutation $\pi \in S_n$ the* NbrSwap *operator is defined as*

$$\mathrm{NbrSwap}(\pi) = \{(\pi_1, \pi_2, \ldots, \pi_{k-1}, \pi_{k+1}, \pi_k, \pi_{k+2}, \ldots, \pi_n) \mid 1 \leq k < n\}$$

This operator swaps two adjacent elements in a permutation.

Definition 5 (Element Swap (Swap)). *For a permutation $\pi \in S_n$ the* Swap *operator is defined as*

$$\mathrm{Swap}(\pi) = \{(\pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_j, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i, \pi_{j+1}, \ldots, \pi_n) \mid 1 \leq i < j \leq n\}$$

This is one of the most used operators, it interchanges two elements in a permutation.

Definition 6 (Element Shift (Shift)). *For a permutation* $\pi \in S_n$ *the* Shift *operator is defined as*

$$\begin{aligned}\mathrm{Shift}(\pi) = &\{(\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_{i+k}, \pi_i, \pi_{i+k+1}, \ldots, \pi_n) \mid k > 0\} \cup \\ &\{(\pi_1, \ldots, \pi_{i+k-1}, \pi_i, \pi_{i+k}, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n) \mid k < 0\}\end{aligned}$$

This operator moves an element to another position in the permutation.

Definition 7 (Substring Reversal (SStrRev)). *For a permutation* $\pi \in S_n$ *the* SStrRev *operator is defined as*

$$\mathrm{SStrRev}(\pi) = \{(\pi_1, \ldots, \pi_{i-1}, \pi_j, \pi_{j-1}, \ldots, \pi_{i+1}, \pi_i, \pi_{j+1}, \ldots, \pi_n) \mid 1 \leq i < j \leq n\}$$

This operator reverses a substring of the permutation.

Definition 8 (Substring Shift (SStrShift)). *For a permutation* $\pi \in S_n$ *the* SStrShift *operator is defined as*

$$\begin{aligned}\mathrm{SStrShift}(\pi) = \{&(\pi_1, \ldots, \pi_{i-1}, \pi_{j+1}, \ldots, \pi_{j+k}, \pi_i, \ldots, \pi_j, \pi_{j+k+1}, \ldots, \pi_n) \\ &\mid 1 \leq i + k < j + k \leq n\}\end{aligned}$$

This operator moves a substring of the permutation to another position. For some features of these five operators see table 1.

2.2 Fitness Distance Correlation and Point Quality

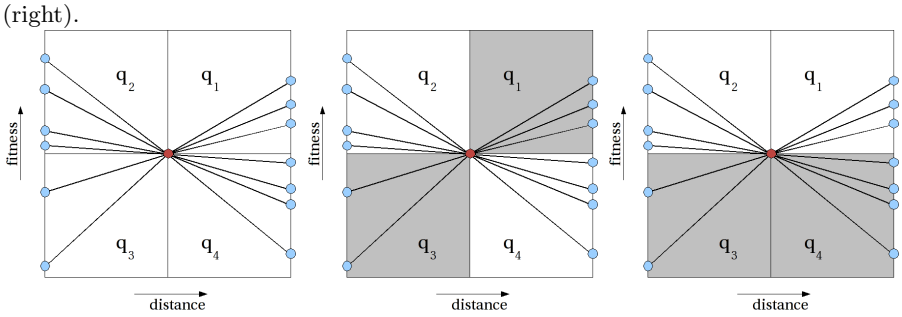
The FDC measures the correlation of fitness and distance to the nearest global optimum opt. Originally [6] it was applied to bit string coded optimization problems and in many cases ([3], [15]) this measure successfully predicts the performance of a genetic algorithm. The correlation coefficient for a sample M of size m is calculated as

$$\mathrm{FDC} = \frac{C_{\mathrm{FD}}}{s_f s_d} \quad \text{where} \quad C_{\mathrm{FD}} = \frac{1}{m} \sum_{i \in M} (f_i - \bar{f})(d(i, \mathrm{opt}_i) - \bar{d})$$

is the covariance of f and d and s_f, s_d, $\bar{f}$ and $\bar{d}$ are the standard deviations and means of f and d.

