Parameterized Runtime Analyses of Evolutionary Algorithms for the Planar Euclidean Traveling Salesperson Problem

Andrew M. Sutton sutton@cs.colostate.edu
Department of Computer Science, Colorado State University, Fort Collins, CO, USA

Frank Neumann frank.neumann@adelaide.edu.au
Optimisation and Logistics, School of Computer Science, University of Adelaide, Adelaide, Australia

Samadhi Nallaperuma samadhi.nallaperuma@adelaide.edu.au
Optimisation and Logistics, School of Computer Science, University of Adelaide, Adelaide, Australia

Abstract

Parameterized runtime analysis seeks to understand the influence of problem structure on algorithmic runtime. In this paper, we contribute to the theoretical understanding of evolutionary algorithms and carry out a parameterized analysis of evolutionary algorithms for the Euclidean traveling salesperson problem (Euclidean TSP).

We investigate the structural properties in TSP instances that influence the optimization process of evolutionary algorithms and use this information to bound their runtime. We analyze the runtime in dependence of the number of inner points k.

In the first part of the paper, we study a $(\mu + \lambda)$ EA in a strictly black box setting and show that it can solve the Euclidean TSP in expected time $O\big(n \cdot A(\epsilon) \cdot \max\big\{(\mu/\lambda) \cdot n^2, 1\big\} + (\mu/\lambda) \cdot n^{4k}(2k-1)!\big)$ where A is a function of the minimum angle ϵ between any three points. Based on insights provided by the analysis, we improve this upper bound by introducing a mixed mutation strategy that incorporates both 2-opt moves and permutation jumps. This strategy improves the upper bound to $O\big(n \cdot A(\epsilon) \cdot \max\big\{(\mu/\lambda) \cdot n^2, 1\big\} + (\mu/\lambda) \cdot n^{2k}(k-1)!\big)$.

In the second part of the paper, we use the information gained in the analysis to incorporate domain knowledge to design two fixed-parameter tractable (FPT) evolutionary algorithms for the planar Euclidean TSP. We first develop a $(\mu + \lambda)$ EA based on an analysis by Theile (2009) that solves the TSP with k inner points in $O(\max\{2^k k^2 n^2 \lambda^{-1}, n\})$ generations with probability $1 - e^{-\Omega(n)}$. We then design a $(\mu + \lambda)$ EA that incorporates a dynamic programming step into the fitness evaluation. We prove that a variant of this evolutionary algorithm using 2-opt mutation solves the problem after $O(\max\{(k-2)!k^{2k-2}\lambda^{-1}, 1\})$ steps in expectation with a cost of $O(nk)$ for each fitness evaluation.

Keywords

Evolutionary algorithms, runtime analysis, parameterized analysis, combinatorial optimization.

1 Introduction

In many real applications, the inputs of an NP-hard combinatorial optimization problem may be structured or restricted in such a way that it becomes tractable to solve

in practice despite having a worst-case exponential time bound. Parameterized analysis seeks to address this by expressing algorithmic runtime in terms of an additional hardness parameter that isolates the source of exponential complexity in the problem structure. In this paper, we study the application of evolutionary algorithms (EAs) to the Euclidean Traveling Salesperson Problem (TSP) and consider the runtime of such algorithms as a function of both problem size and a further parameter that influences how hard the problem is to solve by an EA.

1.1 The Euclidean Traveling Salesperson Problem

Iterative heuristic methods (such as local search and evolutionary algorithms) that rely on the exchange of a few edges such as the well-known 2-opt (or 2-change) operator are popular choices for solving large scale TSP instances in practice. This is partly due to the fact that they have a simple implementation and typically perform well empirically. However, for these algorithms, theoretical understanding still remains limited. Worst-case analyses demonstrate the existence of instances on which the procedure can be inefficient. Chandra, Karloff, and Tovey [6], building on unpublished results due to Lueker, have shown that local search algorithms employing a k-change neighborhood operator can take exponential time to find a locally optimal solution. Even in the Euclidean case, Englert, Röglin, and Vöcking [15] have recently shown that a local search algorithm employing inversions can take worst-case exponential time to find tours that are locally optimal.

If the search operator is restricted to specialized 2-opt moves that remove only edges that intersect in the plane, van Leeuwen and Schoone [36] proved that a tour that has no such planar intersections can be reached in $O(n^3)$ moves, even if a move introduces further intersecting edges. If the vertices are distributed uniformly at random in the unit square, Chandra, Karloff, and Tovey [6] showed that the expected time to find a locally optimal solution is bounded by $O(n^{10} \log n)$. More generally, for so-called ϕ-perturbed Euclidean instances, Englert, Röglin, and Vöcking [15] proved that the expected time to find a locally optimum solution is bounded by $O(n^{4+1/3} \log(n\phi)\phi^{8/3})$. These results also imply similar bounds for simple ant colony optimization algorithms as shown in [20].

To allow for a deeper insight into the relationship between problem instance structure and algorithmic runtime, we appeal in this paper to the theory of parameterized complexity [11, 16]. Rather than expressing the runtime solely as a function of problem size, parameterized analysis decomposes the runtime into further parameters that are related to the structure of instances. The idea is to find parameters that partition off the combinatorial explosion that leads to exponential runtimes [12].

In the context of TSP, a number of parameterized results currently exist. Deĭneko et al. [8] showed that, if a Euclidean TSP instance with n vertices has k vertices interior to the convex hull, there is a dynamic programming algorithm that can solve the instance in time bounded by $g(k) \cdot n^{O(1)}$ where g is a function that depends only on k. This means that this parameterization belongs to the complexity class FPT, the class of parameterized problems that are considered fixed-parameter tractable. Of course, membership in FPT depends strongly on the parameterization itself. For example, the problem of searching the k-change neighborhood for metric TSP is hard for W[1] due to Marx [23]. FPT $\subseteq$ W[1], but the containment is conjectured to be proper [16].

1.2 Computational Complexity of Evolutionary Algorithms

Initial studies on the computational complexity of evolutionary algorithms consider their runtime on classes of artificial pseudo-Boolean functions [14, 17, 18, 39]. The goal of these studies is to consider the impact of the different modules of an evolutionary algorithm and to develop new methods for their analysis. This early work was instrumental in establishing a rigorous understanding of the behavior of evolutionary algorithms on simple functions, for identifying some classes of problems that simple EAs can provably solve in expected polynomial time [14], and for disproving widely accepted conjectures (e.g., that evolutionary algorithms are always efficient on unimodal functions [13]).

More recently, classical polynomial-time problems from combinatorial optimization such as minimum spanning trees [26, 27] and shortest paths [31, 10, 4] have been considered. In this case, one does not hope to beat the best problem-specific algorithms for classical polynomial solvable problems. Instead, these studies provide interesting insights into the search behavior of these algorithms and show that many classical problems are solved by general-purpose algorithms such as evolutionary algorithms in expected polynomial time.

Research on NP-hard combinatorial optimization problems such as makespan scheduling, covering problems, and multi-objective minimum spanning trees [25, 38] show that evolutionary algorithms can achieve good approximations for these problems in expected polynomial time. In the case of the TSP, Theile [35] has proved that a $(\mu+1)$ EA based on dynamic programming can exactly solve the TSP in at most $O(n^3 2^n)$ steps when μ is allowed to be exponential in n. For a comprehensive presentation of the different results that have been achieved see, e.g., the recent text of Neumann and Witt [28].

Algorithmic runtime on NP-hard problems can be studied in much sharper detail from the perspective of parameterized analysis, and this has only recently been started in theoretical work on evolutionary algorithms. Parameterized results have been obtained for the vertex cover problem [22], the problem of computing a spanning tree with a maximal number of leaves [21], variants of maximum 2-satisfiability [32], and makespan scheduling [34].

1.3 Our Results

In this paper, we carry out a parameterized complexity analysis for evolutionary algorithms on the planar Euclidean TSP. Specifically, using the parameterization of Deĭneko et al. [8], we prove upper bounds on the expected time for certain evolutionary algorithms to solve planar TSP instances as a function of n, the number of points in the instance, and of k, the number of *inner points*, i.e., the points that lie on the interior of the convex hull of the planar TSP instance. Our motivation is to use this parameterization from the classical algorithms community to understand the influence of the number of inner points on the runtime of a black box evolutionary algorithm, and then to design evolutionary algorithms that exploit this problem structure to obtain fixed-parameter tractable runtimes.

In the first part of the paper, we investigate evolutionary algorithms applied to the planar Euclidean TSP entirely from a black box perspective. We consider the parameterized runtime bounds on algorithms that do not have access to any concrete problem information. Specifically, in the black box scenario, algorithms can only access the fitness value of each candidate solution as measured by tour length. Our results are for the $(\mu + \lambda)$ EA which operates on a population of μ permutations (candidate

Hamiltonian cycles) and produces λ offspring in each generation using a mutation operator based on 2-opt. This analysis provides further insights into the optimization process that allows us to design a mixed mutation operator that uses both 2-opt moves and permutation jumps and improves the upper bound on the expected runtime of the $(\mu + \lambda)$ EA. This part of the paper is an extension of a paper presented at AAAI 2012 [33].

In the second part of the paper, we leave the black box setting and consider evolutionary algorithms that have access to both fitness values and the ordering of the points that lie on the convex hull. This ordering can be precomputed in polynomial time. Using this approach, we design two fixed-parameter tractable (FPT) evolutionary algorithms for the parameterization of Deĭneko et al. This first algorithm builds on the previous study of Theile [35] and the general framework for evolutionary algorithms for dynamic programming presented in [9]. It works with a large population where each individual can be seen as an entry of the dynamic programming table of the classical approach. The population size grows exponentially with the number of inner points and we assume that this result is mainly of theoretical interest.

While the first FPT algorithm is interesting mostly from a theoretical perspective, our second FPT algorithm is highly practical as a heuristic search algorithm. Instead of searching for a permutation of the n given points, we apply evolutionary search only to find a good permutation of the k inner points. The outer points (lying on the convex hull) have to appear in cycling order γ according to the convex hull in any optimal solution. For a fixed permutation x on the inner points, the best possible tour according to γ and x can be computed in time $O(kn)$ using dynamic programming. We make use of these ideas and propose to use evolutionary algorithms that work with permutations of the inner points. Our theoretical investigations show that commonly used mutation operators for permutations such as inversions, jumps, and exchanges yield fixed-parameter evolutionary algorithms. Furthermore, our experimental investigations confirm that this approach leads to significantly better results if the number of inner points is not too large. This part of the paper is an extension of a paper presented at CEC 2013 [24]

This paper is organized as follows. In Section 2 we introduce the problem, state some preparatory material, and discuss parameterized analysis. In Section 3 we study the runtime of a $(\mu + \lambda)$ EA on Euclidean TSP instances in a black box setting. We first consider the time it takes for such algorithms to solve instances whose points lie in convex position, i.e., have no inner points. We then prove rigorous runtime bounds for the algorithms as a function of the number of inner points in an instance. In Section 4 we design two evolutionary algorithms with access to some instance-specific information and prove that they are fixed-parameter tractable EAs, i.e., they solve the planar TSP with k inner points in expected time $g(k) \cdot n^{O(1)}$ where g depends only on k. We conclude the paper in Section 5.

2 Preliminaries

Let V be a set of n points in the plane labeled as $[n] = \{1, \ldots, n\}$ such that no three points are collinear. We consider the complete, weighted Euclidean graph $G = (V, E)$ where E is the set of all 2-sets from V. The weight of an edge $\{u, v\} \in E$ is equal to $d(u, v)$: the Euclidean distance separating the points. The goal is to find a set of n edges of minimum weight that form a Hamiltonian cycle in G. A candidate solution of the TSP is a permutation $x : V \to V$. We will sometimes associate a permutation x with its *linear form*, which is simply the length-n sequence $(x(1), x(2), \ldots, x(n))$. The

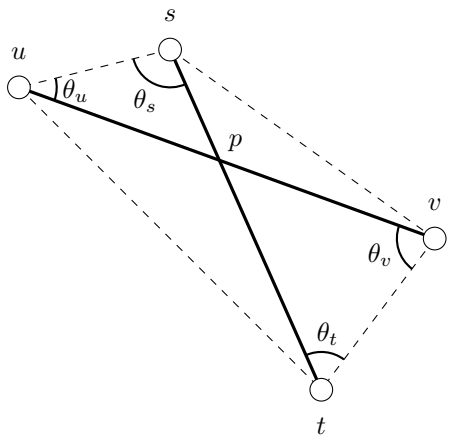

Figure 1: Edges that intersect at a point p form the diagonals of a convex quadrilateral in the plane. Interior edges define nondegenerate triangles with given angles.

Hamiltonian cycle in G induced by a permutation x is the set of n edges

$$C(x) = \{\{x(1), x(2)\}, \{x(2), x(3)\}, \ldots, \{x(n-1), x(n)\}, \{x(n), x(1)\}\}.$$

The optimization problem is to find a permutation x which minimizes the fitness function

$$f(x) = \sum_{\{u,v\} \in C(x)} d(u, v). \tag{1}$$

Geometrically, it will often be convenient to consider an edge $\{u, v\}$ as the unique planar line segment with end points u and v. We say a pair of edges $\{u, v\}$ and $\{s, t\}$ *intersect* if they cross at a point in the Euclidean plane. An important observation, which we state here without proof, is that any pair of intersecting edges form the diagonals of a convex quadrilateral in the plane (see Figure 1).

