Theoretical Analysis of Local Search and Simple Evolutionary Algorithms for the Generalized Travelling Salesperson Problem

Mojgan Pourhassan mojgan.pourhassan@adelaide.edu.au
Optimisation and Logistics, The University of Adelaide, Adelaide, SA 5005, Australia

Frank Neumann frank.neumann@adelaide.edu.au
Optimisation and Logistics, The University of Adelaide, Adelaide, SA 5005, Australia

Abstract

The generalized travelling salesperson problem is an important NP-hard combinatorial optimization problem for which meta-heuristics, such as local search and evolutionary algorithms, have been used very successfully. Two hierarchical approaches with different neighbourhood structures, namely a Cluster-Based approach and a Node-Based approach, have been proposed by [10] for solving this problem. In this paper, local search algorithms and simple evolutionary algorithms based on these approaches are investigated from a theoretical perspective. For local search algorithms, we point out the complementary abilities of the two approaches by presenting instances where they mutually outperform each other. Afterwards, we introduce an instance which is hard for both approaches when initialized on a particular point of the search space, but where a variable neighbourhood search combining them finds the optimal solution in polynomial time. Then we turn our attention to analysing the behaviour of simple evolutionary algorithms that use these approaches. We show that the Node-Based approach solves the hard instance of the Cluster-Based approach presented in [3] in polynomial time. Furthermore, we prove an exponential lower bound on the optimization time of the Node-Based approach for a class of Euclidean instances.

Keywords

Generalized Travelling Salesperson Problem, Parameterised Complexity Analysis, Bi-level Optimisation, Evolutionary Algorithms, Combinatorial Optimisation.

1 Introduction

Evolutionary algorithms and other metaheuristics have been applied to a wide range of combinatorial optimization problems. Understanding the behaviour of metaheuristics on problems from combinatorial optimization is a challenging task due to the large amount of randomness involved in these algorithms.

During the past decade, lot of progress has been made on the analysis of evolutionary algorithms and ant colony optimization for problems of classical benchmark functions and problems from combinatorial optimization [2, 11]. Results have been achieved for classical polynomially solvable problems such as sorting, shortest path, minimum spanning trees and maximum matching as well as for some of the best known NP-hard combinatorial optimization problems such as vertex cover, makespan scheduling, and the travelling salesperson problem [16, 23].

Furthermore, bio-inspired computing methods have been studied in the context of parameterized complexity [5, 14, 15]. This approach allows to study the runtime in dependence of some structural parameters of the given instances and helps to classify when an instance gets hard for the examined algorithm. Results have been obtained for some of the most prominent NP-hard combinatorial optimization problems such as vertex cover, makespan scheduling [21] and the Euclidean Travelling Salesperson problem [22]. The parameterized analysis has also been used to study the generalized minimum spanning tree problem (GMSTP) and the generalized travelling salesperson problem (GTSP) [3]. This paper aims to investigate the latter problem in more detail.

The generalized travelling salesperson problem (GTSP) is given by a set of cities with distances between them. The cities are divided into clusters and the goal is to find a tour of minimal cost that visits one city from each cluster exactly once. Hu and Raidl [10] have presented two hierarchical approaches for solving the GTSP: *Cluster-Based* approach, which uses a permutation on the different clusters in the upper level and finds the best node selection for that permutation on the lower level, and *Node-Based* approach, which selects a node for each cluster and then works on finding the best permutation of the chosen nodes. Combining the two hierarchical approaches, they have also presented a variable neighbourhood search algorithm for solving the GTSP. With this paper, we contribute to the theoretical understanding of local search methods and simple evolutionary algorithms based on these hierarchical approaches for GTSP. The analysis on local search methods (based on the conference version [19]) is presented in Section 3. We investigate the local search methods by presenting instances for which the two approaches mutually outperform each other. We also present a situation where both Cluster-Based and Node-Based local search approaches stick to a local optimum, but the combination of the two approaches solves the problem to optimality.

After investigating local search methods, this paper extends the conference version [19] by investigating simple evolutionary algorithms in Section 4. A (1+1) EA using the Cluster-Based approach is analysed in [3] by presenting upper and lower bounds for the optimization time of the algorithm. In this paper, we show that the worst case instance presented there for the Cluster-Based approach can be solved in polynomial time by means of the Node-Based approach; hence, there are instances of the problem which the latter approach can solve more efficiently. Then we provide a lower bound analysis of this approach for the Euclidean generalized travelling salesperson problem.

Showing lower bounds for the Euclidean travelling salesperson problem has been shown to be quite difficult. Englert et al. [6] have shown that there are instances of the Euclidean TSP for which finding a local optimal solution takes exponential time by means of a deterministic local search algorithm based on 2-opt. In this paper we present a Euclidean class of instances where a simple evolutionary algorithm using the Node-Based approach requires exponential time with respect to the number of clusters. To our knowledge currently an exponential lower bound for solving TSP by a stochastic search algorithm is available only for ant colony optimization in the non-Euclidean case [13]. Our instance for the GTSP places nodes on two different circles with radius r and r' of a given centre. Exploiting the geometric properties of this instance class, we show by multiplicative drift analysis [4] that the evolutionary algorithm under investigation ends up in a local optimum which has different chosen nodes for almost all clusters. Leaving such a local optimum requires exponential time for many mutation-based evolutionary algorithms and leads to an exponential lower bound with respect to the number of clusters for the investigated algorithm.

The outline of this paper is as follows. Section 2 introduces the problem and the algorithms that are subject to our investigations. Our runtime analysis for local search methods and simple evolutionary algorithms are presented in Section 3 and Section 4 respectively. Finally, we finish with some concluding remarks in Section 5.

2 Problem and Algorithms

The generalized travelling salesperson problem (GTSP) is a combinatorial optimization problem with applications in routing, design of ring networks, sequencing of computer files, manufacture planning [8]. The input is given by a complete undirected graph $G = (V, E, c)$ with cost function $c\colon E \to \mathbb{R}^+$ on the edges and a partitioning of the set of nodes V into m clusters $V_1, V_2, \ldots, V_m$ such that $V = \bigcup_{i=1}^m V_i$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. The aim is to find a tour of minimum costs that contains exactly one node from each cluster.

A candidate solution for this problem consists of two parts. The set of *spanning nodes*, $P = \{p_1, \ldots, p_m\}$ where $p_i \in V_i$, and the *permutation* of the clusters, $\pi = (\pi_1, \ldots, \pi_m)$, which makes a Hamiltonian cycle on $G[P] = G(P, \{e \in E \mid e \subseteq P\}, c)$. Here, $G[P]$ is the sub-graph induced by P consisting of all nodes in P and all edges between them. Following [10], we represent a candidate solution as $S = (P, \pi)$. Let p_{π_i} be the chosen node for cluster V_{π_i}, $1 \leq i \leq m$. Then the cost of a solution $S = (P, \pi)$ is given by $c(S) = c(p_{\pi_m}, p_{\pi_1}) + \sum_{i=1}^{m-1} c(p_{\pi_i}, p_{\pi_{i+1}})$.

There are two hierarchical approaches for solving this problem [10], the Cluster-Based approach and the Node-Based approach. In the former, an upper level algorithm searches for finding the best permutation of clusters, while a lower level algorithm finds the optimal spanning node set. In the two levels of Node-Based approach, these tasks are swapped. In the following, we describe four algorithms that make use of these two hierarchical approaches. We analyse these algorithms with respect to the (expected) number of iterations on the upper level, until they have found an optimal solution and call this the (expected) optimization time of the algorithms.

2.1 Cluster-Based Local Search

In the Cluster-Based approach, constructing the permutation of clusters constitutes the upper level and the node selection is done in the lower level [10]. Let $\pi = (\pi_1, \cdots, \pi_m)$ be a permutation of the m clusters. The 2-opt neighbourhood of π is given by

$$N(\pi) \quad = \quad \{\pi' \mid \pi' = (\pi_1, \cdots, \pi_{i-1}, \pi_j, \pi_{j-1}, \cdots, \pi_i, \pi_{j+1}, \cdots, \pi_m), \quad 1 \leq i < j \leq m\}$$

The Cluster-Based local search (CBLS) algorithm working with this neighbour-

Algorithm 1: Cluster-Based Local Search (CBLS)

```
1 Choose a permutation $\pi = (\pi_1, \ldots, \pi_m)$;
2 Find the optimal set of spanning nodes $P$ with respect to $\pi$ to obtain the
  solution $S = (P, \pi)$;
3 for $\pi' \in N(\pi)$ do
4     Find an optimal set of nodes $P' = \{p'_1, \ldots, p'_m\}$ with respect to $\pi'$ to obtain
      the solution $S' = (P', \pi')$;
5     if $c(S') < c(S)$ then
6         $S = S'$;
7         GO TO 3
```

Algorithm 2: Node Exchange Neighbourhood Local Search (NEN-LS)

```
1 Choose $P = \{p_1, p_2, \ldots, p_m\}, \quad p_i \in V_i$;
2 Let $\pi$ be the permutation of clusters obtained by performing a 2-opt local search
  on $G[P]$ and $S = (P, \pi)$ be the resulting solution;
3 for $P' \in N'(P)$ do
4     Let $\pi'$ be the permutation of clusters obtained from $\pi$ by performing a 2-opt
      local search on $G[P']$ and $S' = (P', \pi')$ be the resulting solution;
5     if $c(S') < c(S)$ then
6         $S = S'$;
7         GO TO 3
```

Algorithm 3: Node Exchange Neighbourhood Local Search* (NEN-LS*)

```
1 Choose $P = \{p_1, p_2, \cdots, p_m\}, \quad p_i \in V_i$;
2 Find a minimum-cost permutation $\pi$ for $G[P]$ and let $S = (P, \pi)$ be the resulting
  solution;
3 for $P' \in N'(P)$ do
4     Find a minimum-cost permutation $\pi'$ for $G[P']$ and let $S' = (P', \pi')$ be the
      resulting solution;
5     if $c(S') < c(S)$ then
6         $S = S'$;
7         GO TO 3
```

hood structure, given in Algorithm 1, starts with an initial permutation of clusters. At each step, a new permutation π' is selected from the 2-opt neighbourhood of π, the current permutation of clusters. Then the lower level uses a shortest path algorithm to find the best spanning node set. Hu and Raidl [10] have applied an incremental bidirectional shortest path calculation for this purpose. The shortest path algorithm of [12] is another option, which is an improved version of dynamic programming algorithm given in [7] for finding an optimal set of spanning nodes for a given permutation in time $O(n^3)$. The new solution $S' = (P', \pi')$ replaces the old one if it is less costly, and the algorithm terminates if no better solution can be found in the 2-opt neighbourhood of π.

2.2 Node-Based Local Search

In the Node-Based approach [10], selection of the spanning nodes is done in the upper level and the lower level consists of finding a shortest tour on the spanning nodes. Given a spanning nodes set P, in the Node-Based local search algorithm, the upper level performs a local search based on the node exchange neighbourhood $N'(P)$, which is defined as

$$N'(P) \quad = \quad \{P' \mid P' = \{p_1, \cdots, p_{i-1}, p'_i, p_{i+1}, \ldots, p_m\}, \ \ p'_i \in V_i \setminus \{p_i\}, \ \ 1 \le i \le m\}$$

Note that the lower level involves solving the classical TSP; it therefore poses in general an NP-hard problem on its own. For our theoretical investigations, we consider two algorithms: NEN-LS (Node Exchange Neighbourhood Local Search) and NEN-LS*, presented in Algorithm 2 and Algorithm 3, respectively. NEN-LS computes

Algorithm 4: Variable Neighbourhood Search (VNS)

```
1 Choose an initial solution $S = (P, \pi)$;
2 $l = 1$;
3 while $l \leq 2$ do
4     for $S' \in N_l(S)$ do
5         if $c(S') < c(S)$ then
6             $S = S'$;
7             $l = 1$;
8             GO TO 3
9     $l = l + 1$
```

a permutation on the lower level using 2-opt local search and is therefore not guaranteed to reach an optimal permutation π for a given spanning node set P. NEN-LS* uses an optimal solver to find an optimal permutation π for a given spanning node set P. Such a permutation can be obtained in time $O(m^2 2^m)$ using dynamic programming [9] and is practical if the number of clusters is small. We use NEN-LS* and show where it gets stuck in local optima even if the travelling salesperson problem on the lower level is solved to optimality.

NEN-LS and NEN-LS* start with a spanning node set P and search for a good or optimal permutation with respect to P. Then each solution $P' \in N'(P)$ together with its permutation π' is considered and $S' = (P', \pi')$ replaces the current solution $S = (P, \pi)$ if it is of smaller cost. Both algorithms terminate if there is no improvement possible in the neighbourhood $N'(P)$ of the current solution P.

2.3 Variable Neighbourhood Search

Now we describe the combination of two approaches into variable neighbourhood search, which is introduced in [10]. Two neighbourhood structures of CBLS and NEN-LS are used in this algorithms, where the NEN-LS neighbourhood is used only when the algorithm is in a local optimum with respect to the CBLS neighbourhood.