The FDC is a global measure as it calculates only one scalar value for a search space defined by problem instance and neighborhood operator. For analyzing and understanding the characteristics of a search space too many informations are lost. A possibility to remedy this shortage can be a local quality measure. Such a measure will provide more opportunities to study characteristics of a search space. Using the idea of FDC, with the two attributes fitness and distance, the neighborhood of any element in the search space is divided into four subsets. This can be illustrated with a 2-dimensional coordinate system and its four quadrants (figure 1).

Fig. 1. All neighbors of a solution can be pictured as elements in quadrants $q_1, \ldots, q_4$ (left). In minimization problems (like TSP) neighbors in q_1 and q_3 are supporting optimization (middle), but elements in q_2 and q_4 are deceptive. As walking through search space of a minimization problem, heuristics are favoring elements in q_3 and q_4 (right).



Definition 9. *For each element $\pi \in S_n$ with known fitness value f_π and distance $d(\pi, \varphi)$ to a nearest optimal solution the point quality is defined as*

$$\mathrm{PQ}(\pi) = \begin{cases} \dfrac{\left|\left\{\pi' | (d(\pi',\varphi) \leq d(\pi,\varphi) \wedge f_{\pi'} \leq f_\pi) \vee (d(\pi',\varphi) \geq d(\pi,\varphi) \wedge f_{\pi'} \geq f_\pi)\right\}\right|}{|\mathrm{OP}(\pi)|} & \textit{if min. problem} \\ \dfrac{\left|\left\{\pi' | (d(\pi',\varphi) \leq d(\pi,\varphi) \wedge f_{\pi'} \geq f_\pi) \vee (d(\pi',\varphi) \geq d(\pi,\varphi) \wedge f_{\pi'} \leq f_\pi)\right\}\right|}{|\mathrm{OP}(\pi)|} & \textit{if max. problem} \end{cases}$$

An element with a high PQ value will support the optimizing process. If all elements has a perfect PQ, then the FDC is pefect, too.

3 Distance Algorithms

If a problem is binary coded, the Hamming-Distance is used for calculating the pairwise distance. It can be calculated in $\Theta(n)$. In genetic algorithms (GA) One-Bit-Flipping is used as (mutation) operator, therefore Hamming-Distance represents the correct distance in search space.

In permutation based optimization algorithms many different operators are used. The calculation of the distance needs to reflect the neighborhood of the operator used – otherwise the results are incorrect for most combinations of operator and approximation algorithms. This was shown in [12] by Schiavinotto and Stützle and they also listed correct distance algorithms for five operators. A distance algorithm for permutations (only) has to calculate the distance to identity $\imath$. Because for calculating the pairwise distance the property $d_{\mathrm{OP}}(\pi, \varphi) = d_{\mathrm{OP}}(\pi^{-1} \cdot \varphi, \imath)$ can be used (also shown in [12]).

Neighbor Swap (NbrSwap): The minimal number of neighbor swaps to sort a permutation is the inversion number of the permutation. In [12] an $O(n^2)$ algorithm is presented. Here a faster $O(n \log n)$ and easy to implement algorithm is used. The idea is a balanced tree which is implemented as an array like a heap.

Element Swap (Swap): With a swap operation the length $|c_i|$ of a cycle c_i can be maximally reduced to a cycle c_i' of length $|c_i'| = |c_i| - 1$ and a second cycle of length $|c_i''| = 1$. The identity $\imath$ has n cycles of length 1. Therefore the element swap distance is $d_{\mathrm{Swap}}(\pi, \imath) = \sum_{c_i}(|c_i| - 1)$. This can be calculated in linear time $O(n)$.

Element Shift (Shift): To calculate the element shift distance all elements that stay on their place have to be detected. These elements form the *longest common subsequence* (LCS) or, in the case of permutations, the *longest increasing subsequence*. Here the algorithm of [13] is used and optimized for permutations. This algorithm needs $O(n \log n)$ time for calculating $d_{\mathrm{Shift}}(\pi, \imath) = n - \mathrm{LCS}(\pi, \imath)$. With van Emde Boas' data structure [16] a complexity of $O(n \log \log n)$ can be achieved.

Substring Reversal (SStrRev): This problem is known as *sorting by reversals* (SBR) and it is $\mathcal{NP}$-hard as proved by Caprara [2]. Therefore an easy to implement 2-approximation algorithm was used to calculate the distance [7] with $O(n^2)$ time complexity. The currently best known approximation guarantee is of 1.375 [1].