Proposition 1. *If $\{u, v\}$ and $\{s, t\}$ intersect at a point p, they form the diagonals of a convex quadrilateral described by points u, s, v, and t. Hence edges $\{s, u\}$, $\{s, v\}$, $\{t, v\}$ and $\{t, u\}$ form a set of edges that mutually do not intersect.*

Definition 1. *A tour $C(x)$ is called* intersection-free *if it contains no pairs of edges that intersect.*

The convex hull of V is the smallest convex set containing V. A point $v \in V$ is called an *inner point* if v lies in the interior of the convex hull of V. We denote as $\mathsf{Inn}(V) \subset V$ the set of inner points of V. A point $v \in V \setminus \mathsf{Inn}(V)$ is called an *outer point*. The set of such points we denote by $\mathsf{Out}(V)$. Hence $\mathsf{Inn}(V)$ and $\mathsf{Out}(V)$ partition V.

Note that every tour $C(x)$ for all permutations x on V corresponds to a set of edges that describe a closed polygon in the plane. The *covertex class* of a polygon is the class of all polygons that have the same vertex set as the given polygon. The following two theorems are due to Quintas and Supnick [29].

Theorem 1. *A planar polygon that is shortest of its covertex class and whose vertices are noncollinear cannot intersect itself.*

Theorem 2. *A planar polygon that is shortest of its covertex class and whose vertices are noncollinear must contain the vertices on the boundary of its convex hull in their cyclic order.*

We call the unique cyclic order that arises from the points in $\mathsf{Out}(V)$ hull-order. We say a permutation x respects hull-order if any two points in the subsequence of x induced by $\mathsf{Out}(V)$ are consecutive in x only if they are adjacent along the boundary of the convex hull of V. Thus, Theorem 2 states that, if x is a minimum of f, then x respects hull-order. We will actually need a slightly stronger result than the one provided by the claim of Theorem 2. This is captured by the following.

Lemma 1. *If $C(x)$ is intersection-free, then x respects hull-order.*

Lemma 1 follows from arguments in the proof of Theorem 2 above. In particular, Quintas and Supnick [29] show that if a polygon does not contain the vertices on the boundary of the convex hull in their cyclic order, then the polygon must intersect itself (and hence, by Theorem 1, cannot be optimal). The claim of the lemma follows then by contraposition.

2.1 Parameterized Analysis

Parameterized complexity theory is an extension to traditional computational complexity theory in which the analysis of hard algorithmic problems is decomposed into parameters of the problem input. This approach illuminates the relationship between hardness and different aspects of problem structure because it often isolates the source of exponential complexity in NP-hard problems.

A parameterization of a problem is a mapping of problem instances into the set of natural numbers. We are interested in expressing algorithmic complexity in terms of both problem size and the extra parameter. Formally, let L be a language over a finite alphabet Σ. A *parameterization* of L is a mapping $\kappa : \Sigma^* \to \mathbb{N}$. The corresponding *parameterized problem* is the pair (L, κ).

For a string $x \in \Sigma^*$, let $k = \kappa(x)$ and $n = |x|$. An algorithm deciding $x \in L$ in time bounded by $g(k) \cdot n^{O(1)}$ is called a *fixed-parameter tractable* (or fpt-) algorithm for the parameterization κ. Here $g : \mathbb{N} \to \mathbb{N}$ is an arbitrary but computable function. Similarly, an algorithm that decides a parameterized problem (L, κ) in time bounded by $n^{g(k)}$ is called an XP-algorithm.

When working with the runtime of randomized algorithms such as evolutionary algorithms, one is often interested in the random variable T which somehow measures the number of generations the algorithm must take to decide a parameterized problem. A randomized algorithm with *expected* optimization time $E(T) \leq g(k) \cdot n^{O(1)}$ (respectively, $E(T) \leq n^{g(k)}$) is a randomized fpt-algorithm (respectively, XP-algorithm) for the corresponding parameterization κ.

In the case of the Euclidean TSP on a set of points V, we want to express the runtime complexity as a function of n and k where $n = |V|$ and k is the number of *inner points*: the vertices that lie in the interior of the convex hull of V. For the corresponding optimization problem, we are interested in the runtime until the optimal solution is located.

3 Black Box EAs

We begin by investigating the parameterized runtime behavior of simple randomized search heuristics on the Euclidean TSP purely from a black box perspective. From this perspective, we assume that the heuristics do not have any access to problem information other than the fitness of an individual as measured by the tour length induced by the corresponding permutation.

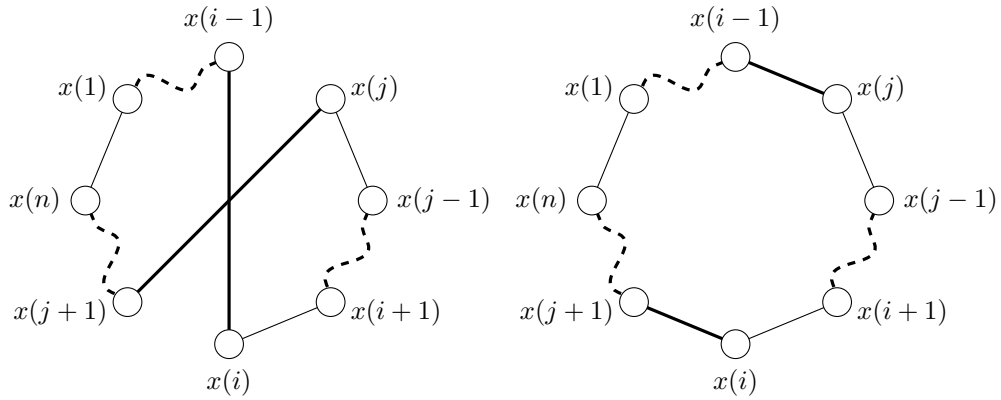

Figure 2: The effect of the inversion σ^{I}_{ij} operation on a Hamiltonian cycle. Inverting a subsequence in the permutation representation corresponds to a 2-opt (or 2-change) move in which a pair of edges in the current Hamiltonian cycle are replaced by a pair of edges not in the current cycle.

3.1 Permutation operators

Randomized search heuristics that are applied to solving TSP instances are usually designed to iteratively search the space of permutations in order to minimize the fitness function defined in Equation (1). Each permutation corresponds to a particular Hamiltonian cycle in the graph. To move through the space of candidate solutions, move or mutation operators are often constructed based on some kind of elementary operations on the set of permutations on V. For the black box algorithms, we will consider two such operations: *inversions* and *jumps* which we define as follows.

Definition 2. *The* inversion *operation σ^{I}_{ij} transforms permutations into one another by segment reversal in their linear forms.*

A permutation x is transformed into a permutation $\sigma^{\mathrm{I}}_{ij}[x]$ by inverting the subsequence of the linear form of x from position i to position j where $1 \leq i < j \leq n$.

$$x = (x(1), \ldots, x(i-1), x(i), x(i+1), \ldots, x(j-1), x(j), x(j+1), \ldots, x(n)),$$
$$\sigma^{\mathrm{I}}_{ij}[x] = (x(1), \ldots, x(i-1), x(j), x(j-1), \ldots, x(i+1), x(i), x(j+1), \ldots, x(n)).$$

In the space of Hamiltonian cycles, the permutation inversion operation is essentially identical to the well-known 2-opt (or 2-change) operation for TSP. The usual effect of the inversion operation is to delete the two edges $\{x(i-1), x(i)\}$ and $\{x(j), x(j+1)\}$ from $C(x)$ and reconnect the tour $C(\sigma^{\mathrm{I}}_{ij}[x])$ using edges $\{x(i-1), x(j)\}$ and $\{x(i), x(j+1)\}$ (see Figure 2). Here and subsequently, we consider arithmetic on the indices to be modulo n, i.e., $1-1=n$ and $n+1=1$. Since the underlying graph G is undirected, when $(i,j) = (1,n)$, the operation has no effect because the current tour is only reversed. There is also no effect when $(i,j) \in \{(2,n), (1,n-1)\}$. In this case, it is straightforward to check that the edges removed from $C(x)$ are equal to the edges replaced to create $C(\sigma^{\mathrm{I}}_{ij}[x])$.

Definition 3. *The* jump *operation σ^{J}_{ij} transforms permutations into one another by position shifts in their linear form. A permutation x is transformed into a permutation $\sigma^{\mathrm{J}}_{ij}[x]$ by moving the element in position i in the linear form of x into position j in the linear form of $\sigma^{\mathrm{J}}_{ij}[x]$ while*

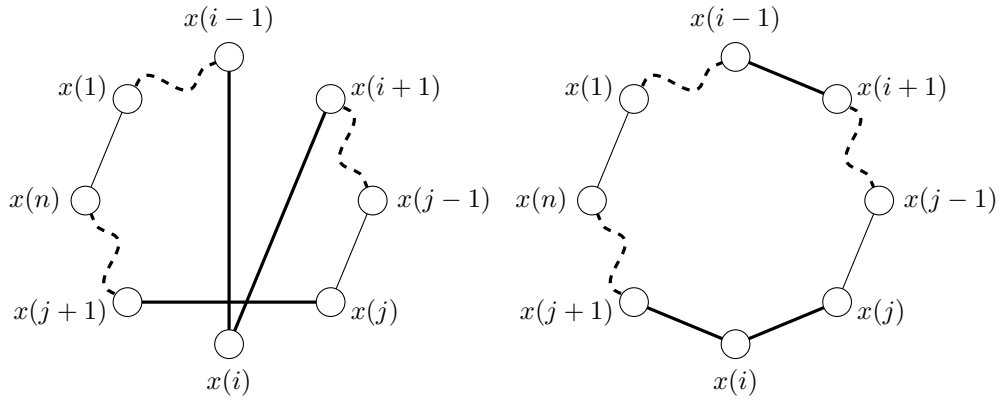

Figure 3: The effect of the jump operation σ_{ij}^{J} on a Hamiltonian cycle.

the other elements between position i and position j are shifted in the appropriate direction. Without loss of generality, suppose $i < j$. Then,

$$x = (x(1), \ldots, x(i-1), x(i), x(i+1), \ldots, x(j-1), x(j), x(j+1), \ldots, x(n)),$$
$$\sigma_{ij}^{\mathrm{J}}[x] = (x(1), \ldots, x(i-1), x(i+1), \ldots, x(j-1), x(j), x(i), x(j+1), \ldots, x(n)).$$

Since $\sigma_{i(i+1)}^{\mathrm{J}}$ and $\sigma_{(i+1)i}^{\mathrm{J}}$ have the same effect, there are $n(n-1)-(n-1) = (n-1)^2$ unique jump operations. The *jump* operator σ_{ij}^{J} was used by Scharnow, Tinnefeld and Wegener [31] in the context of runtime analysis of evolutionary algorithms on permutation sorting problems.

3.2 The $(\mu + \lambda)$ EA

We study the $(\mu+\lambda)$ EA, a population-based evolutionary algorithm that employs only mutation and selection as search operators. The $(\mu + \lambda)$ EA, outlined in Algorithm 1, employs uniform selection for reproduction (cf. line 5 of Algorithm 1), and truncation selection for replacement (see Function 2).

Algorithm 1: The $(\mu + \lambda)$ EA.

```
1 Choose a multiset P of μ random permutations on V;
2 repeat forever
3     P' ← {};
4     repeat λ times
5         choose x uniformly at random from P;
6         y ← mutate(x);
7         P' ← P' ⊎ {y};
8     P ← select(P ⊎ P') ;
```

When designing evolutionary algorithms, it is usually desirable to have a mutation operator that is capable of making sufficiently large changes to candidate solutions. Mutation strategies for combinatorial structures such as paths and permutations can in

Function 2: `select` $(P \uplus P')$

1 Construct a list L of all permutations in $P \uplus P'$ sorted by fitness (ties are broken by first preferring offspring, then any remaining ties are broken uniformly at random);
2 **return** the fittest μ individuals L;

some sense simulate the behavior of simple bitstring mutation by aggregating the mutation operator and applying it a number of times governed by a Poisson process [31]. For the TSP, we introduce here the mutation operator called `2-opt-mutation` (see Function 3) that applies to an individual permutation a sequence of $s + 1$ random inversion operations where s is drawn from a Poisson distribution with unit expectation.

Function 3: `2-opt-mutation` (x)

1 $y \leftarrow x$;
2 draw s from a Poisson distribution with unit expectation;
3 perform $s + 1$ random inversion operations on y;
4 **return** y;

If we restrict the mutation operation to a single inversion move and set $\mu = \lambda = 1$, the resulting algorithm is *randomized local search* (RLS), which is simply a randomized hill-climber in the space of permutations using 2-opt moves. RLS, illustrated in Algorithm 4, operates by iteratively applying random inversion operations to a permutation in order to try and improve the fitness of the corresponding tours. Unlike the $(\mu + \lambda)$ EA, RLS can only generate immediate inversion neighbors so it can become indefinitely trapped in local optima (on the other hand, the $(\mu + \lambda)$ EA can eventually escape the local optima, but possibly only after a very long time).

Algorithm 4: Randomized Local Search (RLS).