Let $S = (P, \pi)$ be a solution to the GTSP. We define the two neighbourhoods N_1 and N_2 based on the 2-opt neighbourhood N and the node exchange neighbourhood N' as

- $N_1(S) = \{S' = (P', \pi') \mid \pi' \in N(\pi), P' = \text{optimal set of nodes with respect to } \pi'\}$
- $N_2(S) = \{S' = (P', \pi') \mid P' \in N'(P), \pi' = \text{order of clusters obtained by 2-opt from } \pi \text{ on } G[P']\}$

Combining the two local searches of Cluster-Based approach and Node-Based approach is done by alternating between N_1 and N_2. Since the computational complexity of finding P' for solutions in neighbourhood N_1 is lower than that of finding π' for solutions in neighbourhood N_2, the first neighbourhood to search is N_1. When a local optimum has been found with respect to that neighbourhood, then N_2 is searched. The resulting variable neighbourhood search (VNS) algorithm is given in Algorithm 4.

2.4 Node-Based (1+1) EA

In the Node-Based approach, selecting the spanning nodes is done in the upper level and the corresponding shortest Hamiltonian cycle is found in the lower level. The

```
Algorithm 5: Node-Based (1+1) EA
1 Let $P = \{p_1, p_2, \dots, p_m\}$, where $p_i \in V_i$ are chosen uniformly at random;
2 Let $\pi$ be the optimal permutation for $G[P]$ and $S = (P, \pi)$ be the resulting
  solution;
3 while termination condition not satisfied do
4     $P' \leftarrow P$;
5     for $i \in \{1, \cdots, m\}$ do
6         with probability $1/m$, sample $p'_i \sim$ Unif($V_i$);
7     Let $\pi'$ be the optimal permutation for $G[P']$ and $S' = (P', \pi')$ be the
      resulting solution;
8     if $c(S') < c(S)$ then
9         $S = S'$;
```

Node-Based (1+1) EA is presented in Algorithm 5. In contrast to Node-Based local search algorithm of Section 2.2, the upper level uses the (1+1) EA to search for the best spanning set instead of a local search method; hence, more than one change on the spanning set is possible on the upper level, at each iteration of the algorithm. The condition for accepting the new solution is a strict improvement.

Note that the lower level consists of an NP-hard problem; hence, when showing polynomial upper bounds on the expected optimization time of this algorithm, we only consider instances where the lower level can be solved in polynomial time. For general case, there exist very effective solvers for TSP such as Concorde [1], that can be used in the lower level. Note that the lower level does not need to solve an NP-hard problem in the Cluster-Based approach. Nevertheless, we prove that there are instances that can be solved in polynomial time with Node-Based (1+1) EA, while the Cluster-Based (1+1) EA [3] needs exponential time to find an optimal solution for them.

3 Local Search Methods

This section presents the analysis on the behaviour of the local search methods. Cluster-Based and Node-Based Local search algorithms and also the variable neighbourhood search algorithm presented in Sections 2.1, 2.2 and 2.3 are investigated in this section.

3.1 Benefits of NEN-LS

In this section, we present an instance of the problem that can not be solved by CBLS. In contrast to this, NEN-LS finds an optimal solution in polynomial time.

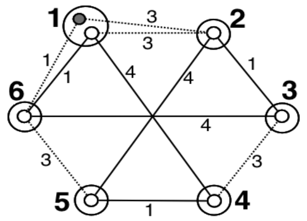


Figure 1: G_1, an easy instance for NEN-LS and a hard instance for CBLS

We consider the undirected complete graph, $G_1 = (V, E)$ which is illustrated in

Figure 1. The graph has n nodes and 6 clusters V_i, $1 \leq i \leq 6$. Cluster V_1 contains $n/12$ white and $n/12$ grey nodes. We denote by V_{1W} the subset of white nodes and by V_{1G} the subset of grey nodes of cluster V_1. Each other cluster V_j, $2 \leq j \leq 6$, consists of $n/6$ white nodes. The node set $V = \cup_{i=1}^{6} V_i$ of G_1 consists of nodes of all clusters. For simplicity, Figure 1 shows only one node for each group of similar nodes with similar edges in the picture. The edge set E consists of 4 types of edges which we define in the following.

- Type A: Edges of this type have a cost of 1. All edges between clusters 2 and 3, and between clusters 4 and 5 and also between clusters 6 and 1, are of this type.

  $$A = \{\{v_i, v_j\} \mid (v_i \in V_{1W} \cup V_{1G} \wedge v_j \in V_6) \vee (v_i \in V_2 \wedge v_j \in V_3) \vee (v_i \in V_4 \wedge v_j \in V_5)\}$$

- Type B: Edges of this type have a cost of 3. All edges connecting the nodes of cluster 1 to cluster 2 are of this type. So are the edges that connect nodes of cluster 3 to 4 and cluster 5 to 6.

  $$B = \{\{v_i, v_j\} \mid (v_i \in V_{1W} \cup V_{1G} \wedge v_j \in V_2) \vee (v_i \in V_3 \wedge v_j \in V_4) \vee (v_i \in V_5 \wedge v_j \in V_6)\}$$

- Type C: Edges of this type have a cost of 4. All edges between nodes of cluster 2 and 5 and also between clusters 3 and 6 are of this type. All edges that connect white nodes of the first cluster to nodes of the fourth cluster are also of this type.

  $$C = \{\{v_i, v_j\} \mid (v_i \in V_{1W} \wedge v_j \in V_4) \vee (v_i \in V_2 \wedge v_j \in V_5) \vee (v_i \in V_3 \wedge v_j \in V_6)\}$$

- Type D: Edges of this type have a large cost of 100. All edges other than those of type A or B or C in this complete graph, including the edges between grey nodes of the first cluster and the nodes of the fourth cluster, are of Type D.

  $$D = E \setminus \{A \cup B \cup C\}$$

We say that a permutation $\pi = (\pi(1), \ldots, \pi(n))$ visits the cities in consecutive order iff $\pi(i+1) = (\pi(i) \bmod n) + 1$, $1 \leq i \leq n$ and say that $\pi = (\pi(1), \ldots, \pi(n))$ visits the cities in reverse-consecutive order iff $\pi(i) = (\pi(i+1) \bmod n) + 1$, $1 \leq i \leq n$.

We now define a property, and then in Theorem 2 we analyse the behaviour of CBLS on G_1.

Property 1. *For the instance G_1, each solution visiting the clusters in consecutive or reverse-consecutive order is optimal.*

Proof. The graph consists of 6 clusters which implies that 6 edges are needed for a tour. The least costly edges are of type A, which are available only between 3 pairs of clusters. The second least costly type of edge is B with weights of 3. This implies that no tour can be less costly than $3 \cdot 1 + 3 \cdot 3 = 12$, which is the cost of every solutions with a permutation in consecutive or reverse-consecutive order. □

Theorem 2. *Starting with the solution consisting of only white nodes and the permutation $\pi = (1, 4, 5, 2, 3, 6)$, CBLS is not able to achieve any improvement.*

Proof. Here we analyse the behaviour of CBLS on G_1, starting with all white nodes and a permutation $\pi = (1, 4, 5, 2, 3, 6)$. The initial solution contains three Type-A edges of cost 1 and three Type-C edges of cost 4. This implies a total cost of 15 which is not optimal. The edges belonging to this tour are marked solid in Figure 1. We claim that this solution is locally optimal, i.e. can not be improved by a 2-opt step.

When a 2-opt move is performed, depending on the different types of edges that are removed from the current tour, we show that the resulting tours have costs greater than 15.

Note, that all 3 edges of cost 1 are already used in the current permutation which means that no additional edge of cost 1 can be added. We inspect the different 2-opt steps with respect to the edges that are removed.

- If two edges of type A which have cost of 1 are removed, two other edges need to be added and the least costly edges that can be added have a weight of 3. This makes the total cost of the resulting solution to be at least $15 - 2 \cdot 1 + 2 \cdot 3 = 19$ which is greater than 15.
- If one edge of type A (weight 1) and one edge of type C (weight 4) are removed, again with the minimum two edges of cost 3 that are added, the total cost is at least $15 - 1 - 4 + 2 \cdot 3 = 16$ which is greater than 15.
- For removing two edges of Type C, there are three options, listed below. In all of them, the operation adds two edges of type D to the solution, making the total cost greater than 15.
  - Remove the edge between cluster 1 and cluster 4 and also the edge between cluster 2 and cluster 5. This 2-opt results in permutation $\pi' = (1, 5, 4, 2, 3, 6)$.
  - Remove the edge between cluster 1 and cluster 4 and also the edge between cluster 3 and cluster 6. This 2-opt results in permutation $\pi' = (1, 3, 2, 5, 4, 6)$.
  - Remove the edge between cluster 2 and cluster 5 and also the edge between cluster 3 and cluster 6. This 2-opt results in permutation $\pi' = (1, 4, 5, 3, 2, 6)$.

We have shown that no 2-opt step is accepted, which completes the proof. □

In contrast to the negative result for CBLS, we show that NEN-LS is able to reach an optimal solution when starting with the same solution.

Theorem 3. *Starting with $\pi = (1, 4, 5, 2, 3, 6)$, NEN-LS finds an optimal solution for the instance G_1 in expected time $O(n)$.*

Proof. Starting with a solution with only white nodes and the permutation of $\pi = (1, 4, 5, 2, 3, 6)$, no improvement can be found by a 2-opt local search (similar to the arguments in the proof of Theorem 2). Therefore, the lower level is already locally optimal and the solution does not change unless a grey node in cluster V_1 is selected.

Let $P = \{p_1, \cdots, p_6\}$ be the current set of spanning nodes. Selecting a grey node p'_1 for cluster V_1 leads to the set of spanning nodes $P' = \{p'_1, p_2, \cdots, p_6\}$. P' in combination with the the current permutation $\pi = (1, 4, 5, 2, 3, 6)$ has a total cost of 111 as there is one edge of type D with cost 100. We now show that starting from this solution and performing a 2-opt local search on the lower level results in an optimal solution.

In order to accept a new permutation on the lower level a solution of cost at most 111 has to be obtained. We do a case distinction according to the different types of edges that are removed in a 2-opt operation. If we only remove edges of type A and C, we reach a solution with total cost of greater than 111 using the arguments in the proof of Theorem 2. Hence, we only need to consider the case where at least one edge of type D is removed.

- There are two possibilities of removing one edge of type D and one of the edge of type C leading to the permutations $\pi' = (1,5,4,2,3,6)$ and $\pi'' = (1,3,2,5,4,6)$. Both have two edges of type D which implies a total cost of greater than 111 and are therefore rejected.
- Considering the case of removing the edge of type D and one of the edges of type A, the only applicable 2-opt move leading to a different permutation results in the permutation $\pi' = (1,2,5,4,3,6)$. The resulting solution has cost 16 and is therefore accepted.

After reaching permutation $\pi' = (1,2,5,4,3,6)$, the only acceptable 2-opt move leads to the global optimum $\pi_{opt} = (1,2,3,4,5,6)$.

The 2-opt neighbourhood for this instance has a constant size, as the number of clusters is constant. Moreover, all permutations that were investigated in the lower level, were either locally optimal with respect to the spanning nodes, or were improved only twice. Therefore, each lower level optimization is done in constant time. Furthermore, it takes expected time $O(n)$ on the upper level to select a grey node for the first cluster. As a result, the expected optimization time is bounded by $O(n)$. $\square$

3.2 Benefits of CBLS

We now introduce an instance where NEN-LS* with a random initial solution finds it hard to obtain an optimal solution, while CBLS with an arbitrary starting solution obtains an optimum in polynomial time. The instance $G_2 = (V, E)$ is illustrated in Figure 2. There are m clusters where $m > 2$, and all the clusters contain only 2 nodes; one white and one black. We refer to the white and black nodes of cluster i, $1 \leq i \leq m$, by v_{iW} and v_{iB}, respectively. We call cluster V_1 the costly cluster as edges connecting this cluster to others are more costly than edges connecting other clusters together. The edge set E of this complete graph is partitioned into 4 different types.

- Type A: Edges of this type have a weight of 1. All connections between white nodes of different clusters except cluster V_1 are of this type.
$$A = \{\{v_{iW}, v_{jW}\} \mid 2 \leq i, j \leq m\}$$
- Type B: Edges of this type have a weight of 2. All connections between black nodes of different clusters are of this type.
$$B = \{\{v_{iB}, v_{jB}\} \mid 1 \leq i, j \leq m\}$$
- Type C: Edges of this type have a weight of m. All edges between white nodes of the costly cluster and white nodes of other clusters are of this type.
$$C = \{\{v_{1W}, v_{iW}\} \mid 2 \leq i \leq m\}$$
- Type D: Edges of this type have a weight of m^2. All edges between a white and a black node are of this type.
$$D = E \setminus \{A \cup B \cup C\} = \{\{v_{iW}, v_{jB}\} \mid 1 \leq i, j \leq m\}$$

We first claim that the optimal solution consists of only black nodes. Then, we bring our main theorems on the runtime behaviour of solving this instance with the two mentioned approaches.