Substring Shift (SStrShift): This problem is known as *sorting by transpositions* (SBT) and it is presumed to be $\mathcal{NP}$-hard as well. For calculating $d_{\mathrm{SStrShift}}$ an easy to implement $O(n^2)$ greedy algorithm was implemented. This algorithm reduces the breakpoints between increasing strips in the permutation. The currently best known approximation guarantee is of 1.375 [4].

Operator	|OP|	Φ_{OP}	other operator names	dist.-alg. complexity
NbrSwap	$n-1$	$\frac{n(n-1)}{2}$	APEX, Swap	$O(n \log n)$
Swap	$\frac{n(n-1)}{2}$	$n-1$	2-cycle, element change, EX	$O(n)$
Shift	$(n-1)^2$	$n-1$	transposition, DSH	$O(n \log n)$
SStrRev	$\frac{n(n-1)}{2}$	$n-1$	2-opt, 2-change, reversal, INV	$O(n^2)$
SStrShift	$\frac{n^3-n}{6}$	$\lceil \frac{n+1}{2} \rceil$ (conjecture)	transposition	$O(n^2)$

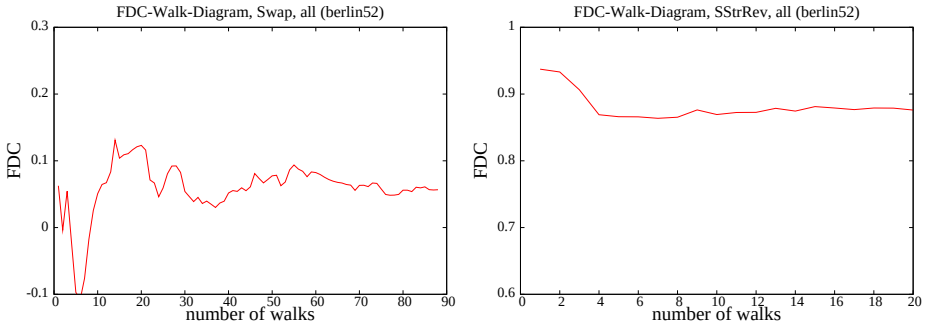
Table 1. Operator overview

4 Data Generation and Test Problems

To simulate the behavior of optimization heuristics the data is generated by (stochastic) hillclimb walks through the search space, because an important problem of random walk or random selection of elements is the small variance in distance to the optimum. The term *hillclimb* is here used in the sense of *go to a better solution* in every step.

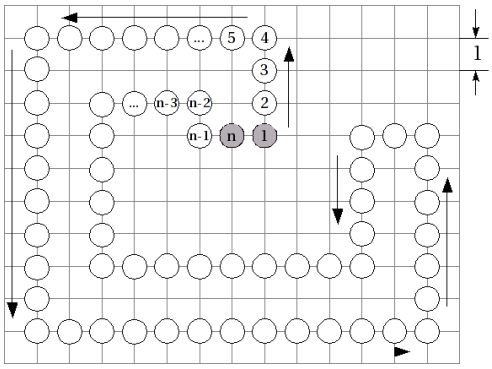
For calculating the FDC value a random element of the search space was selected to start a new hillclimb walk and then a random neighbor was calculated. If this neighbor had a better fitness value this element was selected for the next step. If no better neighbor was found after calculating n^2 elements of the neighborhood, the hillclimb walk was stopped and a new walk was started. After each walk the FDC value was calculated from all elements of all walks. If the difference between old and new calculated FDC value was less than 0.01 for three times the analysis was finished (figure 2).

Fig. 2. Two examples for the convergence of FDC value by calculating it through hillclimb walks.



In order to calculate the data source for PQ analysis the same procedure as for FDC is used. With the only difference, that in each step all n^2 neighbors are calculated. For these n^2 neighbors fitness and distance values are calculated and then the number of elements in quadrants $q_1, \ldots, q_4$ are counted.

Fig. 3. The 2-dimensional euclidian TSP test instances are imported from well known TSPLIB [11]. To have more nontrivial test instances with known optimum, we developed the @-type euclidian TSP. All points are arranged on an equidistant grid with distance 1. The inner and outer side length are calculated in dependence of the length n. The distance to next city is 1 and the distance between inner and outer path is 2. Therefore the identity $\imath$ is the optimum with minimal length n.