1 Choose a random permutation x on V;
2 **repeat** *forever*
3 choose a random distinct pair of elements (i, j) from $[n]$;
4 $y \leftarrow \sigma^{\mathrm{I}}_{ij}[x]$;
5 **if** $f(y) \leq f(x)$ **then** $x \leftarrow y$

Evolutionary algorithms are simply computational methods that rely on random decisions so we consider them here as special cases of *randomized algorithms*. To analyze the running time of such an algorithm, we examine the sequence of best-so-far solutions it discovers during execution $(x^{(1)}, x^{(2)}, \ldots, x^{(t)}, \ldots)$ as an infinite stochastic process where $x^{(t)}$ denotes the best permutation (in terms of fitness) in the population at iteration t. The goal of the runtime analysis is to study the random variable that equals the first time t when $x^{(t)}$ is a candidate solution of interest (for example, an optimal solution). The *optimization time* of a randomized algorithm is the random variable

$$T = \inf\{t \in \mathbb{N} : f(x^{(t)}) \text{ is optimal}\}. \tag{2}$$

In the case of the $(\mu + \lambda)$ EA, this corresponds to the number of generations (iterations

of the mutation, evaluation, selection process) that occur before an optimal solution has been introduced to the population. This is somewhat distinct from the traditional measure of the number of explicit calls to the fitness function [2, 28]. However, in the case of the $(\mu + \lambda)$ EA, this measurement can be obtained from T by $T_f = \mu + \lambda T$, since we need μ fitness function calls to evaluate the initial population and each generation requires evaluating an additional λ individuals. We discuss this further in section 3. In this paper, we will estimate the *expected optimization time* of the $(\mu + \lambda)$ EA. This quantity is calculated as $E(T)$, the expectation of T. However, we will also report the bounds on $E(T_f)$, the expected number of fitness evaluations needed until a solution is found.

Definition 4. *Let α be an indicator function defined on permutations of V as*

$$\alpha(x) = \begin{cases} 1 & \text{if } C(x) \text{ contains intersections;} \\ 0 & \text{otherwise.} \end{cases}$$

Definition 5. *Let β be an indicator function defined on permutations of V as*

$$\beta(x) = \begin{cases} 1 - \alpha(x) & \text{if } f(x) \text{ non-optimal} \\ 0 & \text{otherwise.} \end{cases}$$

The random variable corresponding to optimization time can be expressed as the infinite series

$$T = \sum_{t=1}^{\infty} \big(\alpha(x^{(t)}) + \beta(x^{(t)})\big). \tag{3}$$

In order to characterize the behavior of evolutionary algorithms and express their expected runtime in terms of the number of points n and the number of inner points k, we analyze a Markov chain generated by the algorithm. We construct the Markov chain as follows. Given a point set V, each permutation x on V that corresponds to a tour $C(x)$ that is non-optimal is a unique state in the Markov chain. Finally, every permutation that corresponds to an optimal tour in V is associated with a single absorbing state *opt*. We then bipartition the state space (minus *opt*) into two sets S_α and S_β where

$$S_\alpha = \{x : C(x) \text{ contains intersecting edges}\},$$

and

$$S_\beta = \{x : C(x) \text{ is intersection-free }\} \setminus \{opt\}.$$

Note that $opt \notin S_\alpha$ since the optimal tour cannot contain intersections. In terms of the Markov chain, the optimization time T is the first hitting time of the state *opt*. We will need the following two preparatory lemmas.

Lemma 2. *If there are values $0 < p < 1$ and $c > 0$ with p and c constant with respect to t (but not necessarily n), such that for any $x \in S_\alpha$, the transition probability from x to some x' with $f(x') < f(x) - c$ is bounded below by p, then*

$$\mathbb{E}\left(\sum_{t=1}^{\infty} \alpha(x^{(t)})\right) \leq p^{-1} n (d_{max} - d_{min})/c,$$

where d_{max} and d_{min} are the maximum and minimum distances between any two points in V, respectively.

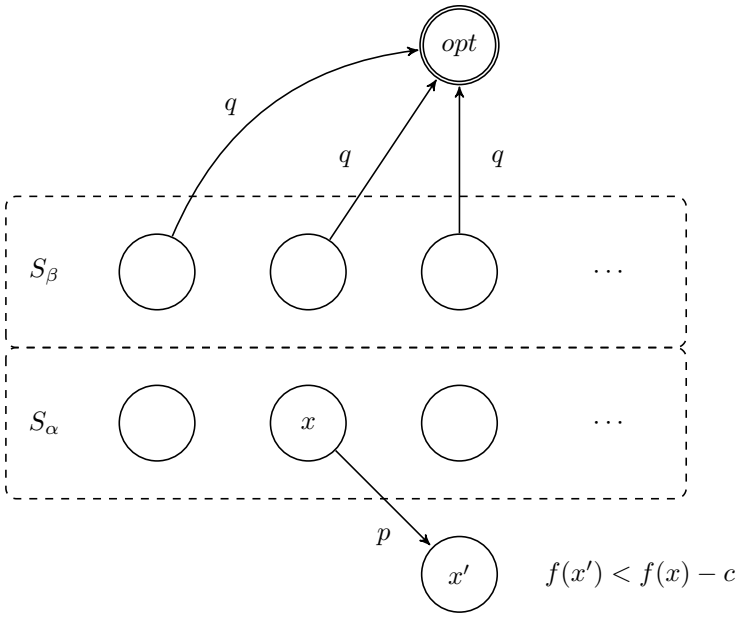

Figure 4: Partitioned Markov chain with transition probability bounds. S_α contains all states corresponding to tours that have intersecting edges. S_β contains all states corresponding to suboptimal intersection-free tours. Note that x' can be in either S_α or S_β.

Proof. Denote as $f_\alpha = \min\{f(x) \mid x \in S_\alpha\}$ to be the minimum cost over all tours with intersections. Furthermore, let $f^\star = \max\{f(x) \mid x \in S_\beta \cup \{opt\}$ and $f(x) < f_\alpha\}$ be the highest cost of all intersection-free tours that have a strictly lower cost than f_α (this is well-defined since at least the optimal tour as a strictly lower cost than f_α). Note that once the algorithm generates a state z with $f(z) \leq f^\star$, it will never return to a state in S_α (since such as state has strictly worse fitness). We define a nonnegative distance function δ as

$$\delta(x) = \begin{cases} f(x) - f^\star & \text{if } x \in S_\alpha; \\ 0 & \text{otherwise.} \end{cases}$$

For all $x \in S_\alpha$, $f(x) \geq nd_{min}$. Moreover, $f^\star \leq nd_{max}$. Hence $\delta(x) \leq n(d_{max} - d_{min})$.

Let $(y^{(1)}, y^{(2)}, \ldots)$ be the stochastic process that is a restriction of $(x^{(1)}, x^{(2)}, \ldots)$ constructed by taking only permutations $y^{(t)} \in S_\alpha$ in the same order. We consider the drift of the random sequence $\{\delta(y^{(t)});\ t = 1, 2, \ldots\}$.

From any solution $y^{(t)}$, the probability to improve the fitness by at least c is bounded below by p. Therefore we have

$$\mathbb{E}(\delta(y^{(t)}) - \delta(y^{(t+1)})) \geq cp.$$

By the Additive Drift Theorem [17], the random variable $T_\alpha = \inf\{t \in \mathbb{N} : \delta(y^{(t)}) = 0\}$ is bounded above by $p^{-1}n(d_{max} - d_{min})/c$. Finally, the series on the LHS of the claimed inequality is the total time the algorithm spends in S_α, or more precisely,

$$\sum_{t=1}^{\infty} \alpha(x^{(t)}) = \sum_{t=1}^{\infty} \alpha(y^{(t)}) = T_\alpha,$$

and hence the claim follows. □

Lemma 3. *If there is a constant $0 < q < 1$ such that for any $x \in S_\beta$, the transition probability from x to opt is bounded below by q, then*

$$\mathbb{E}\left(\sum_{t=1}^{\infty} \beta(x^{(t)})\right) \leq q^{-1}$$

Proof. Again, since $q > 0$ for any $x \in S_\beta$ there is a nonzero probability the algorithm exits the state x and transits to the absorbing state opt. This event is characterized as an independent Bernoulli trial with success probability at least q. It follows that the series in the claim can be estimated by a geometrically distributed random variable with parameter q, and therefore the expected time spent in states contained in the S_β partition is bounded above by q^{-1}. □

In Section 3.3 we will introduce a constraint on the angles between edges that give rise to suitable values for c in the claim of Lemma 2. This, along with corresponding values for p and q (that appear the claims of Lemma 2 and 3, respectively), will allow us to bound the expected runtime in terms of n and k. Before doing so, it will be convenient to state a number of lemmas that will be useful in the analysis. To preserve the flow of the text, we defer their proofs to the appendix.

Lemma 4. *Let x be a permutation such that $C(x)$ is not intersection-free. Then there exists an inversion that removes a pair of intersecting edges and replaces them with a pair of non-intersecting edges.*

Proof. (see Appendix). □

Note that it is still possible that the introduced edges intersect with some of the remaining edges in $C(y)$.

Lemma 5. *Suppose $|\mathsf{Inn}(V)| = k$. Then there are at most $\binom{n}{k}k!$ distinct intersection-free tours.*

Proof. (see Appendix). □

Lemma 6. *Suppose $|\mathsf{Inn}(V)| = k$ and $C(x)$ is an intersection-free tour on V. Then there is a sequence of at most k jump operations that transforms x into an optimal permutation. Similarly, there is a sequence of at most $2k$ inversion operations that transform x into an optimal permutation.*

Proof. (see Appendix). □

3.3 Angle-bounding, quantization, and resulting structural properties

A challenge to the runtime analysis of algorithms that employ edge exchange operations such as 2-opt is that, when points are allowed in arbitrary positions, the minimum change in fitness between neighboring solutions can be made arbitrarily small. Indeed, proof techniques for worst-case analysis often leverage this fact [15]. To circumvent this, we impose bounds on the angles between points, which allows us to express runtime results as a function of trigonometric expressions involving these bounds. Momentarily, we will refine this further by introducing a class of TSP instances embedded in an $m \times m$ grid. In that case, we will see that the resulting trigonometric expression is bounded by a polynomial in m.

We say V is *angle-bounded by* ϵ for some $0 < \epsilon < \pi/2$ if for any three points $u, v, w \in V$, $0 < \epsilon < \theta < \pi - \epsilon$ where θ denotes the angle formed by the line from u to v and the

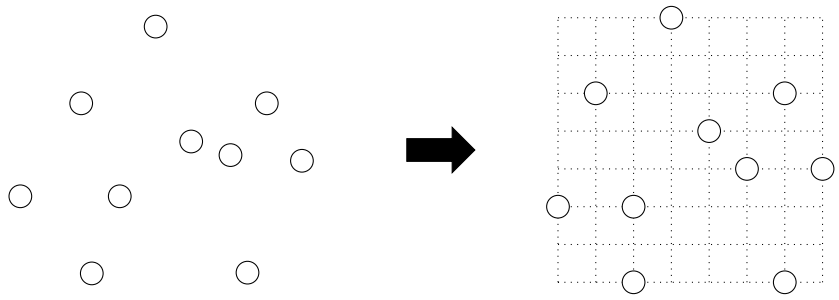

Figure 5: Quantizing a set of n points in the plane onto an $m \times m$ grid. Coordinates are rounded to the nearest value in a set of m equidistant values.

line from v to w. This allows us to express a bound in terms of ϵ on the change in fitness from a move that removes an inversion.

Lemma 7. *Suppose V is angle-bounded by ϵ. Let x be a permutation such that $C(x)$ is not intersection-free. Let $y = \sigma^{\mathrm{I}}_{ij}[x]$ be the permutation constructed from an inversion on x that replaces two intersecting edges in $C(x)$ with two non-intersecting edges.*[1] *Then, if d_{min} denotes the minimum distance between any two points in V, $f(x) - f(y) > 2d_{min}\left(\frac{1-\cos(\epsilon)}{\cos(\epsilon)}\right)$.*

Proof. (see Appendix). □

The lower bound on the angle between any three points in V provides a constraint on how small the change in fitness between neighboring inversions can be. This lower bound is useful in the case of a *quantized* point-set. That is, when the points can be embedded on an $m \times m$ grid as illustrated in Figure 5.

Quantization, for example, occurs when the x and y coordinates of each point in the set are rounded to the nearest value in a set of m equidistant values (e.g., integers). We point out that it is still important that the quantization preserves the constraint on collinearity since collinear points violate a nonzero angle bound. We have the following lemma.

Lemma 8. *Suppose V is a set of points that lie on an $m \times m$ unit grid, no three collinear. Then V is angle-bounded by* $\arctan\left(1/(2(m-2)^2)\right)$.

Proof. (see Appendix). □

Lemma 8 allows us to translate the somewhat awkward trigonometric expression in the claim of Lemma 7 (and subsequent lemmas that depend on it) into a convenient polynomial that can be expressed in terms of m.

Lemma 9. *Let V be a set of n points that lie on an $m \times m$ unit grid, no three collinear. Then, V is angle-bounded by ϵ where $\cos(\epsilon)/(1-\cos(\epsilon)) = O(m^4)$.*

Proof. (see Appendix). □

We are now ready to prove the following technical lemma for 2-opt mutation defined in Function 3. This lemma will be instrumental in proving runtime bounds later in the paper.

[1] Lemma 4 guarantees the existence of such an inversion.

Lemma 10. *Let V be a set of planar points in convex position angle-bounded by ϵ. We have the following.*

(1) For any $x \in S_\alpha$, the probability that 2-opt mutation creates an offspring y with $f(y) < f(x) - 2d_{min}\,(1-\cos(\epsilon))\,/\,(\cos(\epsilon))$ is at least $2/(en(n-1))$.