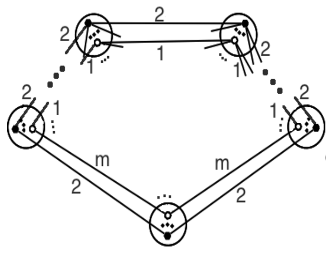


Figure 2: Graph G_2

Property 4. *For the graph G_2 any solution containing all black nodes is optimal.*

Proof. A solution that contains only black nodes has m edges of type B and therefore total cost of $2m$.

Choosing a combination of black and white nodes implies a connection of type D and therefore a solution of cost at least m^2. Choosing all white nodes implies 2 edges of cost m connected to cluster V_1 and $m-2$ edges of cost 1. Hence, the total cost of such a solution is $2m+(m-2)$ which implies that a solution selecting all black nodes is optimal. □

We now show that CBLS always finds an optimal solution due to selecting optimal spanning nodes in time $O(n^3)$.

Theorem 5. *Starting with an arbitrary permutation π, CBLS finds an optimal solution for G_2 in time $O(n^3)$.*

Proof. As mentioned in Property 4, visiting black nodes of the graph in any order is a globally optimal solution. For each permutation π the optimal set of nodes is given by all black nodes and found when constructing the first spanning node set. This set is constructed in time $O(n^3)$ by the shortest path algorithm given in [12]. □

In contrast to the positive result for CBLS, NEN-LS* is extremely likely to get stuck in a local optimum if the initial spanning node set is chosen uniformly at random. Note, that NEN-LS* is even using an exact solver for the lower level.

Theorem 6. *Starting with a spanning node set P chosen uniformly at random, NEN-LS* gets stuck in a local optimum of G_2 with probability $1-e^{-\Omega(n)}$.*

Proof. Selecting $P=\{p_1,\cdots,p_m\}$ uniformly at random, the expected number of white nodes is $\frac{n}{2}$. Using Chernoff bounds, the number of white nodes is at least $n/4$ with probability $1-e^{-\Omega(n)}$. The same applies to the number of black nodes.

Since connecting white nodes to black nodes is costly, the lower level selects a permutation which forms a chain of white nodes and a chain of black nodes connected to each other by only two edges of type D to form a cycle.

Let p_1 be the selected node of the costly cluster V_1. If p_1 is initially white, the lower level places it at one border between the black chain and the white chain to avoid using one of the edges of type C. This situation is illustrated in Figure 3-a. If p_1 is initially black, then the initial solution would look like Figure 3-b, in which the costly cluster is placed somewhere in the black chain. Here we present two auxiliary claims, which will be used in the rest of the proof of Theorem 6.

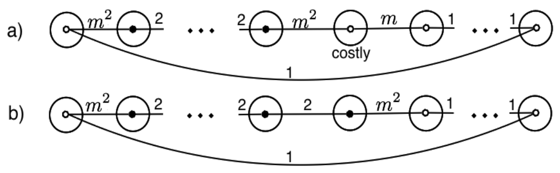


Figure 3: The initial solution for G_2 if a) A white node is selected for the costly cluster. b) A black node is selected for the costly cluster.

Claim 7. *Starting with a random initial solution, with probability $1 - e^{-\Omega(n)}$ for all the clusters $V_i, 2 \leq i \leq m$; a change from black to white is improving while no change from white to black is improving.*

Proof. As mentioned earlier, a random initial node set has both kinds of nodes with probability $1 - e^{-\Omega(n)}$; therefore, the exact solver of the lower level forms a chain of black nodes and a chain of white nodes. Changing a black node $p_i, i \neq 1$ to white results in shortening the chain of black nodes by removing an edge of type B and cost 2, while the chain of white nodes gets longer by adding an edge of type A and cost 1. The new solution is hence improved in terms of fitness and accepted by the algorithm. On the other hand, the opposite move increases the cost of the solution; therefore in a cluster $V_i, i \neq 1$ a change from white to black is not accepted.

The number of selected white nodes for clusters $V_i, i \neq 1$ never decreases; therefore, at all time during the run of the algorithm we have both chains of black nodes and white nodes, until all the black nodes change to white. □

Claim 8. *As long as there is at least one cluster V_i, $i \neq 1$ for which the black node is selected, a change from white to black is accepted for cluster V_1 and the opposite change is rejected.*

Proof. Since there is at least one cluster $V_i, i \neq 1$, for which the black node is selected, we know that the current solution and the new solution both have a chain of black nodes and a chain of white nodes. If the white node of cluster V_1 is selected in the current solution, changing it to black shortens the chain of white nodes by removing the edge of type C and increases the number of black nodes by adding an edge of type B. This move is accepted because the new solution is improved in terms of cost. The result is illustrated in Figure 3-b. Using similar arguments, if the black node of cluster V_1 is selected in the current solution, changing it to white is rejected because it increases the cost. □

Using Claim 7 we can conclude that all nodes $p_i, i \neq 1$ are gradually changed to white in NEN-LS. As long as at least one node $p_i, i \neq 1$ is black, a mutation from white to black for p_1 is accepted, and this node remains black. When all other nodes are changed to white, if p_1 is black at this point, it is connected to two white nodes with edges of type D and cost m^2 as illustrated in Figure 4-a. If it changes to white, these two edges are removed and two edges of type C and cost m are added to the solution (Figure 4-b). This change is accepted because two edges of cost m are less costly than two edges of cost m^2.

This eventually results in a local optimum with all white nodes selected. The algorithm needs to traverse the clusters on the upper level only twice which gives $O(m)$

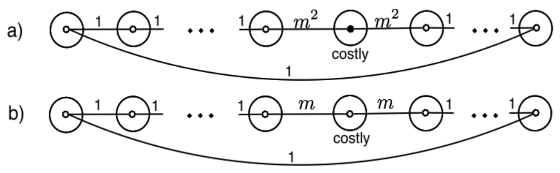


Figure 4: a) All other clusters change to white one by one. b)Local Optimum for G_2.

iterations on the upper level for the algorithm to get stuck in a local optimum. In the first traversal, for all the clusters the white node will be selected except for the costly cluster, V_1. In the second traversal, the white node will be selected for V_1 as well (Figure 4-b). This completes the proof of Theorem 6. □

3.3 Benefits of VNS

In this section we introduce an instance of the problem for which both of the mentioned neighbourhood search algorithms fail to find the optimal solution. Nevertheless, the combination of these approaches as described in Algorithm 4 finds the global optimum.

We consider the undirected complete graph G_3 shown in Figure 5 which has 6 clusters each containing $n/6$ nodes. There are three kinds of nodes in this graph: white, grey and black. The first cluster consists of $n/12$ black, $n/24$ white, and $n/24$ grey nodes. All other clusters contain $n/12$ white and $n/12$ black nodes. We refer to the set of white, black and grey nodes of cluster V_i by V_{iW}, V_{iB}, andV_{iG}, respectively.

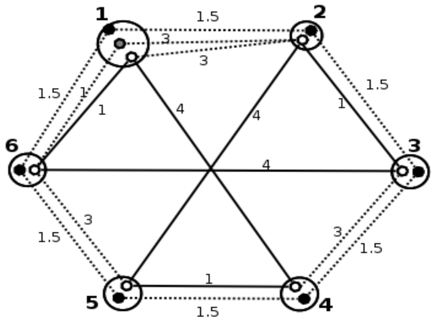


Figure 5: Graph G_3 showing one node of each type for each cluster and omitting edges of cost 100.

There are 5 types of edges in this graph, 4 of which are quite similar to the 4 types of the instance in Section 3.1. The other type, named type D below, includes the edges between two consecutive black nodes with a cost of 1.5.

- Type A: Edges of this type have a cost of 1.

$$A = \{\{v_i, v_j\} \mid \quad (v_i \in V_{1W} \cup V_{1G} \wedge v_j \in V_{6W}) \vee (v_i \in V_{2W} \wedge v_j \in V_{3W}) \vee (v_i \in V_{4W} \wedge v_j \in V_{5W})\}$$

- Type B: Edges of this type have a cost of 3.

$$B = \{\{v_i, v_j\} \mid \quad (v_i \in V_{1W} \cup V_{1G} \wedge v_j \in V_{2W}) \vee (v_i \in V_{3W} \wedge v_j \in V_{4W}) \vee (v_i \in V_{5W} \wedge v_j \in V_{6W})\}$$

- Type C: Edges of this type have a cost of 4.

$$C = \{\{v_i, v_j\} \mid \quad (v_i \in V_{1W} \wedge v_j \in V_{4W}) \vee (v_i \in V_{2W} \wedge v_j \in V_{5W}) \vee (v_i \in V_{3W} \wedge v_j \in V_{6W})\}$$

- Type D: Edges of this type have a cost of 1.5.

$$D = \{\{v_i, v_j\} \mid (v_i \in V_{kB} \wedge v_j \in V_{(k+1)B}\ ,\ 1 \leq k \leq 5) \vee (v_i \in V_{6B} \wedge v_j \in V_{1B})\}$$

- Type F: Edges of this type have a large cost of 100. All edges other than those of type A or B or C or D in this complete graph are of Type F. Note that the edges between grey nodes of the first cluster and the white nodes of the fourth cluster are also of this type.

$$F = \quad E \setminus \{A \cup B \cup C \cup D\}$$

We now show that an optimal solution visits a black node from each cluster in consecutive or reverse-consecutive order. Then in Theorem 10, we show that the algorithms CBLS and NEN-LS may get stuck in local optimums.

Property 9. *The optimal solution for the graph G_3 is visiting all black nodes with the consecutive or reverse-consecutive order.*

Proof. There are three kinds of nodes in this graph; white, grey and black. Any solution that contains black and one other kind of node has at least two edges of type F and weight 100 which makes the total cost of that solution more than 200. A solution that visits all black nodes in consecutive or reverse-consecutive order has 6 edges of type D and a total cost of 9. On the other hand, if we consider only white and grey nodes, our graph is the same as the instance of Section 3.1 with the optimal solution of cost 12. Therefore, visiting all black nodes with the cost of 9 is the optimal solution. □

Theorem 10. *Starting with a spanning node set P consisting of only white nodes and the permutation $\pi = (1, 4, 5, 2, 3, 6)$, CBLS and NEN-LS get stuck in a local optimum of G_3.*

Proof. We first show that the mentioned initial solution is a local optimum for CBLS. The cost of this solution is 15 which is less than any of the edges between black nodes and white or grey nodes. Therefore, any solution consisting of black and another kind of node, cannot be accepted. If we do not consider the black nodes and their edges, then G_3 is similar to G_1, and according to Theorem 2, starting with the initial permutation, no improvements can be achieved with Algorithm 1. Particularly, permutation $\pi' = (1, 2, 3, 4, 5, 6)$ is not achievable by searching the 2-opt neighbourhood of the initial solution. A solution consisting of black nodes is less costly only if they are visited in the optimal order of $\pi' = (1, 2, 3, 4, 5, 6)$ which is proved not to be achievable by CBLS.

Now we investigate the behaviour of NEN-LS which performs a local search based on the Node-Based approach for this instance. We show that this algorithm finds another locally optimal solution. Starting with the initial solution that is specified in the theorem, all black nodes can not be selected in one step and trying any one of the black nodes is rejected, because using two edges of type F are inevitable which makes the solution worse than the initial solution. The only spanning node set left in the NEN has the grey node of the first cluster. For this selection of nodes, the 2-opt TSP solver of the lower level finds the optimal order of clusters similar to what we described in the proof of Theorem 3 of Section 3.1 which form a solution of cost 12. From this point any Node-Exchange-Neighbourhood search fails to find a better solution. □

Using a variable-neighbourhood search that combines the two hierarchical approaches, we are able to escape these local optima. In the following, we show that VNS obtains an optimal solution when starting with the same solution as investigated in Theorem 10.

Theorem 11. *Starting with a spanning node set P consisting only of white nodes and the $\pi = (1, 4, 5, 2, 3, 6)$, VNS obtains an optimal solution in time $O(n^3)$.*

Proof. This approach is supposed to start with Cluster-Based algorithm and alternate between the two algorithms whenever CBLS is stuck in a locally optimal solution. As we saw, from the initial solution, Algorithm 1 can not find any better solutions, because the initial solution is a local optimum for that algorithm. Finding this out requires searching all the 2-opt neighbourhood which can be done in constant time, because the number of clusters is fixed. Then NEN-LS manages to find another solution with the permutation of $\pi' = (1, 2, 3, 4, 5, 6)$. This can also be done in polynomial time as we described in Theorem 3 of Section 3.1. Then CBLS uses this as a starting solution. As $\pi' = (1, 2, 3, 4, 5, 6)$ is an optimal permutation the optimal set of nodes P consisting of all black nodes is found in time $O(n^3)$ on the lower level. □

The investigations of this section have pointed out that a combination of the two hierarchical approaches into a variable neighbourhood search is beneficial, because each approach helps escape local optimum of the other approach.