For asymmetric TSP instances the distances for identity was set to zero, therefore identity $\imath$ is optimal and has length zero. All other weights in the distance matrix were randomly chosen from $\{1, 2, \ldots, n^{\exp}\}$. The parameter exp for asymmetric benchmark generation is used to analyze the impact of the ratio of weights and n to the FDC value.

5 Results

Extensive experiments for the five operators and applicable permutation based codings are performed. The FDC value for symmetric TSP was calculated for nine instances over three orders of magnitude. For asymmetric TSP ten instances are analyzed.

For symmetric TSP the SStrRev operator produces the highest FDC values (less tour length, less distance to optimum). The SStrRev operator performs the minimal change of two edges. The second best operators are the Shift or SStrShift operator (table 2). For this two operators the PQ diagrams reveals, that the Shift operator outclasses the SStrShift operator. This operator leads to better fitness elements with less distance to optimum. For greater n the Shift or SStrShift operators are inducing better FDC values. At instance `pr1002` their FDC values reach the FDC value of the SStrRev operator. This effect should be analyzed in further work.

For asymmetric TSP the SStrRev operator isn't a good choice, this arises from the asymmetric nature of the problem class. Here the SStrShift operator produces best FDC values (table 3). An explanation is that moving good optimized stages (substrings) of a tour to a better position leads to a search space which is better to optimize. The exponent for randomly generated weights in distance matrix does not influence the FDC value. The FDC values of SStrRev and Shift operator are comparable, but as shown in figure 5, the PQ diagrams reveals that the Shift operator leads to better fitness values (top and middle diagram on the left side).

As a PQ diagram example the symmetric TSP (berlin52) instance is shown in figure 4. All five operators are plotted. Three operators, Swap, Shift and SStrRev, result in good fitness values. But only the SStrRev operator does reach the best fitness value and small distance. The Swap and Shift operator reach good fitness values, but these points are mostly far away from optimum.

The PQ diagrams for asymmetric TSP are shown in figure 5. The operators SStrShift and Shift lead to fitness values near to the optimum. The elements generated by SStrShift operator are closest to optimum but doesn't reach the optimum. This can be a symptom why asymmetric TSP are harder to optimize than symmetric TSP. FDC value gives a good hint for the selection of a convenient operator. But in some cases its oversimplifying nature is obscuring differences of the operator specific search space. In table 3 FDC values for Shift and SStrRev operator are comparable. But the PQ diagram in figure 5 (left side) shows, that SStrRev is clearly inferior, because worse fitness values are reached by hillclimb walks.

Fig. 4. An example for PQ analysis for all five operators (TSP berlin52). The diagrams are sorted from worst (top) to best (bottom) FDC value. On the left side the PQ is plotted over fitness (x-axis starting at optimum value) and on the right side over distance (x-axis ending at Φ_{OP}). The optimum located in the upper left corner of each diagram.