(2) For any $x \in S_\beta$, the probability that 2-opt mutation creates an optimal solution is at least $(en^{4k}(2k-1)!)^{-1}$

Proof. For (1), suppose $x \in S_\alpha$. By Lemma 4, there is at least one pair of intersecting edges in $C(x)$ that can be removed with a single inversion operation σ^{I}_{ij}. Let $y = \sigma^{\mathrm{I}}_{ij}[x]$. By Lemma 7, $f(y)$ satisfies the fitness bound in the claim. It suffices to bound the probability that y is produced by `2-opt-mutation(`x`)`.

Let E_1 denote the event that Poisson mutation performs exactly one inversion (i.e., $s = 0$ in line 2 of Function 3). Let E_2 denote the event that the pair (i, j) is specifically chosen for the inversion.

We have $\Pr\{E_1\} = e^{-1}$ from the Poisson density function and $\Pr\{E_2|E_1\} \geq (n(n-1)/2)^{-1}$. Thus the probability that 2-opt mutation creates y from x is

$$\Pr\{E_1 \cap E_2\} = \Pr\{E_1\} \cdot \Pr\{E_2|E_1\} \geq 2/(en(n-1)).$$

For (2), suppose $x \in S_\beta$. Thus, $C(x)$ is intersection-free, and it follows from Lemma 6 that there are at most $2k$ inversion moves that transform x into an optimal solution.

Let E'_1 denote the event that Poisson mutation performs exactly $2k$ inversions (i.e., $s = 2k-1$ in line 2 of Function 3). Let E'_2 denote the event that all $2k$ inversions are the correct moves that transform x into an optimal solution.

Again, from the Poisson density function, $\Pr\{E'_1\} = (e(2k-1)!)^{-1}$. Since $\Pr\{E'_2|E'_1\} \geq (n(n-1)/2)^{-2k} \geq n^{-4k}$, the probability of transforming x into an optimal solution is at least

$$\Pr\{E'_1 \cap E'_2\} = \Pr\{E'_1\} \cdot \Pr\{E'_2|E'_1\} \geq (en^{4k}(2k-1)!)^{-1}. \qquad \square$$

Finally, since the time bounds in the remainder of this paper are expressed as a function of the angle bound ϵ, and the bounds on point distance, it will be useful to define the following function.

Definition 6. *Let V be a set of points angle-bounded by ϵ. We define*

$$A(\epsilon) = \left(\frac{d_{max}}{d_{min}} - 1\right)\left(\frac{\cos(\epsilon)}{1-\cos(\epsilon)}\right)$$

where d_{max} and d_{min} respectively denote the maximum and minimum Euclidean distance between points in V.

3.4 Instances in Convex Position

A finite point set V is in *convex position* when every point in V is a vertex on its convex hull. Deĭneko et al. [8] observed that the Euclidean TSP is easy to solve when V is in convex position. In this case, the optimal permutation is any linear ordering of the points which respects the ordering of the points on the convex hull. Such an ordering can be found in time $O(n \log n)$ [7].

In the context of evolutionary algorithms, the natural question arises, if V is in convex position, how easy is it for simple randomized search heuristics? In this case,

a tour is intersection-free if and only if it is globally optimal, hence finding an optimal solution is exactly as hard as finding an intersection-free tour. Since an intersection can be (at least temporarily) removed from a tour by an inversion operation (cf. Lemma 4), we focus in this section on algorithms that use the inversion operation.

We begin our analysis of instances in convex position by considering RLS. RLS operates on a single candidate solution by performing a single inversion in each iteration. Since each inversion which removes an intersection results in a permutation whose fitness is improved by the amount bounded in Lemma 7, it is now straightforward to bound the time it takes for RLS to discover a permutation that corresponds to an intersection-free tour.

Theorem 3. *Let V be a set of planar points in convex position angle-bounded by ϵ. The expected time for RLS to solve the TSP on V is $O(n^3 A(\epsilon))$ where A is defined in Definition 6.*

Proof. Consider the infinite stochastic process $(x^{(1)}, x^{(2)}, \ldots)$ generated by Algorithm 4. It suffices to bound the expectation of the random variable T defined in Equation (2). Since V is in convex position, any intersection-free tour is globally optimal. Thus, in this case $S_\beta = \emptyset$, and Equation (3) can be written as

$$T = \sum_{t=1}^{\infty} \alpha(x^{(t)}).$$

Consider an arbitrary permutation $x \in S_\alpha$. Since $C(x)$ must contain intersections, by Lemma 4, there is an inversion σ^{I}_{ij} which removes a pair of intersecting edges and replaces them with a pair of non-intersecting edges. Moreover, by Lemma 7, such an inversion results in an improvement of at least

$$2d_{min}\left(1 - \cos(\epsilon)\right) / \left(\cos(\epsilon)\right). \tag{4}$$

If the state x is visited by RLS, the probability that this particular inversion is selected uniformly at random is $2/(n(n-1))$. Thus we have the conditions of Lemma 2 with $p = 2/(n(n-1))$ and c equal to the expression in (4) and the claimed bound on the expectation of T follows. □

Corollary 1 (to Theorem 3)**.** *If V is in convex position and embedded in an $m \times m$ grid with no three points collinear, then RLS solves the TSP on V in expected time $O(n^3 m^5)$.*

Proof. The bound follows immediately from Theorem 3 since V is in convex position and since, by Lemma 8, V is angle-bounded by $\arctan\left(1/(2(m-2)^2)\right)$, $d_{max} \leq (m-1)\sqrt{2}$, and $d_{min} \geq 1$. Appealing to Lemma 9 yields $\cos(\epsilon)/(1-\cos(\epsilon)) = O(m^4)$. Substituting these terms into bound of Theorem 3 completes the proof. □

We now consider the optimization time of the $(\mu + \lambda)$ EA using 2-opt mutation defined in Algorithm 1 applied to a set of points in convex position. Obviously, in this case we will find μ and λ terms appearing in the runtime formulas. We will assume that μ and λ are polynomials in both n and k. We will also find that setting $\lambda = \Theta(\mu n^2)$ ensures that a transition from any state $x^{(t)} \in S_\alpha$ to a state which improves on the fitness by at least a specified amount occurs with constant probability.

Such a setting has the effect of reducing the number of *generations* spent removing intersections from the best-so-far tours. However, it is important to note that the number of calls to the fitness function must be accordingly increased by a factor of $\Theta(\mu n^2)$ in each generation. Nevertheless, the expected number of generations can be a useful

measure when considering parallel evolutionary algorithms. As Jansen, De Jong and Wegener [19] have pointed out, when the fitness evaluation of the offspring can be performed on parallel processors, the number of generations corresponds to the *parallel* optimization time. In such a case, we would observe a quadratic factor improvement in the parallel runtime corresponding to the segment spent in tours with intersections. Nevertheless, we also report the bound on the number of total fitness function evaluations.

The following theorem bounds the number of expected generations the $(\mu+\lambda)$ EA needs to solve the Euclidean TSP on a set of angle-bounded points in convex position.

Theorem 4. *Let V be a set of planar points in convex position angle-bounded by ϵ. The expected time for the $(\mu+\lambda)$ EA using 2-opt mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\big\{(\mu/\lambda) \cdot n^2, 1\big\}\big)$ where A is as defined in Definition 6.*

Proof. The sequence of best-so-far permutations generated by Algorithm 1 is the infinite stochastic process $(x^{(1)}, x^{(2)}, \ldots)$ and we seek the expectation of the random variable T defined in Equation (2). Again, in the case of convex position, there are no non-optimal intersection-free tours so that $S_\beta = \emptyset$ and

$$T = \sum_{t=1}^{\infty} \alpha(x^{(t)}).$$

As long as $x^{(t)}$ is non-optimal, $C(x^{(t)})$ must contain at least one pair of intersecting edges. Hence, in generation t, if $x^{(t)}$ is selected for mutation to create one of the λ offspring, then by Lemma 10, 2-opt mutation must improve the best-so-far solution by at least $c = 2d_{min}\left(1-\cos(\epsilon)\right)/\left(\cos(\epsilon)\right)$ with probability at least $(en(n-1)/2)^{-1}$. The probability that at least one of the λ offspring improves on $x^{(t)}$ by at least this amount is

$$p \geq 1 - \left(1 - \frac{1}{\mu en(n-1)/2}\right)^{\lambda}.$$

We now make the following case distinction on λ.

Case 1: $\lambda \geq \mu en(n-1)/2$. For this setting of λ, we have $p \geq 1 - e^{-1}$, so an intersection is removed in each generation with constant probability. Invoking Lemma 2, the expected time to find an intersection-free tour is at most $O(nA(\epsilon))$.

Case 2: $\lambda < \mu en(n-1)/2$. Here we have

$$1 - \left(1 - \frac{1}{\mu en(n-1)/2}\right)^{\lambda} \geq 1 - e^{-\lambda/(\mu en(n-1)/2)} \geq \frac{\lambda}{\mu en(n-1)}.$$

The final inequality comes from the fact that $e^{-x} \leq 1 - x/2$ for $0 \leq x \leq 1$. Thus, in this case we have $p \geq \lambda/(\mu en^2)$. Applying Lemma 2, the expected time to find an intersection-free tour is at most $O((\mu/\lambda) \cdot n^3 A(\epsilon))$.

Thus the expected time to solve the instance is bounded by $O(\max\{(\mu/\lambda) \cdot n^3 A(\epsilon), nA(\epsilon)\})$ which yields the claimed bound. □

We immediately get the following corollary.

Corollary 2 (to Theorem 4)**.** *Let V be a set of planar points in convex position angle-bounded by ϵ. The expected number of fitness evaluations needed by the $(\mu + \lambda)$ EA using 2-opt mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\{\mu n^2, \lambda\}\big)$, and the expected optimization time of the $(1 + 1)$ EA is bounded above by $O\big(n^3 A(\epsilon)\big)$ where A is as defined in Definition 6.*

Moreover, analogous to the corollary to Theorem 3, we have the following.

Corollary 3 (to Theorem 4)**.** *If V is in convex position and embedded in an $m \times m$ grid with no three points collinear, then the $(\mu + \lambda)$ EA solves the TSP on V in expected time $O\big(nm^5 \cdot \max\{(\mu/\lambda) \cdot n^2, 1\}\big)$*

3.5 Instances with k inner points

We now turn our attention to TSP instances in which $\mathsf{Out}(V) \neq V$ and express the expected runtime in terms of $n = |V|$, the number of points, and $k = |\mathsf{Inn}(V)|$, the number of inner points.

3.5.1 2-opt Mutation

We use the structural analysis in Section 3.3 to show that when there are few inner points, intersection-free tours are somehow "close" to an optimal solution in the sense that relatively small perturbations by the EA suffice to solve the problem. Again, recall that we assume μ and λ are polynomials in both n and k.

Theorem 5. *Let V be a set of points angle-bounded by ϵ such that $|\mathsf{Inn}(V)| = k$. The expected time for the $(\mu + \lambda)$ EA using 2-opt mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{4k}(2k-1)!\big)$.*

Proof. We argue by analyzing the Markov chain generated by the $(\mu + \lambda)$ EA using 2-opt mutation. Let $(x^{(1)}, x^{(2)}, \ldots)$ denote the sequence of best-so-far states visited by the $(\mu + \lambda)$ EA.

In generation t, if $x^{(t)} \in S_\alpha$, then the probability of generating an offspring that improves the fitness by at least $c = 2d_{min}\,(1 - \cos(\epsilon))\,/\,(\cos(\epsilon))$ is identical to that in the proof of Theorem 4. Arguing in the same manner as in the proof of Theorem 4, we have

$$\sum_{t=1}^{\infty} \alpha(x^{(t)}) = O\big(n \cdot A(\epsilon) \cdot \max\{(\mu/\lambda) \cdot n^2, 1\}\big)$$

On the other hand, if $x^{(t)} \in S_\beta$, if it is selected for mutation, then, by Lemma 10, 2-opt mutation produces the optimal solution with probability at least $(en^{4k}(2k-1)!)^{-1}$. Hence, the overall probability of generating an optimal permutation when $x^{(t)}$ is the (intersection-free) population-best permutation is at least

$$q \geq 1 - \left(1 - \frac{1}{\mu e n^{4k}(2k-1)!}\right)^{\lambda} \geq \frac{\lambda}{2\mu e n^{4k}(2k-1)!}.$$

Since this is the probability that the Markov chain transits from a state $x^{(t)} \in S_\beta$ to the optimal state, substituting the value for q into the claim of Lemma 3, we have

$$\mathbb{E}\left(\sum_{t=1}^{\infty} \beta(x^{(t)})\right) = O((\mu/\lambda) \cdot n^{4k}(2k-1)!).$$

The bound on $\mathbb{E}(T)$ then follows from Equation 3 and linearity of expectation. □

We also get the following corollary.

Corollary 4 (to Theorem 5)**.** *Let V be a set of points angle-bounded by ϵ such that $|\mathsf{Inn}(V)| = k$. The expected number of fitness evaluations needed for the $(\mu+\lambda)$ EA using 2-opt mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\{\mu n^2, \lambda\} + \mu n^{4k}(2k-1)!\big)$ and the expected optimization time for the (1+1) EA is $O\big(n^3 \cdot A(\epsilon) + n^{4k}(2k-1)!\big)$.*

Again, from Lemmas 8 and 9 we also have the following corollary.