4 Simple Evolutionary Algorithms

A simple evolutionary algorithm with the Cluster-Based approach for solving GTSP has been studied in [3] and a hard instance is presented there to prove the exponential lower bound on the runtime of that algorithm which holds with high probability. In this section, we analyse the behaviour of Node-Based (1+1) EA presented in Algorithm 5 on that instance (Section 4.1). Moreover, we find a lower bound for optimisation time of Node-Based (1+1) EA in Section 4.2. Our analysis gives an exponential lower bound on the optimization time of the upper level; therefore, implies exponential time even if the lower level is solved efficiently.

4.1 Behaviour of Node-Based (1+1) EA on the Hard Instance of Cluster-Based (1+1) EA

In this section, we show that the hard instance for Cluster-Based (1+1) EA introduced in [3] can be solved in polynomial time by the Node-Based approach. Moreover, we perform experiments in Section 4.1.2, which confirm the theoretical results of this section.

The hard instance of Cluster-Based (1+1) EA [3] is illustrated in Figure 6. In this instance, there are m clusters and all of them comprise two nodes; a white node which represents the suboptimal node, and a black node which is the optimal node. All the white nodes are connected to each other with edges of cost 1, except for the white nodes of consecutive clusters (shown in the picture) which are connected with edges of cost 2. All the edges between a black node and a white node have a cost of m^2. All edges between black nodes also have a cost of m^2, except the ones that connect consecutive clusters (shown in the picture) which have a small cost of $\frac{1}{m}$.

The optimal node selection is to select all the black nodes and the optimal permutation of clusters is a clockwise or anti-clockwise order of them. The cost of edges between black nodes and white nodes in this permutation are $\frac{1}{m}$ and 2 respectively.

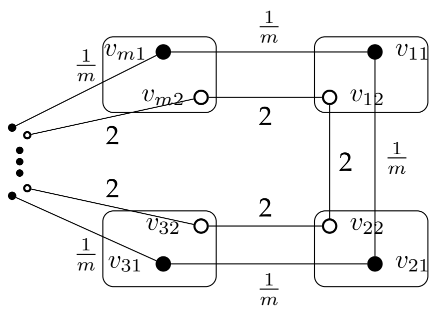


Figure 6: Illustration of G_G, hard instance of Cluster-Based (1+1) EA [3]

Algorithm 6: A Lower Level TSP Solver

```
1: Consider $G'$ a graph consisting of nodes of $G[P]$ with no edges
2: Add the edges of $G[P]$ that have a cost of $1/m$ to $G'$
3: Find a shortest path visiting all white nodes in $G[P]$ and add the edges of that to $G'$.
4: Use edges of cost $m^2$ to make a Hamiltonian cycle out of the paths that are formed in $G'$ and the disconnected black nodes
```

Therefore, optimal solution will consist of all $\frac{1}{m}$ edges and it is shown that the local optimum is selecting all white nodes in an order which does not have any of the 2-weighted edges. For this instance of the problem, it is proved in [3] that with an overwhelmingly high probability, the proposed Cluster-Based (1+1) EA needs exponential time to find the optimal solution.

4.1.1 Theoretical Analysis of Node-Based (1+1) EA on G_G

Here we prove that, with probability $1 - o(1)$, G_G can be solved in polynomial time by the Node-Based approach. We call this probability a high probability, since by definition, $o(1)$ approaches 0 when the input size approaches infinity. In order to prove this, we first need to analyse how an optimal TSP tour can be found on the lower level of this approach. Although solving TSP in general is NP-hard, it can be solved in polynomial time for the instances induced by picking one node of each cluster of the graph G_G. Algorithm 6 provides such a method. In step 3 of this algorithm, if the number of white nodes is at most 3, finding the shortest path can be done by checking all configurations. If the number of white nodes is more than 3, only edges of cost 1 will be used in the shortest path since all white nodes are connected to $m-2$ other white nodes with a cost of 1. Finding this can be done by a depth-first-search and checking all configurations of connecting the last 4 nodes of the path. Therefore step 3 needs time $O(m)$ to find the shortest path on white nodes. Since the required time for other steps of the algorithm is also at most $O(m)$, we can conclude that Algorithm 6 runs in time $O(m)$.

To prove that Algorithm 6 finds the optimal tour with respect to the spanning set fixed on the upper level, we first present two properties on the solutions of lower level. Then in Lemma 14 we show that Algorithm 6 finds the optimal tour.

Property 12. *Let w be the number of white nodes selected on the upper level. If $2 \leq w \leq 3$ and all the selected white nodes are from consecutive clusters, then Step 3 of Algorithm 6 uses one edge of cost 2 (and one edge of cost 1 in case $w = 3$). Otherwise, it only uses edges of cost 1.*

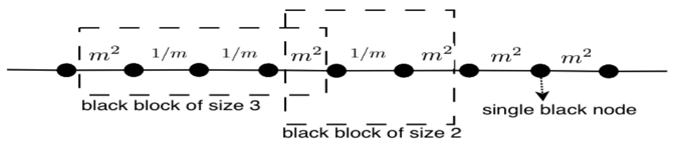


Figure 7: Blocks of black nodes

Property 13. *Let $C(S)$ denote the total cost of a solution S. Also, let Y and X be two solutions with r and s edges of weight m^2 respectively. If $r > s$ then we have $C(Y) > C(X)$.*

Lemma 14. *Let w and $r = m - w$ be the number of white and black nodes selected on the upper level, respectively. Moreover, let s be the number of black nodes where the selected node in proceeding cluster with respect to the optimal solution is also black. Algorithm 6 finds an optimal tour with total cost of*

- $s \cdot \frac{1}{m} + (m - r - 1) + (r - s + 1) \cdot m^2$*; if conditions of Property 12 do not hold*
- $s \cdot \frac{1}{m} + (m - r) + (r - s + 1) \cdot m^2$*; if conditions of Property 12 hold*

Proof. There are r black nodes in the spanning set; therefore, in order to make a Hamiltonian cycle, at least $r + 1$ edges that are connected to these nodes are required. Since all edges connected to black nodes, except for s edges of cost $1/m$, are of cost m^2, at least $r + 1 - s$ edges of cost m^2 are needed, and refusing to select any of edges of cost $1/m$, increases this number. Moreover, according to Property 13, the optimal solution of the lower level has a minimum number of m^2-edges. Therefore, the lower level has to select all edges of cost $1/m$, which is done in step 2 of the algorithm.

On the other hand, in order to minimise the number of white-black connections which are of cost m^2, all white nodes need to form one chain which is done is step 3 of the algorithm. This chain will be connected to two black nodes from its two ends. If conditions of Property 12 do not hold, then only edges of cost 1 are used in forming the white chain. Otherwise, one edge of cost 2 is also required. Therefore, the cost of forming the white chain is $m - r - 1$ in the former case, and $m - r$ in the latter case.

So far, we have formed some chains of black nodes and one chain of white nodes. In order to connect these chains together, we have to use $r + 1 - s$ edges of weight m^2, which is done in step 4 of the algorithm. Summing up, the optimal tour on the selected set of nodes consists of s edges of weight $\frac{1}{m}$and $r - s + 1$ edges of weight m^2. Furthermore, if conditions of Property 12 hold, it contains $m - r - 2$ edges of cost 1 and one edge of cost 2; otherwise, it contains $m - r - 1$ edges of cost 1. All together, these edges give the total cost as stated in the lemma. □

From now on, we only consider the number of iterations on the upper level. Note that the lower level uses Algorithm 6 which adds only a factor of $O(m)$ to our analysis. We start analysing the behaviour of Node-Based (1+1) EA on G_G with a couple of definitions that helps us in describing a TSP tour that the lower level forms. In the following, w denotes the number of white nodes in the solution.

Definition 15. *A **black block of size** l, $l > 0$ an integer, is a path on exactly l consecutive black nodes, which consists of $l - 1$ edges of cost $1/m$.*

The two end nodes of a black block are connected to edges of cost m^2. Black blocks of size 1, 2 and 3 nodes are illustrated in Figure 7.

Definition 16. *A solution is critical if $3 \leq w \leq 4$ and all the selected white nodes are from consecutive clusters.*

Note that a one-bit flip on a white node of a critical solution results in either a solution with a greater number of black blocks, or a solution that fulfils conditions of Property 12. In the rest of this section we prove that with high probability, in time $O(m^2)$ the algorithm either finds the optimal solution, or reaches a critical solution. From a critical solution, we prove that a 2-bit flip can make an improvement, and with high probability, in time $O(m^2 \log m)$ the optimal solution is found. Lemmata 19 and 22 prove the upper bound if we do not face a critical solution, and Lemma 23 investigates the behaviour of the algorithm, otherwise. Lemmata 17 and 18 help us with the proof of Lemma 19.

Lemma 17. *Other than a situation where $w = 3$ and all selected white nodes are from consecutive clusters, w can only increase in a step in which the number of m^2-edges decreases.*

Proof. Having $r = m - w$ black nodes, Lemma 14 gives the total cost of a solution as $s \cdot \frac{1}{m} + (m-r-1) + (r-s+1) \cdot m^2$ when conditions of Property 12 do not hold. Here s is the number of edges of weight $\frac{1}{m}$, $m-r-1$ is the number of edges of weight 1 (that connect white nodes), and $(r-s+1)$ is the number of edges of weight m^2. When w increases, the number of edges of weight 1 increases, and since the total number of edges stay the same, either s has to decrease or $r-s+1$. Decreasing s cannot compensate the increase in the total cost that is caused by adding new edges of weight 1. Therefore, in order to prevent an increase in the total cost, $r-s+1$, which is the number of m^2-edges, has to decrease.

For the situation where $w = 2$ and the selected white nodes are from consecutive clusters, according to Lemma 14 the total cost is $\frac{m-3}{m} + 2 + 2 \cdot m^2$. Observe that all solutions with $w \geq 4$ have a larger cost. For $w = 3$, the solution either needs more than 2 m^2-edges, which is clearly more costly, or all three white nodes need to be from consecutive clusters. In this situation, Lemma 14 gives the total cost as $\frac{m-4}{m} + 3 + 2 \cdot m^2$ which is also more costly and rejected by the algorithm. □

Lemma 18. *In a phase of Cm^2 steps, C a constant, if we do not face a critical solution, with probability $1 - o(1)$, the sum of all increments on the number of white nodes is at most $5m$.*

Proof. From Lemma 17 we know that the number of white nodes can increase only when the number of blocks reduces. Since the number of blocks is bounded by m, this can only happen in at most m steps. At each of those steps, either two black blocks are merged or a black block mutates to white, and some additional nodes may also mutate. We here prove that with high probability no blocks of size larger than 3 mutate in this phase, which results in at most $3m$ white to black mutations. We also prove that the number of additional nodes that mutate at the same steps is with high probability bounded by $2m$. Therefore, we find that in this phase, the sum of all increments on the number of white nodes is at most $5m$.

At each step, each cluster is selected for a mutation with probability $\frac{1}{m}$, and its white node is selected with probability $\frac{1}{2}$. Therefore, the probability that a block of size at least 4 mutates to white in one step is at most $\frac{1}{(2m)^4}$. Since the number of blocks is bounded by m, the probability that at least one block of size at leas 4 mutates to white at one step is at most $\frac{1}{16m^3}$. Hence, the probability that at least one of them mutates in a phase of $C \cdot m^2$ steps is $O(\frac{1}{m})$. Therefore, with probability at least $1 - o(1)$, no black block of size 4 or more mutates to white. In other words, all blocks that mutate to white in a phase of $C \cdot m^2$ steps are of size at most 3. This implies that at most $3m$ nodes can belong to the blocks that mutate from black to white in a phase of $C \cdot m^2$ steps.

However, at each step that the number of blocks is reduced, some additional nodes may also mutate to white. Let X_{ij} be a random variable such that $X_{ij} = 1$ if node j is selected for mutation at step i. Note that we only need to consider the steps in which the number of black blocks is reduced, because according to Lemma 17 a mutation from black to white is not accepted in other steps. Since the number of blocks is bounded by m, there are at most m steps in which the number of blocks reduce. The expected value of $X = \sum_{i=1}^{m} \sum_{j=1}^{m} X_{ij}$ is $E[X] = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{1}{m} = m$ and by Chernoff bounds we get $\text{Prob}(X \geq 2m) \leq e^{-\Omega(m)}$. Therefore, with probability $1 - e^{-\Omega(m)}$ at most $2m$ additional nodes mutate during the steps at which the number of black blocks is reduced, and with probability $(1 - o(1))(1 - e^{-\Omega(m)}) = 1 - o(1)$ at most $2m$ additional nodes mutate during the considered phase. As a result, together with at most $3m$ black to white mutations, we find that with probability $1 - o(1)$ at most $3m + 2m$ black nodes mutate to white in a phase of $C \cdot m^2$ steps. □

Lemma 19. *If we do not face a solution with no black nodes or a critical solution, then with probability* $1 - o(1)$*, in time* $24em^2$ *a solution with* $w = 0$ *is found.*

Proof. According to Lemma 18, with probability $1 - o(1)$ during a phase of $C \cdot m^2$ steps, C a constant, at most $5m$ black nodes turn into white. Since the number of white nodes in the initial solution is at most m, at most $6m$ steps of increasing the number of black nodes is sufficient for reaching a situation with $w = 0$.