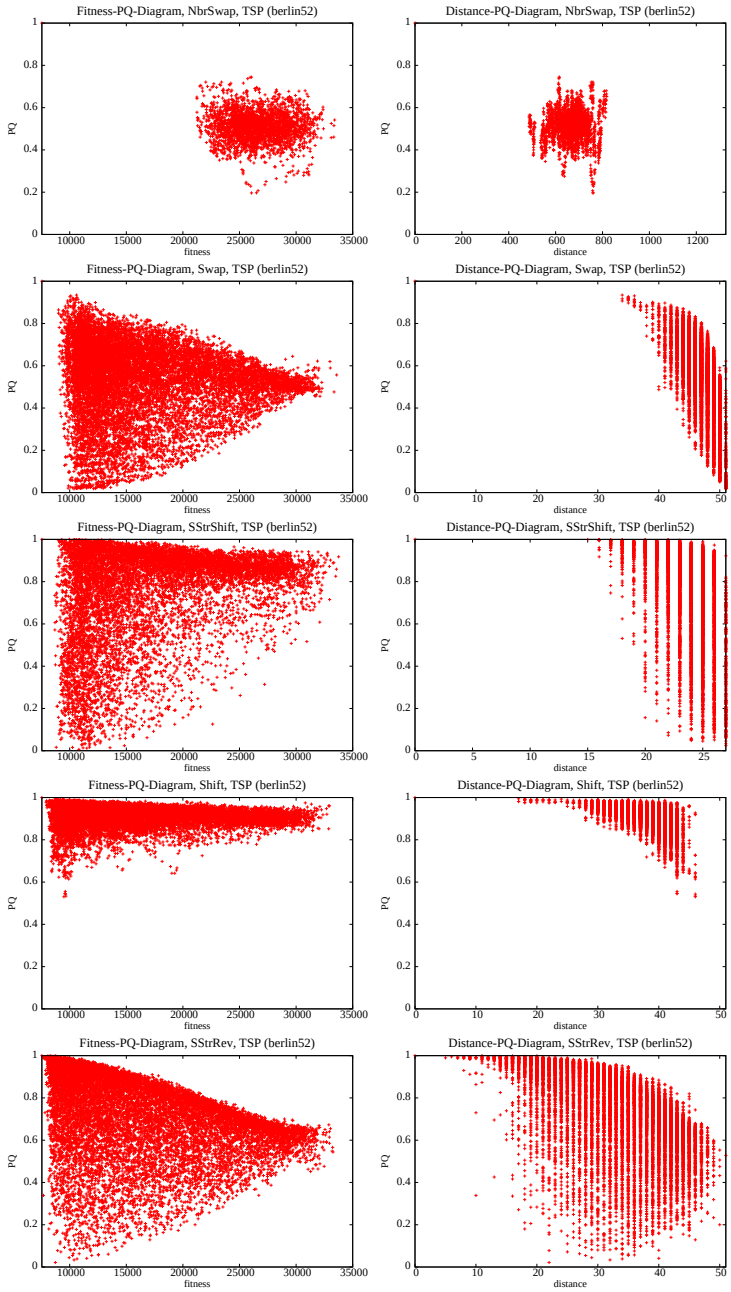


Fig. 5. Example PQ diagrams for AsymTSP with $n = 52$ and exp $= 1$. For asymmetric TSP instances the SStrShift operator produces the best FDC values. The SStrRev and Shift operators have approximately the same FDC values. But the fitness-PQ diagrams on the left side showing, that the Shift operator is outclassing the SStrRev operator.