Corollary 5 (to Theorem 5)**.** *Let V be a set of points quantized on an $m \times m$ grid such that $|\mathsf{Inn}(V)| = k$. The expected time for the $(\mu+\lambda)$ EA using 2-opt mutation to solve the TSP on V is $O\big(nm^5 \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{4k}(2k-1)!\big)$.*

3.5.2 Mixed Mutation Strategies

The Markov chain analysis relies on the inversion operator to construct an intersection-free tour, but then relies on the inversion operator to simulate the jump operator in order to transform an intersection-free tour into an optimal solution. We now introduce a mutation technique called `mixed-mutation` (outlined in Function 5) that performs both inversion and jump operations, each with constant probability. This allows for an upper bound that is smaller by a factor of $\Omega(n^{2k}(2k-1)!/(k-1)!)$. We remark here that we only use mixed mutation as a tool to reduce the upper bound, and cannot claim that it results in a faster algorithm due to the absence of a lower bound.

Similar to Lemma 10, we have the following result for mixed mutation.

Lemma 11. *Let V be a set of planar points in convex position angle-bounded by ϵ. Then,*

(1) For any $x \in S_\alpha$, the probability that mixed mutation creates an offspring y with $f(y) < f(x) - 2d_{min}\,(1-\cos(\epsilon))\,/\,(\cos(\epsilon))$ is at least $(en(n-1))^{-1}$.

(2) For any $x \in S_\beta$, the probability that mixed mutation creates an optimal solution is at least $(2en^{2k}(k-1)!)^{-1}$

Proof. For (1), the probability that mixed mutation selects inversions is $1/2$ and the rest of the claim follows from the argument of Lemma 10.

For (2), suppose $x \in S_\beta$. Thus, $C(x)$ is intersection-free, and it follows from Lemma 6 that there are at most k jump operations that transform x into an optimal solution. The remainder of the proof is identical to that of Lemma 10. The probability that mixed mutation selects jump operations contributes the leading factor of 2^{-1}. □

Function 5: `mixed-mutation(`x`)`

```
y ← x;
draw r from a uniform distribution on the interval [0, 1];
draw s from a Poisson distribution with unit expectation;
if r < 1/2 then perform s + 1 random inversion operations on y;
else perform s + 1 random jump operations on y;
return y;
```

Theorem 6. *Let V be a set of points angle-bounded by ϵ such that $|\mathsf{Inn}(V)| = k$. The expected time for the $(\mu+\lambda)$ EA using mixed mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{2k}(k-1)!\big)$.*

Proof. The proof is identical to the proof of Theorem 5, except we substitute the probabilities from Lemma 11 into Lemmas 2 and 3. □

We immediately get the following corollary.

Corollary 6 (to Theorem 6)**.** *Let V be a set of points angle-bounded by ϵ such that $|\mathsf{Inn}(V)| = k$. The expected number of fitness evaluations needed for the $(\mu+\lambda)$ EA using mixed mutation to solve the TSP on V is bounded above by $O\big(n \cdot A(\epsilon) \cdot \max\{\mu n^2, \lambda\} + \mu n^{2k}(k-1)!\big)$ and the expected optimization time for the (1+1) EA is bounded above by $O\big(n^3 \cdot A(\epsilon) + n^{2k}(k-1)!\big)$.*

As before, from Lemmas 8 and 9 we have the following.

Corollary 7 (to Theorem 6)**.** *Let V be a set of points quantized on an $m \times m$ grid such that $|\mathsf{Inn}(V)| = k$. The expected time for the $(\mu+\lambda)$ EA using mixed mutation to solve the TSP on V is $O\big(nm^5 \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{2k}(k-1)!\big)$.*

4 Fixed-parameter evolutionary algorithms

Moving away from the black box perspective, we now consider designing evolutionary algorithms that are given access to a small amount of problem information. Specifically, we allow the algorithms access to the ordering of points that lie on the convex hull. Using this information, we design two fixed-parameter tractable evolutionary algorithms for the inner point parameterization of Deĭneko et al. [8].

Definition 7. *Let V be a set of points in the plane. We define γ as a linear order on $\mathsf{Out}(V)$*

$$\gamma = (p_1, p_2, \ldots, p_{n-k})$$

such that for all $i \in \{1, \ldots, n-k\}$, p_i and p_{i+1} are adjacent on the boundary of the convex hull of V. We say a permutation x on V is γ-respecting if and only if

$$x^{-1}(p_i) < x^{-1}(p_j) \implies i < j.$$

Note that, by definition, every γ-respecting permutation respects hull-order (as defined in Section 2). The linear order γ can be found in $O(n \log n)$ time using well-known methods from computational geometry [7].

4.1 A population-based approach

In this section we describe a $(\mu + \lambda)$ EA that solves the TSP to optimality in fixed-parameter tractable time. We build on the evolutionary dynamic programming framework for TSP introduced by Theile [35]. We show that a slight modification to Theile's $(\mu + 1)$ EA that carefully maintains the invariant that the points in $\mathsf{Out}(V)$ remain in convex-hull order for each individual allows us to improve the time bound significantly on instances for which the number of inner points is not too large.

Given a set of planar points V, with $|V| = n$ and $|\mathsf{Inn}(V)| = k$, let $\gamma = (p_1, p_2, \ldots, p_{n-k})$ denote the order on $\mathsf{Out}(V)$ given by Definition 7. A permutation x on a subset U of V is a bijection $x : U \to U$. The $(\mu + \lambda)$ EA is described in Algorithm 6. The $(\mu + \lambda)$ EA maintains a population P of μ permutations on subsets of V that have a special structure that we now describe. Let $S \subseteq \mathsf{Inn}(V)$, let $i \in \{1, \ldots, n-k\}$, and let $r \in S \cup \{p_i\}$. An individual $x = x_{(i,S,r)}$ is a permutation on $S \cup \{p_1, p_2, \ldots, p_i\}$ with the following properties.

1. $x(1) = p_1$,
2. $x(|S| + i) = r$,

3. x is γ-respecting, that is, $p_1, p_2, \ldots, p_i$ appear in order.

For a permutation $x_{(i,S,r)}$, we call the set $S \cup \{p_1, p_2, \ldots, p_i\}$ its *ground set* and we call r its *tail vertex*.

The TSP tour defined by a permutation $x_{(i,S,r)}$ is a cycle $(p_1, v_{x(2)}, \ldots, v_{x(|S|+i-1)}, r, p_1)$, i.e., it starts at p_1, runs over all nodes in $(S \cup \{p_2, \ldots, p_i\}) \setminus r$ such that any nodes in $\mathsf{Out}(V)$ are visited in the order they appear in γ, and finally visits r before returning to p_1. The full population of the $(\mu + \lambda)$ EA consists of an individual $x_{(i,S,r)}$ for every $i \in \{1, \ldots, n-k\}$, every $S \subseteq \mathsf{Inn}(V)$, and possible tail vertex r.

The fitness function f assigns a nonnegative real value to each individual $x_{(i,S,r)}$ representing the cost of the tour through the vertices in the ground set of $x = x_{(i,S,r)}$.

$$f(x_{(i,S,r)}) = \sum_{j=1}^{|S|+i} d(v_{x(j)}, v_{x(j+1)}) \tag{5}$$

where the summation indices are taken to be modulo $|S| + i$.

In each generation, λ individuals are selected uniformly at random from the population P and an offspring is created for each individual by applying a mutation operator. This mutation operator attempts to extend each individual by "growing" its ground set. To mutate a single individual $x = x_{(i,S,r)}$, a vertex v is chosen uniformly at random from the remaining vertices in $(\mathsf{Inn}(V) \setminus S) \cup \{p_{i+1}\}$. A new permutation x' is created by concatenating v to the linear order described by x, that is, for $j \in \{1, \ldots, |S| + i + 1\}$,

$$x'(j) = \begin{cases} v & \text{if } j = |S| + i + 1; \\ x(j) & \text{otherwise.} \end{cases}$$

Thus x' is defined on a different (slightly larger) ground set than x using v as the new tail vertex. Following our notation, we have

$$x' = \begin{cases} x'_{(i,S\cup\{v\},v)} & \text{if } v \in \mathsf{Inn}(V); \\ x'_{(i+1,S,v)} & \text{if } v = p_{i+1}. \end{cases}$$

Obviously, when $i = n - k$ and $S = \mathsf{Inn}(V)$, the mutation operator has no effect since the ground set can not be extended for such an individual.

In order to introduce sufficiently large changes to the population in each generation, we utilize the Poisson mutation strategy in the same manner as both Theile's original algorithm and the black-box evolutionary algorithms introduced in Section 3. Specifically, in each generation, for each of the selected λ individuals, we apply the mutation operator defined above $s+1$ times, where s is drawn from a Poisson distribution with unit expectation.

Finally, we again employ truncation selection in the sense that each mutated offspring may only replace the individual in the parent population with the same ground set and tail vertex. This replacement occurs only if the fitness of the mutated offspring is at least as good as the parent individual in question. Thus, at the end of a generation, the parent population size μ is preserved.

We now bound the expected time until the $(\mu + \lambda)$ EA described in Algorithm 6 has solved a TSP instance on n points with k inner points to optimality. We first bound the population size μ with the following lemma.

Algorithm 6: $(\mu + \lambda)$ EA

```
P ← ∅;
foreach i ∈ {1, ..., n − k} do
    foreach S ⊆ Inn(V) do
        foreach r ∈ S ∪ p_i do
            x ← a permutation on the ground set S ∪ {p_1, p_2, ..., p_i} such that
            x(|S| + i) = r and x respects γ;
            P ← P ∪ x;
repeat forever
    P' ← {};
    repeat λ times
        Select an individual z ← x_(i,S,r) ∈ P uniformly at random;
        Draw s from a Poisson distribution with unit expectation;
        Generate z' ← x_(i',S',r') by applying the mutation operator s + 1 times;
        Let f(z') be the cost of TSP tour generated by z';
        P' ← P' ∪ z';
    /* truncation selection based on the same ground set  */
    foreach offspring z' in P' do
        Let z'' ← x_(i',S'r') ∈ P be an individual defined on the same ground set
        as z' having the same end vertex if such an individual exists in the
        population;
        if f(z') ≤ f(z'') then P ← P ∪ z' \ z''
```

Lemma 12. *The parent population size μ is bounded above by $O(2^k kn)$.*

Proof. The population is initialized by constructing, for each possible distinct ground set $(S \subseteq \mathsf{Inn}(V)) \cup (\{p_1, p_2, \ldots, p_i\} \subseteq \mathsf{Out}(V))$ and each legal tail vertex r a length $|S|+i$ permutation $x = x_{(i,S,r)}$ such that $x(1) = p_1$, $x(|S| + i) = r$, and the elements from $\{p_1, p_2, \ldots, p_i\}$ appear in the same order as they do in γ. Note that to maintain the correct ordering, the tail vertex cannot be any point p_j for $j < i$, or the ordering with respect to γ is violated. Therefore, the tail vertex can be any member of $S \cup \{p_i\}$.

We count the number of distinct ground sets $S \cup \{p_1, p_2, \ldots, p_i\}$ along with the number of ways to select a tail vertex legally from the ground set. There are $\binom{k}{|S|}$ ways to choose a distinct set $S \subseteq \mathsf{Inn}(V)$. On the other hand, the order of the outer points is fixed by γ, so there are only $(n - k)$ ways to choose a distinct set of outer points $\{p_1, \ldots, p_i\}$. Finally, for such a ground set, there are $|S| + 1$ ways to choose the tail vertex $r \in S \cup \{p_i\}$. In total, there are

$$(n-k)\sum_{s=0}^{k}\binom{k}{s}(s+1) = O(2^k kn)$$

distinct permutations in the initial population. Since $(\mu + \lambda)$ truncation selection maintains a constant population size throughout the process, the claim holds. □

After the population P is initialized, for each ground set $(S \subseteq \mathsf{Inn}(V)) \cup (\{p_1, p_2, \ldots, p_i\})$ and each legal tail vertex $r \in S \cup \{p_i\}$, there exists some permutation $x = x_{(i,S,r)} \in P$ that respects the order γ.

We consider the function

$$F : \{1, 2, \ldots, n-k\} \times 2^{\mathsf{Inn}(V)} \times V \to \mathbb{R}_+$$

where $F(i, S, r)$ is the length of the shortest path starting at p_1 and ending at r, visiting each point in $\{p_1, \ldots, p_i\} \cup S$ exactly once, and respecting the order γ. Because each Hamiltonian path possesses the optimal substructure property (i.e., subpaths of optimal paths are also optimal paths), F can be defined recursively using the Bellman principle [5, 8],

$$F(i, S, r) = \begin{cases} \min\limits_{q \in S \cup \{p_{i-1}\}} F(i-1, S, q) + d(q, r) & \text{if } r \in \mathsf{Out}(V); \\ \min\limits_{q \in (S \setminus \{r\}) \cup \{p_i\}} F(i, S \setminus \{r\}, q) + d(q, r) & \text{if } r \in \mathsf{Inn}(V). \end{cases}$$

Thus, if there is a permutation with an optimal subpath in the population, there exists some correct mutation that can construct a longer path that is also optimal. Furthermore, a permutation $x_{(i,S,r)}$ that represents a suboptimal path through the ground set $S \cup \{p_1, p_2, \ldots, p_i\}$ can always be replaced by one that represents an optimal subpath through that ground set, i.e., some $x'_{(i,S,r)}$ such that $f(x'_{(i,S,r)}) = F(i, S, r) < f(x_{(i,S,r)})$ by the selection operation. Hence, once an optimal subpath is found for a particular ground set and tail vertex, the subpath optimality is never lost. To bound the optimization time of the $(\mu + \lambda)$ EA, it suffices to bound the time until a particular length-n permutation $x_{(n-k,\mathsf{Inn}(V),r)}$ has its optimal path in the population.