While the number of black nodes is at least one and we have not reached $w = 0$ or a critical solution yet, there is always at least one white node that if it mutates to black, the length of a black block increases. This move is accepted by the algorithm, because it shortens the white path by removing an edge of cost 1, while adds one edge of cost $\frac{1}{m}$ to the black block. At each step, the node of each cluster is mutated to white with probability $\frac{1}{2m}$. Therefore, the probability that only the mentioned mutation happens at one step is at least $\frac{1}{2 \cdot m} \cdot \left(1 - \frac{1}{m}\right)^{m-1} \geq \frac{1}{2em}$, where $\left(1 - \frac{1}{m}\right)^{m-1}$ is the probability that no other mutations happen at that step.

Let $X = \sum_{i=1}^{T} X_i$, where X_i is a random variable such that $X_i = 1$ if a white node mutates to black at step i, and $X_i = 0$ otherwise. At each step i, before reaching $w = 0$ or a critical solution, $\text{Prob}(X_i = 1) \geq \frac{1}{2em}$. Considering a phase of $T = 24em^2$ steps, by linearity of expectation we get $E[X] \geq 24em^2 \cdot \frac{1}{2em} = 12m$. Using Chernoff Bounds we get $\text{Prob}\left(X \leq (1 - \frac{1}{2})12m\right) \leq e^{-\Omega(m)}$. As a result, in a phase of $24em^2$ steps, we either find a solution with $w = 0$, or with probability $1 - e^{-\Omega(m)}$ at least $6m$ white nodes mutate to black, which results in a situation with $w = 0$ because $6m$ is an upper bound on the number of white to black mutations. Overall with probability $1 - o(1)$, the a solution with $w = 0$ is reached in time $24em^2$. □

Lemma 20. *The initial solution, chosen uniformly at random, has at least* $\frac{m}{48}$ *single black nodes with probability* $1 - e^{-\Omega(m)}$

Proof. Considering the consecutive clusters with respect to their optimal permutation, for any specific cluster, a black (or white) node may be selected for its following cluster with a probability of $1/2$. As a result, any selection of nodes in 3 consecutive clusters can happen with probability $(1/2)^3$. There are at least $m/3$ separate sets of consecutive clusters; therefore, the expected number of single black nodes is at least $\frac{m}{3 \cdot 8}$. Using

Chernoff bounds and considering X to be the number of single black nodes in the initial solution, we have: $P(X < (1-1/2)\frac{m}{3\cdot 8}) \leq e^{-\frac{m}{3\cdot 8}\cdot\frac{1}{8}}$

As a result, with a probability $1 - e^{-\Omega(m)}$ the initial solution has at least $\frac{m}{48}$ single black nodes as described. □

In the proof of the next lemma, we use the Simplified Drift Theorem (Theorem 21) presented in [17, 18]. Consider a random variable X_t, $t \geq 0$ with positive values that is changed in a stochastic process. Also consider an interval of $[a, b]$, $a \geq 0$. The simplified drift theorem shows that the lower limit of the interval is not reached by X_t with high probability, if the starting point is above b, the average drift of the value of the random variable is positive, and the probability of having big changes on it is small. In this theorem, F_t denotes a filtration on states. In the proof of Lemma 22, we analyse the changes on the size of a large black block, and the filtration is done according to the steps where an accepted change happens on the size of that block.

Theorem 21. *(Simplified Drift Theorem [18]) Let X_t , $t \geq 0$, be real-valued random variables describing a stochastic process over some state space. Suppose there exist an interval $[a, b] \subseteq \mathbb{R}$, two constants $\delta, \varepsilon > 0$ and, possibly depending on $l := b - a$, a function $r(l)$ satisfying $1 \leq r(l) = o(l/\log(l))$ such that for all $t \geq 0$ the following two conditions hold:*
1. $E[X_{t+1} - X_t \mid F_t \wedge a < X_t < b] \geq \varepsilon$,
2. $\text{Prob}(|X_{t+1} - X_t| \geq j \mid F_t \wedge a < X_t) \leq \frac{r(l)}{(1+\delta)^j}$ for $j \in \mathbb{N}$.
Then there is a constant $c^* > 0$ such that for $T^* := min\{t \geq 0 : X_t \leq a | F_t \wedge X_0 \geq b\}$ it holds $\text{Prob}(T^* \leq 2^{c^* l/r(l)}) = 2^{-\Omega(l/r(l))}$.

Lemma 22. *With probability $1 - o(1)$, the number of black nodes is at least one during $24e \cdot m^2$ steps of the Node-Based (1+1) EA.*

Proof. Let r be the number of all black blocks in the solution. From Lemma 20 we know that with high probability, the initial solution consists of at least $\frac{m}{48}$ single black nodes. As a result, in the initial solution $r = \Omega(m)$.

In order to reach a solution in which all nodes are white, the number of black blocks needs to reduce. Let's consider the step when for the first time $r \leq m^\epsilon$, where $0 < \epsilon < 1$ is a small constant. At this step, $r \geq \frac{m^\epsilon}{2}$; otherwise, at least $\frac{m^\epsilon}{2}$ mutations have to had happened at one step which is exponentially unlikely.

We first show that we either have a block of size greater than one at this stage, or we will reach such a situation. Let us assume that all of the blocks at this stage are of size one. For any single black node, there exist two adjacent white nodes that can extend the size of that block, by mutating to black. The probability that a white node is selected and mutated to black is $\frac{1}{2m}$; therefore, with probability $P_1^+ \geq \frac{2}{2\cdot e\cdot m}$ the size of that block is extended. On the other hand, the probability that this single black node mutates to white is $P_1^- \leq \frac{1}{2\cdot m}$. Therefore, if a change happens on the size of this block, it would be an increase with probability at least

$$P_{1N}^+ = \frac{P_1^+}{P_1^+ + P_1^-} \geq \frac{\frac{1}{em}}{\frac{1}{em} + \frac{1}{2m}} \geq \frac{2}{2+e}$$

Therefore, the probability that none of these blocks experience an increase in the size when they change for the first time, is $\left(1 - \frac{2}{2+e}\right)^r \leq \left(1 - \frac{2}{2+e}\right)^{\frac{m^\epsilon}{2}} = e^{-\Omega(m^\epsilon)}$. As a result, with probability $1 - e^{-\Omega(m^\epsilon)}$, we reach a stage at which there are $r \leq m^\epsilon$ blocks and one of the blocks is of size at least 2. We refer to this block as the large block.

Now we show that in a phase of $m^{1+\epsilon}$ we reach this stage. Since each single black node has a probability of $P_1^+ + P_1^- \geq \frac{1}{em}$ to change at each step, the expected number of steps that is required to make a change on each single black node is at most em. Therefore, by Markov's inequality we know that the probability of not changing each single black node in a phase of $2em$ is at most $\frac{1}{2}$. Considering a phase of $m^{1+\epsilon}$ steps, we see that the probability of not changing each single black nodes is at most $e^{-\Omega(m^\epsilon)}$. There are at most m^ϵ single black nodes, and by union bound we can see that with probability at most $m^\epsilon \cdot e^{-\Omega(m^\epsilon)} = e^{-\Omega(m^\epsilon)}$ at least one of them does not change. Therefore, with probability $1 - e^{-\Omega(m^\epsilon)}$ all these nodes face a change in the mentioned phase.

For a black block of size $l \geq 2$, there is a probability of $P_l^+ \geq \frac{2}{2 \cdot e \cdot m}$ that a white node mutates to black and extends the size of that block. But to decrease the size of the block, either the whole block needs to mutate at one step (with probability at most $\frac{1}{(2m)^l}$), or one improving move needs to happen somewhere else at the same step that a black node of either end of the large block is mutating to white (with probability at most $\frac{2}{2m}$). An improving move can be a mutation on a white node that extends a black block, which happens with probability at most $\frac{2}{2m}$ for each block, or a mutation on all black nodes of a block, the probability of which is upper bounded by $\frac{1}{2m}$ for each block. Since the number of blocks is at most m^ϵ, the probability of an improving move to happen, is at most $\frac{2 \cdot m^\epsilon}{2m} + \frac{m^\epsilon}{2m}$. Overall, the probability of decreasing the size of the large block is

$$P_l^- \leq \frac{1}{(2m)^l} + \frac{2}{2m} \cdot \left(\frac{2 \cdot m^\epsilon}{2m} + \frac{m^\epsilon}{2m} \right) \leq \frac{1}{(2m)^l} + \frac{1}{m} \cdot \frac{3 \cdot m^\epsilon}{2m} \leq \frac{4m^\epsilon}{2m^2}$$

Now consider a phase of $m^{\frac{3}{2}}$ steps. With probability at most $\frac{4m^\epsilon}{2m^2} \cdot m^{\frac{3}{2}} = \frac{2m^\epsilon}{\sqrt{m}}$ the size of the large block is decreased at least once. Therefore, with probability $1 - O(m^{\epsilon - 1/2}) = 1 - o(1)$ its size is not decreased in the mentioned phase.

On the other hand, there is a probability of at least $P_l^+ \geq \frac{1}{e \cdot m}$ at each step, that the size of the block is increased. Let X_i be a random variable such that $X_i = 1$ if the size of large block is increased at step i, and $X_i = 0$ otherwise. The expected number of increases in the size of that block in a phase of $m^{\frac{3}{2}}$ steps is $\sum_{i=0}^{m^{3/2}} X_i \geq \frac{\sqrt{m}}{e}$. Moreover, by Chernoff bounds we have $\sum_{i=0}^{m^{3/2}} X_i \geq \frac{\sqrt{m}}{2e}$ with probability at least $1 - e^{-\Omega(\sqrt{m})}$, which means with probability $1 - o(1)$, the size of the large block is at least $\frac{\sqrt{m}}{2e}$ after a phase of $m^{\frac{3}{2}}$ steps.

After this phase, we consider a phase of $24e \cdot m^2$ steps and show that with high probability, the large black block does not lose more than half of its nodes. In order to show this, we use the Simplified Drift Theorem [17, 18] presented in Theorem 21. Let t_0 be the first step after the previous phase has finished and let L be the largest block at that time. We define X_t, $t \geq 0$, as

$$\begin{aligned} X_t := & \quad \text{size of } L \text{ at } t_0 \\ + & \quad \text{the number of steps increasing size of } L \text{ from } t_0 \text{ until } t_0 + t \\ - & \quad \text{the number of nodes removed from } L \text{ from } t_0 \text{ until } t_0 + t. \end{aligned}$$

Note that X_t always represents a lower bound on the size of L at time $t + t_0$. We filter the steps and only consider the relevant steps, i.e. the steps in which a change happens on the size of L. Moreover, we set $a = \frac{X_0}{2}$, $b = X_0$, $r = 1$, $\varepsilon = \frac{1}{4e}$ and $\delta = 1$.