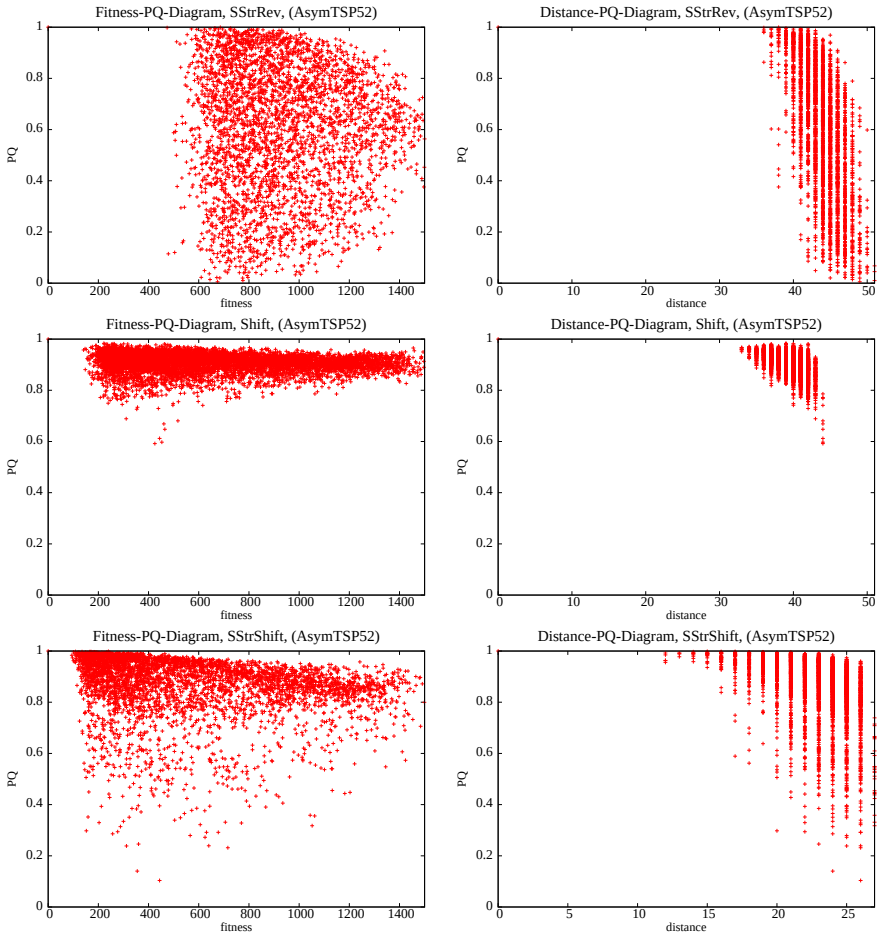


Table 2. As expected, best choice for symmetric TSP instances is the SStrRev operator. Data from hillclimb walks induce highest FDC value in all experiments. This operator performs minimal change (two edges) on TSP tour. The Shift and SStrShift operators are changing three edges whereas the Shift operators specific neighborhood is a subset of the SStrShifts neighborhood.

problem instance	size n	Nbr Swap	Swap	Shift	SStr Rev	SStr Shift
@	26	0.01	0.16	0.26	**0.86**	0.11
@	50	0.01	0.05	0.43	**0.87**	0.37
eil51	51	0.0	0.07	0.39	**0.89**	0.37
berlin52	52	0.05	0.06	0.47	**0.88**	0.24
@	100	0.02	0.08	0.52	**0.87**	0.65
ch130	130	−0.01	0.06	0.52	**0.88**	0.64
tsp225	225	0.04	0.07	0.61	**0.85**	0.75
@	250	0.12	0.11	0.60	**0.84**	0.8
pr1002	1002	0.0	0.06	0.74	**0.75**	**0.76**

Table 3. For all tested asymmetric TSP instances the SStrShift operator leads to highest FDC values. The SStrRev and Shift operators have comparable FDC values, but PQ analysis shows that the SStrRev operator is inferior.

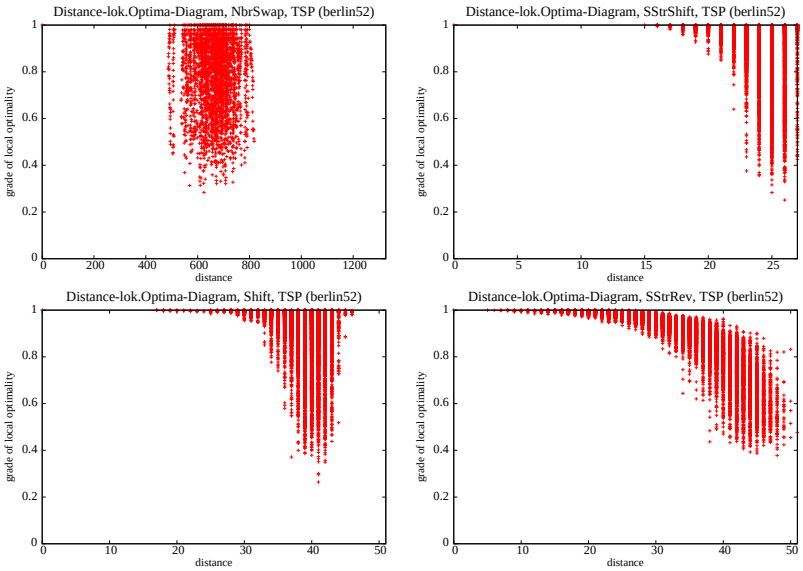
problem instance	size n	Nbr Swap	Swap	Shift	SStr Rev	SStr Shift
ATSP exp=1	26	0.03	0.06	0.36	0.30	**0.73**
ATSP exp=1	52	−0.02	0.01	0.27	0.22	**0.75**
exp=2	52	0.0	0.0	0.25	0.30	**0.74**
exp=3	52	0.02	0.02	0.32	0.27	**0.71**
exp=5	52	0.0	−0.01	0.24	0.27	**0.74**
ATSP exp=1	100	0.0	0.03	0.25	0.23	**0.71**
exp=5	100	0.01	0.02	0.21	0.19	**0.72**
ATSP exp=1	200	0.0	−0.01	0.14	0.18	**0.72**
ATSP exp=1	500	0.0	−0.02	−0.04	0.19	**0.67**
ATSP exp=1	1000	−0.03	0.0	−0.05	0.20	**0.60**

To analyze the distribution of local optima in search space and examine the difference between good and bad performing operators, the proportion of neighbors with worse fitness values (elements in q_1 and q_2) are plotted over distance. As can be seen in figure 6, the local optima with best performing operator SStrRev are heaping near the global optima. Local optima for operator Shift and SStrShift are uniformly distributed in search space. Local optima for operator NbrSwap are heaping at $\Phi_{\mathrm{OP}}/2$, but not in the direction of global optima.