The proof of the runtime is a straightforward extension of Theile's [35] argument that the $(\mu + 1)$ EA can simulate dynamic programming on a function similar to F in a randomized fashion.

Theorem 7. *Let V be a set of n points in the Euclidean plane with $|\mathsf{Inn}(V)| = k$. After $O(\max\{2^k k^2 n^2 \lambda^{-1}, n\})$ generations, the $(\mu+\lambda)$ EA has solved the TSP on V to optimality in expectation and with probability $1 - e^{-\Omega(n)}$.*

Proof. Suppose there exists an individual $x = x_{(i,S,r)}$ such that $f(x_{(i,S,r)}) = F(i, S, r)$. Since selection for reproduction occurs uniformly at random, with probability at least $1/\mu$, x is selected for reproduction to produce one of the λ offspring by mutation in each subsequent generation. Furthermore, with probability e^{-1}, only a single mutation step is performed on x (i.e., $s = 0$ following line 11 of Algorithm 6). Finally, the probability that x is extended in an optimal fashion is at least $1/(k+1)$, that is, the probability that the correct vertex q is chosen out of $(\mathsf{Inn}(V) \setminus S) \cup \{p_{i+1}\}$ to create an optimal subpath on the ground set $S \cup \{p_1, p_2, \ldots, p_i\} \cup q$.

Assuming an optimal subpath of length $m-1$ already exists in the parent population, let π denote the probability that at least one of the λ offspring corresponds to an optimal subpath of length m by applying the correct mutation to the parent individual representing optimal subpath of length $m-1$. We have

$$\pi \geq 1 - \left(1 - \frac{1}{e\mu(k+1)}\right)^{\lambda}$$

In the base case, the permutation $x_{(p_1,\{\},p_1)}$ corresponds to an optimal length-1 path and is present after initialization in lines 1-2 of Algorithm 6. Since optimal paths through a given ground set and tail vertex are never lost, it follows by induction that the expected number generations until an optimal path of length m exists in the population is at most $m\pi^{-1}$. Thus, the expected number of generations until an optimal solution enters the population is bounded above by $n\pi^{-1}$. We make a case distinction on λ.

Case 1: $\lambda < e\mu(k+1)$. In this case, we have

$$\pi \geq 1 - \left(1 - \frac{1}{e\mu(k+1)}\right)^{\lambda} \geq 1 - e^{-\lambda/(e\mu(k+1))} \geq \frac{\lambda}{e\mu(k+1)},$$

again since $e^{-x} \leq 1 - x/2$ for $0 \leq x \leq 1$. The expected number of generations until the optimal solution is found is at most

$$n\pi^{-1} \leq en\mu(k+1)\lambda^{-1} = O(2^k k^2 n^2 \lambda^{-1}).$$

In the last step, we have applied the result of Lemma 12.

Case 2: $\lambda \geq e\mu(k+1)$. For this setting of λ we have $\pi \geq 1 - e^{-1}$ and the number of generations until the optimal solution is found is at most $n\pi^{-1} = O(n)$.

To prove the concentration result, for some constant $c > 0$ and any $m \in \{1, \ldots, n-1\}$, let $t := \alpha em\mu(k+1)$ be the number of generations needed to produce an optimal subpath of length m where $\alpha = cn/m$. We define a sequence of random indicator variables $X_i \in \{0,1\}$ for all $i \in \{1, \ldots, t\}$ such that $X_i = 1$ if the correct mutation has occurred to extend an optimal subpath in the i-th generation. Let $X = \sum_{i=1}^{t} X_i$. The expectation of X is $E(X) = \sum_{i=1}^{t} \Pr(X_i = 1) = t\pi \geq cn$, by the above bounds on π.

If $X < m$ the $(\mu+\lambda)$ EA has not created an optimal subpath in t fitness evaluations, and the probability of this event is at most $e^{-cn/2+m}$ by Chernoff bounds. Suppose there are ℓ individuals in the population that are length m. Clearly $\ell \leq m\binom{n-1}{m}$. The probability that at least one of the ℓ individuals in P has not been optimized within t generations is, by the union bound, at most

$$\sum_{i=1}^{\ell} e^{-cn+m} \leq m\binom{n-1}{m} e^{-c'n} \leq \exp\left(m\ln(n-1) - m\ln m - c'n + \ln m\right) = e^{-\Omega(n)}.$$

where $c' \geq c/2 - m/n$ is a constant. This completes the proof. □

Corollary 8 (to Theorem 7)**.** *The expected number of fitness evaluations is* $E(T_f) = \mu + \lambda E(T) = O(\max\{2^k k^2 n^2, \lambda n\})$.

4.2 Searching for permutation of inner points

In this section, we propose an approach that makes use of insights in the context of parameterized complexity. By Theorem 2, the outer points $\mathsf{Out}(V)$ are always visited in hull-order in every optimal solution. In the following, we assume that γ is the fixed order on the outer points given in Definition 7. Let x be a permutation on the inner points $\mathsf{Inn}(V)$. Using the dynamic programming algorithm of Deĭneko et al., one can compute the shortest Hamiltonian cycle respecting γ and x in time $O(kn)$.

These ideas suggest that using a heuristic to only search for an optimal permutation x^* on the inner points can significantly reduce the effective size of the search space, and thus lead to quicker solution times for evolutionary algorithms. For each permutation on the inner points x, the quality can be evaluated by running a dynamic programming algorithm to compute the best possible tour using γ and x.

Let $\gamma = (p_1, p_2, \ldots, p_{n-k})$ be the fixed order of the outer points and $x = (q_1, \ldots, q_k)$ be an arbitrary permutation of the inner points. We can compute the shortest tour respecting γ and x using the dynamic programming approach of [8] which we state in the following.

The dynamic programming algorithm computes a three-dimensional table F with entries $F[i, j, m]$ where $i \in \{1, \ldots, n-k\}$, $j \in \{0, 1, \ldots, k\}$, and $m \in \{Inn, Out\}$. The algorithm computes for each triple (i, j, m) the cost of the shortest path from p_1 through all nodes $p_1, \ldots, p_i$ and $q_1, \ldots q_j$ respecting the orders γ and x. The path ends at p_i iff $m = Out$ determines that the last node is an outer point and ends at q_j iff $m = Inn$ determines that the last node is an inner point.

The length of the shortest tour respecting γ and x is given by

$$Dyn(x) = \min\{F[n-k, k, Out] + d[p_{n-k}, p_1], F[n-k, k, Inn] + d[q_k, p_1]\}$$

The computation of the dynamic program starts with $F[1, 0, Out] = 0$ and we get

$$F[i, j, Inn] = \min\{F[i, j-1, Out] + d[p_i, q_j], F[i, j-1, Inn] + d[q_{j-1}, q_j]\}$$

for $i \in \{1, 2, \ldots, n-k\}$ and $j \in \{1, \ldots, k\}$. Also,

$$F[i, j, Out] = \min\{F[i-1, j, Out] + d[p_{i-1}, p_i], F[i-1, j, Inn] + d[q_j, p_i]\}$$

for $i \in \{2, 3, \ldots, n-k\}$ and $j \in \{0, \ldots, k\}$. For impossible states, the entries are set to ∞, namely, $F[1, j, Out] = \infty$ if $j \in \{1, \ldots, k\}$ and $F[i, 0, Inn] = \infty$ if $i \in \{1, \ldots, n-k\}$.

The time computing the F array is proportional to its size which is $2(n-k) \cdot (k+1)$. Therefore, the fitness $Dyn(x)$ for a given permutation x of the inner points can be computed in time $O(kn)$.

Algorithm 7: $(\mu + \lambda)$ EA^k

```
Choose a multiset P of μ random permutations on V;
repeat forever
    P' ← {};
    repeat λ times
        Choose x uniformly at random from P;
        Draw s from a Poisson distribution with unit expectation;
        Construct x' from x by applying s + 1 random basic operations;
        Let f(x') be Dyn(x');
        P' ← P' ∪ x';
    P ← select(P ⊎ P') ;
```

For our theoretical investigations, we study $(\mu + \lambda)$ evolutionary algorithms working on permutations of the inner points. The $(\mu + \lambda)$ EA^k (see Algorithm 7) starts with μ permutations on the set of inner points and creates in each iteration λ offspring by mutation. The mutation operator applies $s + 1$ basic operations where s is chosen according to the Poisson distribution with unit expectation. Then truncation selection is applied to preserve the best μ individuals out of all individuals for the next generation.

For mutation, we employ the inversion (2-opt) and permutation jump mutation operators introduced in Section 3.1. In the case of inversions, however, it is no longer clear whether this is the correct choice for the new search space, which is restricted only to orderings on the inner points. In addition to inversions and jumps (see Definitions 2 and 3), we also consider the *exchange* operator, which exchanges two elements (chosen uniformly at random) in the ordering of a permutation. We formalize the exchange operator in the following definition.

Definition 8. *The* exchange *operation σ^{E}_{ij} takes a permutation x and produces a permutation $\sigma^{\mathrm{E}}_{ij}[x]$ such that*

$$\sigma^{\mathrm{E}}_{ij}[x](\ell) = \begin{cases} x(j) & \text{if } \ell = i \\ x(i) & \text{if } \ell = j \\ x(\ell) & \text{otherwise.} \end{cases}$$

We denote by $(\mu+\lambda)$ EA^k_i, $(\mu+\lambda)$ EA^k_j, and $(\mu+\lambda)$ EA^k_e, the variants of the $(\mu+\lambda)$ EA^k using inversions, jumps, and exchanges, respectively.

In the following, we investigate the number of operations needed in order to obtain an optimal solution in dependence of the chosen mutation operator. This will lead to different upper bounds on the runtime of the algorithm. For all the operators, we show that they yield fixed-parameter tractable results.

Theorem 8. *Let V be a set of n points in the Euclidean plane with $|\mathsf{Inn}(V)| = k$. The expected number of generations needed to solve the TSP on V for both the $(\mu + \lambda)$ EA^k_j, and the $(\mu + \lambda)$ EA^k_e is bounded by $O(\max\{(k-1)!k^{2k}\lambda^{-1}, 1\})$. For the $(\mu + \lambda)$ EA^k_i, the bound is $O(\max\{(k-2)!k^{2k-2}\lambda^{-1}, 1\})$.*

Proof. Let π be a lower bound on the probability to transform an arbitrary non-optimal solution into an optimal solution. Suppose there are no optimal solutions in the population. The probability that at least one of the λ offspring is optimal is bounded below by $1 - (1-\pi)^\lambda = \min\{\lambda\pi, 1 - e^{-1}\}$.

The probability that a mutation operation consisting of a specific sequence of ℓ basic operations occurs is at least

$$\frac{1}{e(\ell-1)!k^{2\ell}}.$$

Thus, if ℓ is the maximum number of operations needed to transform an arbitrary solution into the optimal, the expected waiting time (in terms of generations) until such an event is at most $O(\max\{(\ell-1)!k^{2\ell}\lambda^{-1}, 1\})$. The number of inversions needed to transform an arbitrary permutation on k elements into any other permutation on the same elements is at most $k-1$ (this is due to a conjecture by Gollan that was later proved by Bafna and Pevzner [3]). For the jump and exchange operators, the number of these operations required is obviously at most k. The claimed bounds on the expected number of generations for each variant follow from these arguments, and the proof is complete. □

As usual, the expected number of fitness evaluations $E(T_f)$ can be derived by $T_f = \mu + \lambda E(T)$ where $E(T)$ is bounded by Theorem 8.

The previous theorem shows that simple evolutionary algorithms are fixed-parameter tractable evolutionary algorithms for the considered problem when working with permutations on the inner points and applying dynamic programming as part of the fitness evaluation. Note that dynamic programming takes time $O(kn)$. This is more expensive than computing the cost of a given tour on n cities, which can be done in $O(n)$ time. However, it is still feasible to use the dynamic programming approach as part of each fitness evaluation. In the next section, we show that this also leads to a significantly better optimization process than working with the standard approach that searches through the set of permutations on all n cities.

4.3 Experiments

In this section, we carry out experimental investigations for the approach proposed in the previous section. We consider a simple (1+1) EA, a specific case of the above studied $(\mu + \lambda)$ EA where the population consists of one individual and one offspring is generated at each iteration. We study the three variants of the (1+1) EA^k working on permutations of the set of inner points and compare them to the standard (1+1) EA^n (see Algorithm 8) approach working on permutations of the whole set of the given n points. We denote by (1+1) EA$_i^n$, (1+1) EA$_j^n$, and (1+1) EA$_e^n$, the variants of (1+1) EA^n using inversions, jumps, and exchanges, respectively.

Algorithm 8: $(1 + 1)$ EA^n

```
Choose a random permutation x on V;
repeat forever
    Draw s from a Poisson distribution with unit expectation;
    Construct x' from x by applying s + 1 random basic operations;
    if f(x') ≤ f(x) then x ← x';
```

We consider several planar Euclidean TSP instances of various sizes (25, 50 and 100) and, within each size class, control the percentages of inner points. The instance generation process is described in detail in the next section. We calculate the solution quality reached within a given time limit on each instance. On these instances, we run the evolutionary algorithms using each mutation strategy for both the (1+1) EA^n and the (1+1) EA^k. As a baseline, we also compute the value of the optimal solution on each instance using the exact TSP solver CONCORDE [1].