Earlier, we found an upper bound on P_l^- and a lower bound on P_l^+. An upper bound on the latter is $P_l^+ \leq \frac{2}{2m}$, because in order to increase the size of a black block, at least one of the two white neighbours of it need to mutate to black. Using these bounds, we get upper and lower bounds for $P_{rel} = P_l^+ + P_l^-$, the probability of each step to be a relevant step:

$$\frac{1}{e \cdot m} \leq P_{rel} \leq \frac{1}{m} + \frac{4m^\epsilon}{2m^2} \leq \frac{2}{m}$$

The last inequality holds for sufficiently large m, because $\epsilon < 1$. At each step, with probability at least $\frac{1}{em}$, an increase happens on the size of L; hence, the positive drift on X_t is $\frac{1}{em}$. Considering conditional probability, at each relative step, the probability of an increase on the size of L is $\frac{1/em}{P_{rel}}$. Therefore, the positive drift on X_t in the relevant steps is:

$$\Delta^+ \geq \frac{1}{em} . \frac{1}{P_{rel}} \geq \frac{1}{em} \cdot \frac{m}{2} \geq \frac{1}{2e}$$

Similarly, the expected decrease in the number of black nodes of that block, in the relevant steps is

$$\Delta^- \leq \left(\frac{l}{(2m)^l} + \left(\sum_{k=1}^{m} k \cdot \frac{k+1}{(2m)^k} \right) \cdot \left(\frac{2 \cdot m^\epsilon}{2m} + \frac{m^\epsilon}{2m} \right) \right) \cdot \frac{1}{P_{rel}}$$

$$\leq \left(\frac{1}{(2m)^l} + \frac{2}{m} \cdot \frac{3 \cdot m^\epsilon}{2m} \right) \cdot \frac{1}{P_{rel}} \leq \frac{4m^\epsilon}{m^2} \cdot \frac{1}{P_{rel}} \leq \frac{4m^\epsilon}{m^2} \cdot em \leq \frac{4em^\epsilon}{m}$$

where k is the number of black nodes that are removed from the large block, and $\frac{k+1}{(2m)^k}$ is the probability of such mutations happening in one step. Here, $k+1$ is the number of possible ways that L can lose k nodes, since all these nodes have to be taken from the two ends of the block. Also, $\frac{1}{(2m)^k}$ is the probability that those nodes mutate to white. Moreover, $\sum_{k=1}^{m} k \cdot \frac{k+1}{(2m)^k} = \frac{1}{m} + \frac{2\times3}{(2m)^2} + \frac{3\times4}{(2m)^3} + \cdots + \frac{m\times(m+1)}{(2m)^m} \leq \frac{1}{m} + \frac{1}{2m} + \frac{1}{2^2m} + \cdots + \frac{1}{2^{k-1}m} + \cdots + \frac{1}{2^{m-1}m} \leq \frac{2}{m}$ holds for $m \geq 3$. Using Δ^- we find the total expected difference of

$$E[X_{t+1} - X_t \mid F_t \wedge a < X_t < b] = \Delta^+ - \Delta^- \geq \frac{1}{2e} - \frac{4em^\epsilon}{m}$$

Therefore, the first condition of the simplified drift theorem holds for an appropriate choice of ε. The second condition also holds because at each step X_t can be increased by at most 1 and the probability of decreasing it by j is

$$\text{Prob}(X_t - X_{t+1} \geq j \mid F_t \wedge a < X_t) \leq \frac{j+1}{(2m)^j} \cdot \frac{1}{P_{rel}} \leq \frac{1}{2^j}.$$

Therefore, the conditions of simplified drift theorem hold and we get

$$\text{Prob}(T^* \leq 2^{c^* \cdot X_0/2}) = 2^{-\Omega(X_0/2)},$$

As a result, with probability $1 - 2^{-\Omega(\sqrt{m})}$, the size of the large block does not decrease to less than a, in a phase of $c \cdot m^2$ steps. Overall, with probability $1 - o(1)$, the number of black blocks is at least one during the mentioned phase. □

Lemma 23. *From a critical solution, with probability $1 - o(1)$, the optimal solution is reached in time $O(m^2 + \log m)$, where $\epsilon > 0$ is a constant.*

Proof. The cost of a critical solution with $w = 3$ is $\frac{m-4}{m} + 3 + 2 \cdot m^2$, and it can be observed from Lemma 17 that all solutions with $w \geq 5$ have a larger cost and can not replace this solution. Therefore, only a solution with $w \leq 2$ or a critical solution with $w = 4$ can replace this solution which can be obtained by a 1-bit flip.

From a critical solution with $w = 4$, the number of white nodes does not increase, because there are only two m^2-edges which connect black and white chains in this situation, and according to Lemma 17, in order to increase the number of white nodes, all black nodes need to mutate to white at one step, which is exponentially unlikely. Moreover, a non-critical solution with the same number of white nodes is not accepted after a critical solution either, because if the selected white nodes are not from consecutive clusters, more than two m^2-edges are required in the tour.

Here we show that there exists a 2-bit flip in a critical solution that reduces the number of white nodes by two, and results in a solution with one chain of $n - 2$ black nodes and one chain of 2 white nodes. From that solution, similar to our argument in the above paragraph, increasing the number of white nodes is exponentially unlikely, and according to Lemma 19, with probability $1 - o(1)$, the optimal solution is found in time $O(m^2)$.

From Lemma 14 we know that the cost of a critical solution is $(m-5) \cdot \frac{1}{m} + 3 + 2 \cdot m^2$. By flipping two white nodes of one end of the white chain, conditions of Property 12 hold and the cost of the new solution is $(m - 3) \cdot \frac{1}{m} + 2 + 2 \cdot m^2$, which is better than the cost of the critical solution with respect to the fitness function. Therefore, this solution is accepted by the algorithm. This move has a probability of $\frac{1}{m^2}$. Therefore, the expected time until it happens is m^2, and with probability at least $\frac{1}{2}$ it happens in $2 \cdot m^2$ steps. Considering $\log m$ phases of $2 \cdot m^2$ steps, by Markov's inequality we get that with probability $1 - (\frac{1}{2})^{\log m} = 1 - \frac{1}{m}$, this 2-bit flip happens in time $O(m^2 \log m)$, which completes the proof. □

Theorem 24. *Starting from an initial solution chosen uniformly at random, the Node-Based (1+1) EA finds the optimal solution of G_G in time $O(m^2 \log m)$ with probability $1 - o(1)$.*

Proof. Lemma 22 shows that in a phase of $c \cdot m^2$, $c = 24e$, steps, the number of black nodes does not decrease to 0 with probability $1 - o(1)$. Therefore, due to Lemma 19, if we do not face a critical solution, the optimal solution is found in time $O(m^2)$ with probability $1 - o(1)$. Moreover, if we face a critical solution, according to Lemma 23, with probability $1 - o(1)$ it takes $O(m^2 \log m)$ additional steps to find the optimal solution. Overall, with probability $1 - o(1)$, in time $O(m^2 \log m)$, the optimal solution is found by the Node-Based (1+1) EA. □

4.1.2 Experimental Results

In this section, we present experimental results that confirm our theoretical analysis on the behaviour of Nodes-Based (1+1) EA optimizing G_G. We have run the algorithm for instances of different sizes, and we have done that 30 times with a maximum of 10^7 iterations for each instance. The results are summarised in Table 1. The first and second columns indicate the input size and the percentage of runs that result in the optimal solution. The average and the maximum number of iterations until finding this solution are presented in the third and fourth columns. Observe that $\%100$ of all runs find the optimal solution and the maximum runtime is in $O(m^2)$, which confirm the theoretical results.

Table 1: Experimental results of Node-Based (1+1) EA on G_G

Input Size (m)	**%Optimum**	**Average Runtime**	**Maximum Runtime**
20	100	495	3954
50	100	1873	10800
100	100	4076	19252
200	100	27817	190882
500	100	33280	280873
1000	100	110186	2518061

4.2 Lower Bound Analysis for Node-Based (1+1) EA

In this section we prove an exponential lower bound on the optimization time of Node-Based (1+1) EA. In Section 4.2.1 we introduce an instance of the Euclidean GTSP, G_S, that is difficult to solve by means of our algorithm and discuss some geometric properties of it. In Section 4.2.2 we show how the algorithm reaches a local optimum in our instance and discuss how it can reach the global optimum after reaching the local optimum. Consequently we find a lower bound for the optimization time of the algorithm. In Section 4.2.3, using an efficient algorithm that solves the lower lever problems of this instance, we present some experimental results that confirm the obtained lower bound.

4.2.1 A Hard Instance and its Geometric Properties

The hard instance presented in this section, which is partly illustrated in Figure 8, is composed of m clusters. Let $a > 1$ be a constant. Only $\frac{m}{a}$ of these clusters have one node. Other clusters contain m nodes which makes the total number of nodes $n = m(m-\frac{m}{a})+\frac{m}{a}$. All nodes are connected to each other and the cost of travelling between them is their Euclidean distance.

In the clusters that have m nodes, $m-1$ nodes are placed on the small circle and are shown by a star in the picture. We refer to them as white nodes or inner nodes. For simplicity we assume that the inner nodes of each cluster all lie on the same position. The same result can be obtained by placing the nodes within a small circle having an arbitrarily small radius ϵ. The remaining node of each cluster, shown black in the picture, is placed on the larger circle. Other $\frac{m}{a}$ clusters do not have any nodes on the small circle and have only one black node on the larger circle. The figure demonstrates how the clusters are distributed on the two circles. The arc between black nodes of two consecutive clusters subtend an angle of $\frac{2\pi}{m}$, while the arc between two consecutive one-node clusters subtend an angle of $a \cdot \frac{2\pi}{m}$.

If we represent the radius of inner and outer circles by r and r' respectively, then a black node and a white node have distance at least $r'-r$ and the length of edges between two adjacent black nodes is $2r'\sin(\frac{\pi}{m})$. The minimum length of edges between two black nodes of one-node clusters is also quite similar to previous formula with a greater angle: $2r'\sin(\frac{\pi}{\frac{m}{a}})$.

Here we define a characteristic for the introduced instance, which is required for proving the exponential lower bound on the optimization time of Node-Based (1+1) EA. This characteristic shows the ratio of r to r' and is defined by the following inequality.

$$r < \frac{1}{2}\left(2\sin\left(\frac{\pi}{m}\right) - \sin\left(\frac{2\pi}{m}\right)\right) r' \tag{1}$$

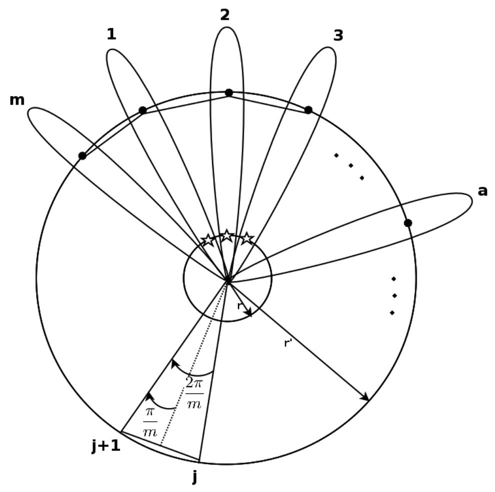


Figure 8: Euclidean hard instance, G_S, for Node-Based (1+1) EA

We now prove that if the introduced instance has this characteristic, then for $m \geq 8a$, the best tour on any spanning set that has at least one white node, contains only 2 edges between outer and inner circles. We also show that the optimal solution consists of all the black nodes, but with high probability, the Node-Based (1+1) EA reaches a plateau of local optimums with $\frac{m}{a}$ black nodes and $m - \frac{m}{a}$ white nodes. Note that in such local optimums, selecting $\frac{m}{a}$ black nodes is a must, since there's no other choice for those clusters. The local optimum in this instance is a basin of attraction since the distance between white nodes are smaller than the distance between the black nodes.

In the proof of the following property, we have used a couple of theorems from [20]. Given a set of vertices, it is stated in Theorem 1 of that paper that the shortest spherical or planar polygon does not intersect itself. Moreover, Theorem 2 of that paper proves that the shortest polygon contains the vertices on the boundary of its convex hull in their cyclic order.

Property 25. *The best tour on a spanning set that has at least one white node, contains only two edges between nodes on the inner and outer circle for $m \geq 8a$.*

Proof. We first take into account the tour on a node set consisting of only the black nodes of one-node clusters. There is no other choice except selecting those nodes because their clusters have no other node. For such a node set, due to Theorem 2 of [20], the optimal tour is to visit all the nodes in the order they appear on the convex hull. This order will be respected in an optimal tour even if there are some inner nodes to visit as well, because according to Theorem 1 in [20] the optimal solution cannot intersect itself. In other words, if some white nodes are selected in the upper level, while visiting the outer nodes with respect to their convex hull order, a solution occasionally travels the distance between outer circle and inner circle to visit some inner nodes, and then travels roughly the same distance back to the outer circle, to continue visiting the remaining outer nodes. As illustrated in Figure 9, this can be done generally in two ways:

1. Case 1: Leaving the outer circle only once and visiting all inner nodes together.

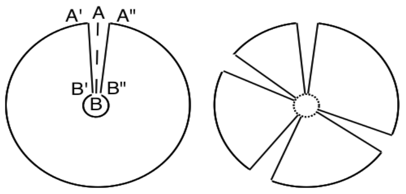


Figure 9: Left side: Case 1, Right side: Case 2

2. Case 2: Leaving the outer circle more than once and visiting some of the inner nodes each time.

We now show that there exists a solution for Case 1 that is less costly than all the solutions of Case 2. As a result, the best tour on a spanning set with at least one white node travels the distance between two circles only twice.

If we represent the number of times a tour leaves the outer circle to visit some nodes in the inner circle with k, then for the solutions of Case 1, $k = 1$ and for solutions of Case 2, $k \geq 2$. For both cases the number of edges connecting the two circles is $2k$. The picture at the left side of Figure 9 illustrates a solution with $k = 1$ for which we find an upper bound of the tour cost as the following:

$$C(1) < 2\pi r' + 2\pi r + 2(r' - r) \tag{2}$$

The last part of this formula is two times of the length of edge AB which is a direct line from the inner circle to the outer circle along their radius. The lengths of edges $A'B'$ and $A"B"$ are actually more than that because their ends are not from same clusters. Nevertheless, formula 2 presents an upper bound of the total cost of the tour, because we are considering the complete circumference of both circles. In other words, the distance between A and A' is included in the circumference of the large circle and the distance between B and B' is included in that of the small circle and according to quadrilateral inequality $|A'B'| < |A'A| + |AB| + |BB'|$.