6 Conclusion

To extensively analyze fitness distance correlation, we implemented algorithms for measuring the real (minimal) distance between permutations for five operators, as demanded by [12]. Instances for symmetric and asymmetric TSP were analyzed. The best operator for symmetric TSP is the well known SStrRev operator. For asymmetric TSP the SStrShift operator leads to highest FDC values. We proposed PQ as an extension of FDC to measure local quality in search space. PQ diagrams can visualize additional characteristics of search space. The PQ diagrams can help to identify good operators even if the FDC values are approximately the same (examples shown in figure 4 and 5).

Fig. 6. Here the grade of local optimality is plotted (TSP berlin52). The y-axis indicates the proportion of neighbors which appear in quadrant q_1 or q_2. If all neighbors are in these quadrants a local optima is reached (the end of a hillclimb walk). It's visible how the local optima are distributed in search space.



In future work we will use our framework to analyze further permutation based optimization problems, like bin packing or quadratic assignment problem (QAP). Additionally, more problem specific operators and their distance algorithms will be implemented to understand, what makes a search space easy to optimize and to identify characteristics of such operators. Possibly another way to gain knowledge of search spaces is to analyze PQ diagrams of artificial and misleading functions, like the so called ridge functions [9].

References

1. Berman, P., Hannenhalli, S., Karpinski, M. 1.375-Approximation Algorithm for Sorting by Reversals, Electronic Colloquium on Computational Complexity (ECCC) vol. 8, num. 47 (2001)
2. Caprara, A. Sorting by Reversals is Difficult, Proceedings of the first annual international conference on Computational molecular biology 75–83 (1997)
3. Collard, P., Gaspar, A., Clergue M., Escazut, C. Fitness Distance Correlation, as Statistical Measure of Genetic Algorithm Difficulty, Revisited, In Proceedings of the European Conference on Artificial Intelligence 650–654 (1998)
4. Elias, I., Hartman, T. A 1.375-Approximation Algorithm for Sorting by Transpositions, IEEE/ACM Trans. Comput. Biol. Bioinformatics vol. 3, num. 4, 369–379 (2006)
5. Fonlupt, C., Robilliard, D., Preux P. Fitness Landscape and the Behavior of Heuristics, in Evolution Artificielle 97 (1997)
6. Jones, T., Forrest, S. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms, Proc. of the 6th International Conference on Genetic Algorithms 184–192 (1995)
7. Kececioglu, J., Sankoff, D. Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement, Algorithmica vol. 13, num. 1/2 180–210 (1995)
8. Merz, P., Freisleben, B. Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem, IEEE Trans. Evolutionary Computation vol. 4, num. 4 337–352 (2000)
9. Quick, R.J., Rayward-Smith, V.J., Smith, G.D. Fitness Distance Correlation and Ridge Functions, Proc. of the International Conference on PPSN V 77–86, (1998)
10. Reeves, C.R., Landscapes, Operators and Heuristic Search, Annals of Operations Research vol. 86 473–490 (1999)
11. Reinelt G. TSPLIB - A Traveling Salesman Problem Library, INFORMS Journal on Computing vol. 3, num. 4 376–384 (1991)
12. Schiavinotto, T., Stützle, T. A Review of Metrics on Permutations for Search Landscape Analysis, Comput. Oper. Res. vol. 34 num. 10 3143–3153 (2007)
13. Szymanski, T., Hunt, J., A fast Algorithm for Computing Longest Common Subsequences, Commun. ACM, vol. 20, num 5 350–353 (1977)
14. Thierens, D. Predictive Measures for Problem Representation and Genetic Operator Design, (2002)
15. Tomassini, M., Vanneschi, L., Collard, Ph., Clergue, M. A Study of Fitness Distance Correlation as a Difficulty Measure in Genetic Programming Journal Evol. Comput. vol. 13, num. 2 213–239 (2005)
16. P. van Emde Boas Preserving Order in a Forest in Less Than Logarithmic Time and Linear Space Inf. Process. Lett. vol. 6, num. 3 80–82 (1977)