For each run, we set time limits based on the instance size. Hence, we run each EA for 1 second on instances with 25 points, 3 seconds for instances with 50 points, and finally 15 seconds for instances with 100 points. We measure the *iteration count* which is the number of fitness function calls for each EA. The (1+1) EA^n runs roughly 200,000 iterations within the given time limits for all instance sizes. On the other hand, the (1+1) EA^k performs in the range of 1000 – 5000 iterations. The number of iterations performed by the (1+1) EA^k decreases as a function of the ratio of outer points to inner points in the instance. The experiments are conducted on a PC with 2.4GHz Intel Core i5 CPU and 4GB 1333MHz DDR3 RAM.

4.3.1 Parameterized Instances

Since we need to study the impact of the number of inner points on algorithm performance, we cannot use standard benchmark instances, since it is not clear how to cleanly control the number of inner points in such problem sets. Instead, we generate random instances controlling for the percentage of points that lie in the interior of the convex hull. For each size category $\{25, 50, 100\}$, we generate a set of 30 instances each with inner point percentages 5%, 10%, 20%, 30% and 40%. For example, the set $50/10\%$ contains 30 TSP instances on $n = 50$ planar points such that exactly $k = 5$ lie in the interior of the convex hull. To construct such instances, we generate first the outer points on a convex hull, and then add the required number of inner points. We create a circle with a given radius and randomly generate outer points on the circumference by producing a random angle and then deriving x and y coordinates from the angle with respect to the y axis. This process continues until we have constructed the required values for $n - k$. We then identify the convex hull and generate random points inside the hull

instance	exchanges		inversions		jumps		
n / % inner	(1+1) EA^n_e	(1+1) EA^k_e	(1+1) EA^n_i	(1+1) EA^k_i	(1+1) EA^n_j	(1+1) EA^k_j	CONCORDE
25 / 5%	81937	53141	53665	53141	59173	53141	53141
25 / 10%	82592	57904	59190	57896	66070	57483	57482
25 / 20%	86646	63005	64222	63132	70232	61787	61324
25 / 30%	83869	67864	67570	67325	72586	65761	64165
25 / 40%	84941	70912	68486	69363	72649	67229	65370
50 / 5%	128640	61090	63258	61098	73293	60936	60936
50 / 10%	131391	67155	69625	67012	79230	66389	65831
50 / 20%	131027	79682	81414	78429	87586	77019	74809
50 / 30%	133533	89645	86474	86662	95599	83730	79785
50 / 40%	131902	97231	89271	92597	98999	90102	82469
100 / 5%	210234	69996	74392	69895	81839	69466	68990
100 / 10%	211502	83840	86301	82720	95024	81613	79529
100 / 20%	208611	104295	100239	98373	113363	98855	90840
100 / 30%	215322	121505	109335	112168	125907	110406	98853
100 / 40%	212083	138134	117594	124852	135720	128331	105432

Table 1: Average minimum tour length values (rounded) obtained by the six algorithms ((1+1) EA^n_e, (1+1) EA^k_e, (1+1) EA^n_i, (1+1) EA^k_i, (1+1) EA^n_j, (1+1) EA^k_j)and exact value by CONCORDE (from left to right) for the set of 30 instances in each category having sizes 25, 50 and 100 and inner points percentages 5,10,20,30 and 40

within a distance of zero to the radius. Here, we accept a randomly generated inner point only if it falls inside the convex hull.

4.3.2 Results

We measured the solution quality obtained within the allocated time for the six (1+1) EA variants described in this paper. We also compared each with the optimal solution computed by CONCORDE. The data set consists of 30 instances from each group representing each instance size and each inner node percentage.

The results suggest that the (1+1) EA^k approach consistently leads to more effective optimization when there are not too many inner points. In Table 1, we see that the average values of solution quality obtained for the instances in each instance size and inner point percentage. For some cases, variants of (1+1) EA^k (for example the set of $n = 25$ with 5% inner points) reaches the optimal values computed by the exact solver CONCORDE, while in most instances, optimizes the instances more effectively than the corresponding variants that search through permutations on all n points ((1+1) EA^n). In a few cases, where the inner point percentage is high, the new approach tends to perform poorly. This is expected, since the construction procedures are leveraging the constraints on the convex hull. Therefore, we expect the performance to degrade as the number of inner points becomes large.

To further investigate the effect of mutation strategy, we examine the three variants of the (1+1) EA^k for the problem set $n = 100$ and 40% inner points. The solution quality reached in the allocated time (15 seconds) is plotted as a function of instance for each variant in Figure 6.

To determine whether the (1+1) EA^k outperforms the (1+1) EA^n with statistical significance, we perform a one-tailed Wilcoxon signed rank test [37] on the solution quality measurements obtained after the allocated runtime. The resulting positive rank sum (W) and confidence (p-values) are recorded. We perform the test for each mutation

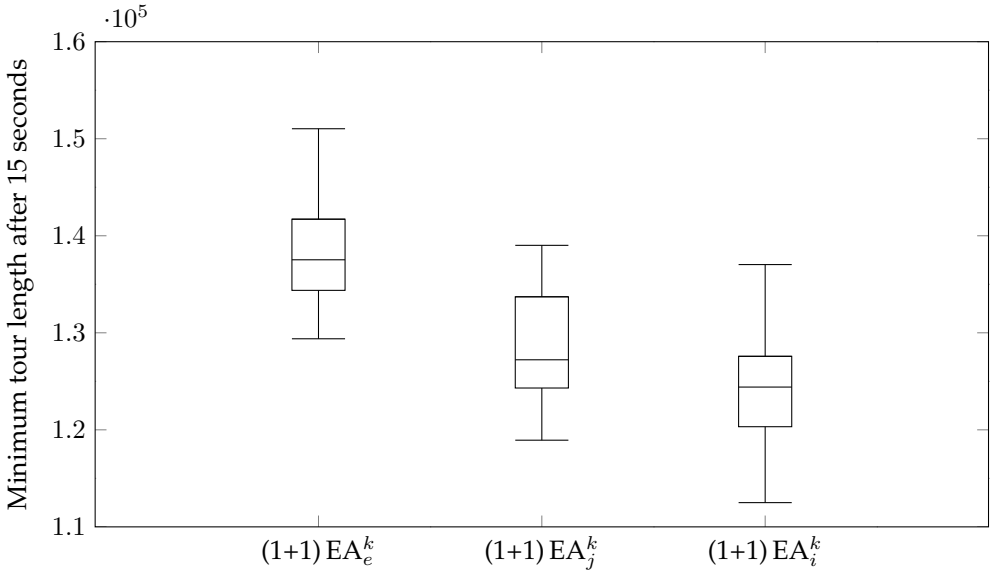

Figure 6: Comparison of solution quality (minimum tour length values) of (1+1) EA$_e^k$, (1+1) EA$_i^k$ and (1+1) EA$_j^k$ obtained within 15 seconds on 30 instances of size 100 and inner point percentage 40

strategy on each instance, and observe the effect of inner point percentage and size. To perform these analyses, we use the R statistical package [30]. The results are summarized in Table 2 where the control variables (instance size and inner point percentage) are indicated in the first column.

In the first result column of Table 2, we see that the (1+1) EA$_e^k$ consistently outperforms the (1+1) EA$_e^n$ with statistical significance on every TSP instance ($p < 0.001$). In such cases, with at least 99.9% confidence we reject the null hypothesis that the (1+1) EA$_e^k$ does not perform better than the (1+1) EA$_e^n$ in favor of the alternative hypothesis that (1+1) EA$_e^k$ performs better than (1+1) EA$_e^n$. The W-values indicate positive rank sums of the difference of two pairs ((1+1) EA$_e^k$and (1+1) EA$_e^n$). Greater values mean higher differences in algorithm performance.

For the inversion operator, the rank sum values in Table 2 are increasing as a function of instance size. This is a result of the widening gap between solution quality in these experiments and shows the scalability of our approaches compared to the classical (1+1) EA. Furthermore, we observe that when the inner point percentage increases to 40%, the performance of the (1+1) EA^k begins to degrade. This is reflected in the large p-values for these instances where we can no longer conclude that the (1+1) EA^k variant is outperforming the classical (1+1) EA. We expect such behavior as the inner point density increases. Interestingly, the (1+1) EA^k using the exchange operator and jump operator is still performing surprisingly well compared to the classical EA for this level of inner point percentage.

To assess the effect of the mutation strategies on the performance of the (1+1) EA^k, we conduct another one-tailed Wilcoxon signed rank test for the pairwise comparison of the three variants of the (1+1) EA^k. We present these results in Table 3. We cannot conclude from this test that the inversion operator performs better than the jump op-

instance	exchanges		inversions		jumps	
n / % inner	W	p	W	p	W	p
25 / 5%	465	*	91	*	253	*
25 / 10%	465	*	336	*	465	*
25 / 20%	465	*	366	0.003	465	*
25 / 30%	465	*	293	0.11	465	*
25 / 40%	465	*	172	0.894	465	*
50 / 5%	465	*	335	*	435	*
50 / 10%	465	*	435	*	465	*
50 / 20%	465	*	431	*	465	*
50 / 30%	465	*	217	0.627	465	*
50 / 40%	465	*	33	1	465	*
100 / 5%	465	*	460	*	465	*
100 / 10%	465	*	462	*	465	*
100 / 20%	465	*	399	*	465	*
100 / 30%	465	*	147.5	0.960	461	*
100 / 40%	465	*	56	1	394	*

Table 2: Results of Wilcoxon signed rank tests for minimum tour length values within time limits (2, 3 and 15 seconds for 25, 50 and 100 sizes accordingly) for (1+1) $\text{EA}_e^n >$ (1+1) EA_e^k (exchanges), (1+1) $\text{EA}_i^n >$ (1+1) EA_i^k (inversions) and (1+1) $\text{EA}_j^n >$ (1+1) EA_j^k (jumps), positive rank sums (W) and confidence (p) values are displayed accordingly. A * denotes $p < 0.001$.

eration. However, the p-values in the final column suggest that, in most cases, jump mutation is performing better than exchange mutation with statistical significance.

5 Conclusion

In this paper, we have studied the runtime complexity of evolutionary algorithms on the Euclidean TSP. We have carried out a parameterized analysis that studies the dependence of the hardness of a problem instance on the number of inner points in the instance. Moreover, we have shown that under reasonable geometric constraints (low angle bounds), simple evolutionary algorithms solve planar TSP instances that are in convex position in polynomial time.

We have bounded the expected number of generations for a black-box $(\mu + \lambda)$ EA to solve a TSP instance with k inner points as $O(n \cdot A(\epsilon) \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{4k}(2k-1)!)$. Using the analysis, we have also introduced a mixed mutation strategy based on both permutation jumps and 2-opt moves which attains an improved bound on the expectation of $O(n \cdot A(\epsilon) \cdot \max\{(\mu/\lambda) \cdot n^2, 1\} + (\mu/\lambda) \cdot n^{2k}(k-1)!)$. Hence, with this paper, we have shown that the $(\mu + \lambda)$ EA in a black-box setting is a randomized XP-algorithm for the inner point parameterization of Deĭneko et al. [8].

Moving out of the black-box setting, we have shown that the addition of a small amount of domain knowledge (namely, the hull-respecting order of the outer points) allow us to design two fixed-parameter evolutionary algorithms for the TSP. The first approach is based on a recently proposed dynamic programming framework for evolutionary computing. It works with a very large population and is mainly of theoretical interest. Our second approach searches for a permutation of the inner points and uses

instance	test 1		test 2	
n / % inner	W	p	W	p
25 / 5%	0	1	0	1
25 / 10%	1	0.9688	14	0.0625
25 / 20%	64	0.999	315	*
25 / 30%	96	0.9981	454	*
25 / 40%	70	0.9998	463	*
50 / 5%	0	1	28	0.007812
50 / 10%	108	0.9713	304	0.0001238
50 / 20%	90	0.9988	464	*
50 / 30%	64	0.9999	465	*
50 / 40%	49	1	465	*
100 / 5%	109	0.9382	221	0.05741
100 / 10%	77	0.9996	460	*
100 / 20%	278	0.1799	456	*
100 / 30%	194	0.786	456	*
100 / 40%	310	0.02423	446	*

Table 3: Results of Wilcoxon signed rank tests for minimum tour length values within time limits (2, 3 and 15 seconds for 25, 50 and 100 sizes accordingly) for (1+1) EA_i^k $<$ (1+1) EA_j^k (test 1), (1+1) EA_j^k $<$ (1+1) EA_e^k (test 2) , positive rank sums (W) and confidence (p) values are displayed accordingly. A * denotes $p < 0.001$.

a dynamic programming algorithm that runs in time $O(kn)$ as part of the fitness evaluation. Although the theoretical bound for the second fixed-parameter algorithm is worse than the one for the large population fixed-parameter EA, we consider it more applicable in practice. Our experimental investigations for the (1+1) evolutionary algorithms analyzed in this paper suggest that working with a permutation on the inner points can lead to significantly better results.

References

[1] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. Concorde TSP solver [Computer software]. `http://www.tsp.gatech.edu/concorde`.

[2] Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Company, 2011.

[3] Vineet Bafna and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal of Computing*, 25(2):272–289, 1996.

[4] Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur, and Frank Neumann. Computing single source shortest paths using single-objective fitness functions. In Thomas Jansen, Ivan Garibay, R. Paul Wiegand, and Annie S. Wu, editors, *Proceedings of the Tenth International Workshop on Foundations of Genetic Algorithms (FOGA'09)*, pages 59–66, Orlando, USA, 2009. ACM Press.

[5] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63, January 1962.