On the other hand, a lower bound of the tour cost in all solutions of Case 2 is:

$$C(k) > \left(\frac{m}{a} - k\right) 2\sin\left(\frac{\pi}{m/a}\right) r' + 2k(r' - r) \tag{3}$$

In the above formula, $2\sin(\frac{\pi}{m/a})r'$ is the length of the edges connecting two consecutive clusters with one black node. These edges are the longest edges that can be removed from the tour when we add two edges connecting inner and outer circles. There are initially at least m/a of these edges and in this formula we have omitted k of them from the tour.

We can rewrite the right side of inequality 3 as:

$$C(k) > \left(\frac{m}{a}\right) 2\sin\left(\frac{\pi}{m/a}\right) r' - k \cdot 2\sin\left(\frac{\pi}{m/a}\right) r' + 2k(r' - r).$$

Since for $m > 8a$, $\sin(\frac{\pi}{m/a}) < 0.39$ and $(\frac{m}{a})\sin(\frac{\pi}{m/a}) > 3.06$ the above expression is at least:

$$2(3.06)r' - 2k \cdot 0.39r' + 2k(r' - r)$$

This expression is monotone increasing in k when $r \leq 0.61r'$; therefore, setting $k = 2$ we get the smallest lower bound of $C(k)$ for $k \geq 2$:

$$C(k) > 4.56r' + 4(r' - r)$$

Now if we prove that the upper bound we found for $C(1)$ in Inequality 2 is less than the above expression, we can then conclude that $C(1) < C(k)$ for $k \geq 2$. Therefore, we should prove that:

$$\begin{aligned}
&2\pi r' + 2\pi r + 2(r' - r) \leq 4.56r' + 4(r' - r) \\
&\Leftrightarrow 2\pi r' + 2\pi r \leq 4.56r' + 2r' - 2r \\
&\Leftrightarrow (\pi + 1)r \leq (-\pi + 3.28)r' \\
&\Leftrightarrow r \leq \frac{-\pi + 3.28}{\pi + 1} r' \approx 0.033r'
\end{aligned}$$

The latest inequality holds, because the constraint we introduced on the value of r in Equation 1 is quite tight and we can see that for $m \geq 8$ it gives us $r < 0.03r'$ which is a tighter bound for r than what the right side of equation above gives us. □

Property 26. *An optimal solution chooses all black nodes and visits them in clockwise or anti-clockwise order when $m \geq 7a$.*

Proof. The tour comprising all black nodes has a cost strictly less than $2\pi r'$ which is the length of the circumference of the circle with radius r'. Therefore, we can state that $2\pi r'$ is an upper bound on the cost of the optimal solution. Besides, in Property 25 we saw that the best tour when at least one white node is selected has only two edges connecting the two circles. Therefore, as a lower bound on the cost of a solution with any spanning set other than all black nodes, we can use Formula 3 with $k = 1$ and get

$$C(1) \geq \left(\frac{m}{a} - 1\right) 2 \sin\left(\frac{\pi}{\frac{m}{a}}\right) r' + 2(r' - r).$$

We here show that with the assumptions we have on the value of r, this lower bound is greater than the upper bound we found for the cost of optimal solution. By replacing r with its maximum value from Equation 1 we have:

$$C(1) \geq \left(2\left(\frac{m}{a} - 1\right)\sin\left(\frac{\pi}{\frac{m}{a}}\right) + 2 - \left(2\sin\left(\frac{\pi}{m}\right) - \sin\left(\frac{2\pi}{m}\right)\right)\right) r'$$

since for $m \geq 7a$

$$\left(\frac{m}{a} - 1\right)\sin\left(\frac{\pi}{\frac{m}{a}}\right) > 2.60$$

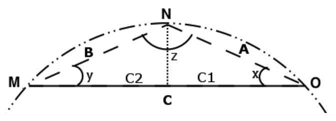
and for $m \geq 4$

$$\left(2\sin\left(\frac{\pi}{m}\right) - \sin\left(\frac{2\pi}{m}\right)\right) \leq 0.42$$

we can conclude that for $m \geq \max\{4, 7a\}$

$$C(1) \geq (2(2.60) + 2 - (0.42))r' = 6.78r' > 2\pi r'$$

As a result, for $m \geq 7a$ the minimum cost of such tours, is greater than $2\pi r'$ which is the maximum cost when all black nodes are selected. Hence the tour consisting of all black nodes is the optimal solution and since they comprise the convex hull the optimal Hamiltonian cycle on them would be visiting them in the order they appear in the convex hull. □



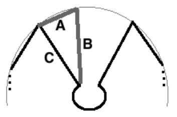


Figure 10: Left: Case 1, adding a new outer node between two outer nodes. Right: Case 2, adding a new outer node just before inner nodes

Property 27. *Let P and P' be non-optimal spanning sets and $P_{out} \subset P$ and $P'_{out} \subset P'$ be their subset of outer nodes. Moreover, let S and S' be optimal solutions with respect to P and P' respectively. If $P_{out} \subset P'_{out}$ and $|P'_{out}| = |P_{out}| + 1$ then $C(S) < C(S')$.*

Proof. The main idea behind this property is that distances in the inner circle are significantly shorter than distances in the large circle and if r is sufficiently smaller than r' (Inequality 1), any single mutation that replaces an inner node with the outer node of the same cluster, increases the cost of the whole tour.

According to Property 25, the permutation chosen in the lower level, has all the inner nodes listed between two black nodes. If one inner node is removed and one outer node is added, the part of total tour that includes all inner nodes gets shorter and the part that connects black nodes gets longer. The edges connecting the inner nodes are at most the size of the diameter of the inner circle. Therefore, the maximum decrease for removing an inner node is upper bounded by $4r$. In the following, we find the minimum increase for adding a black node.

We analyse the increase in two cases. The first case is illustrated in the left picture of Figure 10 in which the new black node is placed between two black nodes in the tour. N is the new node and M and O are its neighbours. The edge connecting M and O will be removed from the tour and the two other edges in the triangle will be added. If we show the length of these edges by C, A and B respectively and the cost of the tour before and after this change by C_{old} and C_{new}, then:

$$C_{\text{new}} = C_{\text{old}} - C + A + B$$

So the increase caused by this change would be:

$$d = C_{\text{new}} - C_{\text{old}} = A + B - C$$

By splitting C with an orthogonal line from N we can write d as:

$$d = (A - C1) + (B - C2) \tag{4}$$

We claim that when A and B have their smallest values, d has also its smallest value. We assume $A' \geq A$ and $B' \geq B$ and show that the corresponding d' will be at least d. When $A' \geq A$ the arc between O' and N' is also greater than or equal to the arc between O and N. Therefore, the angle y', facing that arc, would be also greater than or equal to y. Besides, $C2 = B \cdot \cos(y)$, and all the things we said about $C2$ and y hold for $C1$ and x too. Altogether, we can write d' as:

$$d' = A' - A' \cdot \cos(x') + B' - B' \cdot \cos(y')$$

since $y' \geq y$ and $x' \geq x$, and all of them are acute angles

$$d' \geq A' - A' \cdot \cos(x) + B' - B' \cdot \cos(y)$$

If we represent A' by αA and B' by βB where α and β are real numbers greater than one, then we have:

$$d' \geq \alpha\left(A - A \cdot \cos(x)\right) + \beta(B - B \cdot \cos(y))$$
$$\Rightarrow d' \geq \alpha(A - C1) + \beta(B - C2)$$

By comparing d' with the value of d in Equation 4 it holds that $d' \geq d$.

The shortest edges on the outer circle are from two consecutive clusters and have a length of $A = B = 2\sin(\pi/m)r'$. C has similarly the value of $2\sin(2\pi/m)r'$. As a result, the minimum increase in the convex tour will be:

$$d = A + B - C = 4\sin\left(\frac{\pi}{m}\right)r' - 2\sin\left(\frac{2\pi}{m}\right)r' \tag{5}$$

The second case is when the new node is added just before or after visiting the inner nodes, as illustrated in the right picture of Figure 10. In this case, comparing to the previous case, edge B is longer and the angle between A and B is closer to the right angle. The minimum length of A is also the same as the minimum length of that in previous case. Altogether, with quite similar explanation to what we had for Case 1, the minimum increase in this case is larger than that in Case 1. Therefore the minimum increase in the convex tour, d, which is found in 5 is also less than the minimum increase in Case 2 and can be used for both cases.

On the other hand, as mentioned earlier the maximum decease caused by removing an inner node is $4r$. Therefore the total increase of the tour cost is at least

$$4\sin\left(\frac{\pi}{m}\right)r' - 2\sin\left(\frac{2\pi}{m}\right)r' - 4r$$

From our assumption on the value of r in Equation 1 we can find that the above expression has a positive value; therefore, $C(S) < C(S')$. □

4.2.2 Runtime Analysis

In this section, we give a lower bound on the runtime of Node-Based (1+1) EA. We start by presenting a Lemma about the initial solution that is chosen uniformly at random. Then we introduce the Multiplicative Drift Theorem [4] which is used in our analysis of Lemma 30 to upper bound the time of reaching a locally optimal solution. Then we discuss the main theorem of this section.

Lemma 28. *The initial solution with a spanning set that is chosen uniformly at random, has at least* $0.9\left(1 - \frac{1}{a}\right)(m-1)$ *white nodes with probability* $1 - e^{-\Omega(m)}$.

Proof. For $m - \frac{m}{a}$ clusters that have m nodes, the probability of selecting one of white nodes is $\frac{m-1}{m}$. Therefore the expected number of selected white nodes is

$$E[X] = \left(m - \frac{m}{a}\right)\left(\frac{m-1}{m}\right) = \left(1 - \frac{1}{a}\right)(m-1)$$

By Chernoff bounds we can have:

$$\text{Prob}\left(X < (0.9)(1 - \frac{1}{a})(m-1)\right) \leq e^{-0.005(1-\frac{1}{a})(m-1)} = e^{-\Omega(m)}$$

Therefore, the probability that the initial solution has at least $(0.9)(1 - \frac{1}{a})(m-1)$ white nodes is $1 - e^{-\Omega(m)}$. □

Theorem 29 (Multiplicative Drift [4]). *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum s_{min}. Let $\{X^{(t)}\}_{t \in \mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $X^{(t)} = 0$. Suppose that there exists a real number $\delta > 0$ that*

$$E\left[X^{(t)} - X^{(t+1)} \mid X^{(t)} = s\right] \geq \delta s$$

holds for all $s \in S$ with $\mathrm{Prob}[X^{(t)} = s] > 0$.

Then for all $s_0 \in S$ with $\mathrm{Prob}[X^{(0)} = s_0] > 0$, we have

$$E[T | X^{(0)} = s_0] \leq \frac{1 + \ln(s_0/s_{min})}{\delta}$$

Lemma 30. *Starting with an initial solution chosen uniformly at random, with probability $1 - e^{-\Omega(m)}$, the Node-Based (1+1) EA reaches a local optimum on G_S in expected time of $O(m \ln m)$.*

Proof. For a solution $x^{(t)}$ at time t, we define $X^{(t)}$ to be the number of m-node clusters for which the outer node is selected. Note that this function, as required in Theorem 29, maps the local optimum to zero and all other solutions to positive numbers.

If we assume that the number of m-node clusters that their outer node is chosen in solution $x^{(t)}$ is k, we can find the expected number of that for $x^{(t+1)}$ as follows:

As mentioned in Property 27, if only one mutation operation happens to increase the number of outer nodes, it will increase the cost and the algorithm will refuse it. Therefore, if only one mutation happens that is accepted by the algorithm, it has to change a node from the outer circle to the inner circle and decrease X^t by 1. The probability of this event is at least

$$p_1 = k\left(\frac{1}{m}\right)\left(\frac{m-1}{m}\right)\left(1 - \frac{1}{m}\right)^{m-1} \geq \frac{k}{m}\left(\frac{m-1}{m}\right)\left(\frac{1}{e}\right).$$

In the above formula, $\frac{1}{m}$ is the probability of mutation for any of the nodes in the spanning set and $\frac{m-1}{m}$ is the probability that the new selected node for the mutated cluster is a white node. We need one of k clusters to mutate and all others to stay unchanged. In other words, $(1 - \frac{1}{m})^{m-1}$ in the above formula is the probability of $m - 1$ clusters to stay unchanged.

On the other hand, in some situations, some (one or more) mutations in the opposite direction can happen beside a mutation from outer circle to inner circle. If we want X^t to increase by one, then at least two mutations must happen to change a node from inner circle to outer circle. The probability of this event is at most

$$p_{-1} = \frac{k}{m}\left(\frac{m-1}{m}\right)\binom{m - \frac{m}{a} - k}{2}\left(\frac{1}{m}\right)^2\left(\frac{1}{m}\right)^2 \leq \frac{k}{m}\left(\frac{1}{2!m^2}\right).$$

In the above formula, $(\frac{k}{m})(\frac{m-1}{m})$ is the probability of one node to change from outer circle to inner circle. Then we have the number of different ways we can select two clusters that their selected nodes lies on the inner circle. The first $(\frac{1}{m})^2$ is the probability that the 2 selected clusters mutate and the second $(\frac{1}{m})^2$ is the probability that after mutation the node on the outer circle is selected in those 2 clusters.

Generally, for X^t to increase by q, we need at least one mutation from outer circle to inner circle and $q+1$ mutations in opposite direction and the probability of this even would be at most

$$\begin{aligned} p_{-q} &= \frac{k}{m}\left(\frac{m-1}{m}\right)\binom{m-\frac{m}{a}-k}{q+1}\left(\frac{1}{m}\right)^{q+1}\left(\frac{1}{m}\right)^{q+1} \\ &\leq \frac{k}{m}\left(\frac{1}{(q+1)!m^{q+1}}\right) \end{aligned}$$

As a result, the difference made in X^t by the next step would be at least

$$E[X^{(t)} - X^{(t+1)} \mid X^{(t)} = k] \geq p_1 - \sum_{q=1}^{m} q \cdot p_{-q}$$

By replacing the lower bound of p_1 and upper bounds of p_{-q}, we get

$$\begin{aligned} E[[X^{(t)} - X^{(t+1)} \mid X^{(t)} = k] &\geq \frac{k}{m}\left(\frac{m-1}{m}\right)\left(\frac{1}{e}\right) - \frac{k}{m}\left(\frac{1}{2!m^2}\right) - \dots \\ &\quad -m\frac{k}{m}\left(\frac{1}{(m+1)!m^{m+1}}\right) \\ &\geq \frac{k}{m}\left(\frac{m-1}{em} - \frac{1}{m^2} - \dots - \frac{1}{m^{m+1}}\right) \\ &\geq \frac{k}{m}\left(\frac{m-1}{em} - m \cdot \frac{1}{m^2}\right) \\ &\geq \frac{k}{m}\left(\frac{m-1-e}{em}\right) \end{aligned}$$

For $m \geq 4$ the expression $(\frac{m-1-e}{em})$ is at least $\frac{3-e}{4e}$. So setting $\delta = \frac{3-e}{4em}$ and using the Multiplicative Drift Theorem we find the expected time of reaching the local optimum as:

$$E[T \mid X^{(0)} = 0.1m] \leq \frac{1 + \ln(0.1m/1)}{\frac{3-e}{3em}} = O(m \ln m)$$

In the above formula we have assumed $X^{(0)} \leq 0.1m$ because from Lemma 28 we know that with probability $1-e^{-\Omega(m)}$, the initial solution has less than $0.1m$ black nodes other than the fixed black nodes. □

Theorem 31. *Starting with an initial solution chosen uniformly at random, if $m \geq 8a$, then the optimization time of the Node-Based (1+1) EA presented in Algorithm 5 on G_S is $\Omega\left((\frac{n}{2})^{m-\frac{m}{a}}\right)$ with probability $1 - e^{-\Omega(m^{\delta})}$, $\delta > 0$.*

Proof. In order to prove this theorem, we introduce a phase P in which

1. The algorithm reaches a local optimum with high probability
2. The algorithm does not reach the global optimum with high probability

Then we show that after this phase, only a direct jump from the local optimum to the global optimum will help the algorithm improve the results, probability of which is $\left(\frac{1}{m^2}\right)^{m-\frac{m}{a}}$.

As we saw in Lemma 30, the expected time of Node-Based (1+1) EA to reach the local optimum is $O(m \ln m)$. Let c be the appropriate constant, so that $c \cdot m \ln m$ is an upper bound on the expected time for reaching that local optimum. Now consider a phase of $2c \cdot m \ln m$ steps. If T is the actual time at which the local optimum is reached, by Markov's inequality we have: $\text{Prob}(T > 2c \cdot m \ln m) \leq \frac{1}{2}$. If we repeat this phase for $\frac{m^{\varepsilon}}{\ln m}$ times, $\varepsilon > 0$ a constant, then we get a phase of $P = 2c \cdot m^{1+\varepsilon}$ steps in which the probability of not reaching the local optimum is:

$$\text{Prob}(T > 2c \cdot m^{1+\varepsilon}) \leq \left(\frac{1}{2}\right)^{-\frac{m^{\varepsilon}}{\ln m}} = e^{-\Omega(m^{\delta})},$$

where $0 < \delta < \varepsilon$. As a result, the algorithm reaches the local optimum in phase P with probability $1 - e^{-\Omega(m^{\delta})}$. We here prove that in this phase, the algorithm does not reach the global optimum with probability $1 - e^{-\Omega(m^{\varepsilon})}$.

From Lemma 28 we know that with high probability the initial solution has not too many black nodes other than the fixed black nodes. Here we show that with high probability the number of these nodes does not increase significantly during the phase P; hence, the global optimum will not be reached. The probability of selecting each of the clusters for a mutation is $1/m$ and for clusters with m nodes, the probability of changing the selected node to the black node is $\frac{1}{m}$; therefore, at each step, the probability that each cluster's node is changed from one of its inner nodes to its outer node is $\frac{1}{m^2}$. For $m - \frac{m}{a}$ clusters, at each step the expected number of clusters that face such a mutation is at most $\frac{1}{m}$ and in a phase of $2cm^{1+\varepsilon}$ steps, is $2cm^{\varepsilon}$. If we define X as the number of clusters that will have a mutation like this, then by Chernoff bound we have

$$\text{Prob}(X \geq 3cm^{\varepsilon}) \leq e^{-2cm^{\varepsilon}(0.5)^2/3} = e^{-\Omega(m^{\varepsilon})}.$$

Therefore, with high probability, during the mentioned phase, at most $3cm^{\varepsilon}$ clusters will happen to have a mutation with the result of selecting their black node. Besides, from Lemma 28 we know that with probability $1 - e^{-\Omega(m)}$, the initial solution has at least $0.9(1 - \frac{1}{a})(m - 1)$ white nodes. Hence, with probability $e^{-\Omega(m^{\varepsilon})}$, the algorithm will not reach a state with less than $0.9(1 - \frac{1}{a})(m - 1) - 3cm^{\varepsilon}$ white nodes during phase P. As a result, the probability of having a direct jump to the global optimum in phase P is at most

$$2c \cdot m^{1+\varepsilon} \left(\frac{1}{m^2}\right)^{0.9(1-\frac{1}{a})(m-1)-3cm^{\varepsilon}} = m^{-\Omega(m)}.$$

Consequently, with high probability, the global optimum will not be reached during phase P. According to Property 27, no mutation from the inner circle to the outer circle can decrease the tour cost when the resulting solution is not the optimal solution. Hence, such a change may only be accepted by the algorithm when another mutation on the other direction happens at the same step. At the local optimum, there is no black node other than the fixed black nodes and no mutation from the outer circle to the inner circle can happen; therefore, a mutation from the inner circle to the outer circle can not happen either. As a result, after reaching a local optimum, only a direct jump to the global optimum can help moving towards the global optimum and the probability of such a jump is $\left(\frac{1}{m^2}\right)^{m-\frac{m}{a}}$. We now consider $(\frac{m^2}{2})^{m-\frac{m}{a}}$ steps following phase P. The probability of reaching the optimum solution is by union bound at most: $\left(\frac{m^2}{2}\right)^{m-\frac{m}{a}} \left(\frac{1}{m^2}\right)^{m-\frac{m}{a}} = (\frac{1}{2})^{m-\frac{m}{a}}$. Hence the probability of not reaching the global optimum in the mentioned phase is $1 - (\frac{1}{2})^{m-\frac{m}{a}} = 1 - e^{-\Omega(m)}$. Altogether, with probability $1 - e^{-\Omega(m^{\delta})}$, the optimization time is at least $\left(\frac{m^2}{2}\right)^{m-\frac{m}{a}}$. □

Table 2: Experimental results of Node-Based (1+1) EA on G_S

Input Size (m)	**$\%$LO**	**Average Runtime to Reach Local Optimum**	**Maximum Runtime to Reach Local Optimum**
20	100	15	69
50	100	36	275
100	100	66	346
200	100	82	770
500	100	197	1300
1000	100	680	3120

4.2.3 Experimental Results

In this section we include experimental results that confirm the exponential lower bound that we have proved in Section 4.2.2 for the optimization time of the studied instance. We try the algorithm with a maximum of 10^6 iterations on instances of 6 different input sizes, and we show that they all stick to the local optimum. For the lower level optimization we have developed an algorithm that visits all the selected black nodes in the order they appear on the large circle, then visits all selected white nodes in the order they appear on the small circle, and goes back to the first black node to form a Hamiltonian circuit. This algorithm assures Property 25 of the optimal lower level solution, which was important in our theoretical analysis. We have set $r = 1$ and $r' = 10^8$ so that the inequality of Equation 1 holds for the maximum input size that we are running the algorithm with. The results, based on 30 runs of the algorithm, are summarised in Table 2. The first and second columns indicate the input size and the percentage of runs that stick to the local optimum, respectively. The average and maximum number of iterations until finding the local optimum are presented in the third and fourth columns, respectively. As the table suggests, $\%100$ of the runs for all input sizes find the local optimum and stick to that until the maximum iteration number is reached. This confirms the theory results of Section 4.2.2.

5 Conclusion

Evolutionary algorithms and local search approaches have been shown to be very successful for solving the generalized travelling salesperson problem. We have investigated two common hierarchical representations together with local search algorithms and simple evolutionary algorithms from a theoretical perspective. In the first part of this paper, which is based on the conference version [19], the focus is on local search approaches. By presenting instances where they mutually outperform each other, we have gained new insights into the complimentary abilities of the two approaches. Furthermore, we have presented and analysed a class of instances where combining the two approaches into a variable-neighbourhood search helps to escape from local optima of the single approaches.

In the second part we have investigated the behaviour of hierarchical evolutionary algorithms for this problem. We have proved that there are instances which Node-Based (1+1) EA solves to optimality in polynomial time, while Cluster-Based (1+1) EA needs exponential time to find an optimal solution for them. Then we have presented a Euclidean instance of the GTSP to find an exponential lower bound on the optimization time of this algorithm. Our lower bound analysis for a geometric instance shows that the Euclidean case is hard to solve even if we assume that the lower level TSP is solved to optimality in no time.

References

[1] D. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. Concorde tsp solver. *http://www.math.uwaterloo.ca/tsp/concorde/index.html*, 2013.

[2] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., 2011.

[3] D. Corus, P. K. Lehre, F. Neumann, and M. Pourhassan. A parameterized complexity analysis of bi-level optimisation with evolutionary algorithms. *Evolutionary Computation (to appear), doi: 10.1162/EVCO_a_00147*, 2015.

[4] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 530 pp.

[6] M. Englert, H. Röglin, and B. Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. *Algorithmica*, 68(1):190–264, 2014.

[7] M. Fischetti, J. J. S. González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378, 1997.

[8] G. Gutin and A. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer-Verlag New York, Inc., 2007.

[9] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, pages 71.201–71.204, New York, NY, USA, 1961. ACM.

[10] B. Hu and G. R. Raidl. Effective neighborhood structures for the generalized traveling salesman problem. In J. I. van Hemert and C. Cotta, editors, *EvoCOP*, volume 4972 of *Lecture Notes in Computer Science*, pages 36–47. Springer, 2008.

[11] T. Jansen. *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Natural Computing Series. Springer, 2013.

[12] D. Karapetyan and G. Gutin. Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *European Journal of Operational Research*, 219(2):234–251, 2012.

[13] T. Kötzing, F. Neumann, H. Röglin, and C. Witt. Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence*, 6(1):1–21, 2012.

[14] S. Kratsch, P. K. Lehre, F. Neumann, and P. S. Oliveto. Fixed parameter evolutionary algorithms and maximum leaf spanning trees: A matter of mutation. In *Parallel Problem Solving from Nature - PPSN XI, 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*, pages 204–213, 2010.

[15] S. Kratsch and F. Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. *Algorithmica*, 65(4):754–771, 2013.

[16] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization:Algorithms and Their Computational Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[17] P. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386, 2011.

[18] P. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. Technical report, http://arxiv.org/abs/1211.7184, 2012.

[19] M. Pourhassan and F. Neumann. On the impact of local search operators and variable neighbourhood search for the generalized travelling salesperson problem. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 465–472, New York, NY, USA, 2015. ACM.

[20] L. V. Quintas and F. Supnick. On some properties of shortest hamiltonian circuits. *The American Mathematical Monthly*, 72(9):977–980, 1965.

[21] A. M. Sutton and F. Neumann. A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling. In *Proceedings of the Twelfth Conference on Parallel Problem Solving from Nature (PPSN 2012)*, pages 52–61. Springer, 2012.

[22] A. M. Sutton, F. Neumann, and S. Nallaperuma. Parameterized runtime analyses of evolutionary algorithms for the planar euclidean traveling salesperson problem. *Evolutionary Computation*, 22(4):595–628, 2014.

[23] M. Theile. Exact solutions to the traveling salesperson problem by a population-based evolutionary algorithm. In C. Cotta and P. Cowling, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*, pages 145–155. Springer Berlin Heidelberg, 2009.