[6] Barun Chandra, Howard Karloff, and Craig Tovey. New results on the old k-opt algorithm for the traveling salesman problem. *SIAM Journal on Computing*, 28(6):1998–2029, 1999.

[7] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, third edition, 2008.

[8] Vladimir G. Deĭneko, Michael Hoffman, Yoshio Okamoto, and Gerhard J. Woeginger. The traveling salesman problem with few inner points. *Operations Research Letters*, 34:106–110, 2006.

[9] Benjamin Doerr, Anton V. Eremeev, Frank Neumann, Madeleine Theile, and Christian Thyssen. Evolutionary algorithms and dynamic programming. *Theoretical Computer Science*, 412(43):6020–6035, 2011.

[10] Benjamin Doerr, Edda Happ, and Christian Klein. Tight analysis of the (1+1)-EA for the single source shortest path problem. *Evolutionary Computation*, 19(4):673–691, 2011.

[11] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[12] Rodney G. Downey and Michael R. Fellows. Parameterized complexity after (almost) 10 years: Review and open questions. In *Proceedings of Combinatorics, Computation and Logic, DMTCS'99 and CATS'99*, volume 21 of *Australian Computer Science Communications*, pages 1–33, Singapore, 1999. Springer-Verlag.

[13] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the optimization of unimodal functions with the (1 + 1) Evolutionary Algorithm. In A. E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN'98)*, volume 1498 of *Lecture Notes in Computer Science*, pages 13–22. Springer, 1998.

[14] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[15] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SIAM'07)*, pages 1295–1304. Society for Industrial and Applied Mathematics, 2007.

[16] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer-Verlag, 2006.

[17] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[18] Jun He and Xin Yao. Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1-2):59–97, 2003.

[19] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. On the choice of offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13(4):413–440, 2005.

[20] Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt. Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence*, 6(1):1–21, 2012.

[21] Stefan Kratsch, Per Kristian Lehre, Frank Neumann, and Pietro Simone Oliveto. Fixed parameter evolutionary algorithms and maximum leaf spanning trees: A matter of mutation. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *Proceedings of the Eleventh Conference on Parallel Problem Solving from Nature (PPSN'10)*, volume 6238 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 2010.

[22] Stefan Kratsch and Frank Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. In Franz Rothlauf, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 293–300. ACM Press, 2009.

[23] Dániel Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters*, 36(1):31–36, 2008.

[24] Samadhi Nallaperuma, Andrew M. Sutton, and Frank Neumann. Fixed-parameter evolutionary algorithms for the euclidean traveling salesperson problem. In *IEEE Congress on Evolutionary Computation (CEC'13)*, pages 2037–2044. IEEE, 2013.

[25] Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

[26] Frank Neumann and Ingo Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[27] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

[28] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[29] Louis V. Quintas and Fred Supnick. On some properties of shortest Hamiltonian circuits. *The American Mathematical Monthly*, 72(9):977–980, 1965.

[30] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[31] Jens Scharnow, Karsten Tinnefeld, and Ingo Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3(4):349–366, 2004.

[32] Andrew M. Sutton, Jareth Day, and Frank Neumann. A parameterized runtime analysis of evolutionary algorithms for MAX-2-SAT. In Terence Soule and Jason H. Moore, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'12)*, pages 433–440. ACM, 2012.

[33] Andrew M. Sutton and Frank Neumann. A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI'12)*, pages 1105–1111. AAAI Press, 2012.

[34] Andrew M. Sutton and Frank Neumann. A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling. In *Proceedings of the Twelfth International Conference on Parallel Problem Solving from Nature (PPSN'12)*. Springer, 2012. To appear.

[35] Madeleine Theile. Exact solutions to the traveling salesperson problem by a population-based evolutionary algorithm. In *Proceedings of the 9th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP'09)*, EvoCOP '09, pages 145–155, Berlin, Heidelberg, 2009. Springer-Verlag.

[36] Jan van Leeuwen and Anneke A. Schoone. Untangling a traveling salesman tour in the plane. In *Proceedings of the Seventh International Workshop on Graph-Theoretic Concepts in Computer Science (WG'81)*, 1981.

[37] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[38] Carsten Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proceedings of the Twenty-Second Symposium on Theoretical Aspects of Computer Science (STACS'05)*, volume 3404 of *Lecture Notes on Computer Science*, pages 44–56. Springer, 2005.

[39] Yang Yu and Zhi-Hua Zhou. A new approach to estimating the expected first hitting time of evolutionary algorithms. In *Proceedings of the Twenty-First Conference on Artificial Intelligence (AAAI'06)*. AAAI Press, 2006.

Appendix

Proof of Lemma 4. Suppose $\{x(i-1), x(i)\}$ and $\{x(j), x(j+1)\}$ intersect in $C(x)$. Let $y = \sigma^{\mathrm{I}}_{ij}[x]$. Then

$$C(x) \setminus C(y) = \{\{x(i-1), x(i)\}, \{x(j), x(j+1)\}\}, \qquad \text{and}$$
$$C(y) \setminus C(x) = \{\{x(i-1), x(j)\}, \{x(i), x(j+1)\}\}.$$

By Proposition 1, since $\{x(i-1), x(i)\}$ and $\{x(j), x(j+1)\}$ intersect, the two new edges introduced to $C(y)$ by σ^{I}_{ij} do not intersect. □

Proof of Lemma 5. We first claim that the number of permutations on n points in which some subset of $p < n$ points all remain in the same fixed order is $\binom{n}{n-p}(n-p)!$. To construct such a permutation, each of the $n-p$ "free" points can be placed before all p points, or after any of the p points. Thus, for each of the $n-p$ free points, there are $p+1$ positions relative to the non-free points from which we can choose (with replacement) yielding $\binom{(p+1)+(n-p)-1}{n-p} = \binom{n}{n-p}$ choices for relative placement. For each such choice, there are $(n-p)!$ distinct ways to order the free points.

Fix some point $v \in \mathsf{Out}(V)$. Suppose $C(x)$ is intersection-free. There is a transformation (specifically, either the identity, or a permutation with exactly one cycle and no

fixed points) that maps x to a sequence z in which v appears to the left of any other element in $\mathsf{Out}(V)$, yet $C(x) = C(z)$. Hence the number of distinct intersection-free tours is bounded by the number of sequences of points in V where the elements of $\mathsf{Out}(V)$ appear in hull-order, and v appears to the left of any other element of $\mathsf{Out}(V)$. By the above claim, there are exactly $\binom{n}{k}k!$ such permutations and the lemma is proved. □

Proof of Lemma 6. We begin by proving that there are at most k jump operations to transform x into an optimal permutation. By Lemma 1, since $C(x)$ is intersection-free, x respects hull-order. Let $x^\star$ be an optimal permutation such that the elements in $\mathsf{Out}(V)$ have the same linear order in $x^\star$ as they do in x. Then x can be transformed into $x^\star$ by moving each of the k inner points into their correct position.

The claim that at most $2k$ inversions suffice to transform x into an optimal permutation immediately follows from the fact that any jump operation can be simulated by at most two consecutive inversion operations. In particular,

$$\sigma^{\mathrm{J}}_{ij}[x] = \begin{cases} \sigma^{\mathrm{I}}_{ij}[x] & \text{if } |i-j| = 1; \\ \sigma^{\mathrm{I}}_{i(j-1)}[\sigma^{\mathrm{I}}_{ij}[x]] & \text{if } i-j < 1; \\ \sigma^{\mathrm{I}}_{(j+1)i}[\sigma^{\mathrm{I}}_{ji}[x]] & \text{if } i-j > 1. \end{cases}$$

Since a sequence of at most k jump operations can be simulated by a sequence of at most $2k$ inversion operations, the claim is proved. □

Proof of Lemma 7. The inversion σ^{I}_{ij} removes intersecting edges $\{u, v\}$ and $\{s, t\}$ from $C(x)$ and replaces them with the pair $\{s, u\}$ and $\{t, v\}$ to form $C(y)$. We label the point at which the original edges intersect as p.

Denote as θ_u the angle between the line segments that join u to p and u to s. Similarly, denote as θ_s the angle between the line segments that join s to p and s to u (see Figure 1). Since all angles are strictly positive, the points u, s, and p form a nondegenerate triangle with angles θ_s, θ_u, and $(\pi - (\theta_s + \theta_u))$. By the law of sines we have

$$\frac{d(s,u)}{\sin\left(\pi - (\theta_s + \theta_u)\right)} = \frac{d(s,u)}{\sin\left(\theta_s + \theta_u\right)} = \frac{d(u,p)}{\sin(\theta_s)} = \frac{d(s,p)}{\sin(\theta_u)}.$$

Hence,

$$d(u,p) + d(s,p) = d(s,u)\left(\frac{\sin(\theta_s) + \sin(\theta_u)}{\sin\left(\theta_s + \theta_u\right)}\right). \tag{6}$$

Since u, s, and p form a triangle, $0 < (\theta_s + \theta_u) < \pi$ and we have

$$\begin{aligned} &0 < \sin(\theta_s) < 1 && \text{since } 0 < \theta_s < \pi, \\ &0 < \sin(\theta_u) < 1 && \text{since } 0 < \theta_u < \pi, \\ &0 < \sin\left(\theta_s + \theta_u\right) < 1 && \text{since } 0 < \theta_s + \theta_u < \pi. \end{aligned}$$

Furthermore, since V is angle-bounded by $0 < \epsilon < \pi - \epsilon$, by (6),

$$d(u,p) + d(s,p) > d(s,u)\left(\frac{\sin(\epsilon) + \sin(\epsilon)}{\sin(\epsilon + \epsilon)}\right) = d(s,u)\sec(\epsilon) > d(s,u). \tag{7}$$

The rightmost inequality holds since $\epsilon > 0$ and $\epsilon + \epsilon < \pi$ and the function $\sec(x)$ is strictly positive in this regime. Since there is also a nondegenerate triangle formed by the points t, v, and p, a symmetric argument holds and thus

$$d(t,p) + d(v,p) > d(t,v)\sec(\epsilon) > d(t,v). \tag{8}$$

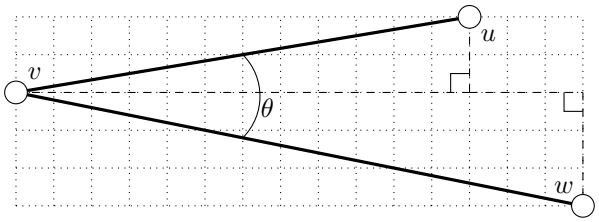

Figure 7: If the slope of the lines from v to u and v to w are of opposite sign, they form the hypotenuses of two right triangles and $\theta \geq 2\arctan((m-1)^{-1})$.

Combining Equations (7) and (8) we have

$$\begin{aligned} f(x) - f(y) &= [d(u,v) + d(s,t)] - [d(t,v) + d(s,u)] \\ &= d(u,p) + d(v,p) + d(t,p) + d(s,p) - (d(t,v) + d(s,u)) \\ &> (d(t,v) + d(s,u))\sec(\epsilon) - (d(t,v) + d(s,u)) > 0 \end{aligned}$$

The constraint that the difference is strictly positive follows directly from Equations (7) and (8) (this constraint prevents a trivial lower bound, which would become problematic in later proofs). Hence,

$$\begin{aligned} f(x) - f(y) &> [d(t,v) + d(s,u)]\,(\sec(\epsilon) - 1) \\ &\geq 2d_{min}\,(\sec(\epsilon) - 1) = 2d_{min}\left(\frac{1-\cos(\epsilon)}{\cos(\epsilon)}\right). \end{aligned}$$

□

Proof of Lemma 8. The grid imposes a coordinate system on V in which the concept of line slope is well-defined. Let $u, v, w \in V$ be arbitrary points. We consider the angle θ at point v formed by the lines from v to u and v to w. Let s_1 and s_2 denote the slope of these lines, respectively. If the slopes are of opposite sign, then $\theta \geq 2\arctan((m-1)^{-1})$ since the lines form hypotenuses of two right triangles with adjacent sides of length at most $m-1$ and opposite sides with length at least 1 (see Figure 7).

We now consider the case where the slopes are nonnegative. The nonpositive case is handled identically (or by simply changing the sign of the slopes by the appropriate transformation). Without loss of generality, assume $s_1 > s_2 \geq 0$. Equality is impossible since u, v, and w cannot be collinear. Since the points lie on an $m \times m$ grid, s_1 and s_2 must be ratios of whole numbers at most $m-1$, say $s_1 = a/b$ and $s_2 = c/d$. The angle at point v is $\theta = \arctan(a/b) - \arctan(c/d) = \arctan\left(\frac{ad-cb}{bd+ac}\right)$. The minimum positive value for the expression $(ad-cb)/(bd+ac)$ over the integers from 0 to $m-1$ is $\frac{1}{2(m-2)^2}$. Since the inverse of the tangent is monotone, the minimum nonzero angle must be $\theta \geq \arctan\left(1/(2(m-2)^2)\right)$. □

Proof of Lemma 9. It follows from Lemma 8 that the angle bound on V is $\epsilon = \arctan\left(1/(2(m-2)^2)\right)$. Since $\cos(\arctan(x)) = 1/\sqrt{1+x^2}$ we have

$$\frac{\cos(\epsilon)}{1-\cos(\epsilon)} = \frac{2(m-2)^2}{\sqrt{1+4(m-2)^4} - 2(m-2)^2}.$$

and since $z/(\sqrt{1+z^2} - z) = O(z^2)$, setting $z = 2(m-2)^2$ completes the proof. □